Design and Simulation of a high performance Emergency data delivery Protocol

Kevin Swartz and Di Wang
Computer Science and Engineering division at Informatics and Mathematical Modelling
Technical University of Denmark (DTU), 2800 Lyngby, Denmark
{swartzk,diwangbruce}@gmail.com

Abstract

The purpose of this project was to design a high performance data delivery protocol, capable of delivering data as quickly as possible to a base station or target node. This protocol was designed particularly for wireless network topologies, but could also be applied towards a wired system. An emergency is defined as any event with high priority that needs to be handled immediately. It is assumed that this emergency event is important enough that energy efficiency is not a factor in our protocol. The desired effect is for fast as possible delivery to the base station for rapid event handling.

1. Introduction

This protocol is intended for use with wireless sensor networks as sort of a back up protocol, or second state for the general topology. Since it is not designed to be easy on power consumption, it should not be used as the general system protocol; rather it should switch to this protocol in the event of an emergency, and then revert back to its normal state once the emergency event has been dealt with. It is also important that data is reliably transferred to its destination, as if it is an emergency; the data is likely important and time sensitive.

The network simulation program ns-2 was decided upon to be the best simulation program for this stage of protocol development. As of present time, ns-2 offers the best testing environment and tools for simulation. NS-2 is able to simulate complex networks of wireless topologies, as well as offer an effective analysis package and online information resource.

There are many applications for this protocol; one such example being structural weakness detection, in which pressure sensors are placed inside the structural supports of a building to detect abnormal symptoms. Another application could be in the area of automobiles, in the sense that the temperature and condition of the car could be monitored in order to gracefully handle a wearing part or low fluid.

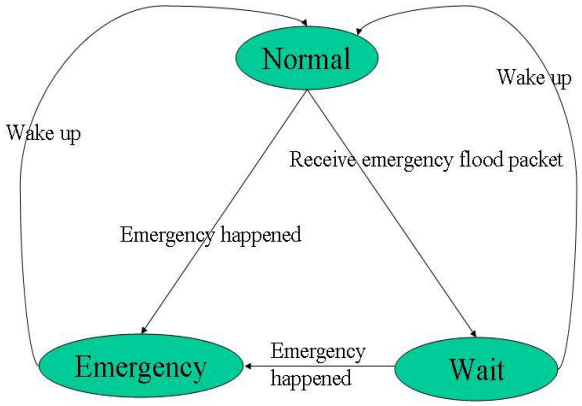

Figure 1: Emergency operation cycle

It should also be noted that this protocol can be used to varying degrees of effectiveness based on the underlying 'normal' mode of operation. For example, if the normal mode of operation involves a protocol that stores optimized paths to the base station, this protocol will be faster than with a normal mode of operation that doesn't. The reason for this will be discussed in later sections.

2. The Protocol

A. Assumptions

There are three main assumptions for this protocol.

1. Power is not a Factor

It is assumed that in an emergency event, the emergency that has occurred has a greater priority over energy conservation, which is sacrificed in order to deliver emergency information to its destination as quickly and reliably as possible.

2. A base station cannot go down.

This must be an assumption with any network, as a network without a base station has no external control and no purpose if not reporting to something outside of itself. This is generally considered a fatal error for any network, and this is particularly true for a network running under this protocol.

3. Data can flow across a network faster if no other nodes are broadcasting information.

This is the central idea of this protocol. If no other nodes are transmitting, then the only limitations set upon transferring data between nodes are the physical characteristics of the system. No collisions yield the effect of no needless repeat broadcasting, which will keep wasted time to a minimum.

.

B. One Emergency

Since this protocol functions as a back-up, or secondary protocol, there will be a finite amount of time in which the network operations under this protocol. The time it takes to complete one cycle of this protocol will be defined as the time from when a node first detects an emergency, until the time when the network has returned to a normal state of operation.

Assume that an event occurs at a certain point in the network. The node detecting this event decides that this event is an emergency. This node immediately sets itself into an emergency state, in which no messages pertaining to normal protocol operation are sent. At this point, the node, or nodes which detected the emergency send out a special type of packet, which for now will be called the emergency packet.

This emergency packet causes any node that receives it to set itself to emergency mode as well, and while in this state will also not send any packets pertaining to 'normal' operation. This packet is fully flooded across the network, with only a small impact from collisions due to the small size of this packet. Once a node is shut down it will not broadcast anything except for more emergency packets and data packets from the node which initiated the emergency.

