FAULT TOLERANT TOPOLOGY CONTROL WITH UNRELIABLE FAILURE DETECTORS

Hannes Stratil*
Institute of Computer Engineering
Vienna University of Technology
email: hannes@ecs.tuwien.ac.at

ABSTRACT
Wireless ad-hoc networking has emerged as its own discipline over the past decade and topology control is an important problem in such networks. The aim of topology control is to construct an appropriate topology to improve the efficiency and performance of the overlying algorithms (e.g., routing). Fault tolerance is a necessary feature of a topology to be applicable in realistic scenarios. This paper presents an analysis of the requirements to tolerate crash failures in the topology with the help of Failure detectors. We show that the class of perfect Failure detectors is strong enough to form a reliable underlay for routing and to avoid message loops.

KEY WORDS
Failure detector, Ad-hoc Network, Topology Control.

1 Introduction

A *wireless ad-hoc network* consists of a collection of communication nodes. The nodes are randomly dispersed over some area of interest and communicate with each other over a wireless medium in the absence of a fixed infrastructure and any centralized control. Direct communication between two nodes is only guaranteed if the distance between them is less than the *communication range*. For the communication between non-neighboring nodes a wireless ad-hoc network needs a *multi-hop routing* protocol. A well known example of a reliable routing algorithm is the *greedy/perimeter-routing* approach [1, 2]. Perimeter routing requires a planar[1] underlying *topology* to guarantee the delivery of messages. Therefore a distributed *position-based topology control* algorithm is needed to compute an efficient, planar topology. This algorithm uses the construction rule of a proximity graph, e.g., *Relative neighborhood graph* [3], *Gabriel graph* [4], *Delaunay triangulation* [5], for the computation.

Fault tolerance is, in addition to planarity, another important feature of topologies. The computation of a topology must be done localized. Each node computes locally its contribution to the overall topology using only information about its single hop neighbor nodes. Therefore it is essential for the correctness of the topology that the status (correct or faulty) of a specific node is agreed upon all of its neighbor nodes.

Asynchronous distributed systems, like wireless ad-hoc networks, are characterized by the fact that there is no bound on the time it takes for a process to execute a computation step, or for a message to be transmitted from its sender to its receiver. Because of this “time freedom” is it impossible for neighbor nodes to determine whether a node has actually crashed or is only very slow. Chandra and Toueg have introduced the notion of *unreliable Failure detectors* to solve this dilemma in the context of the consensus problem [6].

The aim of this work is to analyze the requirements that are necessary for a topology control algorithm to tolerate *crash failures*. A crashed node results in many cases in an inconsistency between the local contributions of the overall topology. This inconsistency can yield to a loss of planarity in the topology which can have disastrous implications on routing. A message can get into a loop or can get lost. However, a planar topology is not absolutely necessary for the successful operation of perimeter routing: failure detectors can be used to tolerate some of the inconsistencies. We show that the class of perfect Failure detectors is strong enough to compute a “sufficiently accurate” topology (*safety property*). The safety property guarantees that disastrous implications of forwarding decisions (e.g., routing loops) cannot occur. If the network is stable over a long enough period of time, the topology becomes planar and the chosen routing path is the most efficient path – according to the appropriate proximity graph and the used routing algorithm (*liveness property*). This work is, to our knowledge, the first attempt to formally define and prove the requirements for a fault tolerant topology control algorithm with Failure detectors.

Organization of the paper. The description of topology control, proximity graphs, Failure detectors, crash failure model and greedy/perimeter routing is given in Section 2. The requirements for fault tolerant topology control and the correctness proof are presented in Section 3. Finally, Section 4 concludes the paper.

*This research is supported by the OEAW — Austrian Academy of Sciences — through a DOC scholarship and partially supported by the Austrian Science Fund (FWF) under the grant P18264-N04 (SPAWN-Project).

[1]A graph is called planar if its edges only intersect at their common vertices.

