

# **Analysis & Development of Dynamic TLPs in the use of Smart Mobile Device Communication**

K. D. A. D. Wickramaratne

(IT12045204)

Degree of Bachelor of Science

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

August 2015

# **Analysis & Development of Dynamic TLPs in the use of Smart Mobile Device Communication**

Kandebaduge Don Ashan Dulaj Wickramaratne

(IT12045204)

Dissertation submitted in partial fulfillment of the requirements for the degree  
of Science

Department of Information Technology

Sri Lanka Institute of Information Technology  
Sri Lanka

August 2015

## Declaration

“I declare that this is my own work and this dissertation<sup>1</sup> does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

K. D. A. D. Wickramaratne	IT12045204		
Name	Registration No.	Signature	Date

The above candidate has carried out research for the B.Sc Dissertation under my supervision.

Mr. Dhammika H. De Silva		
Name	Signature	Date

## **Abstract**

Smart Mobile devices has revolutionized the human behavior with the introduction of different apps that helps it users in day today activities. We often see Smart devices are in use to help with not just ordinary activities but also for users with special needs. Data transferring in real time path navigation systems with obstacles has to be high speed, efficient and reliable. This requires information with minimal time to reach its destination to avoid hazardous encounters discover the required path, identify real time changes on the path in the current map and give out alternative routes. Our research will look into models and methods to facilitate the bounded timing of which it takes the minimum time for path finding application for people with vision disabilities.

Given a comprehensive analysis of the requirements that are put forward in path navigation applications, it was identified the existing transport layer protocols are not the best option and is ideal for the requirements that we are looking into. I.e. transporting the data with urgent flags, congestion control in the use of Smart Mobile communication. Our research investigates existing TLPs and proposes extensions and modifications to facilitate the required short comings in the use of Smart Mobile communication and will implement Dynamic TLP to incorporate.

## **Acknowledgement**

We would like to express our deepest gratitude to those who helped in with various inputs, suggestions and wisdom when completing this task of Analysis & Development of Dynamic TLPs in the use of Smart Mobile Device Communication.

Our first and foremost thank you goes to our Supervisor, Mr. Dhammika H De Silva for being our guiding start providing us with all the literature and help needed when understanding the concept of a Research Project and its application for better prospects in our lives in the aspect of learning through research. If it's not for him we would not be able to perform the tasks oriented through this research and complete it with fullest confidence in our capacity as undergraduates. He has been always available for us with directions and action points to proceed through the correct path of navigation.

We would also like to thank our lecturer in charge of the module CDAP for keep us focused on the outcomes of a research and encouraging us to pursue the light of a patch to become a confident graduate.

Finally we would like to thank colleagues who helped the group in preparing for this assignment with their inputs, suggestions, valuable hints and for the encouragement which played a major role in completing this research project.

## Table of Contents

<b>Declaration.....</b>	<b>iii</b>
<b>Abstract.....</b>	<b>iii</b>
<b>Acknowledgement .....</b>	<b>v</b>
<b>List of Tables .....</b>	<b>viii</b>
<b>List of Figures.....</b>	<b>ix</b>
<b>1. Introduction .....</b>	<b>1</b>
<b>1.1 Literature Survey .....</b>	<b>2</b>
<b>1.1.1 Background information and overview of previous literature survey .....</b>	<b>2</b>
<b>1.1.2 TCP implementation in Linux.....</b>	<b>4</b>
<b>1.1.3 Behaviour of Data Transfers .....</b>	<b>5</b>
<b>1.1.4 Identification and significance of the problem.....</b>	<b>6</b>
<b>1.2 Research Gap.....</b>	<b>11</b>
<b>1.2 Research Objectives .....</b>	<b>12</b>
<b>1.3.1 Main Objective.....</b>	<b>12</b>
<b>1.3.2 Sub Objectives.....</b>	<b>12</b>
<b>2. Methodology.....</b>	<b>13</b>
<b>2.1 Implementation of Dynamic TCP Header .....</b>	<b>13</b>
<b>2.2 Steps that need to be carried out.....</b>	<b>13</b>
<b>2.3 How it could be done.....</b>	<b>14</b>
<b>2.3.1 Standard TCP header .....</b>	<b>14</b>
<b>2.3.2 Urgent TCP header .....</b>	<b>15</b>
<b>2.4 Implementation of the functions .....</b>	<b>16</b>
<b>2.4.1 tcp_gen().....</b>	<b>16</b>
<b>2.4.2 sendpkt() .....</b>	<b>17</b>
<b>2.5 NS2 Emulation.....</b>	<b>18</b>
<b>2.6 Host machine configuration .....</b>	<b>19</b>
<b>2.7 Emulating the updates to the TCP via the TCL script .....</b>	<b>20</b>
<b>2.7.1 Steps in creating the tcl script.....</b>	<b>20</b>
<b>3. Research Findings.....</b>	<b>21</b>
<b>4. Results and Discussion .....</b>	<b>22</b>
<b>4.1 Evidence .....</b>	<b>22</b>

<b>4.1.1</b>	<b>Generated Output via the TCL .....</b>	<b>22</b>
<b>4.1.2</b>	<b>Output Using the Standard Header .....</b>	<b>23</b>
<b>4.1.2</b>	<b>Output Using the Customized Header .....</b>	<b>24</b>
<b>5.</b>	<b>Conclusion .....</b>	<b>25</b>
<b>6.</b>	<b>References.....</b>	<b>26</b>
<b>7.</b>	<b>Appendix.....</b>	<b>28</b>

## List of Tables

Table 1.Comparison of transport layer protocols .....	7
Table 2. Comparison of different data types over network .....	10



## List of Figures

Figure 1.1. Packet Reception 1 .....	4
Figure 1.2. Packet Transmission 2 .....	4
Figure 2. WAP Protocol Suite .....	9
Figure 3. Standard TCP Header .....	13
Figure 4. Standard TCP header structure .....	14
Figure 5. Urgent TCP header structure .....	15
Figure 6. tcp_gen() function .....	16
Figure 7. sendpkt() function.....	17
Figure 8. NS2 Emulation .....	18
Figure 9. Host Configuration .....	19
Figure 10. TCL Script .....	20
Figure 11. TCL Script is run .....	22
Figure 12. Standard Header with 20 Bytes .....	23
Figure 13. TCP with the urgent Header 16 Bytes .....	24

## **1. Introduction**

Modern world revolves around networks which have access to details, information any corner in world and can be accessible fast. We often use this information to ease our day to day activities be updated with new things, be connected to people around the world and many more.

We have evolved from using information for the purpose of simple/advance activities to real time quick updates for the purpose of medical reasons, navigation, bank transactions, aviation industry etc.

We find it's very important to focus on transport layer protocols which possess reliable, high speed and efficient data transfer methods which are vital factors in real time path navigation systems which require information within a short time duration to discover paths, avoid hazardous encounters providing alternatives. There's always a demand for reliable information that is accessible in quick successions without having to wait for long. A second delay can cause a lot of damage in a real time system. We identify the current protocols that are being used in TLP has certain faults/ draw backs that prevents the quick succession of transfer of data or retransmit to serve information to its uses.

