

Online Routing in Faulty Meshes with Sub-linear Comparative Time and Traffic Ratio

Stefan Rührup* and Christian Schindelbauer**

Heinz Nixdorf Institute, University of Paderborn, Germany
{sr, schindel}@uni-paderborn.de

Abstract. We consider the problem of routing a message in a mesh network with faulty nodes. The number and positions of faulty nodes is unknown. It is known that a flooding strategy like expanding ring search can route a message in the minimum number of steps h while it causes a traffic (i.e. the total number of messages) of $\mathcal{O}(h^2)$. For optimizing traffic a single-path strategy is optimal producing traffic $\mathcal{O}(p+h)$, where p is the perimeter length of the barriers formed by the faulty nodes. Therefore, we define the comparative traffic ratio as a quotient over $p+h$ and the competitive time ratio as a quotient over h . Optimal algorithms with constant ratios are known for time and traffic, but not for both. We are interested in optimizing both parameters and define the combined comparative ratio as the maximum of competitive time ratio and comparative traffic ratio. Single-path strategies using the right-hand rule for traversing barriers as well as multi-path strategies like expanding ring search have a combined comparative ratio of $\Theta(h)$. It is an open question whether there exists an online routing strategy optimizing time and traffic for meshes with an unknown set of faulty nodes. We present an online strategy for routing with faulty nodes providing sub-linear combined comparative ratio of $h^{\mathcal{O}\left(\sqrt{\frac{\log \log h}{\log h}}\right)}$.

1 Introduction

The problem of routing in mesh networks has been extensively studied. Even if the mesh contains faulty nodes, standard routing techniques can be applied. But as an *on-line problem*, where the location of faulty nodes is unknown, this task becomes challenging, because some communication overhead has to be invested for the exploration of the network and it is not clear, how much exploration is worth the effort. With this formulation of the problem, online route discovery gets related to graph exploration and motion planning problems. The definition of an appropriate measure for analyzing the efficiency and the design of an algorithm that beats known routing techniques is the contribution of this paper.

We consider the problem of routing (i.e. route discovery) in a mesh network with faulty nodes. If a node fails, then only the direct neighbors can detect this failure. The

* DFG Graduiertenkolleg 776 “Automatic Configuration in Open Systems”

** Supported by the DFG Sonderforschungsbereich 376: “Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen.” and by the EU within the 6th Framework Programme under contract 001907 “Dynamically Evolving, Large Scale Information Systems” (DELIS)

information about a failure can be spread in the network, but this produces unnecessary communication overhead, if there is no demand for routing a message. Therefore we concentrate on a reactive route discovery, which implies that the locations of the faulty nodes are not known in advance: The routing algorithm has no global knowledge and must solve the problem *online*, i.e. it can send a message in a promising direction but it does not know whether the target can be reached or the message gets stuck in a labyrinth of faulty nodes. As groups of faulty nodes can obstruct a path leading to the target, we call them *barriers*. If there are barriers, the routing algorithm has to perform some kind of exploration, i.e. some paths are possibly dead ends so that other paths have to be investigated. This exploration can be done sequentially which is time-consuming or in parallel which increases the traffic (i.e. the total number of messages used).

An example for a sequential strategy is the following *barrier traversal* algorithm: (1.) Follow the straight line connecting source and target node. (2.) If a barrier is in the way, then traverse the barrier, remember all points where the straight line is crossed, and resume step 1 at that crossing point that is nearest to the target. This algorithm needs $\mathcal{O}(h + p)$ steps, where h is the length of the shortest barrier-free path and p the sum of the perimeter lengths of all barriers. This bound holds also for the traffic, because this is a *single-path* strategy. One can show, that this traffic bound is optimal. An example for a *multi-path* strategy is *expanding ring search*, which is nothing more than to start flooding with a restricted search depth and repeat flooding while doubling the search depth until the destination is reached. This strategy is asymptotically time-optimal, but it causes a traffic of $\mathcal{O}(h^2)$, regardless of the presence of faulty nodes.

We observe that both approaches are efficient with respect to *either* time or traffic. The problem is to find an appropriate measure to address *both* the time and the traffic efficiency. We introduce the *combined comparative ratio*, which is motivated by the following two observations: The first observation is that the shortest barrier-free path can be lengthened by adding faulty nodes that cause a detour; this affects the time behavior. We use the so-called *competitive ratio* [4] which compares the performance of the algorithm with the optimal offline solution: The *competitive time ratio* is the ratio of routing time and optimal time. The second observation is that additional barriers increase the exploration costs; this affects the traffic efficiency. Here, by a comparison with the best offline strategy the exploration costs would be disregarded. So we use a *comparative ratio* which compares the traffic caused by the algorithm with the online lower bound for traffic of $\mathcal{O}(h + p)$. Finally, the *combined comparative ratio* is the maximum of time and traffic ratio. These ratios are defined in Section 3.

