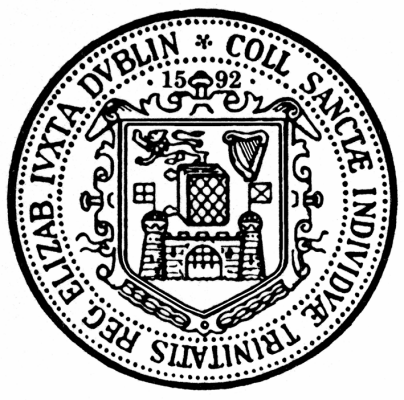
A neurologically-inspired transport layer solution to data transmission in mobile ad hoc networks



Alistair Morris

Trinity College

University of Dublin

A thesis submitted for the degree of

Master of Science (MSc)

August 2010

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Alistair Morris, BSc Lond. (Hons)

September 13, 2010

Abstract

This dissertation proposes a new transport layer protocol called Axon. This is a UDP-based data transport protocol suited for a range of mobile wireless networks inspired by what occurs between neurons in biology. This dissertation shows, through preliminary real world experiments, that this shows promise in terms of performance as well as suitability with existing transport layer protocols over wireless networks.

In particular the content addresses the concepts of intra-protocol fairness, network latency bias, and Transmission Control Protocol (TCP) friendliness which have not been considered at the time of writing. Such concepts are not feasible through the deployment of existing Internet data transport protocols including TCP and UDP.

We hope that Axon will be suited to the emerging data intensive applications we see today. This includes applications such as media streaming where this commonly involves a diminutive number of connections that share a capable amount of wireless network bandwidth.

Axon has been implemented at the application level as a C++ library and the related documented process forms a substantial feature of this dissertation. There are two related but orthogonal parts in the work presented by this dissertation: the Axon protocol and its congestion control algorithm.

By definition Axon is an application level, end-to-end, unicast, reliable, connection-oriented streaming data transport protocol. The protocol is specially designed for efficient high-speed data transfer over networks that may include Wireless links and potentially be in an Ad-Hoc configuration.

The primary aim of this dissertation is to gave an insight into the design and implementation of Axon. Specifically we focus on the tailored congestion control algorithm. This has been inspired by the biological action of neurotransmission in the nervous system: A high speed, multi node network that we all depend on.

To my friends

Acknowledgements

I would like to take this opportunity to thank my supervisor Dr. Stefan Weber for his advice and support without which this would this dissertation would not have been possible.

I would also like to thank my friends and family both for their patience and support during the last year. It really has not been easy to spend so much time away.

Contents

List of Figures vii

List of Tables ix

1 Introduction 1

1.1 The Present Day . . . 1

1.2 The Motive . . . 3

1.3 The Current State of Research . . . 3

1.3.1 Ongoing research . . . 4

1.3.2 The need for further research . . . 5

1.4 Aims of the Project . . . 5

1.4.1 Final aim . . . 6

1.4.2 Preliminary aims . . . 7

1.5 A Reader's Guide . . . 7

2 State of the Art 9

2.1 Transmission Control Protocol - Past it's best . . . 11

2.1.1 The varying versions of TCP . . . 11

2.1.1.1 Tahoe and Reno - A differential overview . . . 12

2.1.1.2 NewReno - An improvement on the familiar . . . 13

2.1.1.3 TCP SACK - Reno and Selective Acknowledgements . . . 13

2.1.1.4 BiC - Suiting the Long and Fat (Networks) . . . 13

2.1.1.5 TCP CUBIC - Bic^3 . . . 14

2.1.2 Shortfalls of TCP . . . 14

2.1.2.1 Poor link utilization in high Bandwidth Product Networks . . . 15

2.1.2.2 Unfairness at long round-trip times . . . 16

2.1.2.3 Confused by lossy links . . . 16

2.1.3 Summary 16
2.2 Previous Work 17
2.2.1 TCP Modifications 17
2.2.1.1 Highspeed TCP 17
2.2.1.2 Scalable TCP 19
2.2.1.3 Wireless TCP 20
2.2.1.4 FAST TCP 20
2.2.1.5 MulTCP 22
2.2.2 Explicit Control Protocol (XCP) 24
2.2.3 Datagram Congestion Control Protocol (DCCP) 25
2.2.4 Ant-Colony Based Routing 26
2.3 Biology - A possible source of inspiration? 27
2.3.1 Biology and Data Communications - The Link 28
2.3.2 Biology and Data Communications - The How 29
2.4 Contributions 31

3 Design **33**

3.1 Introduction 34
3.2 Neurology - An inspiration 35
3.2.1 The Neuron - A technical perspective 35
3.2.2 The Focus - How neurons communicate between each other? 36
3.2.3 An in-depth analysis 38
3.2.4 The borrowed concepts 40
3.3 Implementation Structure 41
3.4 Data Sending and Receiving 44
3.5 Packet Structures 44
3.6 Timers 46
3.6.1 Acknowledge 46
3.6.2 Loss 47
3.6.3 Expiry 47
3.6.4 Send 47
3.7 Connection Management 47
3.7.1 The Handshake 48
3.7.2 The Client/Server Connection Process 48

3.7.3 Rendezvous Connection Setup 49
3.7.4 Acknowledgements 49
3.8 Loss Reports 50
3.9 Connection Maintenance 51
3.10 Connection Shutdown 52
3.10.1 The Sender's Algorithm 52
3.10.2 The Receiver's Algorithm 52
3.10.3 Acknowledgement Event Processing 53
3.10.4 Loss Report Event Processing 55
3.10.5 Expiry Event Processing 55
3.10.6 On receiving an Acknowledgement Packet 56
3.10.7 On receiving a Loss Report packet 56
3.10.8 On Acknowledgement of Acknowledgement (AoA) packet received 57
3.10.9 On Keep-alive packet received 57
3.11 Flow Control 57
3.12 Congestion Control 58
3.12.0.1 MIMD with decreasing increases and increasing decreases (DIID) . . 59
3.12.0.2 MIMD (DIID) in Axon 60
3.12.1 An outline of the benefits DIID brings 62
3.12.2 Rate implied Congestion Control 62
3.12.3 The Determination of Bandwidth 63
3.12.4 Being sensible about Packet Loss 64
3.13 An Overview 64

4 Materials and Methods **67**

4.1 Efficiency and Throughput 67
4.2 Inter-protocol Fairness 67
4.3 TCP Friendliness 68
4.4 Stability (Oscillations) 69
4.5 CPU Usage 69
4.6 Test bed 69