## 1.1 Literature Survey

### 1.1.1 Background information and overview of previous literature survey

In computer networking, the transport layer or layer 4 provides end to end communication services for applications within a layered architecture of network components and protocols. The transport layer provides convenient services such as connection oriented data stream support reliability, flow control and multiplexing. A comparison of prominent transport layer protocols used in this research is shown below. [1]

	TCP	UDP	DCCP	SCTP
Connection Oriented	Yes	No	No	No
Reliable	Yes	No	No	Yes
Congestion Control	Yes	No	Yes	Yes
Ordered/Unordered delivery	Ordered	Unordered	Unordered	Both
Retransmission flow	End to End	End to End	End to End	End to End

Table 1.Comparison of transport layer protocols

The project team conducted a thorough literature survey focusing on research papers such as “Impact of Mobile Wireless Links on TCP Performance” by Haowei Bai *et al* [2] which states that TCP will still be the dominant end-to-end reliable transmission control protocol at least in the near future; However, TCP was initially designed to perform well in networks with reliable wired links and stationary hosts, where packet losses are mainly due to network congestion. TCP assumes that all packet losses are due to network congestion.

“Internet Accessibility In High-Speed Vehicles” by Hala Eidaw *et al* [3] was another paper analysed which mentioned that computer and wireless communication require internet

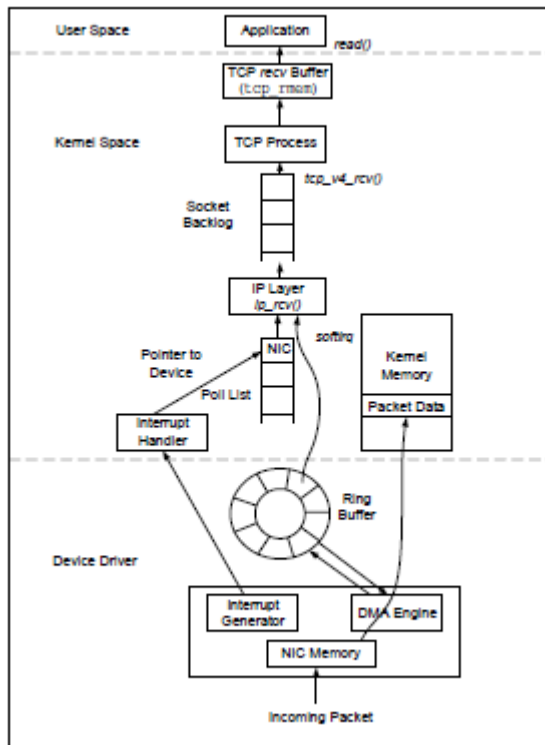
accessibility at anytime and anywhere, this includes in a high-speed mobile station such as in speedy trains, fast moving cars as vehicle-to-infrastructure communication. This increased the development of numerous schemes concerning the need of smooth handover of the mobile nodes.

Moreover focusing on the issues in these protocols, Ibtissam El Khayat *et al* [4] states in his research “Improving TCP in wireless networks with an adaptive machine-learnt classifier of packet loss causes” that nowadays, many applications use TCP as their transport protocol and hence pass through wireless links, which become common in the Internet. Over these links, packet losses are not due anymore only to overthrows but can also be caused by link errors. TCP, which has no mechanism to distinguish packet loss causes, reduces its rate at each packet loss. As a solution the researchers propose to apply a particular learning algorithm called decision tree boosting to automatically design a model for discriminating the two possible packet loss causes and then use this model at best to improve the performance of TCP in wired/wireless networks.

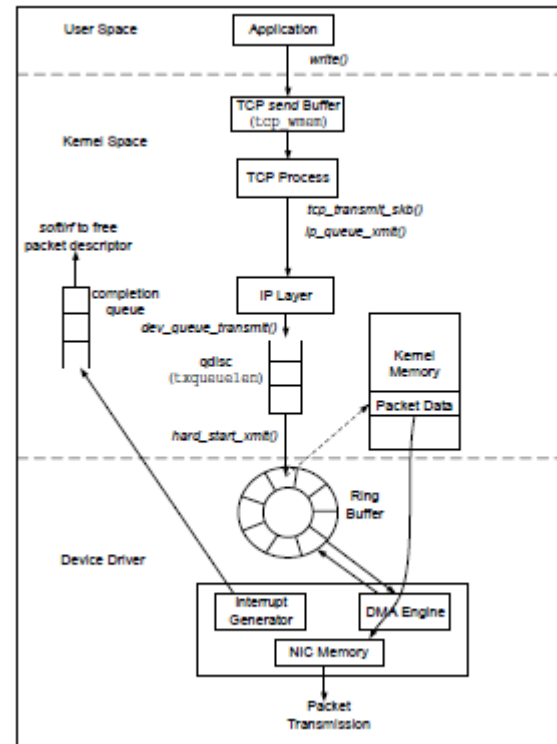
In “Transport Layer Fairness and Congestion Control in Multihop Wireless Networks” it states that experiments and research showed that TCP's congestion control algorithm performs very poorly over wireless ad hoc networks with degraded throughputs and severe unfairness among flows. This paper studies TCP's fairness and throughput issues in wireless ad hoc access networks, and designs an improved congestion control algorithm based on the characteristics of the wireless ad hoc access networks. The protocol is designed as extension to DCCP (datagram congestion control protocol) with a new congestion control component. The research was done by Kunz.T *et al* [5].

“Improving TCP Startup Performance using Active Measurements:Algorithm and Evaluation” by Ningning Hu *et al* [6] is another such study analysed which mentions that TCP slow start exponentially increases the congestion window size to detect the proper congestion window for a network path. This often results in significant packet loss,while breaking off Slow Start using a limited slow start threshold may lead to an overly conservative congestion window size. This problem is especially severe in high speed networks. In this paper we present a new TCP startup algorithm, called Paced Start that incorporates an available bandwidth probing technique into the TCP startup algorithm.

### 1.1.2 TCP implementation in Linux



### Figure 1.1 Packet Reception 1



### Figure 1.2 Packet Transmission 2

Figures 1 and 2 show the internals of the TCP implementation in Linux kernel. Fig. 1 shows the path taken by a new packet from the wire to a user application. The Linux kernel uses an *sk buff* data structure to describe each packet. When a packet arrives at the NIC, it invokes the DMA engine to place the packet into the kernel memory via empty *sk buffs* stored in a ring buffer called *rx ring*. An incoming packet is dropped if the ring buffer is full. When a packet is processed at higher layers, packet data remains in the same kernel memory, avoiding any extra memory copies. Once a packet is successfully received, the NIC raises an interrupt to the CPU, which processes each incoming packet and passes it to the IP layer. The IP layer performs its processing on each packet, and passes it up to the TCP layer if it is a TCP packet. The TCP process is then scheduled to handle received packets. Each packet in TCP goes through a series of complex processing steps. The TCP state machine is updated, and finally the packet is stored inside the TCP *recv* buffer.

