Self-Healing Wireless Ad Hoc Networks Based On Adaptive Node Mobility

Matthew Smith and Bernd Freisleben

Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Str., D-35032 Marburg, Germany
E-Mail: {matthew,freisleb}@informatik.uni-marburg.de

Abstract

The movement of nodes in a wireless ad hoc network has a significant effect on network integrity. To guarantee application-specific dynamic quality of service demands, a novel approach to node mobility is needed to cope with varying network density without disrupting the primary task of the network nodes. In this paper, a novel self-healing mechanism utilizing node mobility in a non-disruptive manner is introduced. The self-healing mechanism of the network is enabled by cross-layer interaction between the application and network layer. To illustrate our approach, we present an application in which multiple simulated mobile robots clean a minefield, coordinating their effort using a wireless self-healing ad hoc network.

Keywords: wireless ad hoc networks, self-healing, mobility.

1 INTRODUCTION

Mobility is a vital aspect of a wireless ad hoc network, since the movement of nodes has a significant effect on the integrity of the network. Most current mobile ad hoc network research is done with simulators and deals with mobility in a static way, viewing it as something which happens to the network in a pre-defined scenario-based way [1]. This reduces mobility to something the network has to handle as an outside influence. In the case of mobile robots or other autonomous mobile nodes, this is usually not true. Mobile robots, sensors or people with wireless devices have a task or goal to fulfill, so they will not move around randomly while avoiding obstacles but will follow a path which lets them fulfill their task. Furthermore, as the network is used for the task at hand, availability and quality of service of the network can actively influence the movement of the network participants.

In this paper, a novel self-healing mechanism utilizing node mobility to repair a wireless ad hoc network is presented. It allows the primary task of the nodes to be fulfilled while at the same time trying to preserve the network. Since our self-healing mechanism needs to be able to adaptively influence node mobility based on the current network state, the static mobility patterns used in existing network simulators and emulators such as NS2 [2], GloMoSim [3] and MobiEmu [4] are not suitable to investigate its behavior. To allow node mobility to be influenced during simulation time, we extended the MobiEmu [4] network emulator and coupled it with the Stage [5] mobile robot simulator. To illustrate our approach, a scenario in which a number of virtual robots cooperatively explore an unknown environment searching for mines using range limited wireless communication is presented. As the number of nodes is relatively small in comparison to the area to be covered, the self-healing mechanism is utilized to ensure message delivery.

2 SELF-HEALING VIA ADAPTIVE NODE MOBILITY

To allow node mobility to be controlled by the application executing the main task of a node and at the same time allow the network state to influence node mobility, all aspects of a node's physical behavior were separated into simple modules called *motivations* which can then be combined freely to form complex behaviors. Each motivation handles one specific type of situation like avoiding obstacles or heading to a target. The combination of the two allows a node to head towards a target while avoiding obstacles. This is achieved by gathering action *suggestions* from each motivation. A suggestion is a vector of acceptance values for each possible action created by the motivation based on sensor inputs from the node. A behavioral arbitrator then merges all suggestions, paying greater attention to critical motivation suggestions. Based on the merging of all the different acceptance levels, one action as the best compromise is chosen.

Figure 1 shows how three suggestions cast their vote for three different actions. Broken lines represent a negative and unbroken lines a positive acceptance level. Suggestion 1 has a high priority, suggestions 2 and 3 have normal priorities. Since suggestion 1 gives actions 1 and 2 the same acceptance level, action 2 is taken, as the overall vote is swayed by suggestion 2 and 3 strengthening action 2 and suggestion 3 weakening action 1.

The pseudo code fragment below shows how a motivation creates a suggestion to head towards a position. The motivation sets the acceptance levels for all possible actions a node can take. If the destination lies to the left of the current position, the suggestion will set a high acceptance level to go left and a negative value to go right. If the distance is still large, a neutral value is given to straight ahead. If the target is near, a negative value is given, because a turn becomes essential to reaching the destination. The

same is done for the other possibilities.

```
Suggestion makeDriveSug(Position pos)
  Suggestion sug = new Suggestion()
  if(pos is to left)
    sug.setGoLef(max)
    sug.setGoRight(-max)
    if(distance large)
      sug.setGoStraight(none)
    else
      sug.setGoStraight(-mid)
    ...
  else if(pos is straight)
    sug.setGoStraight(max)
    sug.setGoBack(-max)
    ...
  ...
  return sug;
```

