Efficient Wireless Adhoc Node to Node Occupancy Grid Sharing for Mobile Platforms in Urban Environments

Matthew Cork

Abstract

My abstract goes here...

Acknowledgements

Ackowledgements go here

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Chapter 2: Literature Review

Occupancy Grid's (OG) are an important part of mapping while operating mobile platforms in a ever changing Urban environment. One of the major problems with mapping is the need for multiple devices to map simultaneously. This allows for a fastest and more accurate map to be created but creates a new set of problems. Some of these problems consist of an efficient way to shared the OG to ensure that all the mobile platforms have the same updated map and a lack of infrastructure for wireless communication.

Adhoc networks were designed to eliminate the infrastructure requirements and allow for communication between the mobile platforms without the need of a base station.

Pfingsthorn and Birk researched into the efficiency of sharing of the Occupancy Grid (OG) through the three following methods; Sending the occupancy grid periodically, sending the cells when they are alter or sending the sensor data used to create the OG. According to their research, the show that the most efficient method to ensure reliability and to eliminate bandwidth hogging. Each node uses the combination of data sent to generate its own OG map onboard. This method proved to be almost seven times faster than sending the cells as they are modified and over seventy times faster than sending the sections periodically. However the main disadvantage of this method is that all hosts need to have the same data set to have the same occupancy grid. As nodes join and leave they would only get a portion of the data and thus the OG wouldn't be the same. This also would decrease the advantage of having a MAV go offline to explore an area and then synchronise the generated map upon re-entry to the network.

Pham Van's research investigates the issues of bandwidth-hogging and time-sensitivity that is required when streaming video (high volume data) over an adhoc network by proposing a new proactive architecture. The paper analyses the re-transmission time and explains how the new architecture handles the incoming packets. The procedure for handling the incoming packets includes checking if the round trip time is less than the threshold of delay and if it isn't the packet gets dropped as a smaller path already exist within the node architecture.The packets that have less delay are selected for retransmission to the next node in the sequence. The loss of packets is handles by an NACK which reduces the ACK messages on the bandwidth.

Peak Signal to Noise Ratio is the measurable error between the sent file and the received file. Through experimental testing the paper proves that the Peak Signal to Noise Ratio (PSNR) is a 2.27dB improvement on the conventional case of 32.67dB. The paper clearly presents an architecture that allows the efficient streaming of video in real time. The strength of this approach is that the data is sent over the quickest path and and thus doesn't flood the network. It also handles the loss of packets effectively. The weakness is that as all nodes know if they are part of the quickest route if the structure changes suddenly their is no redundancy for this data to get through. For use in my application we can't assume that the recipient is available and that each node knows the fastest path.

Putta, Prasad, Ravilla, Nath and Chandra outlined the different algorithms used for routing in Mobile Adhoc Networks (MANETS). The paper cover two different reactive algorithms; Adhoc on Demand Distance Vector (AODV) andDynamic

Source Routing (DSR), and the main proactive protocol Optimised Link State Routing (OLSR). These algorithms were compared based on the following criteria; Packet Delivery Ratio, Mean end-to-end delay and Routing load. The paper concluded that for all high bandwidth sources proactive protocols (OLSR, etc) allowed for the greatest reliability while for low bandwidth sources, the reactive protocols allow for greatest reliability. Neither reactive protocol proved to be the superior than the other. The paper shows that if the OG was high bandwidth is would be more efficient and reliable to use the proactive protocol. For the control data the more efficient protocol would be either of the reactive protocols. The tests of these protocols however didn't allow for a drastically changing MANET.

Abdel-Hardy and Ward also tested the difference between the proactive protocol (OLSR) and the reactive protocols AODV and Dynamic Manet On Demand (DYMO). They tests include one video stream with five UDP data connections, one video stream with 50 UDP connections and multiple video streams. It the first test it was showed that the DYMO performed better than the AODV while OLSR performed the worse. The second case showed that the AODV was more efficient than the DYMO while both reactive protocols were more efficient than the OLSR. In both these test background data was introduced to measure the effect. For the last test they had multiple video sources transmitting data. This test was consistent with their early two test which showed that for smaller dynamic MANETS the DYMO was more efficient. This paper showed that in a constantly changing MANET the DYMO protocol was the most efficient method of communication for both single and multiple video streams.