### 1.1.3 Behaviour of Data Transfers

In the research “An investigation in to Dynamic TLP’s for Smartphone Communication” It is said, The way finding application designed for Vision Impaired People requires efficient data transfer in between Building Information Model (BIM) and the person with disability - Infrastructure Network and among the peers (Ad-hoc network). This research identified the size of data, data transfer frequency, reliability, transfer direction and type of the network. Table I shows the summery of identified data transfers and behaviour of data in this system [1]

The mobile device of the vision impaired person will capture the GPS, Wi-Fi and sensor data depending on availability and processed data will be send client to server. This data has very small payload. Due to the very high data transfer frequency the end-to-end reliability is not required. If one segment lost, there is another within very short bounded time. [1]

In the case of a change in the existing map (Obstacle-Fixed), mobile device will be triggered to transfer data to the server. [1]

Data	Size	Transmission Frequency	Delivery Reliability	Direction	Network
GPS - Processed	Very Small	High	No	C → S	I
Wi-Fi - Processed	Very Small	High	No	C → S	I
Obstacle - Fixed	Medium	Triggered	Yes	C → S	I
Obstacle – non stationary	Small	Urgent	Yes	C → S C → C	A, I
Map Data	Large	Triggered	Yes	S → C, C → S	I
User Location	Very Small	High	No	C → S C → C	A, I

<b>Sensor data - Processed</b>	Very Small	High	No	$C \rightarrow S$	A, I
<b>Instruction</b>	Small	Urgent	Yes	$S \rightarrow C$ $C \rightarrow C$	A, I

Table 2.Comparison of different data types over network

C- Client, S-Server, A – Ad-hoc, I-Infrastructure

As a summary of the literature review conducted formerly, it was understood that these transport layer protocols were originally developed for wired networks. However due to many reasons such as security, stability, convenience and unlimited roaming and range the protocols were used in wireless networks as well. Yet many shortcomings began to occur and as alternatives, modified versions of these same protocols were developed such as TCP-Reno, UDP-lite.

#### 1.1.4 Identification and significance of the problem

- **Packet losses during transmission**

TCP has been deployed in the 1980s. Its congestion control is based on the fact that packet losses are mainly due to buffer overthrows and it works quite well in such situations. However, nowadays, many applications use TCP as their transport protocol and hence pass through wireless links, which become common in the Internet. Over these links, packet losses are not due anymore only to overthrows but can also be caused by link errors. TCP, which has no mechanism to distinguish packet loss causes, reduces its rate at each packet loss. This reduction is not justified when there is no congestion and the consequence is that the throughput of TCP over wireless link is lower than what it could be. Research conducted by Ibtissam El Khayat [7].

- **Congestion control in data transmission**

TCP (Transmission control protocol) is a reliable, end-to-end transport protocol, which is widely used for data services and is very efficient for wired networks. However, experiments and research showed that TCP's congestion control algorithm performs very poorly over wireless ad hoc networks with degraded throughputs and severe unfairness among flows by Dr. Thomas Kunz [8].

In fact, the simplest way to solve congestion is to employ the principle of packet conversation in which the sender stops sending a new packet until the previous one is successfully delivered to the receiver. For this reason, the TCP protocol, one of the core protocols of the Internet protocol suite, employs the concept of congestion control which dynamically controls the flow of packet inside a network, and prevents network performance collapse. Research conducted by van Jacobson [9].

- **Error control**

This paper we propose a new scheme that can efficiently deliver multimedia over wireless Internet. Packet losses due to congestion or random error can be distinguished in our scheme. Proper congestion control and error control are performed, which are suited for multimedia applications. Featured with retransmission mechanism, our scheme can also reduce the packet loss ratio caused by fading and random errors, conducted by Fan Yang [10].

- **Improve wireless TCP performance**

This paper, they describe the design and implementation of a simple protocol to alleviate this degradation and present the results of several experiments using this protocol. Their aim is to improve the end-to-end performance on networks with wireless links without changing existing TCP implementations at hosts in the fixed network and without recompiling or relinking existing applications. They achieve this by a simple set of modifications to the network-layer (IP) software at the base station. These modifications consist mainly of caching packets and performing local retransmissions across the wireless link by monitoring the acknowledgments to TCP packets generated by the receiver. Their experiments show speedups of up to 20 times over regular TCP in the presence of bit errors on the wireless link. They have also found that their protocol is significantly more robust at dealing with multiple packet losses in a single window as compared to regular TCP. Conducted by Hari Balakrishnan [11].



This paper makes two important contributions. First, they design a network-based solution called the Window Regulator that maximizes TCP performance for any given buffer size at the congested router. Second, they present a scheduling and buffer sharing algorithm that reduces the latency for short flows while exploiting user diversity, thus allowing the wireless channel to be utilized efficiently [12].

This research, explore a new way to make TCP adapt to frequent route changes without relying on feedback from the network. It is based on TCP detecting out-of-order delivery events and inferring route changes from these events. They call it Detection of Out-of-Order and Response (DOOR). Their study has shown that this approach can significantly improve TCP performance over mobile ad-hoc networks [13].

- **Flow Control**

The performance of TCP degrades over wireless links due to high rate of data losses, which are falsely perceived as network congestion state. TCP performance metrics also diminish due to low data rate, since large delays may occur in last link i.e. wireless link. Similarly in heterogeneous wireless network, packet loss may also occur due to mobility-events that can cause burst-losses, service-disconnection. This motivates to reevaluate TCP control operations and embed some mobility related services to optimize its performance for new generation of wireless networks. TCP for mobile, wireless environment with the help of link-layer triggers which shall be standardized by IEEE 802.21 standard, through its media independent handover (MIH) services. These services are used to detect link layer events in the TCP control operations, and act accordingly to adjust flow and congestion control in a way that does not seriously hamper TCP performance by Muhammad Saeed Akbar. [14]

- **Performance Enhancement of TCP**

Nodes in are distributed and can be statics or mobile. The main advantage of Ad-hoc Networks is that its nodes can be self-organize allowing nodes to connect to each other. The connections between the nodes do not need to establish preexisting infrastructure like other networks. Each node can work as a router to route data to its neighbor nodes. Ad – hoc networks can be used in places that can be difficult to prepare it with infrastructures like open areas. [15]

- **Improving Transport Layer Performance in Multihop Ad Hoc Networks**

TCP congestion control has an implicit assumption, i.e., any packet loss is due to network congestion. However, this assumption is no longer valid in the ad hoc networks as packet losses may well be due to channel errors, medium contention, and route failures.

Several works have pointed out that greedy TCP can result in severe congestion in ad hoc networks and hence performance degradation. Subsequently the TCP source will reduce congestion window size before it becomes excessively large. To avoid congestion, Chen et al. dynamically adjusted the congestion window limit according to path length of TCP flows, a neighborhood RED scheme was proposed to alleviate TCP fairness problem by adjusting marking/dropping probability in light of observed channel information. [16]

In terms of how novel the research topic is it could be considered different and unique since currently such protocols designed for wireless networks within the transport layer are rarely used.

One such protocol is the Wireless Application Protocol (WAP) which is a technical standard for accessing information over a mobile wireless network. A WAP browser is a web browser for mobile devices such as mobile phones that uses the protocol.

The WAP standard described a protocol suite allowing the interoperability of WAP equipment, and software with different network technologies, such as GSM and IS-95(also known as CDMA). [8]

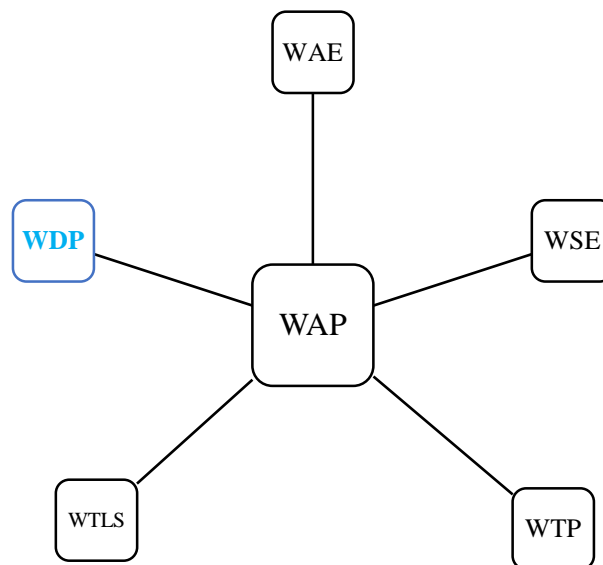


Figure 2.WAP Protocol Suite

However the only transport layer protocol within this suite is WDP (WAP Datagram Protocol).