Under this measure the disadvantages of the two strategies described above, namely that time is optimized at the expense of the traffic or vice versa, can be expressed: Both strategies have a linear combined comparative ratio. In this paper we present an algorithm that has a sub-linear combined comparative ratio. The algorithm is described and analyzed in Section 4.

1.1 Related Work

The routing problem in computer networks, even if restricted to two-dimensional meshes, can be investigated under various aspects, like fault-tolerance, reliability of the message delivery, message size, complexity of a pre-routing stage etc. In this paper we focus on

fault-tolerance, routing as an online problem, competitive analysis, and traffic efficiency with respect to the difficulty of the scenario. These aspects are not only regarded in the field of networking.

A similar model is used by Zakrevski and Karpovski [20]. They also investigate the routing problem for two-dimensional meshes, in which the message are passed in a store-and-forward fashion, and present a routing algorithm that is based on constructing fault-free rectangular clusters in an offline pre-routing stage. Connections between these clusters are stored in a connectivity graph which is used in a routing stage to guide a message to its destination.

Wu [18] presents algorithms for two-dimensional meshes which use only local information and need no pre-routing stage. The underlying model is restricted in that sense that the faulty regions in the network are assumed to be rectangular blocks. In [19] Wu and Jiang present a distributed algorithm that constructs convex polygons from arbitrary fault regions by excluding nodes from the routing process. This is advantageous, if a message consists of several packets, because the routing algorithm needs relatively few virtual channels if the fault regions are convex. The use of virtual channels and also the deadlock-freedom are key aspects for the design of wormhole routing algorithms (see [13] for a survey). We will not deal with these aspects as we consider the store-and-forward model. Faulty mesh networks have been studied in the field of parallel computing, e.g. by Cole et al. [6]. Here, the the ability of a network to tolerate faults and emulate the original network is studied. Emulation schemes are usually based on an embedding of the fault-free network into the faulty network. Measures for the quality of the emulation are load, congestion and dilation of the network, on which the slow down of the emulated computation in the faulty network depends. The general goal is the construction of a routing scheme rather than performing an online path selection.

The problem of finding a target in an unknown environment has been investigated in position-based routing as well as in online robot motion planning.

Position-based routing is a reactive routing used in wireless networks, where the nodes are equipped with a positioning system, such that a message can be forwarded in the direction of the target (see [12] for a survey). Due to the limited range of the radio transceivers, there are local minima and messages have to be routed around void regions (an analog to the fault regions in the mesh network). There are various single-path strategies, e.g. [8, 5, 9]. Position-based strategies have been mainly analyzed in a worst case setting, i.e. the void regions have been constructed such that the connections form a labyrinth. In this case the benefit of a single-path strategy, namely the traffic efficiency compared to flooding, ceases.

The algorithm presented in this paper is a compromise of single-path routing and flooding. By analyzing the time in a competitive manner and by including the perimeters of fault regions we can express performance beyond the worst case point of view. This paper improves the results of the authors presented in [16] where a combined comparative ratio of $\mathcal{O}(h^{1/2})$ has been shown.

In online robot motion planning, the task is to guide a robot from a start point to a target in a scenario with unknown obstacles. This is analogous to the position-based routing problem, except for the possibility to duplicate messages in networks. The motion planning problem for an unknown environment is also known as “online searching”

or “online navigation” (see [2] for a survey). It has been addressed by Lumelsky and Stepanov [11] that the performance of navigation strategies depends on the obstacles in the scenario. The proposed strategies are also suitable for traversing mazes. In such cases, and also in some position-based routing strategies, the well known *right-hand rule* is used: By keeping the right hand always in touch of the wall, one will find the way out of the maze. See [10, 15] for an overview of maze traversal algorithms. Analog to the network model with rectangular fault blocks described in [18], there are robot navigation strategies for obstacles of rectangular or polygonal shape. A competitive analysis of such algorithms is presented by Papadimitriou and Yannakakis [14] and Blum, Raghavan and Schieber [3], where the ratio of the length of the path chosen by the algorithm and the shortest barrier-free path is considered as performance measure. This complies with the most common definition of the competitive ratio. Lumelsky and Stepanov [11] as well as Angluin, Westbrook and Zhu [1] use a modified competitive measure that uses the sum of the perimeters of the obstacles in the scene instead of the optimal path length as a benchmark. In this paper we use the competitive ratio for the length of the routing path — which is, here, regarded as routing time. For the induced traffic, we use a comparative measure that credits the cost of exploring new obstacles (which in fact every online algorithm has to pay) to the algorithm (cf. Section 3).