The negligible effect of collisions can be deduced from having an extremely small packet size for this emergency packet. The point of this packet is to shut down the network as fast as possible, so it must be small and fast. At this point the node detecting the emergency could conceivably re-broadcast the emergency packet again in another flood, in an attempt to shut down as many nodes as possible that weren't shut down by the first broadcast. This second broadcast may be determined to be beneficial through network simulation and testing.

After this emergency packet flood has been initiated, the node at which the emergency originated begins to send the data pertaining to the emergency. The protocol used to locate and route to the base station or data target can be customized depending on the 'normal' operational mode.

For example, if the normal operational mode utilizes some form of ant algorithm [3], it is very possible that the node at which the emergency originated will already know of a fast path to the base station or target node. This will make for quick and reliable data transfer. This is just one situation; many other underlying protocols could be used successfully for the correct balance of speed and reliability. For the ease of testing, a network flood is used to transfer data packets, but this is not considered optimal, and is used more to detect the worst case of the protocol.

These data packets will travel through the network mostly without interference; as most if not all nodes have set themselves to emergency mode, and are waiting on any packets from the emergency originator it might receive. If any other packets not pertaining to an emergency are broadcast, they are ignored completely.

Once the emergency originating node has finished broadcasting the emergency data, it stays in emergency mode, and awaits further instructions. These instructions could be information about other existing emergencies, or a wake up call from the base station or target node. Once the target node has handled the emergency, it floods the network with a wake up packet, which returns all affected nodes in the network to their normal network state.

C. Inner workings

There are several types of packets sent across in this scenario whose details were not discussed. It is important to note the contents of these three different types of packets. The initial emergency flood packet contains only the ID of the node which encountered the emergency, and a special emergency code. The emergency code is interpreted by any receiving nodes as a signal to set itself to emergency mode, and forward only packets pertaining to emergencies. The point of this packet is to be as small as possible, to ensure quick propagation throughout the system.

The second set of packets sent is not as important to this paper's discussion. They contain emergency data that for all relevant purposes need no more than to arrive at the correct destination quickly. However this packet needs to still include the emergency code and node ID of emergency-originating node, else it will be ignored.

The last type of packet, containing the wake up command, contains only the wake up command, as interpreted by the nodes, and also the node ID of the emergency originator. The node ID is included in the wake-up packet in order to handle multiple emergencies, which are discussed in the following section.

D. Multiple Emergencies

This protocol also needs to be able to handle multiple emergencies, or possibly a distributed emergency that is caught by multiple nodes. This is possible due to several features of the protocol.

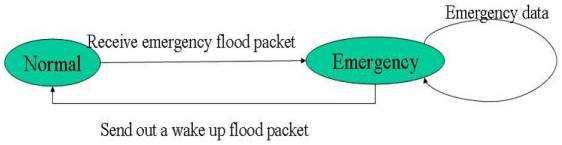

Figure 2: Target node states

When a node receives an emergency packet, it becomes dormant, responding only to packets that contain the node id of the originating node. However it can also respond to other emergency packets it receives, forwarding them accordingly. Each node stores the originating node ID inside of a queue of some kind for each different emergency encountered. As each node receives wake up packets containing node IDs, the node ID entry is removed from each node's list, resuming normal activity only once the list is empty.

The base station, or target node reacts similarly. It goes into an idle state if it receives an emergency packet and waits for emergency data, and similarly wakes up upon emptying its queue of emergency processes. The only difference in this case is that the target node can also send out wake up packets if one of the current emergencies has been dealt with. If there are multiple emergencies with data intended for the same base station, it sends out a wake up packet for each emergency.

It may also be necessary to keep a timeout value for each emergency entry in each nodes emergency queue list. This is to protect nodes on the network from becoming non-functional in the case that wake up packets are lost or unsent. It will be important to fine tune the time-out time for the entries in the emergency queue. A time-out too short could prevent correct handling of an emergency, while too long a time could significantly decrease network performance.

As an example of a multiple emergency situation, let two emergencies occur at the same time. Each node detecting the emergency will send out a wave of emergency packets, setting the nodes to stand-by. This is possible because nodes set to emergency mode will only transmit other emergency packets that it hasn't seen before. The nodes set to emergency mode will then have two node IDs in their respective queues, and will not re-activate until wake-up packets for each node ID has been received, or there is a timeout on the queued node IDs.