Wireless Datagram Protocol defines the movement of information from receiver to the sender and resembles the User Datagram Protocol in the Internet protocol suite. It is a protocol in WAP architecture, covers the Transport Layer Protocols in the Internet model. By letting only the transport layer deal with physical network-dependent issues, global interoperability can be acquired using mediating gateways.

Disadvantages of the WAP architecture:

- Thin client architecture
- Normal web technology cannot be used in WAP client
- WAP clients are limited to handheld wireless devices such as mobile phones
- Slow speed of access
- Limited availability

In the research done by Sankara Krishnaswamy on “Wireless Communication Methodologies & Wireless Application Protocol” it is termed that during the study 70% of the users rejected the idea of WAP enabled phones. Some of the disadvantages of WAP clearly made the users to decide not to like this. Because of the misguided use of design principles from traditional Web design, the usability of the **current WAP services is reduced considerably**. WAP is facing the same problem as WEB designs faced in 1994 during the evolution of the Internet. For example, some of the WAP designs that use more screens to display information could have been displayed in a lesser number of screens. This kind of design may work on the Web if users have a big-screen PC, but on a small-screen device, designers must cut short each service down to its essence and show much less information. The time taken to perform a query on the Internet through the WAP is also not acceptable by the users.

Thereby it is vital that an accurate wireless protocol suited for the transport layer is designed. It should be able to avoid the existing drawbacks of the above mentioned wired and wireless protocols. This would be the height of significance this research project is aimed at.

## 1.2 Research Gap

These existing Transport Layer Protocols (TLP's); mainly TCP since it has a high rate of data packet loss and high retransmission time when operating in wireless networks.

There are various additions and enhancements of both TCP and UDP, such as Multipath TCP[15], Datacenter TCP[16] or UDP-Lite. These protocols are often designed for a specific use case or mitigate a common drawback of the original transport protocol. TCP does more than just break things up into packets. It also makes sure that the data arrives, resending packets where necessary. But for a question that fits in a single packet, we don't need all the complexity of TCP to do this. There's always a demand for reliable information that is accessible in quick successions without having to wait for long. A second delay can cause a lot of damage in a real time.

All new TCP-like protocols are only modification of basic TCP - Protocols. In them there are no essential changes. The behavior of these new Protocols is more or less good, but all of them have not only advantages but also lacks. Process of replacement is very slow and should be very long.

We identify the current protocols that are being used in TLP has certain faults/ draw backs that prevents the quick succession of transfer of data or retransmit to serve information to its uses. There's always a demand for reliable information that is accessible in quick successions without having to wait for long. A second delay can cause a lot of damage in a real time system.

In our proposed system we focus to develop a protocol that support minimal packet drops and takes a low retransmission to transmit data in quick successions with in a wireless environment. This has been identified as a very important factor in real time systems which are focusing on path navigation applications and other related areas which gather information on critical information like patients, avoid accidents for disabled while providing alternate routes in navigation systems in instances of impediments.

## **1.2 Research Objectives**

### **1.3.1 Main Objective**

To design an enhanced dynamic transport layer protocol which adapts itself according to the data that's being transmitted across a network. For an example if there's any urgent data that needs to be transmitted immediately, a different approach will be used rather than using the conventional transport layer protocol, TCP, which is proven to be inefficient in such situations.

### **1.3.2 Sub Objectives**

- Analysis & classification of application layer data in order to determine which method/functionality to be used in transmitting those data.
- Examining the currently available transport layer protocols & analyzing their behavior to identify their drawbacks.
- Handover method analysis to significantly improve the connection establishment process & to reduce the no of times a particular client has to establish a connection.
- Analyzing the modifications that can be applied to the currently available protocols.

## 2. Methodology

### 2.1 Implementation of Dynamic TCP Header

TCP adds 20 bytes of standard header and up to 60 bytes with options [1], It was noted that this is well suited to transferring large amount of data.

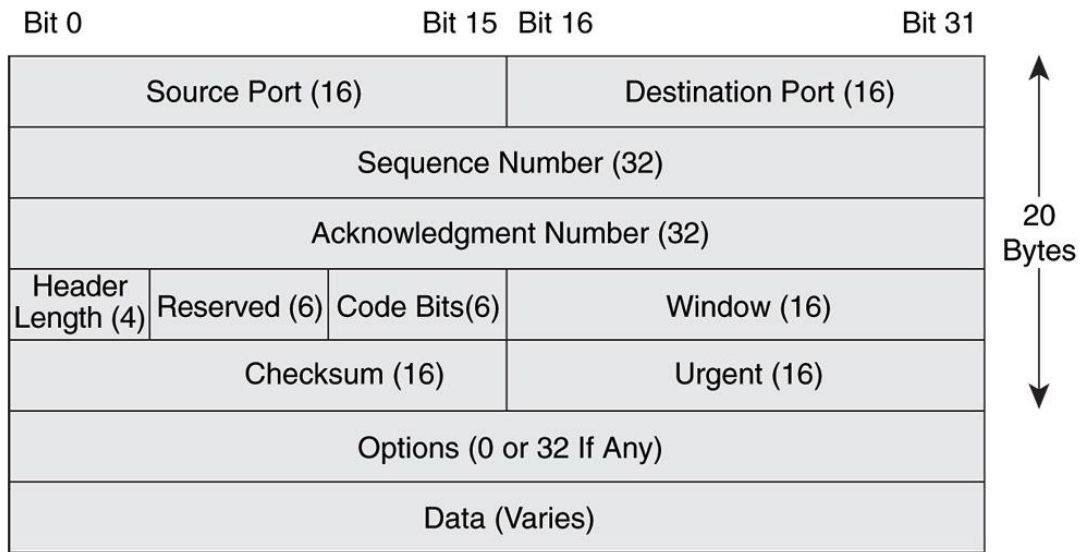


Figure 3 : Standard TCP Header

The mobile device of the vision impaired person will capture the GPS, Wi-Fi and sensor data depending on availability and processed data will be send client to server. [Urgent Data]. This data has very small payload. However for smaller amount of application layer data, TCP can impose considerable overhead leading to significant additional delay effecting data transfer efficiency [2].

### 2.2 Steps that need to be carried out

- Develop a Dynamic TCP Header.
- Which changes its size according to data coming from application layer.
- Urgent data TCP header has to be less than 20 bytes
- Non-urgent data TCP header = 20 bytes

## 2.3 How it could be done

- Removing non-critical information from standard TCP header where those are not needed such as in a scenario where urgent data transfer is required.
- This results in a size reduction of the TCP header.
- To achieve this a new header structure needs to be defined in the Linux kernel.