The problem studied in this paper has a strong relation to online search and navigation problems. However, it is not clear how unbounded parallelism can be modelled for robot navigation problems in a reasonable way — usually navigation strategies are only considered for a constant number of robots. Therefore, we consider a mesh network with faulty parts as underlying model, which enables us to study the impact of parallelism on the time needed for reaching the destination.

2 Barriers, Borders and Traversal

In this paper we consider a two-dimensional mesh network with faulty nodes. The network is defined by a set of nodes $V \subseteq \mathbb{N} \times \mathbb{N}$ and a set of edges $E := \{(v, w) : v, w \in V \wedge |v_x - w_x| + |v_y - w_y| = 1\}$. A node v is identified by its position $(v_x, v_y) \in \mathbb{N} \times \mathbb{N}$ in the mesh. There is no restriction on the size of the network, because we analyze time and traffic with respect to the position of the given start and target node in the network. We will see that the major impact on the efficiency of the routing algorithm is not given by the size of the network.

We assume a synchronized communication: Each message transmission to a neighboring node takes one *time step*. For multi-hop communication we assume the messages to be transported in a store-and-forward fashion. We also assume that the nodes do not fail while a message is transported — otherwise a node could take over a message and then break down. However, there is no global knowledge about faulty nodes. Only adjacent nodes can determine whether a node is faulty.

Important terms and definitions: The network contains *active* (functioning) and *faulty* nodes. Faulty nodes which are orthogonally or diagonally neighboring form a *barrier*. A barrier consists only of faulty nodes and is not connected to or overlapping with other barriers. Active nodes adjacent to faulty nodes are called *border nodes*. All the nodes in the neighborhood (orthogonally or diagonally) of a barrier B form the

perimeter of B . A path around a barrier in (counter-)clockwise order is called a *right-hand (left-hand) traversal path*, if every border node is visited and only nodes in the perimeter of B are used. The *perimeter size* $p(B)$ of a barrier B is the number of directed edges of the traversal path. The *total perimeter size* is $p := \sum_{i \in \mathbb{N}} p(B_i)$.

The perimeter size is the number of steps required to send a message from a border node around the barrier and back to the origin, whereby each border node of the barrier is visited. It reflects the time consumption of finding a detour around the barrier.

3 Competitive and Comparative Ratios

When designing online algorithms one typically asks for the best solution an algorithm can provide online or even offline. The comparison of the performance of an (online) algorithm with the performance of an optimal offline algorithm is called *competitive analysis* (see [4]). In faulty mesh networks the offline algorithm has global knowledge and can deliver the message on the shortest barrier-free path (we denote the length of this path with h). Therefore, both the offline traffic and the offline time bound is h . Comparing the traffic of an online algorithm with this lower bound yields a competitive ratio that is not very informative because it disregards the cost of exploration. In fact every online algorithm produces traffic of $\Omega(h + p)$ in some worst case situation: Consider a scenario where the faulty nodes form long corridors (containing active nodes). The source node sees only the entrances of these corridors. Yet, only one corridor leads to the target; the others are dead ends. Every online routing strategy has to examine the corridors (i.e. explore all the barriers) in the worst case, because an adversary can place the exit at the corridor which is examined as the last. This consideration leads to a lower traffic bound of $\Omega(h + p)$, regardless whether the exploration is done sequentially or in parallel.

The optimal time behavior depends on the type of strategy: A single-path strategy has to traverse the barriers sequentially. This leads to a lower time bound of $\Omega(h + p)$ for all single-path strategies. A multi-path strategy like expanding ring search (repeated flooding with increasing search depth) can do this exploration in parallel. Therefore the trivial lower bound of $\Omega(h)$ can be achieved using such a strategy.

	Time	Traffic
Online lower bound, single-path	$\Omega(h + p)$	$\Omega(h + p)$
Online lower bound, multi-path	$\Omega(h)$	$\Omega(h + p)$
Best offline solution	h	h

For the time analysis, we use the the performance of the best offline algorithm as a benchmark. Regarding time, the best offline performance is of the same order as the online lower bound.