2 Preliminaries

2.1 Topology Control

The technique to compute an appropriate topology is called topology control. Topology control approaches can be divided into two major paradigms. In the first one, each node modulates its transmission power to achieve a sufficient connectivity and to minimize the power consumption [7]. This problem is very challenging because the two goals are self-contradictory and it is sometimes impossible to find an adequate trade-off. For the second approach we assume that each node can communicate with the nodes within a certain neighborhood and the aim of topology control is a reasonable restriction of the available communication links to a small number of beneficial links. The restriction to a small number of beneficial links reduces the interference and therefore improves the efficiency of the wireless ad-hoc network. A subset of the second paradigm is the class of *position-based topology control algorithms*. This class of topology control algorithms uses the positions of the nodes in the wireless ad-hoc network to compute an appropriate topology according to a proximity graph (Subsection 2.4). The node positions are an indispensable requirement for the computation of a planar topology and hence for the usage of perimeter routing. Proximity graphs are not directly applicable to wireless ad-hoc networks. We present in the following subsections the assumptions, the requirements and the methods to make proximity graphs applicable for a distributed topology control algorithm.

2.2 Assumptions

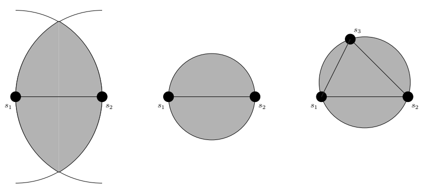
We assume that all nodes in the network have negligible difference in altitude, so they can be considered roughly in a plane. We stipulate the existence of a position service that provides all network participants with their location (e.g., by using the Global Position Service (GPS)) and we assume that the position information are consistent on all nodes in the network. We assume the existence of a Medium Access Control (MAC) protocol (e.g., CSMA/CA) which creates reliable point-to-point connections between the nodes in the wireless network. This forms the basic infrastructure needed for wireless hop-by-hop communication. We further assume a common communication range, an obstacle free environment and symmetric communication links (i.e., the sender and the receiver should observe the same channel properties such as interference, path loss, and fading). Hence, all nodes can communicate with all nodes within the communication range and all communication channels are bidirectional.

2.3 Network Model

The above specified network can be modeled as an undirected communication graph $CG(S, E)$ in the plane, with a set of sites S and a set of edges E. Each site s_i of the set $S := \{s_0, \ldots, s_{N-1}\}$ represents a node of our wireless network. The total number of sites is $N = |S|$. An edge $(s_i, s_j) \in E$, $s_i, s_j \in S$, represents a wireless link of the network in $CG(S, E)$. An edge (s_i, s_j) is present in $CG(S, E)$ if and only if $\|s_i, s_j\|$ is less than or equals the communication range, where $\|s_i, s_j\|$ denotes the Euclidean distance between s_i and s_j. The neighborhood of a node $s \in S$, denoted by $\mathcal{N}(s)$, is the set of nodes within the communication range of node s. The topology returned by the topology control algorithm is denoted by $T(S)$. $T(S)$ must be a subgraph of $CG(S, E)$.

2.4 Proximity Graphs

Proximity graphs are well known in computational geometry. They represent neighbor relationships between geometric points in the Euclidean plane. Three famous examples of proximity graphs are the *Relative neighborhood graph* [3], the *Gabriel graph* [4], and the *Delaunay triangulation* [5] (see Figure 1).



(a) *Relative neighborhood graph* (b) *Gabriel graph* (c) *Delaunay triangulation*

Figure 1. *Proximity Graphs*

Wireless ad-hoc networks use proximity graphs as a basic "construction plan" for the computation of the underlying topology. Different computation algorithms and the features of these graphs are well studied. The three presented proximity graphs, Relative neighborhood graph, Gabriel graph and Delaunay triangulation, are planar. A formal definition of these proximity graphs is given by:

Definition 1 (Relative neighborhood graph). *An example of a Relative neighborhood graph [3] $RNG(S)$ on set S is shown in Figure 1(a). An edge (s_i, s_j) is in $RNG(S)$ if and only if the lune*[2] *of s_i and s_j does not contain any site of S.*

Definition 2 (Gabriel graph). *The Gabriel graph [4] $GG(S)$ of a set S shown in Figure 1(b) consists of all edges (s_i, s_j) with the property that no other site of S is inside or*

[2]The lune of a pair of sites $s_i, s_j \in S$ is the intersection of two open discs of radius $\|s_i, s_j\|$, one centered at s_i and the other centered at s_j.

on the circle through s_i and s_j which has $\|s_i, s_j\|$ as a diameter.