### 2.3.1 Standard TCP header

The following figure illustrates the Standard TCP header which is being used in Unix-like operating systems.

```

/*
 * TCP header.
 * Per RFC 793, September, 1981.
 */
struct tcphdr {
    u_short th_sport;           /* source port */
    u_short th_dport;          /* destination port */
    tcp_seq th_seq;             /* sequence number */
    tcp_seq th_ack;             /* acknowledgement number */
#if BYTE_ORDER == LITTLE_ENDIAN
    u_char  th_x2:4,            /* (unused) */
           th_off:4;           /* data offset */
#endif
#if BYTE_ORDER == BIG_ENDIAN
    u_char  th_off:4,           /* data offset */
           th_x2:4;            /* (unused) */
#endif
    u_char  th_flags;
#define TH_FIN  0x01
#define TH_SYN  0x02
#define TH_RST  0x04
#define TH_PUSH 0x08
#define TH_ACK  0x10
#define TH_URG  0x20
#define TH_ECE  0x40
#define TH_CWR  0x80
#define TH_FLAGS (TH_FIN|TH_SYN|TH_RST|TH_PUSH|TH_ACK|TH_URG|TH_ECE|
TH_CWR)
#define PRINT_TH_FLAGS " \20\1FIN\2SYN\3RST\4PUSH\5ACK\6URG\7ECE\10CWR"

    u_short th_win;            /* window */
    u_short th_sum;            /* checksum */
    u_short th_urp;            /* urgent pointer */
};|

```

Figure 4 : Standard TCP header structure

### 2.3.2 Urgent TCP header

Below figure shows the newly implemented urgent TCP header structure in the kernel to be used in the use of urgent data transfer needs.

```

/*
 * Urgent TCP header.
 */
struct urgtcp_hdr {
    u_short th_sport;
    u_short th_dport;
    tcp_seq th_seq;
    tcp_seq th_ack;
#if BYTE_ORDER == LITTLE_ENDIAN
    u_char th_x2:4,
           th_off:4;
#else
    u_char th_off:4,
           th_x2:4;
#endif
    u_char th_flags;
#define TH_FIN 0x01
#define TH_SYN 0x02
#define TH_RST 0x04
#define TH_PUSH 0x08
#define TH_ACK 0x10
#define TH_URG 0x20
#define TH_ECE 0x40
#define TH_CWR 0x80
#define TH_FLAGS (TH_FIN|TH_SYN|TH_RST|TH_PUSH|TH_ACK|TH_URG|TH_ECE|TH_CWR)
#define PRINT_TH_FLAGS " \20\1FIN\2SYN\3RST\4PUSH\5ACK\6URG\7ECE\10CWR"

    u_short th_win;
};

```

Figure 5 : Urgent TCP header structure



## 2.4 Implementation of the functions

### 2.4.1 tcp\_gen()

- tcp\_gen() function is implemented to generate a new TCP segment which can be sent through a live network using the Emulator functionality in NS2.
- Newly implemented TCP header is being used.
- Values are assigned to its fields.

```

void
TCPTapAgent::tcp_gen(char *packet, unsigned short sport, unsigned short
dport,
                    Packet *nsp)
{
    struct urgtcphdr *tcpheader;

    hdr_tcp* ns_tcphdr = HDR_TCP(nsp);

    tcpheader = (struct urgtcphdr *) packet;
    //printf("size : %d", sizeof(struct urgtcphdr));
    memset((char *)tcpheader, '\0', sizeof(struct urgtcphdr));

#ifdef LINUX_TCP_HEADER

    tcpheader->th_sport = htons(sport);
    tcpheader->th_dport = htons(dport);

    tcpheader->th_seq = htonl(ns_tcphdr->seqno_);
    tcpheader->th_ack = htonl(ns_tcphdr->ackno_);

    tcpheader->th_off = (TCP_HEADER_LEN / 4);

    tcpheader->th_win = htons(adv_window);

    tcpheader->th_flags = 0;
    if (ns_tcphdr->tcp_flags_ & TH_FIN)
        tcpheader->th_flags |= TH_FIN;
    if (ns_tcphdr->tcp_flags_ & TH_SYN)
        tcpheader->th_flags |= TH_SYN;
    if (ns_tcphdr->tcp_flags_ & TH_RST)
        tcpheader->th_flags |= TH_RST;
    if (ns_tcphdr->tcp_flags_ & TH_PUSH)
        tcpheader->th_flags |= TH_PUSH;
    if (ns_tcphdr->tcp_flags_ & TH_ACK)
        tcpheader->th_flags |= TH_ACK;
    if (ns_tcphdr->tcp_flags_ & TH_URG)
        tcpheader->th_flags |= TH_URG;

```

Figure 6 : tcp\_gen() function

### 2.4.2 sendpkt()

- sendpkt() function implementation has been illustrated in the below figure.
- This function will convert NS2 type network packets into real world network packets for it to be able to transfer it across a live network.
- Uses the packets generated by tcp\_gen() function described earlier.

```

int
TCPTapAgent::sendpkt(Packet* p)
{
    int byteswritten, datalen;
    unsigned char *packet;
    unsigned char received_ttl;
    int hlength = IP_HEADER_LEN + TCP_HEADER_LEN;
    struct urtcphdr *tcpheader;

    if (net_>mode() != O_RDWR && net_>mode() != O_WRONLY) {
        fprintf(stderr,
            "TCPTapAgent(%s): sendpkt called while in read-only mode!\n",
            name());
        return (-1);
    }

    // send packet into the live network
    hdr_cmh* ns_cmnhdr = HDR_CMN(p);
    if (net_ == NULL) {
        fprintf(stderr,
            "TCPTapAgent(%s): sendpkt attempted with NULL net\n",
            name());
        drop(p);
        return (-1);
    }

    hdr_tcp* ns_tcphdr = HDR_TCP(p);

    hdr_ip * ns_iphdr = HDR_IP(p);
    received_ttl = ns_iphdr->ttl_;

    // Here we check if ns has sent any data in the packet.
    datalen = ns_cmnhdr->size() - ns_tcphdr->hlen();
    packet = (unsigned char *) calloc (1, sizeof(unsigned char) *
                                       (hlength + datalen));

    if (packet == NULL) {
        fprintf(stderr,
            "TCPTapAgent(%s): Failed allocating memory\n", name());
    }
}

```

Figure 7 : sendpkt() function

## 2.5 NS2 Emulation

- NS2 simulator functionality is not intended to be faithful replicas of real-world TCP implementations. They do not contain a dynamic window advertisement, there is no SYN/FIN connection establishment/teardown, and no data is ever transferred (e.g. no checksums or urgent data).
- Hence, Emulator functionality of ns2 is being used.
- This maps ns2 packet header into the network packet header.
- Then injects it into a live network.
- Which will be consumed by a TCP server in another host.