Definition 1. An algorithm A has a competitive ratio of c , if $\forall x \in \mathcal{I} : C_A(x) \leq c \cdot C_{\text{opt}}(x)$, where \mathcal{I} is the set of all instances of the problem, $C_A(x)$ the cost of algorithm A on input x and $C_{\text{opt}}(x)$ the cost of an optimal offline algorithm on the same input.

Definition 2. Let h be the length of the shortest barrier-free path between source and target. A routing algorithm has competitive time ratio $\mathcal{R}_t := T/h$ if the message delivery is performed in T steps.

Regarding traffic, a comparison with the best offline behavior would be unfair, because this bound cannot be reached by any online algorithm. So, we define a *comparative ratio* based on a *class* of instances of the problem, which is a modification of the comparative ratio introduced by Koutsoupias and Papadimitriou [7]: In this paper, we compare the algorithm A w.r.t. the competing algorithm B using the cost of each algorithm in the worst case of a class of instances instead of comparing both algorithms w.r.t. a particular instance that causes the worst case ratio. The reason is the following: On the one hand for every online algorithm A there is a placement of barriers with perimeter size p such that A is forced to pay extra cost $\mathcal{O}(p)$ (cf. the labyrinth scenario described above). On the other hand in the class of online algorithms there is always one algorithm B that uses the path which is the shortest path in this particular scenario. Therefore B has extra knowledge of the scenario.

Definition 3. An algorithm A has a comparative ratio $f(P)$, if

$$\forall p_1 \dots p_n \in P : \max_{x \in \mathcal{I}_P} C_A(x) \leq f(P) \cdot \min_{B \in \mathcal{B}} \max_{x \in \mathcal{I}_P} C_B(x),$$

where \mathcal{I}_P is the set of instances with can be described by the parameter set P , $C_A(x)$ the cost of algorithm A and $C_B(x)$ the cost of an algorithm B from the class of online algorithms \mathcal{B} .

With this definition we address the difficulty that is caused by a certain class of scenarios that can be described in terms of the two parameters h and p . For any such instance the online traffic bound is $\min_{B \in \mathcal{B}} \max_{x \in \mathcal{I}_{\{h,p\}}} C_B(x) = \Theta(h + p)$. Note, that for any choice of a scenario one can find an optimal offline algorithm: $\max_{x \in \mathcal{I}_{\{h,p\}}} \min_{B \in \mathcal{B}} C_B(x) = h$.

This requires the modification of the comparative ratio in [7] in order to obtain a fair measure. So, we use the online lower bound for traffic to define the *comparative traffic ratio*.

Definition 4. Let h be the length of the shortest barrier-free path between source and target and p the total perimeter size. A routing algorithm has comparative traffic ratio $\mathcal{R}_{Tr} := M/(h + p)$ if the algorithm needs altogether M messages.

Under these ratios we can formalize the intuition telling that flooding as a multi-path strategy performs well in mazes and badly in open space, while some single-path strategy performs well in open space, but bad in mazes. The following table shows the competitive time ratio \mathcal{R}_t and the comparative traffic ratio \mathcal{R}_{Tr} of a single-path and a multi-path strategy. We consider the barrier traversal algorithm described in Section 1 as traffic-optimal single-path strategy, and expanding ring search as time-optimal multi-path strategy.

Multi-path	\mathcal{R}_t	\mathcal{R}_{Tr}	Single-path	\mathcal{R}_t	\mathcal{R}_{Tr}
General	$\frac{\mathcal{O}(h)}{h}$	$\frac{\mathcal{O}(h^2)}{h+p}$	General	$\frac{\mathcal{O}(h+p)}{h}$	$\frac{\mathcal{O}(h+p)}{h+p}$
Open space ($p < h$)	$\mathcal{O}(1)$	$\mathcal{O}(h)$	Open space ($p < h$)	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Maze ($p = h^2$)	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Maze ($p = h^2$)	$\mathcal{O}(h)$	$\mathcal{O}(1)$

This comparison shows that by these basic strategies the time is optimized at the expense of the traffic or vice versa. To address both the time and the traffic efficiency, we define the *combined comparative ratio*:

Definition 5. The combined comparative ratio is the maximum of the competitive time ratio and the comparative traffic ratio: $\mathcal{R}_c := \max\{\mathcal{R}_t, \mathcal{R}_T\}$

For both strategies compared above this ratio is linear, i.e. $\mathcal{R}_c = \mathcal{O}(h)$.