Definition 3 (Delaunay triangulation). *A triangulation of a set S of sites in the plane is called a Delaunay triangulation $DT(S)$ of S shown in Figure 1(c), if the circumcise of each of its triangles does not contain any site of S in its interior*[3].

2.5 Localized computed graphs

The concepts of the Relative neighborhood graph, the Gabriel graph and the Delaunay triangulation are originated in Computational Geometry — one ”unit” computes the proximity graph using knowledge about all node positions in the network. However, algorithms for wireless networks must be localized. It is therefore impossible to compute anything in a centralized manner. Moreover, the edges in our network model must be shorter than the communication range, whereas the length of edges in the Relative neighborhood graph, as well as in the Gabriel graph and in the Delaunay triangulation, are only depending on the geographic positions and are hence not bounded by any communication range. Various algorithms are proposed in literature for *localized* computed versions of the presented proximity graphs [1, 8–10].

Each node $s_i \in S$ computes a *local proximity graph* only with the nodes of its neighborhood $\mathcal{N}(s_i)$. Each node s_i computes further a subgraph of its local proximity graphs containing only edges originating at s_i. For topologies based on Relative neighborhood graph or Gabriel graph, the union of these subgraphs is the localized computed graph of the desired proximity graph [1]. The localized computed graphs contain all edges of the centralized computed graphs that are shorter than the communication range. The distributed computation of a topology based on the Delaunay triangulation is a little bit more complicated. The union of the local subgraphs is not necessarily planar. Some additional computation is required to attain planarity. The localized computed graph is not a subgraph of the centralized computed graph. It contains all edges of the centralized computed graphs that are shorter than the communication range and some other edges. For detailed information about the localized computed Delaunay triangulation we refer to [8–10].

2.6 Crash Failure Model

Fault tolerance is an important issue in the context of wireless ad-hoc networks. Particularly for topology control algorithms it is important to tolerate node failures. We concentrate in our analysis on crash failures. Such crashes can lead to undesired states where some neighbors suppose node s is still alive while others have already detected the crash of s.

[3]We consider that the interior of a circle is an open disc, i.e., the boundary is excluded.

The communication network is assumed to be reliable, i.e., it does not lose or generate messages. A node in the network can only fail by permanently crashing. Its state is correct until it crashes. A node that does not crash during the execution is said to be correct; otherwise it is faulty.

2.7 Failure detector

A wireless ad-hoc network is a distributed asynchronous system. A system is asynchronous, if there are no bounds on message delay, clock drift, or the time necessary to execute a step. Thus, to say that a system is asynchronous is to make no timing assumptions whatsoever. This model has several advantages, e.g., simplicity, portability. Unfortunately, there are also drawbacks in asynchronous systems as soon as there are failures. It is impossible to determine in a fully asynchronous system whether a node has actually crashed or is only “very slow”. To circumvent this problem without restricting the asynchronous model more than necessary, Chandra and Toueg [6] introduced the concept of unreliable Failure detectors.

Failure detector modules monitor the system and inform the algorithm about nodes they suspect to have failed. A Failure detector can be seen as a distributed oracle related to the detection of failures. Their essential characteristic is related to the guess they provide about failures. As defined by Chandra and Toueg, a Failure detector is basically defined by two properties, namely, a *completeness property* and an *accuracy property*. Completeness is on actual detection of failures, while accuracy restricts the mistakes a Failure detector can make.

Chandra and Toueg identified multiple classes of Failure detectors, distinguished by their accuracy property:

- *Perfect ($\mathcal{P}$)*: No process is suspected before it crashes.
- *Strong ($\mathcal{S}$)*: Some correct process is never suspected.
- *Eventually Perfect ($\diamond\mathcal{P}$)*: There is a time after which correct processes are not suspected by any correct process.
- *Eventually Strong ($\diamond\mathcal{S}$)*: There is a time after which some correct process is never suspected by any other correct process.