5 Implementation **71**

5.1 Software Architecture 71

5.2 Even Distribution of Processing 72

5.3 Loss Information Management 73

5.4 Memory Copy Avoidance 75

5.5 Assuring accurate Rate Control 76

5.5.1 High Precision Timer 76

5.6 Speculation of Next Packet 77

5.7 Threading 78

5.8 Related Work 79

5.9 Concluding Remarks 80

6 Discussion **81**

6.1 Efficiency, Fairness and Stability 81

6.2 TCP Friendliness 86

6.3 Implementation Efficiency 89

6.4 Performance in Real World Applications 91

6.5 Concluding Remarks 92

7 Conclusions **95**

7.1 Contributions 96

7.1.1 A High Performance Data Transport Protocol and a Implementation of 96

7.2 An Efficient and Fair Congestion Control Algorithm 96

7.3 Limitations and Future Research Direction 97

7.3.1 Bandwidth Estimation in the Context of Transport Protocol 97

7.3.2 Implementation Optimization 98

7.3.3 Tuning and Configuration 98

8 Concluding Remarks **99**

References **101**

List of Figures

3.1 A diagram illustrating Neurotransmission 37
3.2 Graph showing the relation between Action Potential and Stimulus 39
3.3 The relation between Stimulus and Neuron Firing rate 40
3.4 Axon in the layered network protocol architecture 42
3.5 Relationship between Axon sender and receiver 43
3.6 Axon Data Packet Format 44
3.7 Axon Control Packet Format 45
3.8 Packet Type 0 - Connection Handshake 48
3.9 Packet Type 1 - Acknowledgement 50
3.10 Packet Type 2 - Loss Report 51
3.11 Packet Type 3 - Stay Alive 51
3.12 Packet Type 4 - Shut Down 52
3.13 Ion flow in action potential down the length a Neural Axon 61

5.1 Diagram of the Axon Protocol Architecture 71
5.2 The figure shows a loss list with loss 1, 2, 3, and 10. Each node on the list has a start value and an end value. The list uses a static list data structure 74
5.3 The sender's buffer as a list of both protocol allocated buffers 75
5.4 The receiver's buffer as a protocol buffer 75

6.1 The variation of throughput in Axon over time 82
6.2 The variation of throughput in TCP over time 83
6.3 The tuning of the packet sending rate in Axon 83
6.4 The variation of throughput in Axon over time in a high delay environment 84
6.5 The weak correlation between network latency and throughput 85

6.6 The fair and stable sharing of throughput between Axon flows over time 85
6.7 The fairness of concurrent Axon flows 87
6.8 The stability of concurrent Axon flows 87
6.9 The tuning of the Packet Sending Period in concurrent Axon flows 88
6.10 The weak correlation between Network Latency and Throughput in concurrent Axon flows 88
6.11 The throughput of concurrent Axon and TCP flows 89
6.12 The fairness of concurrent Axon and TCP flows 90
6.13 CPU usage of the reference Axon Implementation 90
6.14 Real world performance of Axon in file transfer 92

List of Tables

2.1 The relation between Computer Networks and Neurology 29
2.2 A description of shared concepts between Neurotransmission and Data Communications with descriptions 30

6.1 TThe fairness of multiple Axon flows 86

1

Introduction

1.1 The Present Day

Today, particularly in the western world, we find ourselves living in bandwidth rich times where the speed of networks seems to be ever increasing. More revolutionary is fast growth of wireless networks found in the home and office place. Computer networks are now truly mobile and network bandwidth today, even that over wireless links, has been expanded to speeds measured in the Megabytes per second (Lehr and McKnight, 2003). This, excitingly, has allowed the development of many data intensive applications that were impossible in the past (Samueli, 2000). It could even be argued that applications such as Internet Protocol TeleVision (IPTV) and on-demand media have stimulated the deployment of such high-speed wireless networks.

Living in bandwidth rich times we have seen the once hyped term 'broadband' has found almost compulsory usage in the marketing material issued by ISPs worldwide where it's a label that can be given to any high-speed connection. However, the term also takes on several meanings in relation to the fast evolution of carrier technologies - from DSL (Digital Subscribed Lines) to fibre-optic, from WiMax to the next generation networks such as 4G. This has meant users Japan can boast connection speeds exceeding 60Mb/s and even 1Mb/s is offered in some of the remotest parts of the Philippines.

Interestingly we are living in a world of exponentially increasing data. The concept of storing data in either optical or flash storage ready for delivery to the, as is apparent in DVD and Blue-Ray films is becoming a dated one. In many situations this apparently old fashioned method of shipping data on storage media makes it impossible meet the requirements of modern web-based 'on-demand' applications (Little and Venkatesh, 1994). Examples of applications that are growing

hungrier for bandwidth may include Internet gaming and Video-on-Demand. This is in turn fed by us, the user, who in demanding network service providers to provide a better Quality of Experience (QoE) have produced a demand for the network to transport ever increasing amounts of data (Fulp and Reeves, 2004).

Moving away from the home user we can see that researchers in high-energy physics, astronomy, bioinformatics, and other high performance computing areas have started to use both wired and wireless high-speed networks to transfer vast quantities of data. Media streaming is another application that in particular carries the demand of high data throughput over wireless networks. This has been driven exclusively by the Internet and in particular it's ubiquity. This has undoubtedly made it an attractive platform for the delivery bandwidth intensive applications such as streaming. Although this been nothing short of revolutionary socially, in the western world at least, a particularly elusive goal has proven to be the deployment of a effective streaming solution.

Such a solution arguably can not be implemented with the de facto transport protocol of the Internet - the Transmission Control Protocol (TCP). There are cases where it substantially underutilises network bandwidth over high-speed connections with long delays (Motwani and Gopinath, 2005). In considering the viability of TCP for streaming video the recent proliferation of work on TCP-friendly streaming should be considered. One of the most important issues today in research is whether service model of TCP need to change. Much work needs to be done but it does seem the unfortunate case the high-speed networks that are very available today may not be used to their potential by these applications.

Coupled with that fact that high-speed and long delay network conditions are now common in wireless networks both at the local and wide area. These are becoming ever popular due to their convenience and lack of infrastructure costs from the perspective of the user and the owner of the network infrastructure. A new transport protocol is therefore a timely solution to meet this challenge. As with any new protocol though it should be expected to be easily deployed and easily integrated with the applications, in addition to utilizing the bandwidth efficiently and fairly.

This dissertation hence recognizes the emerging requirements of wireless networks in general and proposes a new high performance neurologically inspired network transport protocol called Axon. This dissertation will describe in detail the design and implementation of Axon and through extensive theoretical and experimental work demonstrate that Axon satisfies the requirements demanded by data intensive applications.

1.2 The Motive

The way that network transport protocols are deployed are integral to a user's Quality of Service (QoS) and Quality of Experience (QoE) on a network. This respectively refers to both resource reservation control mechanisms as well as the perceived service quality. Both concepts prove to be very important in applications ranging from the simplest File Transfer Protocol client to the most complex Peer-to-Peer media streaming service.

The problem exists where the established transport layer communication protocols such as TCP were developed without the requirement to provide connectivity in both typical Wireless and Mobile Ad hoc NETworks (MANETs). There were instead designed and implemented to provide a fundamental transport layer service to hard wired network infrastructures. As such these protocols cannot be applied effectively to MANETs or even the most standard Wireless Network where the characteristics of infrastructure networks are fundamentally different. With Wireless Network we encounter a whole new range of problems that can be attribute to unpredictable nature of radio transmission.

Two characteristics that are arguably crucial are the potentially large number of hops of a communication, the lack knowledge of the topology and membership of the network. The later issues can be notably be applied to the most general of Wireless Networks where knowledge can be used to counter a number of specific problems such as packet loss. These are notably not addressed in the mainstream transport protocols we see today.