4 The Algorithm

The basic idea of the routing algorithm is to use flooding only if necessary. Therefore we identify regions where flooding is affordable. This is done by a subdivision (partitioning) of a quadratic search area into smaller squares. Messages are sent in parallel to each square using only paths on the borders of the squares. If a message on such a path encounters a barrier, it tries to circumvent the barrier in the interior of a square. However, if too many barrier nodes are detected while traversing the border of the barrier, then the algorithm floods the whole square.

4.1 Incremental BFS

We observe that flooding defines a bread-first search (BFS) tree by the message paths. The BFS tree contains all the active nodes that are reachable from the root node. Obviously, both the size and the depth of the BFS tree are bounded by the number of nodes in the network. We use an incremental BFS which works in $\log d(T)$ iterations, where $d(T)$ is the depth of the BFS tree T : In the i -th iteration BFS is started with a search depth restricted to 2^i .

In every iteration i we observe *open paths* and *closed paths* in the tree. Open paths are paths with length larger than 2^i where the exploration is stopped after 2^i steps because of the search depth restriction. Closed paths end in a leaf of the tree. In the next iteration (assumed that the target is not found in this iteration) we do not need to investigate nodes on closed paths again. Thus, we have to remember and revisit only open paths for continuing the search. The following lemma shows that the steps needed for this BFS strategy are linear in the size of the tree.

Lemma 1. Given a tree T , let T_i denote a sub-graph of T which contains all paths from the root to a leaf with length greater than 2^i , which are pruned at depth 2^{i-1} . Then for all trees T with depth $d(T)$ it holds $\sum_{i=0}^{\log d(T)} s(T_i) \leq 3s(T)$, where $s(T)$ denotes the size of the tree, i.e. the number of nodes.

A proof is given in [17].

4.2 The Online Frame Multicast Problem

In the following we define a message multicast problem for a quadratic mesh network. The solution of this problem will lead to the solution of the routing problem.

Definition 6. The frame of a $g \times g$ mesh is the set of framing nodes $F = \{v \in V_{g \times g} : v_x \in \{1, g\} \vee v_y \in \{1, g\}\}$. For a $g \times g$ mesh and a set of entry nodes $s_1, \dots, s_k \in F$, the frame multicast problem is to multicast a message (starting from the entry nodes) to all nodes u on the frame which are reachable, i.e. $u \in F \wedge \exists s_i : \text{dist}(s_i, u) \leq g^2$ where $\text{dist}(s_i, u)$ denotes the length of the shortest barrier-free path from s_i to u .

Definition 7. Given a $g \times g$ mesh with entry nodes s_1, \dots, s_k which are triggered at certain time points t_1, \dots, t_k . A routing scheme is called c -time-competitive if for each active node u on the frame of the mesh a message is delivered to u in at most $\min_{i \in [k]} \{t_i + c \cdot \text{dist}(s_i, u)\}$ steps where $\text{dist}(s_i, u)$ denotes the length of the shortest barrier-free path from s_i to u .

It is obvious that a simple flooding algorithm is 1-time-competitive. The main disadvantage of flooding is the large number of messages, namely $\mathcal{O}(g^2)$, regardless whether few or many barrier nodes are involved. In the following we describe an algorithm that uses a simple technique to reduce the traffic with only constant factor slow down.

The Traverse and Search Algorithm The basic idea of the Traverse and Search algorithm for the online frame multicast problem is to traverse the frame and start a breadth-first search if a barrier obstructs the traversal. The algorithm works as follows:

The message delivery is started at each of the entry nodes as soon as they are triggered. An entry node s sends two messages in both directions along the frame, i.e. a message is forwarded to both neighboring frame nodes (if present) that is in turn forwarded to other frame nodes (frame traversal). If the message is stopped because of a barrier (i.e. a frame node is faulty) then a flooding process is started in order to circumvent the barrier and return to the frame. We call the nodes that coordinate the flooding process *exploration nodes* (cf. Figure 2).

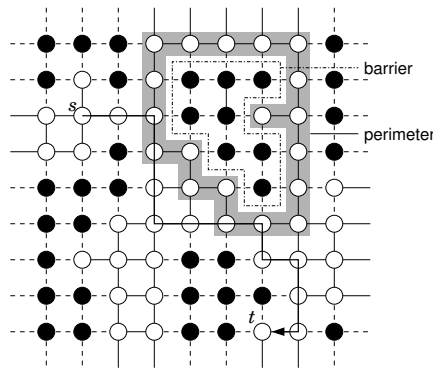


Fig. 1. Optimal routing path from s to t in a mesh network with faulty nodes (black)

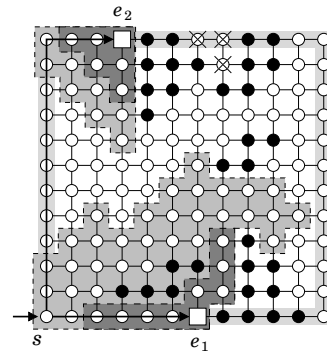


Fig. 2. Execution of the Traverse and Search Algorithm. Entry node s , exploration nodes e_1, e_2 , explored nodes: light background, occupied nodes: dark background.