If the emergency target nodes are different nodes, the emergencies are handled separately, and wake-up packets are sent independently. If the emergency target nodes are the same node, then the emergencies will be handled as they are received, and wake-up packets will be sent out accordingly.

There is yet one unresolved issue with multiple emergency handling, which is that since emergencies are identified by the node ID that detected it, only one emergency at a time can be handled per a single node. If a node in the network detects an emergency, and then the same node detects a second emergency right after it, the network will have no way of keeping track of two different emergencies for the same node. The first emergency would have to be taken care of before the same node could create a second emergency. This issue could be dealt with by adding an extra field to the emergency packet, which is incremented if the node as an unresolved emergency tied to its node ID already.

3. Simulation

A. NS-2

NS-2 has the potential to be a very powerful simulation program. The scripts are written in TCL, which is a fairly versatile language, in addition to being fairly readable. It can be installed on any Linux system, as well as Cygwin under Windows.

There were some difficulties in installing ns-2. It was found to be necessary to manually change some of the configuration files. A find and replace for .relid`` with .relid` was necessary in order for it to installed on Debian Linux, or under Cygwin.

NS-2 also comes as a package, with several different packages and visualization programs in order to make the ns-2 output more readable. NAM is an important part of ns-2, in that it animates the events of the network, and is usually started directly in the ns-2 TCL script.

There were also some difficulties with the trace file output of ns-2. One of the waveform generating programs packaged with the ns-2 all-in-one package, Xgraph, was difficult to get working at first due to

some incorrect parameters initially set in the ns-2 script.

B. Network simulation

The network topology and program design is designed with help from the flooding protocol example in the ns-2.29/tcl folder, enclosed in the installation package. This was found to create a good network topology that would provide a good environment to test the emergency protocol. This basic topology is pictured in Figure 3.

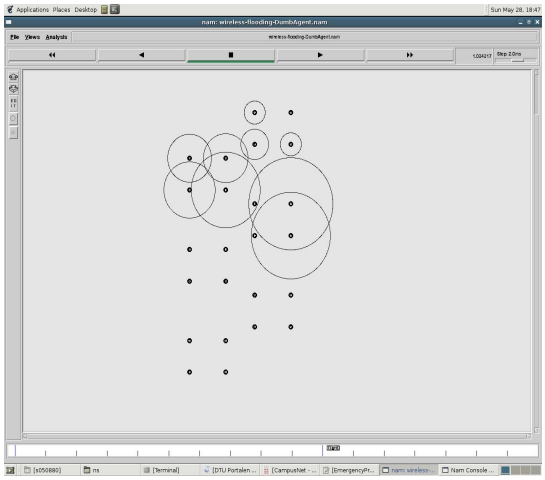

Figure 3: Network Topology

This is a good topology due to the many bottlenecks in the network grid, set on the side of ever four node cluster. The node at the bottom left was said to be the origin of the emergency, while the node at the top right was said to be the emergency target node.

It should be noted that the emergency data transfer protocol is set to be a simple network flood in this simulation. This was chosen because a network flood is the worst case scenario for transferring the data. It is possible to design or implement custom protocols using C++, but the timeframe was not enough to write and synchronize the C++ and ns-2 scripts.

The main test program involved a few stages of programs. The first program simply tested that the sequence of events occurred and finished correctly. This meant that no other network activity was detected on the system after the time of the emergency packet flood. Subsequent testing involved anywhere from small amounts of network traffic to massive storms of network traffic at precisely the same time as the emergency flood occurred, thus covering best to worst case scenarios.

Difficulties were encountered when it was found that in ns-2 simulations, the animation is the same regardless of packet size. Since one of the key ideas to this protocol involves the size difference in emergency and data packets, another method of performance testing had to be found for this simulation.

The technique used involved timing two methods of delivering emergency data. Method one involved using the discussed emergency protocol, while method two involved simply sending the data as a normal packet to the base station. The generated network traffic is the same in both cases, in order to keep the tests as equal as possible.