This thesis will propose areas where MANETs and Wireless Networks in general exhibit similar characteristics to the communication patterns that occur between neurons. Such comparisons maybe drawn where a many to many relationship is observed in the communication between individual nodes to other nodes. Interestingly the interaction between them is, also like in Wireless Networks, intermittent. Hence this dissertation will pay particular focus on how other such observations maybe used to improve the interaction between nodes at the transport layer.

1.3 The Current State of Research

The shortfalls of existing transport protocols, namely the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP), are the topic of ongoing debate both in the academic and industrial research community. Specifically TCP's deployment over mobile wireless networks comes up against a very critical view across the research community (Nahm et al., 2005). The main issue lies in the fact that TCP cannot distinguish a wireless specific transmission error from a congestion issue. UDP

respectively also comes up against similar criticism where the sense of using an unreliable protocol over an inherit unreliable wireless network is very debatable. Interestingly though it has been observed that when UDP does work in an mobile environment the throughput can be exceptional (Garg and Kappes, 2003) and this forms but one area of ongoing research which will be examined further on.

1.3.1 Ongoing research

Currently the beginnings of promising solutions to the problem of transport data across wireless link maybe found in the concepts of 'split' TCP, TCP aware link protocols and Explicit Notification Schemes. 'Split' TCP can be used to be describe the ideas behind Indirect TCP and Selective Repeat Protocol (SRP).

Indirect TCP itself uses two TCP connections. One from the fixed host to the base station and from the base station to the mobile host (Bakre and Badrinath, 1997). Independent flow control can be found on two connections and the data packets are themselves buffered in the base. SRP however takes a much simpler approach. Like indirect TCP two connections are used from the fixed host to the base station and from the base station to the mobile host (Yavatkar and Bhagawat, 1994). Unlike indirect TCP however it uses standard TCP from the fixed host and SRP over UDP from the base station and mobile host.

Snoop is a well known member of the TCP Aware Link Layer Protocols. Snoop works using split connection and link level retransmission. In this set-up the base station monitors the returning acknowledgements (Balakrishnan et al., 1995). It thus retransmits on duplicate acknowledgements and subsequently drops them.

Explicit notification schemes are themselves less complex but conceptually very different. One major scheme includes Explicit Loss Notification (ELS) which is used with mobile host sources whose first link on the path is wireless (Balakrishnan et al., 1997). Here the base station keeps track of the missing packet for the mobile host.

Although all very different and promising these are all topics of ongoing research and maybe not be suitable for use in the real world. Bold this statement maybe but these examples are themselves tied down to such a specific scenario and therefore are quite limited in terms of potential areas of implementation. They presume the mass take up of any new technology that may develop from the research where there are essentially extensions of existing technologies - most notably IP. Giving version six of the IP (Internet Protocol) as the most infamous example where we are still no where near the point of mass deployment you simply cannot expect such a wide and diverse user audience

such as those connected to the Internet to accept a new approach quickly. This especially the case when the current approach seems to work just fine to the unaware user.

We should however never attempt to remove the scope to suggest alternatives where this forms the focus of this thesis and research in general. Many have already taken on this task as seen in the examples Snoop or SRP that have produced some very promising results in some scenarios. This drive to research and develop further alternatives is continued in the following section.

1.3.2 The need for further research

Many of the problems with current Transport Layer Protocols are due to the fact that mobile wireless networks arrived on scene all too recently. Protocols such as TCP were simply never designed to accommodate wireless transmission and this exposes a number of issues. It is important to also consider that these were designed for weird networks whose performance were measured in *Bytes per second*. Such a increase in performance would have extensively examined during their design.

Current research has suggested modifications to a transport this has often required a modification to the underlying workings of TCP. However such modifications would have to be made to the TCP stack found deep in the foundations of the current range of Operating Systems. From a practical perspective this is simply not an attractive option due to the entailed complexity. From this we see there is a definite need for an alternative to such alternatives.

Although the likes of TCP and UDP do seem to adequate for the current breed of wireless applications the likelihood is that this will not be the case as applications become increasingly more data intensive. Once technologies such as Wireless Local Area Networks (WLANs) and Mobile Broadband start to mature and become more common place we may need to look at the suitability and the improvement of TCP over such networks. In keeping with this idea we will argue that efficiency should form a common design objective in such further revisions and innovations in transport layer protocols. Efficiency is obvious in the major majority of cases but we should highlight the cases where this can decrease as the bandwidth delay product increases.

We define the bandwidth delay product as the product of the link capacity and the round trip time of a packet. This thesis will highlight other considerations, such as fairness and stability, that sometimes make it difficult to realise the goal of optimum efficiency over wireless links.

1.4 Aims of the Project

Here we will attempt to assess the both the final and preliminary aims of the project.

1.4.1 Final aim

Today, the emergence and spread of wide area wireless networks has imposed new challenges on transport protocol design. While the transport layer is an unlikely candidate for application performance woes it can become a problem. When we consider that the transport protocols in broad use today were designed in 1981 (Ruthfield, 1995) issues do become apparent. The demands of Today's application and network topologies differ greatly from the networks of the early 1980s. TCP really shows it's age when you consider that 300 baud was considered state-of-the-art at the time it saw initial deployment.

When we contemplate the networks that TCP was originally design to work on we see that congestion was largely due to a handful of nodes on a shared network of limited scale. This is very different to the complex high-speed networks that are plagued with over subscription aggregation and millions of concurrent users each contending for available bandwidth we see today. Applications at the time TCP was deployed were often text-oriented applications but today even the most ill-equipped corporate user can easily move files that are tens upon hundreds of megabytes in size at a click of a button.

Although the network has inevitable evolved, the fact remains that TCP is very relevant in today's dynamic and ever-changing network environment. TCP has had to undergo minor changes in the past 25 years and those changes are in the form of extensions rather than thorough rewrites. Although there are some more modern transport protocols that have roots in TCP such as WTCP, many are considered developmental projects only and currently have limited deployment in the mainstream. Although some have met with some success we should concede that it is unrealistic to deploy a new protocol in the transport layer in a comparatively short period of time.

Ideally we should be able to gradually deploy new transport protocols on a network. These should be friendly to existing protocols while achieving superior performance. An application level solution is often more desirable in this where it may be ported into the lower layer gradually if the protocol proves to be successful. This is arguably one of original purposes of the standard UDP protocol where it allows new data transport mechanisms to be built on top of it if needed. One example could include the RTP (Real-time Transport Protocol) currently being used increasing to stream multimedia.

Follow this idea we will present both experiences and critical observations in the implementation of a high performance wireless network transport protocols for data intensive applications at the application level. In particular it will focus on solves the issues that have not been addressed previously with traditional protocols such as TCP.

This will required the design and implementation of a network transport protocol and a respective library to overcome the efficiency and fairness problems of existing network transport protocols over wireless networks. In keeping with the topic of this dissertation the main concerns will remain in performance in an wireless environment and how any shortcomings may be addressed. It is hoped that any shortcomings can be solved by observing the interactions between neurons at a cellular level where we can draw parallels with the Wireless Ad-Hoc concept.