For the flooding process we use the idea of the incremental BFS described in Section 4.1, i.e. flooding is started in consecutive rounds until the newly found barriers are small or until the flooded area cannot be extended anymore. In the i -th round the search depth is restricted to 2^{i+1} . This is sufficient to circumvent a single faulty frame node in the first round ($i = 1$). If in round i the number of border nodes b_i is greater than a constant fraction of the search depth $\alpha 2^i$ with $0 < \alpha < 1$ then flooding is re-started in the

next round with doubled search depth. Therefore it is necessary to report the number of flooded border nodes (i.e. active nodes adjacent to barrier nodes) to the entry node. These reply messages are sent to the entry node using the reverse of the flooding paths, which are defined by the BFS tree. At the branching points of the BFS tree the replies from all branches are collected and merged. This collection process behaves like the BFS in reverse time order. So, the traffic for collecting information corresponds to the forward search of BFS and is increased only by a factor of 2.

The open paths (see Section 4.1) in the BFS tree are used for flooding a greater area in the next round. If only small barriers are discovered then the flooding process is aborted. Then the flooded frame nodes at which the search depth is reached continue to forward the message on the frame. Due to the location of the barriers or because of the presence of more than one entry node the flooding process is possibly started concurrently from different locations. So we have to ensure that the same regions are not flooded too often by different entry nodes. Otherwise the traffic would disproportionately increase. But if we refrain from flooding a region twice then the flooded regions of one entry node can block the flooding process of another entry node, which would affect the time behavior. Our solution to this problem is as follows: When a node is flooded for the first time, it is marked as *explored*. When it is flooded for the second time (because it is part of an open path in the BFS tree) it is marked as *occupied*. Explored nodes may be explored again by other exploration nodes, but it is forbidden to flood any occupied node again. These two marks are only valid for a specific round of the BFS, that corresponds to a specific search depth. This way we only forbid to flood a region twice with the same search depth.

Time and Traffic Analysis In the following we show that the Traverse and Search algorithm described above has a constant competitive time behavior and requires traffic $\mathcal{O}(g + \min\{g^2 \log g, p^2 \log g\})$, where p is the total perimeter size in the $g \times g$ mesh. This is based on the following ideas:

If we cannot approximate the shortest path by simply traversing the frame then we use the incremental BFS which has a linear asymptotic behavior. Thus, we achieve a constant competitive time ratio. The bound for the number of messages is derived as follows: We allow to flood an area that is quadratic in the number of discovered border nodes. But this area is also bounded by the size of the $g \times g$ mesh. The search from concurrent entry points costs a logarithmic factor. The traversal of the frame costs additional g messages.

Lemma 2. *The Traverse and Search algorithm is $\mathcal{O}(1)$ -time-competitive.*

Lemma 3. *Given a $g \times g$ mesh with total perimeter size p . The Traverse and Search algorithm produces traffic $\mathcal{O}(g + \min\{g^2 \log g, p^2 \log g\})$.*

Proofs for Lemma 2 and Lemma 3 are given in [17].

4.3 Grid subdivision

We will now discuss a technique to apply the Traverse and Search algorithm within a grid subdivision to reduce traffic asymptotically while keeping a constant competitive time behavior.

Lemma 4. *Given a $g_1 \times g_1$ mesh with total perimeter size p . For all g_0 with $1 \leq g_0 \leq g_1$ there is a $\mathcal{O}(1)$ -time-competitive algorithm for the online frame multicast problem with traffic $\mathcal{O}(\frac{g_1^2}{g_0} + p \cdot g_0 \log g_0)$.*

A proof is given in [17].

The technique to subdivide (partition) an area into smaller squares in which the Traverse and Search algorithm is started can be used instead of flooding the area. Thus, it can replace flooding which is used by the incremental BFS. This leads to a modified Traverse and Search algorithm, which is similar to the algorithm described in Section 4.2 except for the BFS:

The multicast process is started with the level-1 square and begins with the frame traversal. If a barrier prevents a frame node from proceeding with the frame traversal, the frame node becomes exploration node and starts the modified BFS, which we call level-1 BFS. The flooding process now uses only the nodes on the grid, i.e. only the frame nodes of level-0 squares are “flooded”, whereas the interior of the frames is controlled by the Traverse and Search algorithm on level 0.