Chapter 3: Technical Report

Network communication can be achieved on five main network layers. These layers are the physical, datalink, Network, transport and the application. The transmission of data from the application layer is passed down, through the transport, network, datalink and then finally to the physical link.

The physical link is the only link that has a direct connection to another devices. This base link depends on the method used to connect the devices. This could be via copper, fibre or even wireless. The physical layer sends its data using electrical pulses and waves on the electromagnetic frequency. This is received by other wireless devices for communication.

The link layer sits above the physical layer. This layer is mostly implemented in the Network Interface Card (NIC). The NIC is the onboard hardware that handles the data passed from the physical layer. The NIC uses the Message Authentication Code (MAC) protocol which gives a unique MAC address for each NIC device. It is responsible for the:

- Flow control,
- Error Detection and Correction,
- Controlling the Duplexity.

The link layer has a buffer has limited size which means that the NIC hardware needs to control the flow from the host to avoid causing the loss of data through data overflow. This is done through two main methods; software and hardware controlled flow. The NIC is in charge of the hardware flow control and will ensure that the data is less than the buffer. The second option is done through software.

The only part of the link layer that is implemented in software is the driver which allows the program to interact with the Operating System (OS). This driver controls the the software flow and allows the user to change the setting as needed. The error detection and correction is accomplished by using various parity methods. The parity method is given a parity value which is can check to see of there is a corruption. By using this method if a corruption has happened the program will be able to detect where the change is and recover from it. These approaches however will not be able to fix major corruption with the data but will be able to detect the change. The cards other main task to to establish wether the user requires a full duplex channel or only a half duplex. If the user only requires one way transmission they can control the duplexity of the channel through the driver. The two types of communication in the link layer are point to point and broadcasting. The point to point communication requires a single transmitter and a single receiver. The broadcasting communication allows for multiple senders and receivers which causes a problem with over saturation of the bandwidth from any user. The main three methods used to ensure that bandwidth is shared are:

- Channel Partitioning (FDM and TDM)
- Random Access
- Turns based sending.

The two methods used in channel partitioning are the Frequency Division Method (FDM) and the Time Division Method (TDM). Both these methods require division of resources to allow for multiple senders and receivers. The FDM divides up the bandwidth frequency with the number of concurrent programs assigning each program a certain the range of the frequency. This then allows the program to use its frequency as it sees fit. The TDM divides the time into frames and assigns each subsection to a different program. Each program is only allowed to transmit during its allocated time.

The random access principle start by transmitting on the full bandwidth. When it detects a collision it will wait a randomly determined time then retransmit all the data. This approach is simple and effective for a small quantity of programs.

The last category is the turn based sending algorithms. There are many different turn based algorithms that are able to be used however the most common and effective are the ALOHA and SLOTTED ALOHA. The SLOTTED ALOHA method divides up the link layer frame into segments based on the bits of each frame divide by the full bandwidth on channel. The programs only transmit on the start of their period. This requires the synchronisation of all programs to ensure that no program transmit at the same time. The ALOHA starts by transmitting on the full bandwidth. When it detects the collision it will determine the wait time based on a probability. This approach doesnt require any of the programs to be synchronised and still it a simple and effective method of collision avoidance. The link layer has many effective means to allow communication between devices as well as error checking/detection built in.

The next layer is the network layer. This layer is implemented entirely in the software and is controlled by the OS. The network layer uses the IP protocol. Due to the dynamic nature of this layer the IP address is often issued by a Dynamic Host Control Protocol (DHCP) Server. However for any system that it fixed, the IP address is often statically defined. This minimises the network overhead for connection and allows for a quicker establishment of communication between the network layers of different devices. The two main functions of this layer is either forwarding or routing.

Network layer forwarding is the ability for the network layer to determine the correct input to the correct output. This layer is able to sustain many connections simultaneously. When an input in received from one of the connections that need to go to another the connection, the network layer forwarding function is responsible to ensure that the correct input is aligned with the correct output.