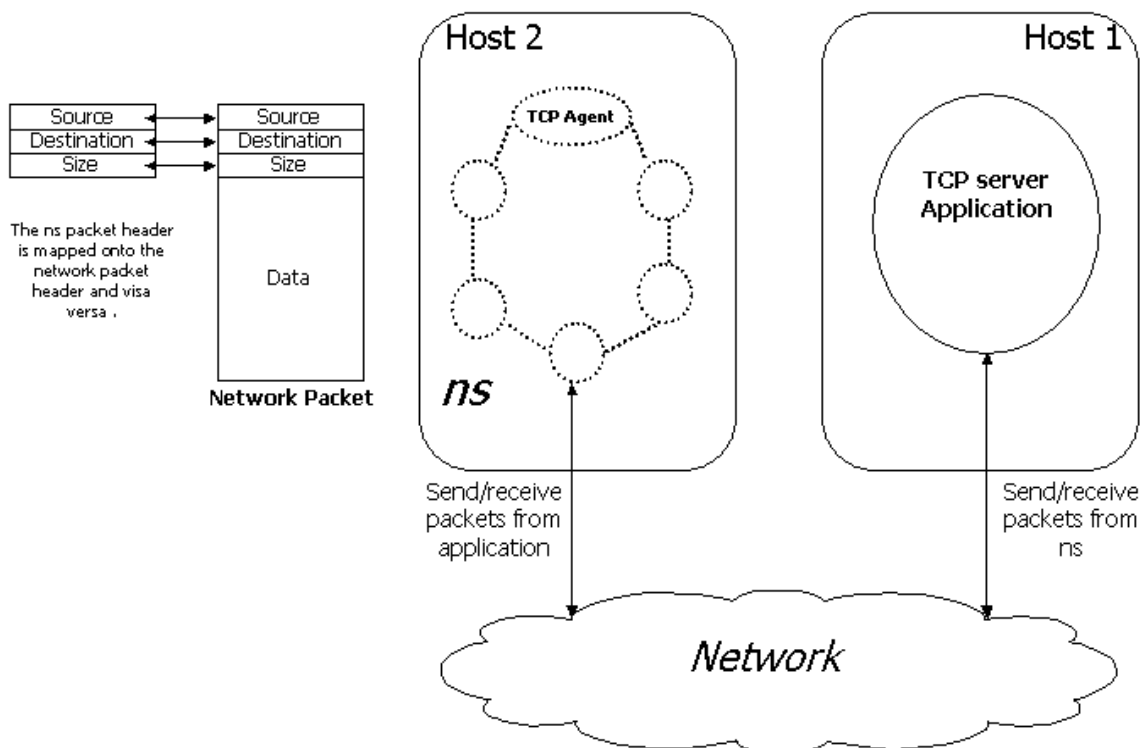


Figure 8 : NS2 Emulation

## 2.6 Host machine configuration

Two host machines should be configured in order to use the emulator with correct IP addresses & port nos. Client which is running NS2 has to be configured with disabled IP forwarding setting whereas TCP server should be configured with disabled IP redirects.

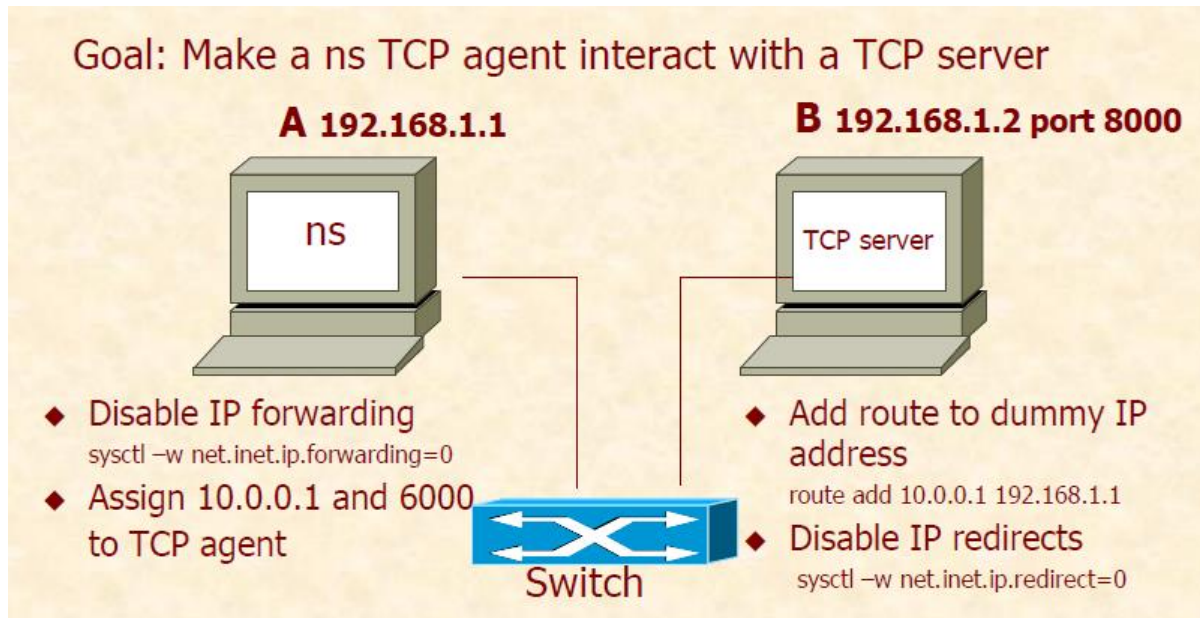


Figure 9 : Host Configuration

## 2.7 Emulating the updates to the TCP via the TCL script

Custom written tcl script is being used to facilitate the NS2 emulation.

### 2.7.1 Steps in creating the tcl script.

1. Activate ns and change to real-time scheduler.
2. Create topology
3. Create TCP Agent
4. Capture & convert to ns format
5. Convert to network format & inject.

```

set ns [new Simulator]
$ns use-scheduler RealTime

set f [open out.tr w]
$ns trace-all $f

set entry_node [$ns node]
set exit_node [$ns node]
set tcp_node [$ns node]

$ns simplex-link $entry_node $tcp_node 10Mb 1ms DropTail
$ns simplex-link $tcp_node $exit_node 10M 1ms DropTail

# Configure the entry node
set tap1 [new Agent/TCPTap];           # Create the TCPTap Agent
set bpf [new Network/Pcap/Live];       # Create the bpf
set dev [$bpf open readonly em0]
$bpf filter "src 192.168.233.135 and src port 5000 and dst 10.0.0.1 and dst
port 6000"
$tap1 network $bpf;                   # Connect bpf to TCPTap Agent
$ns attach-agent $entry_node $tap1;   # Attach TCPTap Agent to the node

# Configure the exit node
set tap2 [new Agent/TCPTap];           # Create a TCPTap Agent
set ipnet [new Network/IP];           # Create a Network agent
$ipnet open writeonly
$tap2 network $ipnet;                 # Connect network agent to tap agent
$tap2 advertised-window 512
$tap2 extipaddr "192.168.233.135"
$tap2 extport 5000
$ns attach-agent $exit_node $tap2;     # Attach agent to the node.

# Configure the TCP agent
set tcp [new Agent/TCP/FullTcp]
$ns attach-agent $tcp_node $tcp

```

Figure 10 : TCL Script

### **3. Research Findings**

In this research our main focus is on catering for the requirements of the way finding applications designed for vision impaired people where current transport layer protocols are not efficient in terms of performance and reliability. To address this issue we have identified four different areas which will further enhance functionality of the current transport layer protocols to better suit for the said requirements.

Depending on the data coming from the application layer the implemented dynamic transport layer will adjust its characteristics such as reliability, flow control, error control as well as its size to be able to provide a better throughput across the network where it is being used.

Secondly our implemented solution will address the issue where having to wait till the connection establishes to transfer urgent data by using techniques to transfer urgent data if needed in the very first step of the connection establishment process itself.