The level-1 BFS tree which is defined by the flooding process consists only of level-0 frame nodes. Especially the leaves of the tree are frame nodes, namely the entry nodes of level-0 squares, because at such points the control is given to the Traverse and Search algorithm on level 0. The BFS-tree will be used for gathering information for the coordinating level-1 entry nodes. This information is the number b_i of newly found border cells in round i , and the paths which should be explored in the next round.

4.4 Recursive subdivision

In the last section we have seen that we can modify the Traverse and Search algorithm such that the flooding phase is replaced by a strategy that subdivides the area that is to be flooded and uses the original Traverse and Search algorithm for the sub-frames. Now, we will see that we can achieve a further reduction of the traffic by applying this subdivision technique recursively.

Let ℓ be the number of recursive subdivisions, g_ℓ the edge length of a top level square. Beginning with the top-level squares, each $g_i \times g_i$ square is subdivided into squares with edge length g_{i-1} ($i = 1, \dots, \ell$). On the lowest level the Traverse and Search algorithm is applied, which produces a traffic of $g_0 + \min\{g_0^2 \log g_0, p^2 \log g_0\}$, where p is the perimeter size in the $g_0 \times g_0$ square. Traffic in the $g_i \times g_i$ square is caused by the trials of surrounding the square and — if the number of thereby observed border nodes is too large — by covering the corresponding area with $g_{i-1} \times g_{i-1}$ sub-squares.

We use the following notation: tr_ℓ denotes the traffic induced by a single $g_\ell \times g_\ell$ square; $\text{Tr}_\ell(a)$ denotes the traffic of an area a which is dissected by level- ℓ squares. Define $G(\ell) := \sum_{i=1}^{\ell} \frac{g_i}{g_{i-1}} \prod_{j=i}^{\ell} \log g_j + g_0 \prod_{i=0}^{\ell} \log g_i$

Theorem 1. *Given a $g \times g$ mesh with total perimeter size p . For all g_i with $1 \leq g_0 \leq \dots \leq g_\ell$ there is a $\mathcal{O}(c^\ell)$ -time-competitive algorithm for the online frame multicast problem with traffic $\mathcal{O}(\frac{g^2}{g_\ell} + p \cdot G(\ell))$.*

A proof is given in [17]. Note, that the recursive subdivision of an area a with ℓ levels produces traffic $\text{Tr}_\ell(a) = \mathcal{O}(\frac{a}{g_\ell} + p \cdot G(\ell))$.

4.5 Expanding Grid

In the previous sections we have discussed the frame multicast problem and developed a basic solution which was refined by the grid subdivision and the recursive subdivision technique, respectively. All these variants solve the original frame multicast problem. Now, we will see how these algorithms can be used for solving the routing problem. Recall that the task is to route a message from a source node s to a target node t and while optimizing the competitive time ratio and the comparative traffic ratio.

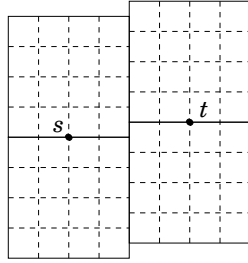


Fig. 3. Search area with grid subdivision

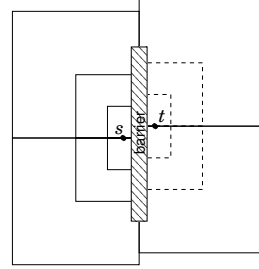


Fig. 4. Enlargement of the search area

For routing a message from s to t , we first define four connected quadratic subnetworks in the environment of s and t which are connected (see Figure 3). Then we only have to apply the frame multicast algorithm to the quadratic subnetworks to route a message. Again we use the idea of expanding ring search: We begin with a small, restricted region which is enlarged if the target cannot be reached because of a barrier (see Figure 4).

Theorem 2. *Given a mesh network of arbitrary size, a source node s and a target node t . There is a $\mathcal{O}(c^\ell)$ -time-competitive routing algorithm with traffic $\mathcal{O}(h^2/g_\ell + p \cdot G(\ell))$, where h is the shortest possible hop distance between s and t , and p the total perimeter size; g_ℓ and $G(\ell)$ are defined as in Section 4.4.*

A proof is given in [17].