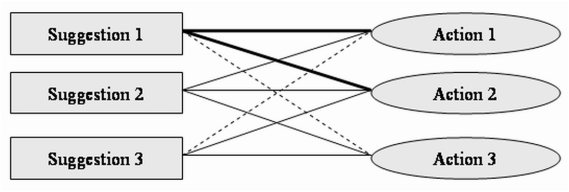

Figure 1. Suggestion-Action Architecture

For the self-healing approach introduced in this paper, it is necessary to base actions on the network state. A cross-layer connection between the application and the network layer is used to gather the information needed. For an overview of cross-layer possibilities in wireless ad hoc networks see [6, 7, 8, 9, 10]. Here, a motivation is added to the arbitration process which also receives inputs from the network layer. Figure 2 shows the motivation/suggestion architecture. Depending on the priority of the cross-layer motivation, its suggestion can override the primary task or merely influence it slightly.

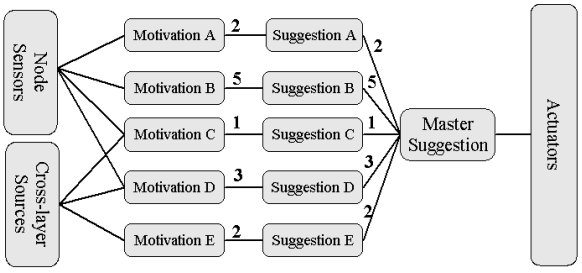

Figure 2. Motivation-Suggestion Architecture

The adaptive self-healing capability is implemented in two separate motivations: The Passive Network Preservation Motivation and the Active Network Preservation Motivation.

The Passive Network Preservation Motivation is a low priority motivation which constantly tries to move the node, so that it stays within communication range of as many other nodes as possible. The information on which nodes are within communication range and the connection strength is gathered from the network layer. Since this motivation has a low priority, it does not interfere with the execution of the main task of the node, because the primary movement is never completely overridden. Through the suggestion-based subsumption architecture, it will however influence the movement of the node, so that within the bounds of the task at hand an optimal position is adopted.

The listing below shows the pseudo code of the passive network preservation motivation. First, the position information of the neighbor nodes is collected from the network layer. Then, a suggestion is created which would drive the node towards each of those nodes. Since the priority of these suggestion is the same, the arbitrator will create the best compromise between all neighbor nodes.

```
Position pos[] = Network.getPos();
for(int i=0;i<pos.length;i++)
  Suggestion sug=makeDriveSug(pos[i],low)
  Arbitrator.addSuggestion(sug)
```

The Active Network Preservation Motivation has a higher priority and is only used if a certain network connection is essential to the node's operation. It also has a higher priority than the normal task of the node and thus can subsume that behavior. The Active Network Preservation notes where its neighbors are headed and shorty before losing connection to them sends a query, asking what their intended destination is. To be able to do this, the routing daemon receives position, direction and speed information from the application layer. This information is then passed to other nodes by the routing daemon. If later those nodes need to be contacted but are not reachable due to network partitioning, an educated guess based on last known position and destination can be made as to where they might be found. The Active Network Preservation Motivation then steers the node in that direction. If possible, other tasks are taken into account, but until network connectivity is reestablished the node will continue searching for the lost connection.

The code fragment below is run continuously as a background task as a pre-requirement for the active network preservation. If there are nodes whose signal strength drops so low that it is probable that the connection will be lost soon, the communication centre requests an expected destination from those nodes. The communication centre is a simulation module responsible for sending and receiving messages within the wireless ad hoc network. It offers convenience methods to request information from other network participants. The `getDestinationFor` method, for instance, contacts the node passed as the parameter and requests its current destination. The position data gathered this way is then stored in a HashMap so it can be retrieved later.

```
if(Network.loosingNeighborNode())
  for(each node)
  ComCenter.getDestinationFor(node);
  store.add(node,destination);
```

If a message needs to be sent to a node which is currently

not reachable, the active network motivation is initialized to contact that node. The code fragment below shows the active network preservation. As long as there is no connection to the target node, the `requestLocationUpdate` will ask the reachable network nodes if a more recent location for the target node is known, otherwise the stored position is used. Based on the current estimated position for the target node, a high priority suggestion is made to drive to that position.

```
loop until exit
  if not(RouteD.canReach(node))
    ComCenter.requestLocationUpdate(node)
    Position pos=store.getDestination(node)
    Suggestion sug = makeDriveSug(pos,high)
  else
    exit
```

3 EXPERIMENTAL RESULTS

To illustrate the mobility based self-healing approach for mobile ad hoc networks introduced in this paper, the following scenario has been implemented. A number of mobile robots are charged to clear an area of mines. All robots can detect mines but only one robot is equipped with a mine clearing device. To complete their task in an efficient manner, the robots must cooperate. The robots are connected via range limited wireless communication, thus forming a mobile wireless ad hoc network.

