3 Routing

So far, we have only looked at networks without dealing with the issue of how to send information in them from one node to another. The problem of sending information in a network is known as *routing*. Routing involves two basic activities:

- The determination of routing paths, and
- the transport of information groups (typically called *packets*) along the paths.

The first item is usually referred to as *path selection*, and the second item is usually called *packet switching*. In this section we only concentrate on the problem of selecting good paths. Strategies for sending packets along the paths will be discussed in the next section.

Certainly, if there is only one source-destination pair in the network that wants to exchange information (and all the edges in the network have the same capacity), then the best solution would be to connect them via a shortest path. But what about multiple source-destination pairs? Choosing a shortest path for each of them may lead to a high congestion and therefore a poor routing performance. For example, consider a complete binary tree in which the leaves are connected so that they form an $n \times n$-mesh. Then for most pairs of nodes the shortest path would lead via the root of the binary tree, but if many nodes want to exchange information, it is much better to use the mesh-edges instead because otherwise the root would become a highly congested point.

So it appears that a certain degree of coordination is necessary among the nodes to arrive at good paths. A naive strategy would be to simply collect information about all the messages that the nodes want to send out, and then to compute a best possible collection of paths for them. But this is certainly not practical in a large network. 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 computed *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 3.1 *An oblivious routing strategy is specified by a* path system $\mathcal{P}$ *and a* weight function $w : \mathcal{P} \rightarrow \mathbb{R}^+$ *with the property that for every source-destination pair* (s,t)*, the system of flow paths* $\mathcal{P}_{s,t}$ *for* (s,t) *in* $\mathcal{P}$ *fulfills* $\sum_{q \in \mathcal{P}_{s,t}} w(q) = 1$.

In the case of a flow problem, the weights indicate how a flow from s to t has to be split among the paths in $\mathcal{P}_{s,t}$, and in the case of a packet routing problem, the weights indicate the probability that a packet from a source s to a destination t chooses some particular path $p \in \mathcal{P}_{s,t}$.

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. 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.

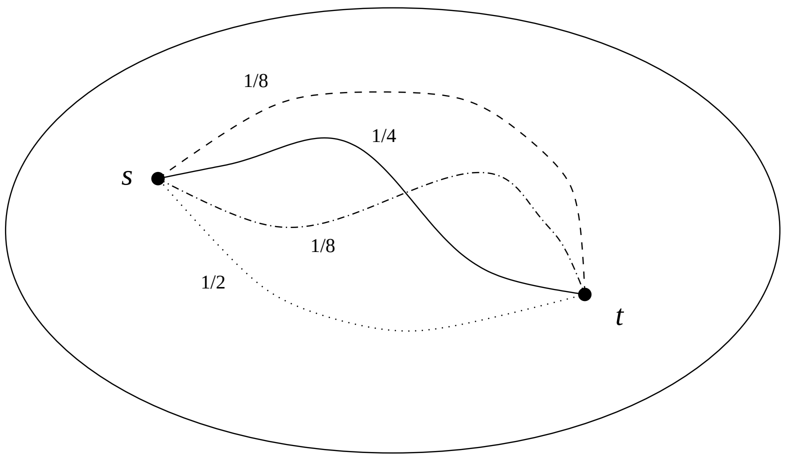

Figure 1: A system of optional paths for the pair (s,t). As can be easily checked, $\sum_{q\in\mathcal{P}_{s,t}} w(q) = 1$, i.e. the weight condition in Definition 3.1 is satisfied.

3.1 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 permutation routing problems? 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, a permutation routing problem can be specified by a permutation $\pi : V \to V$ on the set of nodes V.

Theorem 3.2 *The $x-y$ routing strategy can route arbitrary permutations in an $n \times n$-mesh of unit-capacity edges with congestion at most $2d$ and dilation at most d, where d is the maximum distance of a source-destination pair in the permutation.*

Proof. We only prove the theorem for the worst case, namely, that paths can have a length of up to $2n$. The general case will be an assignment.