Theorem 3. *For $g_\ell = \Theta(h)$ and $g_i = h^{\frac{i+1}{\ell+1}}$, $i = 0, \dots, \ell - 1$ the routing algorithm of Theorem 2 needs time $h \cdot c \sqrt{\frac{\log h}{\log \log h}}$ and causes traffic $\mathcal{O}\left(h + p \sqrt{\frac{\log h}{\log \log h}} h \sqrt{\frac{4 \log \log h}{\log h}}\right)$.*

Proof. We set $g_i := h^{(i+1)/(\ell+1)}$ so that $g_i/g_{i-1} = h^{1/(\ell+1)}$:

$$G(\ell) = \sum_{i=1}^{\ell} h^{\frac{1}{\ell+1}} \prod_{j=i}^{\ell} \log h^{\frac{j+1}{\ell+1}} + h^{\frac{1}{\ell+1}} \prod_{i=0}^{\ell} \log h^{\frac{i+1}{\ell+1}} \leq (\ell + 1) h^{\frac{1}{\ell+1}} \log^{\ell+1} h.$$

In order to minimize this term we set $h^{\frac{1}{\ell+1}} = \log^{\ell+1} h$. By using $\sqrt{\frac{\log h}{\log \log h}} = \ell + 1$ we obtain the traffic of the routing algorithm. \square

Corollary 1. *The routing algorithm of Theorem 3 has competitive time ratio of $c \sqrt{\frac{\log h}{\log \log h}}$ and comparative traffic ratio of $\mathcal{O}\left(\sqrt{\frac{\log h}{\log \log h}} h \sqrt{\frac{4 \log \log h}{\log h}}\right)$, yielding a sub-linear combined comparative ratio of $h^{\mathcal{O}\left(\sqrt{\frac{\log \log h}{\log h}}\right)}$, which is in $o(h^\epsilon)$ for all $\epsilon > 0$.*

References

1. Dana Angluin, Jeffery Westbrook, and Wenhong Zhu. Robot navigation with range queries. In *Proc. of the 28th Annual ACM Symposium on Theory of Computing*, pages 469–478, 1996.
2. Piotr Berman. On-line searching and navigation. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*, pages 232–241. Springer, 1998.
3. Avrim Blum, Prabhakar Raghavan, and Baruch Schieber. Navigating in unfamiliar geometric terrain. *SIAM Journal on Computing*, 26:110–137, 1997.
4. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
5. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.
6. R. J. Cole, B. M. Maggs, and R. K. Sitaraman. Reconfiguring Arrays with Faults Part I: Worst-case Faults. *SIAM Journal on Computing*, 26(16):1581–1611, 1997.
7. Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. *SIAM Journal on Computing*, 30(1):300–317, 2000.
8. E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. 11th Canadian Conference on Computational Geometry*, pages 51–54, 1999.
9. F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proc. of the 6th int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 24–33, 2002.
10. V. J. Lumelsky. Algorithmic and complexity issues of robot motion in an uncertain environment. *Journal of Complexity*, 3(2):146–182, 1987.
11. V. J. Lumelsky and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.
12. M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network Magazine*, 15(6):30–39, November 2001.
13. Prasant Mohapatra. Wormhole routing techniques for directly connected multicomputer systems. *ACM Computing Surveys*, 30(3):374–410, 1998.
14. Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. In *Proc. of the 16th Int. Colloq. on Automata, Languages, and Programming (ICALP'89)*, pages 610–620. Elsevier Science Publishers Ltd., 1989.
15. N. Rao, S. Karetí, W. Shi, and S. Iyenagar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical report, Oak Ridge National Laboratory, 1993. ORNL/TM-12410.
16. Stefan Rührup and Christian Schindelhauer. Competitive time and traffic analysis of position-based routing using a cell structure. In *Proc. of the 5th IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (IPDPS/WMAN'05)*, page 248, 2005.
17. Stefan Rührup and Christian Schindelhauer. Online routing in faulty meshes with sub-linear comparative time and traffic ratio. Technical report, University of Paderborn, 2005. tr-rsfb-05-076.
18. Jie Wu. Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels. *IEEE Transactions on Parallel and Distributed Systems*, 11:149–159, February 2000.
19. Jie Wu and Zhen Jiang. Extended minimal routing in 2-d meshes with faulty blocks. In *Proc. of the 1st Intl. Workshop on Assurance in Distributed Systems and Applications (in conjunction with ICDCS 2002)*, pages 49–55, 2002.
20. Lev Zakrevski and Mark Karpovsky. Fault-tolerant message routing for multiprocessors. In *Parallel and Distributed Processing*, pages 714–730. Springer, 1998.