1.4.2 Preliminary aims

This thesis will attempt to make the following research contributions:

- To suggest an original MIMD rate control algorithm that uses a bandwidth estimation technique to determine the best increase parameter for efficiency. From our experiments we hope to see an increase on the effective throughput of the protocol. We will also look into the advantages of a MIMD rate control algorithm where it should be fair to existing TCP flows. This is important since many networks are shared and many networked applications employ TCP-based communication.

- To propose a new, neurologically inspired, congestion control algorithm that increases fairness and enabling multiple transport flows to coexist over the same path. This is important for concurrent wireless data intensive applications that exist in certain scenarios.

- To advocate the use of dynamic window control to reduce loss and oscillations which are not apparent in transmissions within the nervous system but a problem in wireless network environments. In particular this we will look into why both are desirable from the perspective a typical data intensive application that may be deployed in a wireless network.

These will taken into account during the course and eventually the conclusion of this work.

1.5 A Reader's Guide

This dissertation is divided into carefully divided chapters to give the reader a better insight into the experience of designing and implementing a new network transport protocol in an easy to grasp manner:

- *Chapter 2* will detail the State-the-of-Art and outline the current research that is ongoing in both academia and in industry.

- *Chapter 3* will go into the design of Axon; a neurologically inspired transport layer protocols to solve the problem of low throughput in high bandwidth delay product wireless ad hoc networking environments.
- *Chapter 4* introduces the "Materials and Methods" used to measure the potential success of the new network transport protocol. These will be used essentially as a means to compare the performance of Axon to TCP in a number of common application scenarios.
- *Chapter 5* details the Implementation of Axon using the C++ programming language and the behaviour of the protocol at the transport layer in the network stack. In particular we will focus on some of the measures undertaken to ensure we have a good quality reference implementation library.
- *Chapter 6* is where we will discuss the preliminary measures of performance gained through putting the implemented Axon library through it's paces. For comparison we will compare these results to those gained from the use of TCP using the same experimental setup.
- *Chapter 7* will penultimately outline any conclusions we may draw from the results presented in *Chapter 6*. From this we be able to determine what works well in the protocol and what should be highlighted as areas for future research.
- *Chapter 8* will finally outline any such future work and suggest any specific areas that should be looked at in any further iterations of the design or implementation of Axon

2

State of the Art

In the layered Internet architecture, the transport layer forms the fourth layer found above the network layer and below the application layer. Currently there are two well-known transport protocols at this layer: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), while there are new protocols emerging, including SCTP (Stewart and Metz, 2001) and DCCP (Kohler et al., 2006).

A transport protocol provides various functionalities to the applications, including but not limited to data delivery, data reliability control, and a streaming or messaging service. General-purpose transport protocols should have four fundamental objectives that are usually transparent to applications: efficiency, fairness, convergence, and the ability to be distributed if needed. It should also be efficient and able utilise the available bandwidth as efficiency as possible.

A protocol should therefore accomplish the following two tasks quickly:

- Determine the maximum available bandwidth where we must respect and aspire to this limit
- Recover to the maximum speed after a drop in the sending rate due to congestion or packet loss. Meanwhile, after it reaches maximum speed, it should remain at the current state until the network situation changes. This means that oscillations should be as small as possible. This will increase the aggregate throughput.

We should also expect network bandwidth to be shared fairly among all concurrent flows where it is too restrictive to limit applications to one effective flow. This demand that will look in how we should go about measuring fairness where there are a number of different approaches. The most common one is the max-min fairness where we try to maximize the minimum throughput (Hahne,

1991). The fairness property among all flows belonging to the same protocol is also very relevant and known as the intra-protocol fairness.

Moving toward the next piece of jargon we come to network latency independence. We use this to describe the special case of fairness over topology with different round trip times, a condition not satisfied by TCP. This prove to be very troublesome with wireless links where network latency is rarely static due to the need of retransmission at the MAC Layer where radio transmission is rarely 100% reliable (Cali, 2000). The fairness problem becomes even more difficult to overcome when we consider the cases when heterogeneous protocols coexist. A new transport protocol is required to consider the situation when it coexists with TCP before it is widely deployed on the Internet.

The fairness between TCP and a new protocol such as Axon is known as TCP friendliness. The ideal situation is where the data sending rate should converge to a unique equilibrium from any starting point with any given specific network situation (Yang et al., 2003). It is acceptable that the throughput oscillates around a fixed point because binary feedbacks are usually used to notify changes in the network situation that can be far from ideal. This is known as the global stability property of network transport protocols.

Finally, we must take into account that large networks such as the Internet are essentially a large loosely coupled system. That makes it impossible to have a central server to dispatch the bandwidth in some form of paradigmatic solution. Transport protocols must consequently control their data sending rate at the end hosts with or without assistance from the routers that the traffic passing through. The end-to-end principle states to this effect that, whenever possible, transport protocols operations should only occur at end hosts. The end-to-end principle greatly increases the system's scalability. This even the case with the existence of gateway operators where it is still necessary to retain congestion control functionalities at end hosts (Floyd and Fall, 1999).

Congestion control therefore forms a critical component in a transport protocol in order to realize these objectives. The transport protocol needs to adjust the data sending rate as required using some form of congestion control algorithm. This usually takes the form of a feedback system where this can be either explicitly generated from intermediate nodes such as routers, estimated by packet losses, increase trends in packet delay, or time out events. We can use explicit feedback from routers to gain more accurate information. We will actually consider one protocol, XCP, that takes this approach but at the expense of higher deployment costs.

When trying to work how we may tune the data sending rate we soon see that this can be done through either the inter-packet time or by examining the number of outstanding packets. The former method is called rate-based congestion control and the latter is called window-based

congestion control. A linear system is often applied in a control scheme to tune these parameters because of its simplicity. The most famous example of such a control algorithm is the Additive Increase Multiplicative Decrese (AIMD) algorithm in TCP (Gao and Rao, 2005).

2.1 Transmission Control Protocol - Past it's best

TCP is by all measures the most widely used transport protocol and holds high status as the de facto standard Internet data transport protocol. TCP provides the reliable data streaming service that many applications demand. First proposed during the 1970's (Mowery and Simcoe, 2002) there has since been many updates that are proposed to improve its performance or fix the problems found in previous versions. The success of TCP may be put down to its stability and the widespread presence of short lived reliable flows on the Internet. However the use of network resources in high performance, sometimes distributed, data intensive applications is quite different from that of the more traditional Internet applications we are familiar with.

Data transfer often lasts a very long time at relatively high speeds as we can in streaming platforms such as YouTube or the BBC iPlayer and the various clones these have inspired. Applications are also becoming more distributed and this requires cooperation among multiple data connections. Fairness between flows with different start times and network delays is desirable where ultimately networks are a shared medium of information flow. Now that we has established the general shortfalls we must consider what modifications have been made to TCP adapting for use in modern networking environments.