The second function of the network layer is the controlling the routing between hosts. Each network layer device has a forwarding table that shows the route to all hosts it has encountered. As the network receives data from new hosts this table is automatically updated by the OS. This data is used by the forwarding function to find the correct output to the host required.

The fourth layer is the transport layer. This layer requires the ports of both the destination and source hosts. This layer uses two different protocols User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP).

The UDP protocol is a connectionless transmission. Due to the lack of connection state, UDP overhead is small. However as the connection cant be verified, congestion control on this level is almost impossible and this makes the protocol unreliable for data that requires a pattern.

TCP is the most common connection orientated protocol. It is established using a three way handshake. This handshake ensures the reliability of the data transfer. The protocol has three main methods to ensure reliability These are

- StopnWait (SW)
- Go-Back-N (GBN)

- Selective Repeat (SR)

The TCP protocol can detect and handles congestion in the bandwidth. It does this by throttling the data allowed to be sent by the application layer. The disadvantage of this layer is the limited buffer. To overcome this problem the transport layer provides feedback to the application layer. This requires the application layer to ensure that the data sent is left then the buffer size. The TCP layer is allows for the full duplexity as well. The SW reliability method is the slowest of the three. When it sends a packet, it waits for acknowledgment that the packet has been received.
The GBN method will send a certain number of packets. It will keep sending the packets in the window until it is acknowledged then will move the window up. This is similar to the SW method however it will send a few packets a once and if nothing goes wrong moves the window by the number of packets. The SR method also requires a window that keeps track of the the frames sent. As the sender receives an acknowledgement is marks that packet as being received however the windows will only move to the last unacknowledged packet. The packets will only be resent when a timeout period for that packet is reached. This is the quickest method as it only resends packets that timeout.
The last layer is the application layer. This layer is the program that is using the network functions. The are no common protocols for this layer. The application was chosen for the intelligent sharing of the occupancy grid. This allows for all the safe guards of the lower layers to be used and allowing the program to be written in higher level code with more functionality.

Chapter 4: The Implementation

The implementation was separated into two major logical parts to simplify the system and to allow for each section to be developed and tested individually. It also allows the shared memory communication to be used for multiple purposes not just limited to sharing between nodes.

Shared Memory Protocol

The shared memory protocol was designed to allow separate programs to shared data between them. The protocol was designed to allows for multiple senders to publish data to a shared data set which then could be access by multiple receivers. The protocol creates a file in the local RAM. This file then allows for the ”file” to be treated like a normal file using the read and write functions. This allows the implementation to use the benefits of the read and write system calls to optimise the writing to memory. The structure of the file is shown in the picture below: ¡SHARED MEMORY PICTURE¿ One of the major problems with multiple sources either writing or reading is the chance of data corruption. This is usually solved using mutex or synchronisation locks. This approach however requires significant overhead for small files. The solution chosen is implementing a circular buffer that minimises the chance of data corruption. This buffer array is ten times the size of the data structure being sent. The shared memory library has two built in counters that keep track of the next available read element and the next write element. When data is written to the shared memory library the read counter is updated to that element of the array, the write counter is incremented to the next element in the array. When read is called, the program requesting the data is given the pointer to the latest element. This pointer remain valid for nine writes. If the new data is written a new pointer will be given to any new programs wanting data. With a buffer array of size ten, the probability that the program will have the data overwritten while it is still reading the old element is negligible. The data section below illustrates the structure of the circular buffer. ¡SHARED MEMORY DATA¿ Each time read is called the program gets a new pointer to the latest data. The shared memory library is an efficient and reliable way to ensure communication between programs on the same platform. This protocol allows the OG map to be shared to the network manager while also allowing many other programs to use the local OG Map (such as path planning programs).

Network

This will explain the network side of the

Chapter 5: Experiments

Experiment 1 - Shared Memory

This will outline the experiments

Experiment 2 - Timing

Chapter 6: Results

Experiment 1 - Shared Memory

This will be where i have the results of test

Experiment 2 - Timing

Chapter 7: Discussion

This will be where the discussion about the efficiency is and what could have been done better

Chapter 8: Future Work

This will be future work

Chapter 9: Conclusion

This will be the conclusion

Appendix