The wireless network used by the robots is emulated by our extended MobiEmu system. The robots interact and move within the virtual world provided by the Stage simulator. Based on the position within the virtual world, MobiEmu emulates a wireless network spanning the mobile robots. Thus, robots which are too far apart can not communicate with each other. This novel combination of a virtual world simulator and a network emulator enables the robots to change course to influence the network state dynamically. Figure 3 shows the information flow between the network emulator and virtual world simulator.

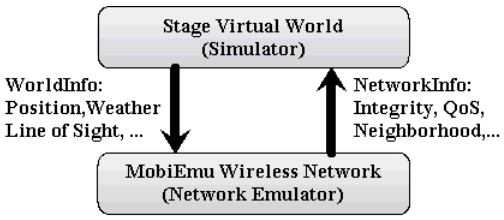

Figure 3. Coupled Simulation/Emulation

The experimental setup is as follows: Five robots are located at the bottom of the Stage playing field littered with obstacles. In the top left hand and top right hand corner, there are two separate minefields. The mines are represented by small black dots. Each robot is equipped with a laser range scanner, a position device and a blob detecting camera. The mine clearing robot D is additionally equipped with a gripper device for defusing mines. The task of robots A through C and robot E is to locate the mines and report the position of them to robot D. The mine clearing robot then must drive to the reported minefield and clear it.

A multi-hop routing daemon routes messages through intermediate robots wherever possible. Due to the fact that the robots have a task to fulfill, the network will become partitioned during the course of the experiment. To ensure that nonetheless communication is possible, the active and passive network preservation motivations introduced in section 2 are added to the robots' motivations.

The following screenshots were taken from the Stage simulator and annotated to illustrate the self-healing mechanism.

Initially, all the robots are set up at the bottom of the playing field. The four mine searching robots A,B,C and E head out across the field in search of mines. The mine clearing robot D does not yet have any task to fulfil so it stays at its starting position.

In Figure 4, the mine clearing robot D is informed by the network layer that it is losing the connection to the network. The large circles around robot A and D show the communication range of the wireless communication device carried by each robot. The passive network preservation motivation kicks in; since the mine clearing robot does not currently have any task, it will ensure that it stays within communication range of the bulk of the network based on information gathered from the network layer. Robot A is also aware that it has lost its connection to the network but since it has already detected the blobs from minefield A, it continues on its way. The passive network preservation motivation does not override the robots primary task. Before A left the network, the active network preservation code queried the network layer in which direction the other nodes are headed and sent messages to the other nodes asking for their destinations, so it can attempt to rejoin the network later.

In Figure 5, robot E has found minefield B. The message is transmitted to the mine clearing robot D by multi-hop routing via robot C. Robot D will now head to the minefield and clear it.

Later, mine searching robot A finds minefield A. Since it is not connected to the network, it can not inform robot D. To regain entry, the active network preservation calls up the information gathered before it left the network and then tries to rejoin the network by moving towards where the other nodes were headed. During this time, robot D is busy clearing minefield B.

In Figure 6, robot A has rejoined the network and can relay its message to the mine clearing robot D via robot C. The mine clearing robot will finish clearing field B and then head to minefield A.