So far, four major versions have been widely deployed: Tahoe (Van Jacobson, 1988), Reno (Mo et al., 1999), NewReno (Barman and Matta, 2002), and SACK (Fall and Floyd, 1996). TCP NewReno and TCP SACK are commonplace today as versions of the TCP protocol. More recently from the TCP BIC (Xu et al., 2004) and TCP CUBIC (Ha et al., 2008) have recently entered the scene bringing their own pros and cons.

2.1.1 The varying versions of TCP

As we have already stated TCP has not been static in terms of it's design. It has had to adapt to cope with the way networks have changed since the protocol came into being. This means that they are plenty of different versions available suiting various network environments, both new and old. We examine this in the following.

2.1.1.1 Tahoe and Reno - A differential overview

TCP Tahoe and Reno are in fact quite similar where they use a multi-faceted congestion control strategy in attempt to avoid congestion collapse. Such an event can occur in any packet switched computer network when it reaches the point where little or no useful communication can happen due to congestion. This requires TCP to maintain a congestion window, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is a concept probably best compared to that of TCP's sliding window used for flow control.

Tahoe and Reno form very much the 'usual suspects' in the history of TCP where it uses a mechanism called slow start (Fall and Floyd, 1996) to increase the congestion window after a connection is initialised and after any time-out that may occur within a network connection. This is primarily so we can avoid congestion with in the network. In such situations packets will go a miss on networks and this cases likelihood of duplicate acknowledgements being received to be very high. It is how Tahoe and Reno differ in the way they detect and react to packet loss which set them apart from the alternatives.

In Tahoe a loss maybe discovered when a time-out expires before an acknowledgement is received. It will then reduce congestion window to that of the Maximum Segment Size, a value indicating the largest amount of data that a network device can receive in a single piece before resetting to slow-start state. The maximum segment size is equal to largest amount of data that a computer or communications device can handle in a non-fragmented piece.

Reno takes a different approach where it halves the congestion window, performs a "fast retransmit", and enters a phase called Fast Recovery in the event of three duplicate acknowledgements being received (Fall and Floyd, 1996). Fast Recovery, by definition, is when TCP retransmits the missing packet that was signalled by three duplicate acknowledgements. After this it will wait for an acknowledgement of the entire sending window before returning to congestion avoidance.

Being critical we should consider that they are cases where no acknowledgements will be received and TCP Reno will, as a consequence, experience a time-out. Here it will enter the slow-start state like Tahoe state and throughput will be limited. This is not only issue though. Reno and Tahoe are designed to reduce congestion window to that of the maximum segment size on a time-out event. Such issues are substantial where wireless networks are involved where packet loss and time-out are more frequent (Holland and Vaidya, 2002).

2.1.1.2 NewReno - An improvement on the familiar

In more recent research TCP New Reno improves retransmission during the fast recovery phase of TCP Reno. The most influential change is in concept of the fast recovery(Mathis et al., 1997). Here we see for every duplicate acknowledgement that is returned to TCP New Reno, a new unsent packet from the end of the congestion window is sent. This in order so that we keep the packet transmission window full. For every acknowledgement that makes partial progress in the sequence space, the sender assumes that the acknowledgement points to a new hole, and the next packet beyond the acknowledged sequence number is sent.

2.1.1.3 TCP SACK - Reno and Selective Acknowledgements

Finally TCP SACK forms a conservative extension of Reno TCP modified to use the concept of Selective Acknowledgements originally proposed by the Internet Engineering Task Force (IETF). It is therefore an attempt to solve Reno TCP's performance problems when multiple packets are dropped where ultimately the absence of selective acknowledgements does impose limits to the performance of TCP.

This specific TCP implementation preserves the properties of Tahoe and Reno TCP of being robust in the presence of out-of-order packets and uses retransmit time-outs as the recovery method as a last resort. The main difference when comparing SACK TCP implementation and the Reno TCP implementation is in the behaviour when multiple packets are dropped from one window of data. This achieved using selective acknowledgements, the data receiver can inform the sender about all segments that have arrived successfully, so the sender need retransmit only the segments that have actually been lost (Allman, 1998).

2.1.1.4 BiC - Suiting the Long and Fat (Networks)

BiC is an implementation of TCP with an optimized congestion control algorithm for high speed networks with high latency (Xu et al., 2004). It uses a similar strategy to the alternatives but it proposes a more complicated method to increase the sending rate. Achieving good bandwidth utilization, BiC TCP also has a better fairness characteristic than Scalable and HighSpeed TCP where it is not as aggressive. In addition, BiC TCP has an upper limit on the increase parameter which makes it less scalable; a potential problem in the ad hoc environment.

2.1.1.5 TCP CUBIC - Bic[3]

CUBIC is a less aggressive and more systematic derivative of BIC, in which the window is a cubic function of time since the last congestion event, with the inflection point set to the window prior to the event. Being a cubic function, there are two components to window growth. The first is a concave portion where the window quickly ramps up to the window size before the last congestion event. Next is the convex growth were CUBIC probes for more bandwidth, slowly at first then very rapidly (Ha et al., 2008). CUBIC spends a lot of time at a plateau between the concave and convex growth region. This allows help the network stabilize before the TCP variant begins looking for more bandwidth.

Another major difference between CUBIC and the more traditional versions of TCP is that it does not rely on the receipt of acknowledgements to increase the window size. This works as CUBIC's window size is dependent only on the last congestion event. With standard TCP, flows with very short round trip times will receive acknowledgements faster and therefore have their congestion windows grow faster than other flows with longer round trip times (Ha et al., 2008). CUBIC also interesting allows for a greater fairness between flows since the window growth is independent of round trip time.

2.1.2 Shortfalls of TCP

Although various versions of TCP have their differences they all share similarities and arguably a flaw. This takes shape of TCP's AIMD-based control algorithm (Jacobson, 1995) that increases the sending rate using the congestion window size approximately one segment per s round trip time. It then halves the window size once there is a loss event. Here if we consider 200ms round trip time, a realistic figure that could be regraded as approximately the network distance between nodes in a wide area wireless network, after a loss event, well over 200 round trip times are required for TCP to increase its window for full utilization of 54Mb/s with 1500-byte packets. This is equal to the payload in a Ethernet frame and entails 9000 packets each round trip time or approximately 4 seconds.

Put simple, to maintain its maximum utilization of 54 Mb/s (75% of the peak throughput), the loss rate cannot be more than 1 loss event per 99,900,000 packets. This is a bit too close for comfort to the theoretical limit of even the bit error rate of a wired connection, let alone a wireless link. This is of special concern in wireless networks where the medium of transmission is, at best, unpredictable.

$$T = \frac{S}{R\sqrt{\frac{2p}{3}} + t_{to}(\sqrt[3]{\frac{3p}{8}})p(1+32p^2)} \tag{2.1}$$

Equation 2.1 (Padhye et al., 2000) indicates that TCP generally becomes ineffective as the network bandwidth and delay increases (Katabi et al., 2002) where T is the throughput, S is the TCP segment size, p is the loss rate and t_{to} is the TCP time out value. This is bad news in the case of Wireless Networks and in particular those that fall into the category of Mobile Ad-hoc NETworks (MANETs) where the loss rate is like to comparatively high. Furthermore, the existence of network latency in the TCP throughput model means that flows with different round trip times may have different throughputs. This is also known as round trip time bias.