Recall that in a permutation routing problem every node is the source and destination of a demand of exactly 1. Thus, every x-dimensional line in the mesh injects a total demand of at most n, and every y-dimensional line in the mesh has to absorb a total demand of at most n. When using the $x-y$ routing strategy, a total demand of at most n can therefore overlap at an edge in x-direction, and a total demand of at most n 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 $1/n$, and therefore the congestion is at most n. 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 $2n$. $\square$

Thus, when using the objective function behind the flow number, i.e. to minimize $\max\{C(S), D(S)\}$ over all feasible solutions S, then the $x - y$ routing strategy is optimal up to a factor of 2 because the congestion never exceeds the dilation by more than a factor of 2.

3.2 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.

Theorem 3.3 ([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 $\mathcal{P} = \{p_{i,j} : \ i, j \in [n]\}$ be any path system with exactly one path for every source-destination pair. 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.

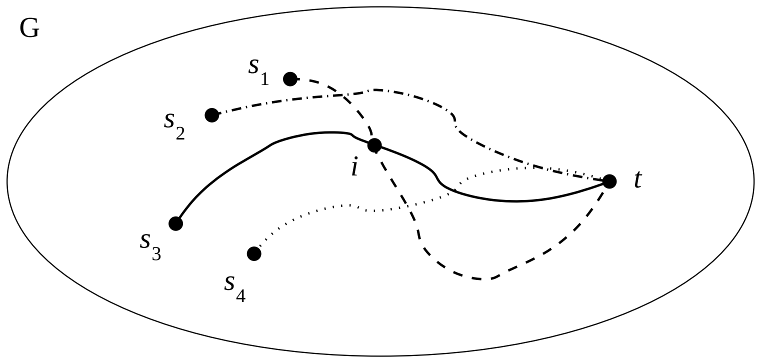

Figure 2: Illustration of the paths to k.

In the following, we will construct a permutation π with a high congestion. First we show that for every 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 3.4 *For every $t \in [n]$, $|A(t, z)| \geq \frac{n}{d \cdot z}$.*

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,

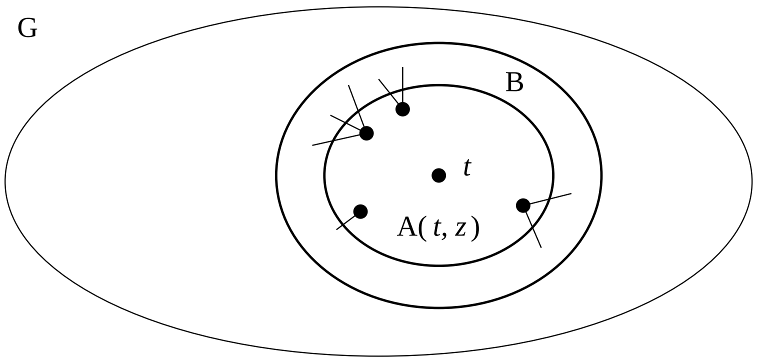

Figure 3: Illustration of $A(t,z)$ and B.

$|B| \cdot (z-1) \geq L$ and therefore

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

□

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

$$|X(z)| = \sum_{t \in [n]} |A(t,z)| \overset{\text{Lemma 3.4}}{\geq} n \cdot \frac{n}{d \cdot z} = \frac{n^2}{d \cdot z} \ .$$

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 $\Theta(\sqrt{n})$. Whereas this is fine for the 2-dimensional mesh, for networks with flow number $O(\log n)$ such as the butterfly this is unacceptably high, since we know from Section 3 that every BMFP and therefore also every permutation routing problem can be solved in the butterfly with congestion and dilation at most $O(\log n)$.