Batch acknowledgment function will introduce sending acknowledgments once for several sequential requests where the time waited for an acknowledgement will be significantly reduced where client has to wait for the acknowledgement from the other end to empty its buffers.

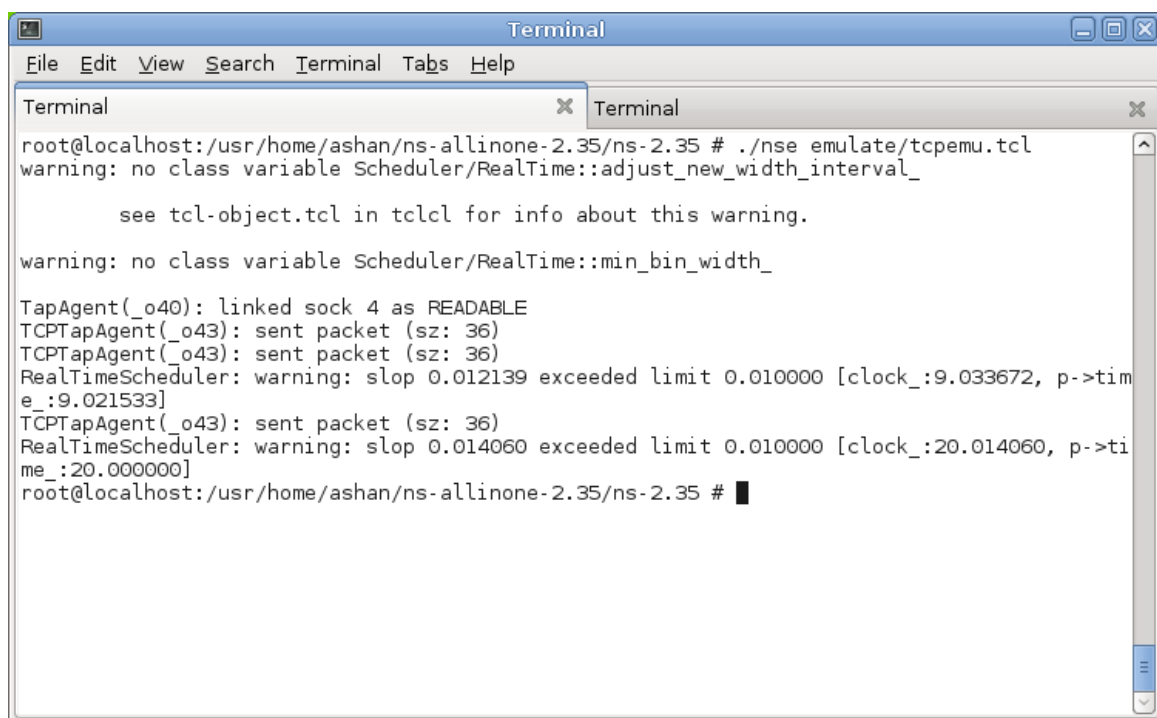
Error control, congestion control which help implement reliability of the TCP protocol has also been looked into in order to eliminate or adjust where those functionalities are no longer needed in the use case of an unreliable/urgent data transfer behavior.

## 4. Results and Discussion

### 4.1 Evidence

#### 4.1.1 Generated Output via the TCL

- Emulation starts once tcl script is run.
- It will send 3 packets to the outside network which will be grabbed by the TCP server and replied with.

A screenshot of a terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). The terminal shows the execution of a TCL script. The prompt is "root@localhost:/usr/home/ashan/ns-allinone-2.35/ns-2.35 #". The command entered is "./nse emulate/tcpemu.tcl". The output includes several warnings about class variables and packet transmissions. The script ends with a prompt "root@localhost:/usr/home/ashan/ns-allinone-2.35/ns-2.35 #".

```
root@localhost:/usr/home/ashan/ns-allinone-2.35/ns-2.35 # ./nse emulate/tcpemu.tcl
warning: no class variable Scheduler/RealTime::adjust_new_width_interval_
        see tcl-object.tcl in tclcl for info about this warning.
warning: no class variable Scheduler/RealTime::min_bin_width_

TapAgent(_o40): linked sock 4 as READABLE
TCPTapAgent(_o43): sent packet (sz: 36)
TCPTapAgent(_o43): sent packet (sz: 36)
RealTimeScheduler: warning: slop 0.012139 exceeded limit 0.010000 [clock_:9.033672, p->time_:9.021533]
TCPTapAgent(_o43): sent packet (sz: 36)
RealTimeScheduler: warning: slop 0.014060 exceeded limit 0.010000 [clock_:20.014060, p->time_:20.000000]
root@localhost:/usr/home/ashan/ns-allinone-2.35/ns-2.35 #
```

Figure 11 : TCL Script is run

### 4.1.2 Output Using the Standard Header

Wireshark application has been used to capture the live packets which are being transmitted to server from client when the tcl script is run. The following figure shows a captured TCP segment which uses the standard TCP header when sending data where reliability is not required.

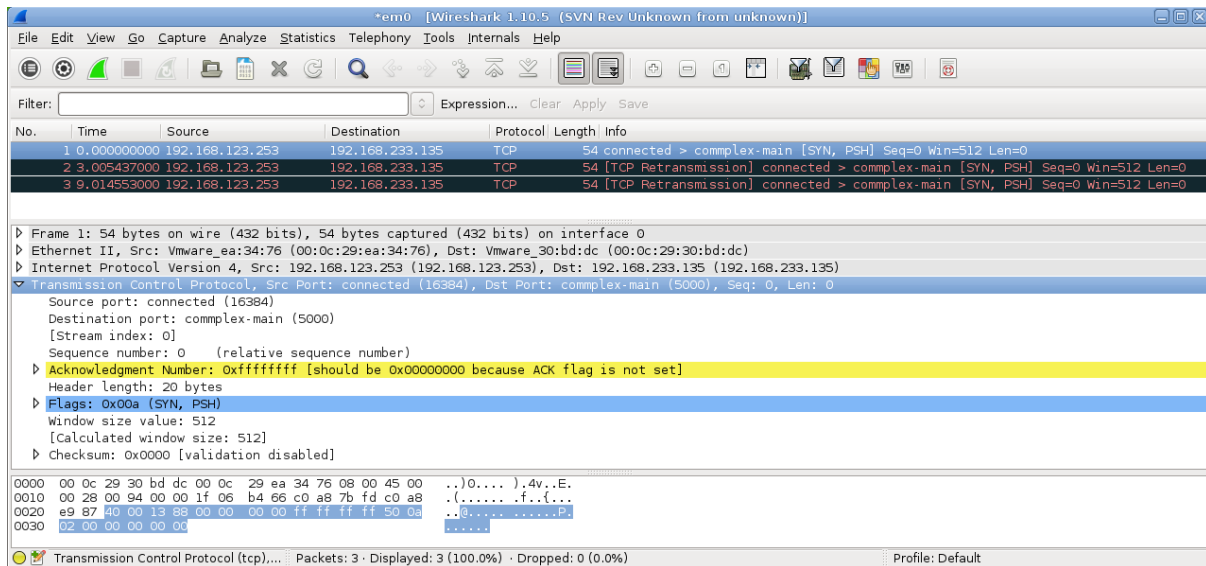


Figure 12 : Standard Header with 20 Bytes



### 4.1.2 Output Using the Customized Header

When there is urgent data which needs to be transmitted it will use the urgent TCP header instead of the standard header which in turn will reduce the size of the network packet. Thus resulting faster transfers of the urgent data. Here in this scenario urgent header size has been reduced to 16 bytes.