Using the discussed emergency protocol, method one, the clock was started at the time of emergency, and was stopped once the network had returned to a normal state of operation. In method two, the clock was started at the time of emergency, but was stopped as soon as the data arrived at the target node, since the network is already functioning in a normal state. However it should be noted that this test still does not test the function of small packet sizes within this protocol.

C. Simulation Results

The emergency protocol was found to respond very well to medium amounts of traffic. For best simulation results, the emergency packet is transmitted only once at the source of the emergency, and then once it is received it gets re-transmitted between nodes 3 times. This is implemented by a simple counter inside the packet receiving function that keeps forwarding emergency packets until it has seen the same packet 3 times. This sounds like it is time consuming, but it appeared not to have a large effect on performance, though it is taxing on network activity.

The graph of this can be seen in Figure 5. This graph is of the number of packets received vs. time for several key nodes in the system. These nodes are: node 0, which originates the emergency wave, node 10, which is in the middle of the topology as one of the bottlenecks, and node 23, which is the emergency target node. This graph represents only the initial emergency packet wave, and no data transfer. It can be observed at the beginning to have normal traffic operating on the network. At a certain time, the traffic spikes, the spike being the emergency wave packet. Now at this point, the network is programmed to still attempt to emit ‘normal’ packets for the rest of the simulation, but it can be seen to have no effect on accurate delivery of emergency data.

The serious test involved creating extremely high traffic across the length of the network. This is done by creating network floods every .01 seconds from

random nodes throughout the network. This high traffic can be seen in Figure 4.

This high traffic scenario was used to test two protocols against each other, the emergency protocol, and a general network flood.

For the network flood test, the network attempted to flood a data packet from the same node (node 0) every .01 seconds for .3 seconds, meaning 30 attempts at transmission. Due to a high volume of collisions the packet was unable to make its way to the base station.

The emergency protocol test was next, and worked wonderfully. Due to its emergency packet re-transmissions, it was able to set the network to emergency mode, send the emergency data, and wake up the network again in less than .06 seconds from the moment of the emergency.

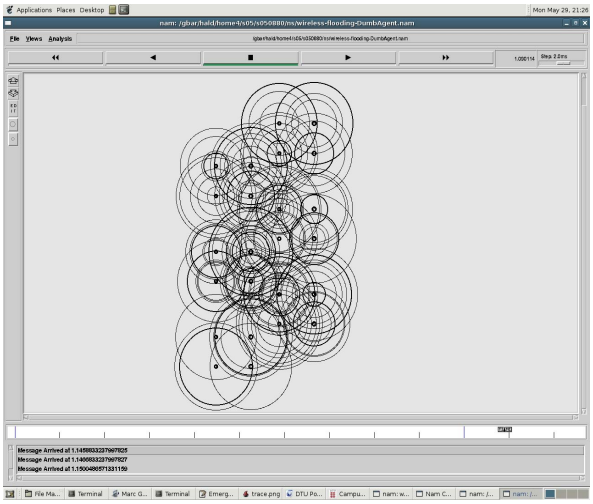

Figure 4: High traffic simulation

The reason the emergency protocol is so successful, is while the regular node flooding encounters a collision and restarts from the beginning, the emergency protocol has already silenced a portion of the network, and so can start over from that same point on the second re-transmission. This is in addition to sending another emergency packet immediately after the first packet collides with some other network activity.

The emergency protocol obviously shows quite an advantage in this area, but it should be noted that the .06 seconds is also probably the fastest time possible (at least simulated on ns-2). The only way that it could be further optimized, is by having a direct path to the base station for data transfer. Since ns-2 could not simulate the difference in packet size this was not able to be tested. It is also assumed in the testing that the data is only contained in one packet, and this was also not otherwise tested.

NS-2 was observed to have some strange behaviors when changing the packet size of the data. The original idea for testing was to make the wave packets as small as possible, while making the data packets as large as possible. It was thought that this would provide a good example, since in theory the emergency packets will finish transmitting faster since they are relatively much smaller. However once the size of the data packet got any larger than 3 times the size of the wave packet, the next node would not flood the packet. The packet would just be dropped past the first or second hop. This could be some kind of inside protocol of ns-2, but we were forced to lower the data packet size to be the same size as the wave packets, which in a physical setting would be erroneous.

We did not test the emergency protocol on another network topology other than this one, because it was thought that this demonstrated a bit of a worst case scenario. It was thought that if it worked on this topology, it could work on any other. The reason this is a worst case scenario, is that there are several crucial bottle necks at the corners of the groups of four nodes. This can be seen clearly in Figure 3.