3.3 Valiant's Trick

We saw in Section 4.2 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 S be the best possible solution for the multicommodity flow problem underlying the definition of F, i.e. $F = \max\{C(S), D(S)\}$. From Theorem 3.12 we know that by applying 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 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 S from s to v and then along a flow path in S from v to t. (If there is more than one path from s to v resp. v to t in 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 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 S from s to v and then along the path in S from v to t.

To demonstrate the effect of this trick, let us consider routing in the d-dimensional hypercube. Suppose that for every source-destination pair $(s_{d-1}, \ldots, s_0), (t_{d-1}, \ldots, t_0) \in \{0,1\}^d$ we use 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. Let these paths form our path system S. Because S has exactly one path for every source-destination pair, it follows from Theorem 3.3 that there must be a permutation with at least $\sqrt{2^d/d}$ paths traversing a node and therefore a congestion of at least $(\sqrt{2^d/d})/d$ at an edge when using S directly. However, if we use Valiant's trick, we arrive at the following result.

Theorem 3.5 *Using Valiant's trick in the d-dimensional hypercube, any BMFP can be routed with congestion at most $2d$ and dilation at most $2d$.*

Proof. Since the d-dimensional hypercube has a diameter of d and S uses shortest paths, the dilation of Valiant's trick must certainly be at most $2d$. Thus, it remains to bound the congestion. First, we determine the number of paths S crossing any edge of the hypercube. Consider some fixed edge e, and suppose that e fixes dimension i for some $i \in \{0, \ldots, d-1\}$, i.e. it connects two nodes v and w in the hypercube that only differ at dimension i. When using the bit adjustment strategy, then there are 2^i possible sources that can reach v before crossing e in the direction of w, and 2^{d-i-1} possible destinations can be reached after crossing e. Also, there are 2^i possible sources that can reach w before crossing e in the direction of v, and 2^{d-i-1} possible destinations can be reached after crossing e. Hence, the number of all possible source-destination pairs whose paths cross e is equal to

$$2 \cdot 2^i \cdot 2^{d-i-1} = 2^d \ . \tag{1}$$

Now, it follows from the definition of the special BMFP $\mathcal{B}$ that the demand of every source-destination pair 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 $(1) \cdot (2) = d$, but every edge can only support a flow of 1. Hence, the maximum concurrent flow value f for $\mathcal{S}$ is $1/d$. This gives a congestion of d for S and therefore a congestion of $2d$ for Valiant's trick, because it doubles the overlap. □

In general, it follows from Theorem 3.12:

Theorem 3.6 *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 $2F$.*

In the case of actually sending packets instead of flows, $2F$ is an upper bound on the expected congestion caused by the packets in the network.

3.4 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.

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.

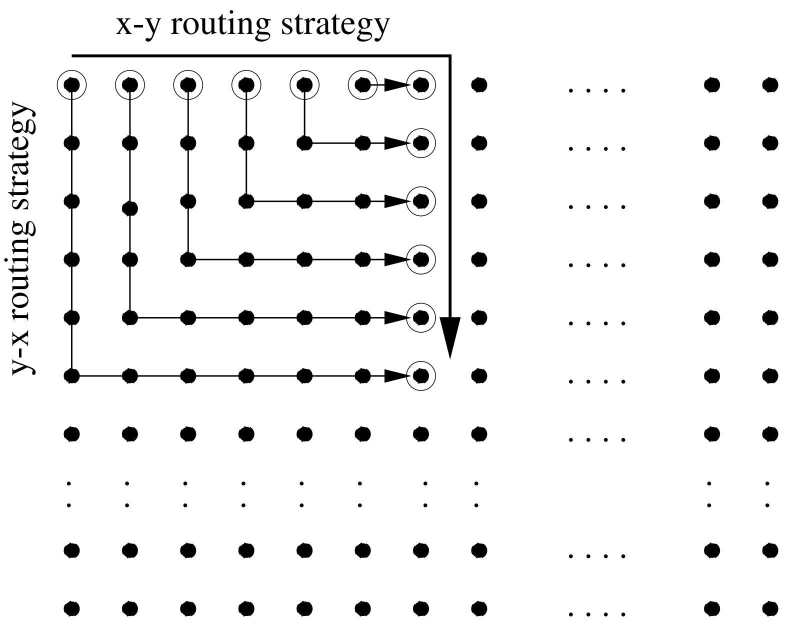

Figure 4: $x-y$ routing vs. $y-x$ routing on a mesh.