Many researchers have made very compelling arguments for such a apparent flaw where some draw the conclusion that this is an acceptable trade-off so we ensure network stability. However some may instead form the opinion that this is only relevant to the yesterday's networks where reliability was of the prime concern. Today this is not as relevant where both the hardware and software has matured to the point where efficiency is primary concern for the user. We are observing networks implement faster links and so we see an increased availability in high performance networking. Unfortunately despite such improvements wireless network users still experience performance penalties because their high bandwidth yet high delay networks are not suited to TCP flows.

When taking into account the current trend of research we se that TCP performs poorly in three notable scenarios:

2.1.2.1 Poor link utilization in high Bandwidth Product Networks

It is difficult to obtain high TCP throughput and, hence, good link utilization for high Bandwidth Delay Product flows traversing end-to-end paths with high bandwidth, high delay, or both. It has been observed that a single TCP flow, in the extreme case, may saturate a 10Gbps link (Bu et al., 2006), that incurs large, oscillatory queues. However, sustaining these high flow rates requires a unrealistically low packet loss where such an event causes TCP to back off all too easily. This is not only limited to infrastructures which such headlining bandwidth figures.

This is often too apparent in the current deployment of wireless networks at the hands of your average user where we rarely see bandwidth used optimally for the same reason. Although (Bu et al., 2006) does present a very extreme scenario we can see how TCP, in it's current form, is likely to become more of a liability as the capability of wireless networks increases over time.

2.1.2.2 Unfairness at long round-trip times

TCP flows with long round-trip times have difficulty obtaining their fair share of bandwidth on a bottlenecked link (Vojnovic et al., 2000). This is strange when you consider the increasing use of mobile, satellite and cellular networks in the present day, all of which carry a substantial amount of TCP traffic. Typically once TCP enters congestion-avoidance mode, long round-trip-time flows slowly open their congestion windows. As a result, short round-trip time flows obtain higher throughput and this causes the troublesome network latency bias.

We can see where this quickly becomes an issue in the ad-hoc environment. Round trip times are dependent on link quality and this is turn related to the distance between nodes. We can not assume either to be constant figure and we should consider that link quality is also at the mercy of radio interference which is periodically random in its occurrence.

This problem is not limited to just the ad-hoc environment. Even in managed wireless networks that include a base station there is a varying round trip time due to the intermittent need of re-transmission at the MAC Layer. There is also the fact that the distance from the base station will vary. Notably none of the TCP variants discussed address the round trip time bias problem and we should make every attempt to look into potential solutions.

2.1.2.3 Confused by lossy links

TCP uses packet loss as a binary indicator of congestion where there is no feedback on the intensity of congestion (Balakrishnan et al., 1997). This works well where packet loss is a reasonable congestion indicator on links with negligible non congestion-related packet loss. However, many wireless links are subject to, for example, uncorrectable bit errors; as a result, TCP may treat them as congested networks and underutilize them.

We need therefore a different approach where by packet loss should not always infer the situation of a congested network link.

2.1.3 Summary

Given the apparent shortfalls in TCP there is a need to look at ways to improve performance in these environments. We place particular emphasis on the first two problems of *Poor link utilization in high Bandwidth Product Networks* and *Unfairness at long round-trip times* where these are the more straight forward to overcome from a practical perspective. There demand a different approach and are also the most common general flaws in wireless networks that have such popularity today.

We should not however ignore the problem where TCP is *Confused by lossy links*. This is an equally relevant issue that should be ideally overcome although it represents a greater challenge. We should always expect lower throughput on a lossy link and we can not always expect control traffic to keep through. This thus limits our options and we may need to infer events from observations such as packet loss or link capacity.

2.2 Previous Work

Before we can set out to fix these problems that have been highlighted it is necessary to look at what research has been conducted. From this we can determine the methods that have been effective in overcoming these issues as well as learn from those have not meet with such success.

2.2.1 TCP Modifications

For a long time researchers have continually worked to improve TCP. Probably the most straightforward approach suggested is to use a larger increase parameter and smaller decrease factor in the AIMD algorithm than those used in the standard TCP algorithm. Scalable TCP (Kelly, 2003) and High Speed TCP (TOKUDA et al., 2003) are the two just typical examples of this class. Others will be examined in the following:

2.2.1.1 Highspeed TCP

Highspeed TCP Changes the way TCP behaves at high speed and in particular focuses on how Congestion Avoidance is conducted. Put plainly it modifies the response function of TCP so that ridiculously low packet drop rates are not required to sustain high throughput. It is claimed that this modification allows Highspeed TCP to roughly emulates n parallel TCP connections (TOKUDA et al., 2003). It should be noted that this is not by design and is not controlled by the protocol. It merely indicates that the throughput of Highspeed TCP is n times that of standard TCP under similar conditions and this is a good thing in terms of network performance.

The best way to see the differences between Highspeed TCP and Standard TCP is in there response functions. If w is the congestion window and p the packet drop rate these would $w = 0.12/p^{0.835}$ and $w = 1.2/sqrt(p)$ for standard TCP. This is respective of whether it is necessary to increase or decrease throughput due to congestion. The way in which Highspeed TCP handles the size of congestion windows is shown in Algorithm 1 using the same symbols.

Algorithm 1 - Highspeed TCP Window Handling

```
while Highspeed TCP connection is open do
  if network is congested then
    w ← w + a(w)/w
  else
    w ← w(1 − b(w))
  end if
end while
```

This algorithmic behaviour is only applicable in the instances where the size of the congestion window is large and where $a(w)$ and $b(w)$ are log functions depending on several measures (TOKUDA et al., 2003). This is not shown as it is beyond the scope of this State of the Art. However from the material available we can see a likely case where if the size of the congestion window is less than 38, $a(w) = 1$ and $b(w) = 0.5$ the behaviour of this protocol is identical to TCP. Unfortunately this makes it less than suitable for the wireless network environment where this does not address the issues that packet loss is all too common.

However we do see that efficiency is ensured where $a(w)$ and $b(w)$ are precomputed and stored in lookup tables. Such an approach to improve the overall performance of the transport layer is though debatable with computing power fast becoming a commodity.

Overall this does seem indeed like the fundamentally correct thing to do as there is a defined need for backwards compatibility and incremental deployment. However another researcher may form the view that this is more a minor modification rather than a novel approach.

Here they have taken a quiet a narrow minded approach to solving the shortcomings of TCP. If we consider wireless networks where lossy links would be frequent due to radio related phenomenon such as interference this derivative of TCP is not likely to perform any better in the real world. It is still too eager to shorten the window size at the first sign of packet loss and this causes it not address the difference between packet loss caused by congestion and packet loss caused by transmission error. Many researchers would probably argue though that this is designed for High Speed Networks and not Wireless Networks. They would probably be right but with innovations such as 4G and 802.11n the dividing line between such networks appears to be getting thinner as time draws on.