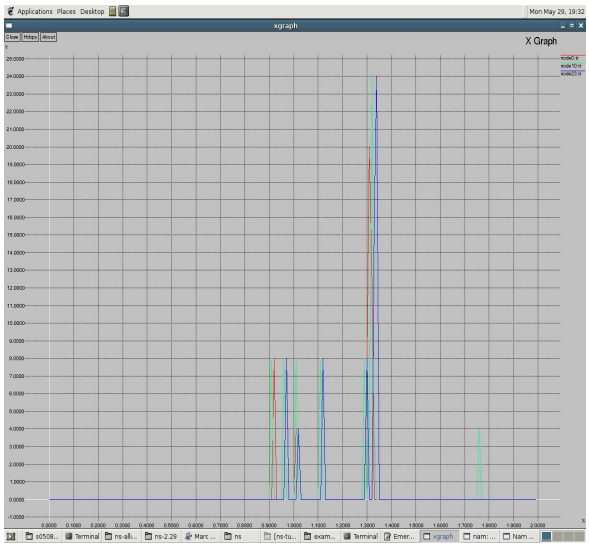

Figure 5: Network activity chart

4. Future Work

This protocol proved to be very useful in simulation, but there are many aspects of it that have yet to be tested. It is necessary to find a method to test the protocol using a program that allows for greater detail when simulating packet sizes. Due to a possible shortcoming of ns-2, or to the shortcoming of this

author's knowledge of ns-2 TCL scripting, it was not possible to simulate a large data packet size.

There was also, of course, no physical testing done. This is the real test of any system, and after some more detailed simulations it would be necessary to test it on a physical system.

Because this was a computer simulation, there were also limits on the number of nodes present in the system. A larger scale test would be highly beneficial. Closely related is also the necessity of test on many different topologies. Although only one topology was used, the authors do not hope to assume that they have thought of everything. In the future it would be necessary to test this protocol over a wide range of topologies to make sure that this works correctly in all situations.

This protocol should also be tested as a secondary protocol for many other 'normal' operating modes. Hidden problems or benefits could arise from being a secondary protocol to different types of 'normal' protocols.

Finally, while the handling of multiple emergencies was mentioned, it was never tested. The authors acknowledge the possibility that hidden problems could arise, in addition to the possibility of a severe negative impact on performance of the system.

5. Conclusion

Although one of the assumptions of this protocol is that energy consumption is not a factor, it is very possible that there are many optimizations that could be made to lessen this protocol's impact on the system. This protocol puts quite a strain on the network bandwidth, as well as completely shutting down most network activity for a period of time. Also, often times, especially in wireless sensor networks, power consumption is a major factor in terms of the life and cost of the system. For this reason, this emergency protocol may not be a viable solution.

It is also important to be careful in defining an emergency per system. While the emergency protocol appears to handle emergencies well, it does this one thing only, sacrificing performance of the rest of the system. If emergencies are triggered easily, the network would never complete any tasks because the nodes would be sleeping in emergency state.

This emergency protocol is definitely in its early stages of development, but the first results are very promising. It remains to be seen if this protocol can withstand the physical testing stage through which any viable protocol must pass.

6. References

[1] M. Greis. *Marc Greis' Tutorial for the UCB/LBNL/VINT Network Simulator "ns"* [Online] Available: http://www.isi.edu/nsnam/ns/tutorial/

[2] A collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. *The ns manual.* [Online]. Available: http://www.isi.edu/nsnam/ns/doc/index.html

[3] Subramanian, D. Druschel, P. Chen, J. Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Data Networks. In Proceedings of IJCAI-97,1997.

[4] D. Braginsky, D. Estrin, "Rumor Routing Algorithm for Wireless Sensor Networks"

[5] Erik Nordström,. Christian Rohner. Interaction between TCP and UDP flows in Wireless Multi-hop Ad hoc Networks Availible: http://user.it.uu.se/~erikn/papers/adhoc05.pdf 5th Scandinavian Workshop on Wireless Ad-hoc Networks (Adhoc'05)

[6] J. Malek, Trace graph - Network Simulator NS-2 trace files analyzer. [Online] Available: http://www.tracegraph.com/