All these properties have in common that eventually every process that crashes is permanently suspected by every correct process (strong completeness property). Chandra and Toueg also present classes of Failure detectors satisfying only a weaker variant of the completeness property (“Eventually every process that crashes is suspected by some correct process.”), but as those Failure detectors can be transformed into Failure detectors satisfying the stronger property, we ignore these variant for our analysis.

Failure detectors are normally defined for fully connected networks. In sparse networks, like the network in our approach (see Section 2.3), we either simulate a fully connected network or we use *local* Failure detectors. Local

Failure detectors, as defined in the work of Hutle and Widder [11], satisfy the same properties as traditional Failure detectors, but just for neighbors. The usage of local Failure detectors is especially adequate in our approach, because the algorithms for the computation of the local proximity graphs have only to communicate with the single hop neighbor nodes.

2.8 Routing

The aim of this paper is the analysis of Failure detector requirements to tolerate topology errors. Topology errors alone are not dangerous, but they can have disastrous implications if a message – forwarded by the routing algorithm – crosses a topology error. We explain in this subsection the assumed reliable position-based routing algorithm – the greedy/perimeter routing approach.

Greedy forwarding is a localized approach: The routing decision at a node is only based on its own position, the position of its single hop neighbor nodes and the position of the destination node. Greedy routing does not require the establishment or maintenance of routes: The nodes neither have to store routing tables nor do they transmit messages to keep the routing tables up-to-date, and no global information about the topology of the network is needed. When an intermediate node receives a message for a specific destination node, it forwards the message to the neighbor node which is closest to the destination node among all its neighbors. Greedy routing requires neither flooding nor the distribution of status information to nodes further than the single hop neighbor nodes. It is used until the message reaches the destination or a node where no neighbor is closer to the destination than the node itself. At such a local minimum, greedy routing is no longer possible and the message is forwarded along the perimeter of the face that is crossed by the (imaginary) straight line from the local minimum node to the destination node. If the edge to the next hop intersects the imaginary line from the local minimum node to the destination node, the message is forwarded along the perimeter of the next face bordering this edge. If a node is reached, whose position is closer to the destination node than the node where the greedy strategy previously failed, the greedy routing algorithm take over control again. The planarity of the topology is an indispensable property to guarantee the reliability of the routing algorithm. The idea of combining greedy and perimeter routing on planar graphs is independently investigated by Karp and Kung [1] and Bose et al. [2].

We assume a connected network for our analysis. Is the network partition-free, the proposed routing algorithm reliably delivers the message to the destination. A network partition has occurred if a perimeter message never reaches a node that is closer to the destination than the node injecting the perimeter message.

3 Fault Tolerant Topology Control with Failure detectors

The nodes in a wireless network decide locally whether an edge becomes part of the topology or not. The nodes make the right decision, if the involved nodes have the same view about the nodes in their neighborhood. The computed topology can be wrong if nodes have a differing opinion of a common neighbor node. In such a case, the stipulated properties of the topology (e.g., planarity) cannot be guaranteed. An erroneous topology has various effects on routing, both greedy and perimeter routing.

In a wireless ad-hoc network, a Failure detector can make two different kinds of mistakes. First, as a result of the completeness property, a faulty node can not or not yet be detected by some neighbor nodes. And second, as a result of the accuracy property, a correct node can be falsely suspected by some neighbors. Both mistakes generate inconsistencies between local topologies and can result in a non-planar topology.

Mistakes in the topology can have the following effects:

- **Case 1:** A faulty node is not or not yet suspected by the Failure detector of node s. Hence, the faulty node and – maybe – the edge to the faulty node is part of the local proximity graph of node s. Moreover, the faulty node can block some other edges to become part of the local proximity graph of node s.

- **Case 2:** A correct node is suspected by the Failure detector of a neighbor node s. Thus, node s excludes the suspected node and computes a new, wrong local proximity graph. Moreover, the exclusion of the wrongly suspected node enables the admission of additional edges. The computed overall topology can become non-planar.