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_{OPT} be the best possible congestion and and D^P_{OPT} be the best possible dilation achievable for P (by possibly different solutions).

Theorem 3.7 *For any multicommodity flow problem P our recursive routing scheme has a congestion of $O(C^P_{\text{OPT}} \cdot \log n)$ and a dilation of $O(D^P_{\text{OPT}})$.*

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 $O(D^P_{\text{OPT}})$.

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 $O(C^P_{\text{OPT}})$. Since there are only $\log n$ different k, this results in a total congestion of $O(C^P_{\text{OPT}} \cdot \log n)$. 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

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

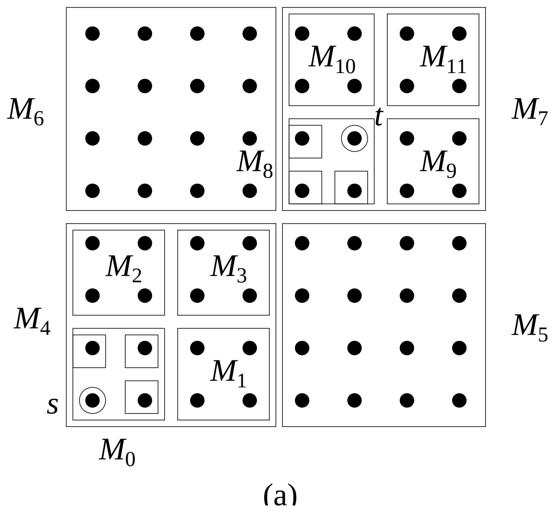

Figure 5: Recursive routing strategy from s to t. (a) illustrates the recursive decomposition into sub-meshes and (b) illustrates the distribution of flow from the shaded sub-mesh to the three other sub-meshes in its next higher mesh.

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. $C_{\text{OPT}}^P = \Omega(d/2^k)$. 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 $O(C_{\text{OPT}}^P)$, 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 $\Omega(C_{\text{OPT}}^P \cdot \log n)$ [2].

3.5 Routing in decomposable networks

The recursive routing technique for the mesh can also be used for other classes of networks. Given any graph $G = (V, E)$ with edge capacities specified by c, a *hierarchical decomposition tree* $T(G)$ of G is a binary tree in which every node v is associated with a subset V_v of V so that the following conditions are met:

- the root represents V,
- for every node u in $T(G)$ with two children v and w, $V_v \cap V_w = \emptyset$ and $V_v \cup V_w = V_u$, and
- for every leaf u in $T(G)$, u is associated with a single node in G.

The subgraph induced by a node u in $T(G)$ is the subgraph $G_u = (V_u, E_u)$ with $E_u = \{\{v,w\} \in E \mid v, w \in V_u\}$. As an example, the decomposition tree of a $n \times n$-mesh may look like in Figure 6.