2.2.1.2 Scalable TCP

Scalable TCP increases its sending rate proportional to the current value, whereas it only decreases the sending rate by 1/8 when there is packet loss. HighSpeed TCP uses logarithmic increase and decreases functions based on the current sending rates (Kelly, 2003). Both of the two TCP variants have better bandwidth utilization, but suffer from serious fairness problems. The MIMD (multiplicative increase multiplicative decrease) algorithm used in Scalable TCP may not converge to the fairness equilibrium, whereas HighSpeed TCP could converge very slowly.

Scalable TCP works through the implements of very simple modifications to algorithmic foundations of TCP. It focuses on the observation that traditional TCP connections can not effectively use large windows. This is bad where often we have a limited amount of sockets able to receive and buffer memory available. This means that they tend not to have a windows greater than a certain size (Kelly, 2003). This maybe referred as the legacy window size and its use is shown in Algorithm 2 where w is equal to the congestion window, lws is equal to the legacy window size.

Algorithm 2 - Scalable TCP Window Handling

```
while Scalable TCP connection is open do
  if network is congested then
    if w ≤ lws then
      Use the traditional window update algorithm.
    end if
    if w ≥ lws then
      if An Acknowledgement Packet is received then
        w ← w + 0.01
      end if
      if A Packet Loss is discovered then
        w ← w − 0.125 × w
      end if
    end if
  end if
end while
```

This response function is effectively MIMD. In a fixed time, in terms of number of round trip times for doubling the sending rate, the calculation for each acknowledgement received is: $w = w + a$ and for each loss: $w = w - b \times w$. Quite intuitively this allows scaling property applies for any choice of values for a and b so giving the TCP variant such a curious name.

This is important where its a wide variety of performance attributes a connection may have.

This includes on impact on legacy traffic, bandwidth allocation properties, flow rate variance, convergence properties as well as how it would be possible to control theoretic stability.

Critically it would seem that Scalable TCP is actually a special case of the Highspeed TCP response function. This makes is much easier to implement, mathematically modelled and less ad-hoc in nature. Benefits may also be found where the protocol ensures that sending rate is fixed in terms of number of round trip times and this independent of the capacity of the link. This, in turn, makes the protocol very scalable and suited to today's data intensive applications.

It is however not the first network technology in existence not to have issues. The choice of a MIMD congestion algorithm does have it inherently more aggressive than protocols which implement a form of AIMD. We see another issue where there does not seem to be enough experimental results available to convinces us that Scalable TCP will not starve standard TCP. Research is though ongoing and it would seem again the problems of TCP over wireless networks is a moot point. It would therefore be conceivable to argue that this protocol would again suffer when deployed on a wireless network. Again this we suspect that this is due to packet losses being mistakes for congestion and as a need to slow down transmission.

2.2.1.3 Wireless TCP

Wireless TCP (WTCP) has a similar approach to that proposed in Indirect-TCP (I-TCP). We see this as it splits the transport connection at the interface between weird and wireless networks, and maintains two TCP connections, one over the wired network, and another over the wireless link (Sinha et al., 2002). This way the poor quality of the wireless link is hidden from the fixed network. WTCP follows a similar line of thinking but maintains the end-to-end TCP semantics and requires no modification to the TCP code running in the fixed host or the mobile host in the wireless next. WTCP effectively shields wireless link errors and attempts to hide the time spent by the base station. This allows locally recover so that the TCP's round trip time estimation at the source is not affected (Sinha et al., 2002). This is critical since otherwise the ability of the source to effectively detect congestion in the fixed wired network will be hindered. It important to consider that this have been done at the expense of the end-to-end principle.

2.2.1.4 FAST TCP

Recently a new method that follows the strategy outlined in TCP Vegas called FAST TCP was proposed. FAST uses an equation-based approach in order to react to the network situation faster.

FAST TCP works where instead of resulting to AIMD congestion control that backs off in response to a single congestion indication we use equation-based congestion control. Such a control equation that explicitly gives the maximum acceptable sending rate as a function of the recent loss event rate. The sender adapts its sending rate, guided by this control equation, in response to feedback from the receiver (Wei et al., 2006). We should also taken into account the queuing delay where the time a job waits in a queue until it can be executed is a necessity in events such as burst packet transmissions. It is in these scenarios where buffering becomes necessary.

Conceptually this is a take on TCP that differs in its use of multiplicative decrease with packet loss. This means that the protocol undergoes exponential reduction of the congestion window when a congestion even takes place. Here packet loss is notably used as a gauge to how congested a network may or may not be. Arguably the interesting fact of FAST TCP is that it is based on flow-level dynamics rather than packet level dynamics (Wei et al., 2006). FAST TCP therefore ensures that most of existing research on data networks wrongly assume a fixed population of TCP connections or flows.

The main algorithm in FAST TCP is in fact quite simple and can be described as a simple number of steps:

1. Estimate target rate
2. Estimate how far away the current rate is from target rate
3. If very far, increase the sending rate aggressively
4. If close by, increase the sending rate very smoothly

The biggest advantage is that avoids oscillations in terms of the amount of network traffic and this should keep router queues stable. FAST TCP achieves this through a intrinsic window control algorithm which is detailed in Algorithm 3.

Algorithm 3 - FAST TCP Window Handling

$w = w \times (brtt/rtt) + k$

In Algorithm 3 w is the congestion window, rtt is the exponential weighted average of round trip time, $brtt$ is the minimum instantaneous round trip time and k is the constant that determines fairness (or inversely the aggressiveness) and convergence rate.

Currently k is computed using the link bandwidth and the link round trip time which is then stored in a look up table. Work continues in trying to compute it automatically. Theoretically, it is

shown to be fair but there is a distinct lack experimental results illustrate this. We should also the fact that the effect delay on congestion indication is neither fully understood or widely deployed in the networks at the time of writing.

Other issues exist where although there has been much theoretical work on Vegas and FAST, many of their performance characteristics on real networks are yet to be investigated. In particular, the delay information needed by these algorithms can be heavily affected by reverse traffic. As a consequence, the performance of the two protocols is very vulnerable to such events.

2.2.1.5 MulTCP

MulTCP is quite different where it was originally proposed for differentiated services with a form of pricing scheme (Nabeshima, 2005). This takes the form of a computer networking architecture that specifies a simple, scalable and coarse-grained mechanism for providing Quality of Service (QoS) guarantees on modern IP networks with a price tag.

Ignoring the business opportunity and concentrating rather on technical detail we see that a typical MulTCP flow behaves as if it was a collection of several virtual flows. This has been achieved through modifications to the slow start, linear increase and multiplicative decrease algorithms of TCP. An interesting aspect to note is that the developer has not been able to modify the manner in which timeouts are handled. This further outlines the difficulty in modifying an existing protocol.

The modified Slow Start algorithm is outlined in Algorithm 4 where w represents the congestion window.

Algorithm 4 - MulTCP Window Handling (in Slow Start State)

```
if i < k then
    w ← w × 3
else
    w ← w × 2
end if
```

All that is happening here is the the congestion window is tripled for each acknowledgement received for the first k round trip times and double after that. The congestion window for MulTCP at the end of k round trip times should therefore be equal to $3k$ and the sum of Congestion window of n separate flows will be $n2k$. This all relies on the choosing $k \in 3k = n2k$.