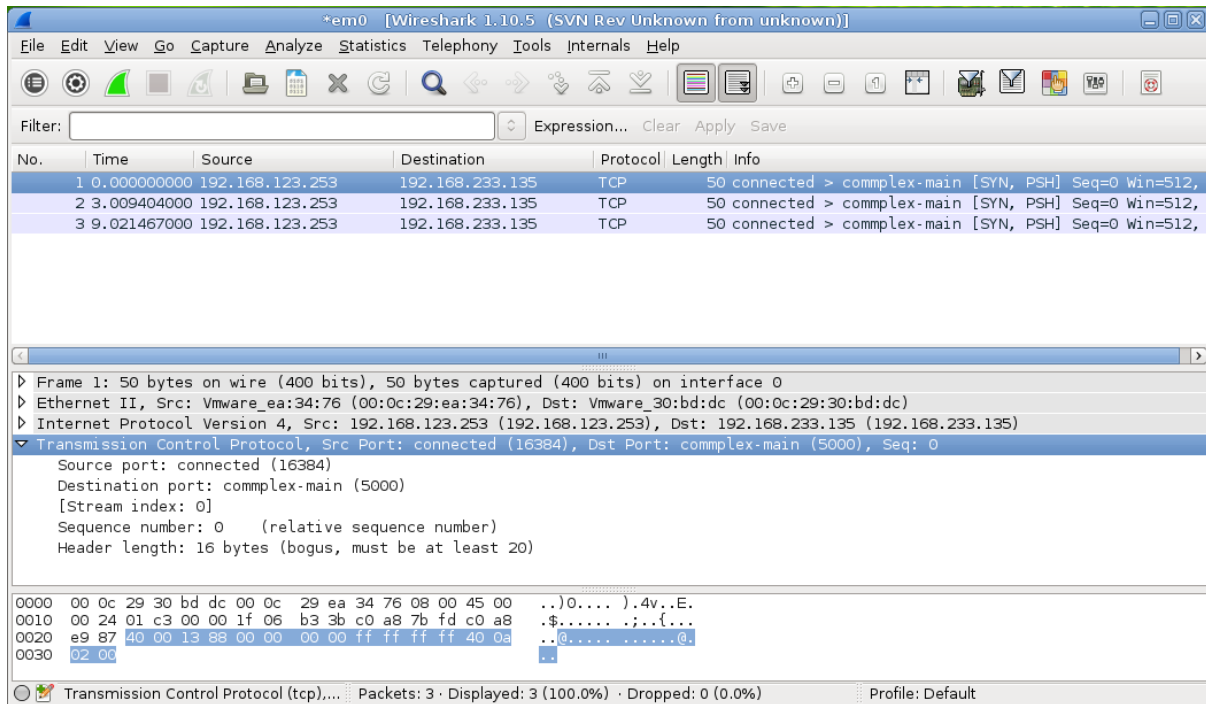


Figure 13 : TCP with the urgent Header 16 Bytes

## 5. Conclusion

As mentioned in the objectives, this research is to design and develop a dynamic Transport Layer (DTL) protocol, which is well suited for low latency, small payload data transfers or large data without the requirement of latency or combination two. Providing data transfer in a timely manner with the reliability will be very useful especially for the visually impaired to avoid dangerous situations such as fire and change shore. This will be useful when the existing maps are changing or new roads are found. These changes should also transfer at one time or an appropriate event to the main system so it will help the next person coming in the same area.

This research identifies the specific data transfer behavior in CUCAT Application Way investigation that requires dynamic and efficient data transfers bounded time for small payloads. Due to the lack of support from existing transport layer protocols to facilitate all requirements of the way finding application, this research will be to continue the change process low latency transport the establishment connection layer, the bit sequence numbers variable and acknowledgment numbers, improved monitoring mechanism, improved congestion control mechanism in order to meet the requirements of the way finding application.

This research may also be important for other fields of application, such as medical systems. They have similar requirements, such as the transfer of the patient's vital signs (heart rate pulse, blood pressure and level of glucose in the blood) in real time, the delivery of large images (CAT scan, X-ray, etc.) and notes the low-priority patients.

## 6. References

- [1]Dhammika H De Silva, Dr.Ian Murray “An investigation in to Dynamic TLP’s for Smartphone Communication”, in International Conference on Advances in ICT for emerging regions ICTer, 2013, p.194 – 197
- [2]Haowei Bai *et al* “Impact of Mobile Wireless Links on TCP Performance”
- [3]Hala Eidaw *et al* “Internet Accessibility In High-Speed Vehicles”
- [4]Ibtissam El Khayat *et al* “Improving TCP in wireless networks with an adaptive machine-learnt classifier of packet loss causes”
- [5]Kunz.T et al “Transport Layer Fairness and Congestion Control in Multihop Wireless Networks”
- [6]Ningning Hu *et al* “Improving TCP Startup Performance using Active Measurements:Algorithm and Evaluation” by
- [7]I. E. Khayat, P. Geurts and G. Leduc “Improving TCP in wireless networks with an adaptive machine-learnt classifier of packet loss causes”
- [8]H. Zhang “Transport Layer Fairness and Congestion Control in Wireless Ad Hoc Access Networks”. July 2006
- [9]V. Jacobson, M. J. Karels “Congestion Avoidance and Control”. November, 1988
- [10] F. Yang, Q. Zhang, W. Zhu and Y. Zhang “An Efficient Transport Scheme for Multimedia over Wireless Internet”
- [11] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz “Improving TCP/IP Performance over Wireless Networks”

[12] M. C. Chan and R. Ramjee “Improving TCP/IP Performance over Third Generation Wireless Networks”

[13] F. Wang, Y. Zhang “Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response”

[14] M. S. Akbar, S. Z. Ahmed, M. A. Qadir, “*Performance Optimization of Transmission Control Protocol in Heterogeneous Wireless Network during Mobility*”, Vol.8 No.8., IJCSNS International Journal of Computer Science and Network Security, 2008.

[15] S. Zaher, “*Performance Enhancement of Transmission Control Protocol over Wireless Ad-hoc Networks*”, Vol.4, International Journal of Emerging Technology and Advanced Engineering, 2014.

[16] Hongqiang Zhai, Xiang Chen, and Yuguang Fang,” *Improving Transport Layer Performance in Multihop Ad Hoc Networks by Exploiting MAC Layer Information*”, Vol.6 No.5, IEEE Transactions on Wireless Communications, 2007.

## 7. Appendix

**Congestion control:** A technique for monitoring network utilization and manipulating transmission or forwarding rates for data frames to keep traffic levels from overwhelming the network medium.

**Mobile nodes:** A node is either a connection point, a redistribution point communication endpoint

**Packet loss:** Packet loss is the failure of one or more transmitted packets to arrive at their destination. This event can cause noticeable effects in all types of digital communications

**QoS:** Quality of Service is the overall performance of a telephone or computer network, particularly the performance seen by the users of the network

**Multi-hop:** Multi-hop or ad hoc, wireless networks use two or more wireless hops to convey information from a source to a destination.

**Wireless ad hoc networks:** The ad hoc network does not rely on a pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is made dynamically on the basis of network connectivity.