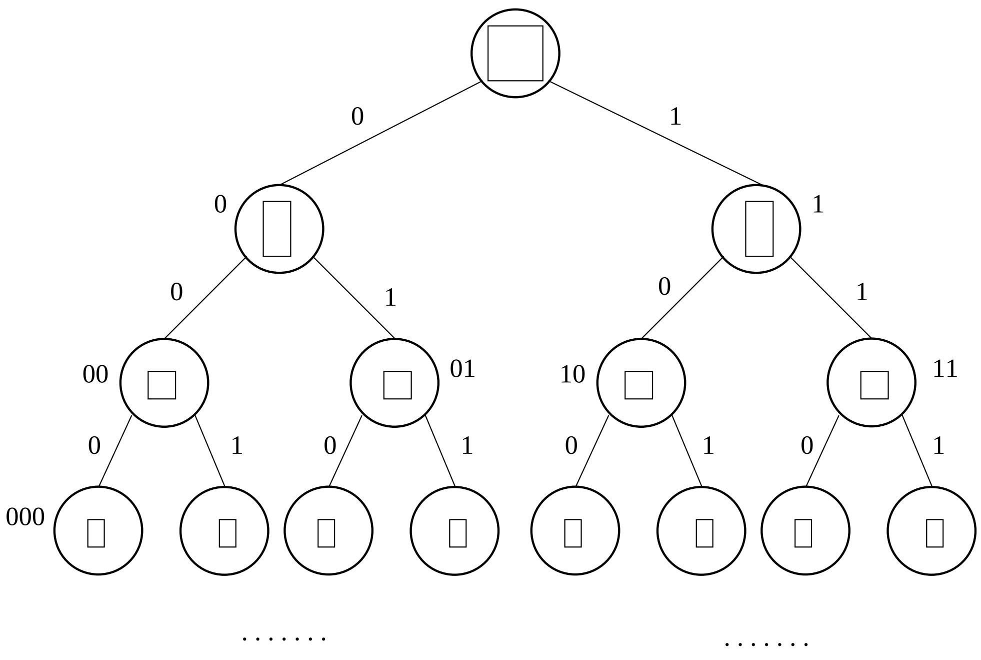

Figure 6: A possible decomposition tree for the $n \times n$-mesh. In every node, the subgraph induced by that node is shown.

We want to use the decomposition tree $T(G)$ in the following way when routing a flow of value f from a node v to a node w in G:

Suppose that we have an oblivious routing strategy R_u for every graph G_u with $u \in T(G)$. Let u_v be the node in $T(G)$ representing v and u_w be the node in $T(G)$ representing w, and let $(u_1 = u_v, u_2, \ldots, u_k = u_w)$ be the unique path from u_v to u_w in $T(G)$. For any $i \in \{1, \ldots, k-1\}$ let M_i be the multicommodity flow problem in which every pair of nodes (x, y) with $x \in V_{u_i}$ and $y \in V_{u_{i+1}}$ is associated with a demand of

$$d_{x,y} = f \cdot \frac{c_{G_{u_i}}(x) \cdot c_{G_{u_{i+1}}}(y)}{c_{G_{u_i}}(V_{u_i}) \cdot c_{G_{u_{i+1}}}(V_{u_{i+1}})}$$

where $c_{G'}(s)$ denotes the capacity of node (set) s in graph G'. Then every node $x \in V_{u_i}$ is the origin of $f \cdot c_{G_{u_i}}(x)/c_{G_{u_i}}(V_{u_i})$ flow and every node $y \in V_{u_{i+1}}$ is the destination of $f \cdot c_{G_{u_{i+1}}}(y)/c_{G_{u_{i+1}}}(V_{u_{i+1}})$ flow, as is easy to check. Hence, solutions to these flow problems can be concatenated to give a solution for sending a flow of f from v to w in G. In order to route the flow for M_i, the oblivious routing strategy R_{u_j} is used where $j = i+1$ if $V_{u_i} \subset V_{u_{i+1}}$ and otherwise $j = i$.

For the strategy above to work, G_u has to be connected for all nodes u in $T(G)$, and for the strategy to work well, every G_u should have a flow number that is as low as possible. This is where our definition of decomposable graphs comes in.

Definition 3.8 *We call a graph G* (nicely) decomposable *if G has a decomposition tree $T(G)$ so that for all nodes u in $T(G)$ with child v it holds for the flow number of G_u that*

$$F(G_u) = O\left(\frac{c_{G_v}(V_v)}{c_G(V_v, \bar{V}_v)}\right)$$

and $|V_v| = \Theta(|V_u|)$.