Obviously, both fault cases presented above are harmless, if the Failure detectors of all neighbor nodes come to the same, right or wrong, decision.

We analyze in the following the consequences that can occur through the mistakes in greedy and perimeter routing. The aim of a fault-tolerant topology is giving a reliable routing algorithm the capability to detect a message loss, i.e., protecting the network from circulating messages. In some of the following instances a message can get lost because a node s sends a message to a faulty node. A message loss can occur in any network and it is the aim of a reliable routing algorithm to detect a message loss (e.g., by the absence of acknowledgment messages). If a message lost is detected, the Failure detector of node s can mark the non-reacting node as faulty, the node updates its local proximity graph and the message is forwarded once again according to the routing principles.

First, we analyze the consequences of fault-case 1 in greedy routing:

- A forwarded messages to a faulty node gets lost. Such a message loss can be detect by the routing algorithm.

- The faulty node can block other edges from becoming part of the topology. The consequences for a sender node are that a message must take a longer path than necessary. This is unpleasant, but has no dangerous effects in greedy routing. On the other hand, a recipient can receive the greedy message over a communication edge which is not part of the computed topology. This is inefficient and not the intention of a topology, but also not very dangerous in greedy routing. The receiver node forwards the message according to the routing principles.

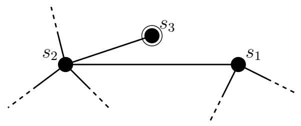


Figure 2. *Faulty Node* s_3

The consequences of fault-case 1 in perimeter routing are the following:

- In perimeter mode, the message is forwarded along the perimeter of the chosen face. If the successor node is faulty, the message gets lost. Again, the routing algorithm can handle this case. The faulty node has no effects, if it is not the subsequent node.

- Figure 2 shows an example where the Failure detector of the receiver node has not suspected a faulty node. Assume node s_1 forwards a perimeter message to s_2, node s_3 is faulty and is not or not yet suspected by node s_2. The edge from s_1 to s_2 can or cannot be part of the local proximity graph of node s_2. According to the perimeter mode, node s_2 forwards the message along the next edge counterclockwise. This edge yields to the faulty node s_3 and the message gets lost. Such message loss can be detected by the routing algorithm.

A wrongly suspected node (fault-case 2) has these consequences in greedy routing:

- The wrongly suspected node is eliminated from the local proximity graph of the sender node, a forwarded message must take a longer path than necessary.

- A recipient can receive a message from the wrongly suspected node! This is a hard challenge for the MAC-protocol of the wireless ad-hoc network.

Finally we show the consequences of fault-case 2 in perimeter routing:

- A wrongly suspected node can have disastrous implications for a message in perimeter mode (see Figure 3). Assume node s_3 is wrongly suspected by node s_1. Hence, s_1 forwards the perimeter message to node s_2 and s_2, according to the routing principles of perimeter routing, onward to node s_3 and node s_3 forwards the message to node s_1. If s_3 is still suspected by s_1, node s_1 receives a message from an unknown node. If node s_1 has retracted the suspicion in the meantime, the message circulates ongoing in the face.

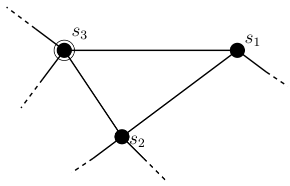


Figure 3. *Wrongly suspected node* s_3

A message loop has disastrous consequences, because it is impossible to detect its occurrence locally and the message circulates perpetually in the network. Hence, we need a Failure detector which prevents the occurrence of such message loops. It is furthermore important to prevent the reception of a message from an “unknown” node, because many modern MAC-Protocols cannot deal with this situation (e.g., TDMA, CDMA)

A message loop, as well as an “unknown” node, can only occur if a correct node is suspected by some neighbor nodes. Therefore, the Failure detectors must avoid wrong suspicions. Only the class of perfect Failure detectors $\mathcal{P}$ fulfills this requirement (see Section 2.7). The other Failure detector classes allow a wrong suspicion in one or another manner; temporarily by the class of $\diamond\mathcal{P}$ or permanently by the classes of $\mathcal{S}$ and $\diamond\mathcal{S}$.