4 CONCLUSIONS

A novel self-healing mechanism based on adaptive mobility control was presented. The self-healing ability is an emergent behavior which is created out of the dynamic merging of the primary task of the network and behavior based on the current network status. The passive network preservation protects network integrity while the primary task is executing, and the active network preservation repairs the network if it has become partitioned. The splitting of the overall self-healing mechanism allows

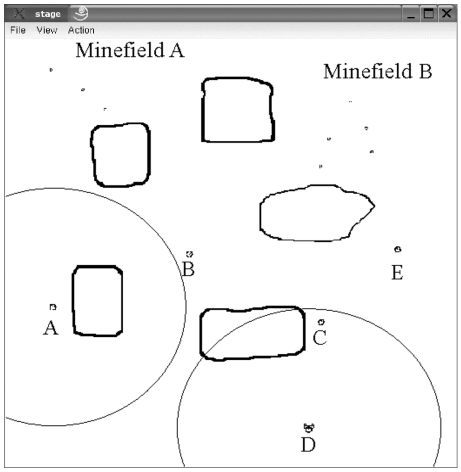

Figure 4. Stage Screenshot 1

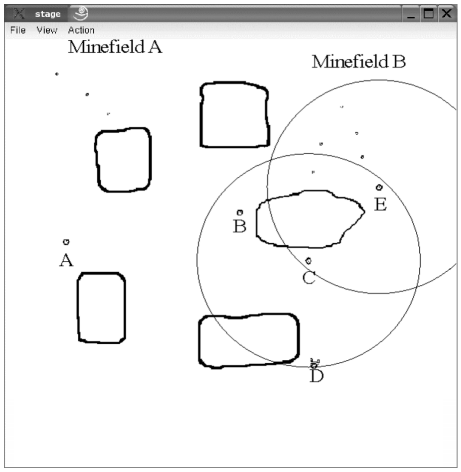

Figure 5. Stage Screenshot 2

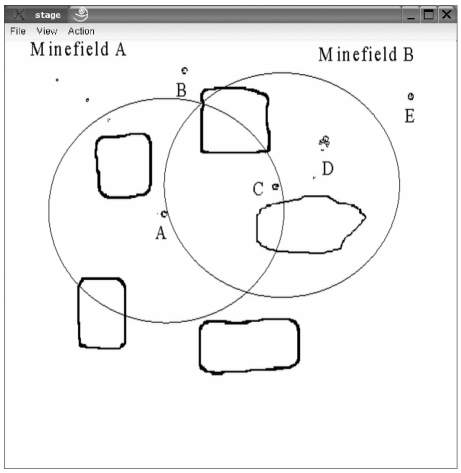

Figure 6. Stage Screenshot 3

the self-healing capacity to be utilized to maximum effect without disrupting the primary network task. Future work will include improving the self-healing mechanism so that the node nearest to the target node's estimated destination changes course to deliver the message instead of the robot trying to send the message. Furthermore, special motivations will be developed to ensure that different quality of service demands can be met in a dynamic wireless mobile environment.

References

[1] A. Jardosh, E. Belding-Royer, K. Almeroth, and S. Suri. Towards Realistic Mobility Models For Mobile Ad hoc Networks. *ACM MobiCom 03*, pages 217–229, 2003.

[2] Ns2. Network Simulator http://www.isi.edu/nsnam/ns/.

[3] Glomosim. GloMoSim Scalable Network Simulator http://pcl.cs.ucla.edu/projects/glomosim/.

[4] Yongguang Zhang and Wei Li. An Integrated Environment for Testing Mobile Ad-Hoc Networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 104–111. ACM Press, 2002.

[5] robotics.usc.edu. Stage. Mobile Robot Simulator http://playerstage.sourceforge.net/stage/stage.html.

[6] A.J. Goldsmith and S.B. Wicker. Design Challenges For Energy-Constrained Ad Hoc Wireless Networks. *IEEE Wireless Communications*, 9(4):8–27, 2002.

[7] Stavros Toumpis and A.J Goldsmith. Performance, Optimization, and Cross-Layer Design of Media Access Protocols for Wireless Ad Hoc Networks. In *International Conference on Communication (ICC) 2003*, pages 2234–2240. IEEE, May 2003.

[8] Izhak Rubin and Arash Behzad. Cross-Layer Routing and Multiple-Access Protocol for Power-Controlled Wireless Access Nets. *Proceedings of IEEE CAS Workshop on Wireless Communications and Networking*, pages 1–7, 2002.

[9] A. K. Singh, V.T. Raisinghani, and S. Iyer. Improving TCP Performance over Mobile Wireless Enviroments Using Cross Layer Feedback. *Proceedings on the IEEE International Conference on Personal Wireless Communications*, pages 81–85, 2002.

[10] S. Shakkottai, T. S. Rappaport, and P.C. Karlson. Cross Layer Design for Wireless Networks. *IEEE Communications Magazine*, 41(10):1–14, 2003.