The $n \times n$-mesh, the hypercube, and the hypercubic networks presented in the previous section can be shown to satisfy these conditions. For these graphs the following result holds.

Theorem 3.9 *If G is decomposable then there is an oblivious routing strategy for G so that any multicommodity flow problem P can be routed with congestion $O(C^P_{\mathrm{OPT}} \log n)$.*

Proof. Suppose that $G = (V, E)$ is decomposable, and let $T(G)$ be the corresponding decomposition tree. For any node u in $T(G)$ let R_u be the oblivious routing strategy resulting from the optimal solution of the balanced multicommodity flow problem for $F(G_u)$ (recall the definition of the flow number).

Consider any multicommodity flow problem P and let (s,t) be any source-destination pair in P with demand d. Our aim will be to show that for every level of $T(G)$, the congestion caused by routing the demand d for (s,t) according to our routing strategy above is bounded by $O(C^{(s,t)}_{\mathrm{OPT}})$. If we can show this, then the congestion of routing P is bounded by $O(C^P_{\mathrm{OPT}})$ in every level. Due to the condition that $|V_v| = \Theta(|V_u|)$ in Definition 3.8, $T(G)$ can have only $O(\log n)$ levels and, hence, the overall congestion of our routing scheme is bounded by $O(C^P_{\mathrm{OPT}} \log n)$, as desired.

Consider any fixed pair (s,t) with positive demand d and let $(u_1 = s, u_2, \ldots, u_k = t)$ be the unique path from s to t in $T(G)$. Let us focus on some fixed stage $i \in \{1, \ldots, k-1\}$ of the routing, and let $v = u_i$ and $w = u_{i+1}$. Without loss of generality we assume that $V_v \subset V_w$ (for $V_w \subset V_v$ we just have to replace V_v with V_w and s with t below). Since $s \in V_v$ but $t \notin V_v$, any routing strategy for (s,t) must send the total demand of d out of G_v, which means that some edge e in the cut $(V_v, \bar{V}_v)$ must have a congestion of at least

$$\frac{d}{c_G(e)} \cdot \frac{c_G(e)}{c_G(V_v, \bar{V}_v)} = \frac{d}{c_G(V_v, \bar{V}_v)}$$

Thus, $C^{(s,t)}_{\mathrm{OPT}} = \Omega(d / c_G(V_v, \bar{V}_v))$. When using our oblivious routing strategy for stage i, we produce a congestion of at most

$$F \cdot \frac{d}{c_{G_v}(V_v)}$$

in G_w because the demands $d_{x,y}$ are defined as

$$d_{x,y} = d \cdot \frac{c_{G_v}(x) \cdot c_{G_w}(y)}{c_{G_v}(V_v) \cdot c_{G_w}(V_w)}$$

and the demands $d'_{x,y}$ of the balanced multicommodity flow problem for $F(G_w)$ are defined as

$$d'_{x,y} = \frac{c_{G_w}(x) \cdot c_{G_w}(y)}{c_{G_w}(V_w)} \geq \frac{c_{G_v}(x) \cdot c_{G_w}(y)}{c_{G_w}(V_w)} = \frac{d}{c_{G_v}(V_v)} \cdot d_{x,y}$$

For $F \cdot d/c_{G_v}(V_v)$ to be bounded by $O(d/c_G(V_v, \bar{V}_v)$ it must hold that

$$F = O\left(\frac{c_{G_v}(V_v)}{c_G(V_v, \bar{V}_v)}\right)$$

which is the case because of Definition 3.8. In this case, the congestion produced by our routing strategy for stage i is bounded by $O(C_{\mathrm{OPT}}^{(s,t)})$, which completes the proof. $\square$

One may ask whether good decompositions only exist for specific classes of graphs or whether any graph is decomposable (when allowing some additional polylogarithmic factor in our condition for the flow number in Definition 3.8). Surprisingly, Räcke has shown that this is always possible, implying the following result:

Theorem 3.10 ([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.