In general, there are two kinds of correctness properties that each algorithm must satisfy: safety and liveness. Intuitively, a safety property specifies that “bad things” do not happen in all executions of a system and a liveness property specifies that “good things” eventually happen in all executions of a system. A topology control algorithm based on a proximity graph requires the following safety and liveness property:

- *Safety:* The topology control algorithm never computes a topology where routing loops can occur.

- *Liveness:* The topology eventually becomes planar and corresponds to the desired proximity graph.

To achieve the desired topology, a *stable period* is required. A stable period is a period of time during which no node crashes or is suspected as faulty and no new nodes are added. If these requirements are fulfilled during a period which lasts long enough, the algorithm will produce the desired topology. If these requirements are not fulfilled, the algorithm does not lose safety but simply does not produce the desired topology. The routing algorithm has always the possibility to forward a message across the network from the source node to the destination node. A message loop cannot occur.

Lemma 1 (Liveness). *The topology satisfies the desired properties after a sufficiently long stable period.*

Proof. The accuracy property of the perfect Failure detector class ensures that no node is suspected before it crashes. Furthermore, by the completeness property, eventually every node that crashes is permanently suspected by every correct node. The neighborhood sets of the nodes are eventually consistent (i.e., if a node n_i is in the neighborhood set of site n_j, then site n_j is in the neighborhood set of n_i). Hence, no conflicts between local proximity graphs can occur. The local proximity graphs become planar, the topology corresponds to the desired proximity graph and the routing algorithm can chose the most efficient path – according to the routing algorithm abilities. □

Lemma 2 (Safety). *The arise of a routing loop is impossible.*

Proof. Let us assume a message circulates along a face f, $f = e_1, e_2, \dots e_k$ $k \geq 3$, of our topology, permanently forwarded by the k nodes of this face. One of these k nodes, say node s_1, was the first one to inject the message into this face; either s_1 received the message in perimeter mode or s_1 changed to perimeter mode, because it was the nearest node towards the message destination. In both cases, the predecessor of node s_1 in face f must be suspected by the Failure detector of s_1. By the accuracy property of a $\mathcal{P}$-class Failure detector, a node will never be suspected before it crashes. □

4 Conclusion

The Failure detector approach was introduced by Chandra and Toueg to solve consensus and similar problems in asynchronous networks. We use this approach for the analysis of the requirements for the computation of an appropriate topology for wireless ad-hoc networks in the context of node crash failures. We show that only the class of perfect Failure detectors is able to guarantee a "sufficiently accurate" topology for routing and prove the correctness of our analysis.

5 Acknowledgments

I would like to thank Martin Biely and Josef Widder for useful discussions and their helpful comments.

References

[1] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the sixth annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, August 2000.

[2] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, November 2001.

[3] Godfried T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[4] K. Ruben Gabriel and Robert R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[5] Boris Nikolajewitsch Delaunay. Neue Darstellung der geometrischen Krystallographie. *Zeitschrift fr Krystallographie*, 84:109–149, 1932.

[6] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[7] Ram Ramanathan and Regina Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. In *Proceedings of the IEEE Conference on Computer Communications INFOCOM*, pages 404–413, March 2000.

[8] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks. In *Proceedings of the second ACM International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc 01)*, pages 45–55, Long Beach, California, October 2001.

[9] Xiang-Yang Li, Gruia Calinescu, Peng-Jun Wan, and Yu Wang. Localized delaunay triangulation with application in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(10):1035–1047, October 2003.

[10] Hannes Stratil. Distributed construction of an underlay in wireless networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks*, pages 176–187, Istanbul, Turkey, January 2005.

[11] Martin Hutle and Josef Widder. On the possibility and the impossibility of message-driven self-stabilizing failure detection. In *Proceedings of the Seventh International Symposium on Self Stabilizing Systems (SSS 2005)*, Barcelona, Spain, October 2005. (to appear).