The Linear Increase algorithm of MulTCP is shown in Algorithm 5, where w is the congestion window:

Algorithm 5 - MulTCP Window Handling (Linear Increase Component)

if An acknowledgement packet is received **then**
$w \leftarrow w + n/w$
end if

If we look at Algorithm 5 we can see that MulTCP simply increases the congestion window by n/w for each incoming acknowledgement. The Multiplicative Decrease algorithm is shown in Algorithm 6 in a similar fashion.

Algorithm 6 - MulTCP Window Handling (when Packets get lost)

if A packet lost is discovered **then**
$w \leftarrow (1/n) \times 0.5 \times w$
end if

This shows that for each loss the the congestion window should be reduced for only one virtual flow (Nabeshima, 2005).

Although this protocol clearly has its advantages where it does improve throughput and adds an intriguing dimension to plain TCP we must consider the shortfall. Currently the variable n has to be set by the user and may not be dynamically influenced by end-to-end congestion conditions. Another argument against it's widespread deployment is that the protocol relies on policing and pricing to ensure that it does not result in congestion collapse. So it does particular shortfalls of TCP but only through a business orientated theorem. This is not always deployable in all networking environments.

MulTCP also suffers where it is more bursty compared to traditional TCP flavours and this causes it to suffers performance loss during time-outs compared to n individual flows (Nabeshima, 2005). The effective gain does not always therefore correspond to the assigned weight n for flavours other than SACK. Some researchers also argue that MulTCP is also more aggressive than should be when compared to more traditional approaches such as TCP SACK in cases $n < 5$. This is a valid argument where is this often one of the approaches used to implement and claim a "faster TCP". We should also consider the real world problem that exists where increasingly aggressive transport protocols may lead back to the point where there is effectively no congestion avoidance. This would unfortunately cause a network to be chronically congested and we, as researchers, would be back to square one.

2.2.2 Explicit Control Protocol (XCP)

Explicit Control Protocol (XCP) adds explicit feedback from routers (Zhang and Henderson, 2005), is a more radical change to the current Internet transport protocol. While those TCP variants mentioned previously have tried many methods to estimate the network situation, XCP is a more radical approach where makes use of explicit information fed back from routers. XCP is novel where as a data packet passes each router, the router calculates an increase parameter or a decrease factor and updates the related information in the data packet header (Zhang and Henderson, 2005). After the data packet reaches its destination, the receiver sends the information back through acknowledgements.

XCP therefore forms an effective congestion-control system suited toward high bandwidth product networks and possibly even those of the wireless variety. This gives XCP the potential to deliver the highest possible application performance over a range of networking infrastructures, including the extremely high-speed and high-delay links not well served by TCP. This brings with it some benefits where it always achieves maximum link utilisation and reduces the wasted bandwidth due to packet loss.

XCP separates the efficiency and fairness policies of congestion control, enabling routers to quickly make use of available bandwidth while conservatively managing the allocation of bandwidth to flows. Built on the principle of carrying per-flow congestion state in packets this allows packets to each carry a small congestion header. This allows a sender to request a desired throughput. Routers then compute a fair per-flow bandwidth allocation without maintaining any per-flow state.

XCP is provably stable and has been shown, through simulation, to scale with numbers of flows, rates of flows, and variance in flow rate and round trip times (Zhang and Henderson, 2005). Simulations show that under these conditions XCP almost never drops packets. This form of congestion control is an opportunity to develop a more systematic framework for managing network resources. This includes link capacity, router memory, and processing power where effectively decouples congestion control from the bandwidth-allocation policy. The router itself comprises of two separate controllers: the Efficiency Controller for preventing congestion and maintaining high utilization and the Fairness Controller for splitting the aggregate bandwidth fairly among connections sharing the link.

If we consider that routers in the core of the Internet typically carry many thousands of flows concurrently we see that algorithms requiring per-flow state have poor scaling properties; XCP provides a form of congestion feedback that is fine grain and flow specific without retaining state in

the routers. The necessary flow specific state is carried in the packets and enables routers execute a few multiplications and additions per packet. The result is an algorithm that is fast, scalable, and robust.

However as XCP requires all routers along the path to participate deployment quickly becomes a concern. XCP may though be deployed on a cloud-by-cloud basis providing some increased link utilization within each cloud. However this does not remove the other issue that should be taken into account is that during an XCP deployment, TCP traffic. It is a very rare case indeed where a network will not carry TCP traffic and this raises the issue of XCP and TCP compatibility, or even XCP and TCP friendliness.

XCP is clearly very in different in it's approach when compared to the more conservative proposed modifications to TCP. It introduces explicit, non binary feedback from the network to the endpoints and so achieves several important functional and performance advantages. First, it enables large bandwidth delay product flows to ramp up to peak rates quicker than current versions of TCP, in both start-up and steady-state operation.

The protocol is also able to obtain the maximum performance supported by the infrastructure under the greatest range of challenging conditions. Rather than being only a modification or tuning of TCP it also introduces a novel framework for resource management. The basic building blocks of XCP may be used by a range of protocols with different semantics, meeting the high-performance communication needs of most networked data intensive applications.

We simply cannot be ignore the fact that overall XCP demonstrates very good performance characteristics. However, it suffers more serious deployment problems than the examined TCP variants where it requires changes in the network infrastructure as well as to the operating systems of end hosts. In addition, recent work showed that gradual deployment where one may update the Internet routers gradually causes a significant performance drop (Zhang and Henderson, 2005). This raises the question if they is any method of estimating the capabilities of a network where making major changes to infrastructure.

2.2.3 Datagram Congestion Control Protocol (DCCP)

Another more novel approach is Datagram Congestion Control Protocol (DCCP), a message-oriented Transport Layer protocol. DCCP implements reliable connection setup, teardown, Explicit Congestion Notification (ECN), congestion control, and feature negotiation Kohler et al. (2006). From these features ECN appears to be the most relevant to wireless networks.

ECN is an attractive concept where this allows the end-to-end notification of network congestion without the need to drop packets. This very unlike the more traditional approach where by IP networks will signal congestion by dropping packets. If and when ECN is successfully negotiated, an compatibility router may start to set bits in IP header instead of dropping a packet in order to signal impending congestion. This would first appear useful where this enables us to know the difference between packet loss due to congestion and packet loss due to transmission error. On closer inspection ECN brings its own issues where it is only effective when supported by the underlying network and there are even cases of outdated equipment of dropping packets with ECN bits set.

It allows for flow-based semantics as with general case of TCP, but does not provide reliable in-order delivery. DCCP is useful for applications with timing constraints on the delivery of data that may become useless to the receiver if reliable in-order delivery combined with network congestion avoidance is used. Such applications include streaming media, Multiplayer online games and Internet telephony. While being useful for these applications, DCCP can also be positioned as a general congestion control mechanism for UDP-based applications, by adding, as needed, a mechanism for reliable and/or in-order delivery on the top of UDP. In this context, DCCP allows the use of different, but generally TCP-friendly congestion control mechanisms. DCCP connection contains control traffic as well as data traffic. Acknowledgements inform a sender whether packets have arrived successfully and, importantly, whether they were marked by a ECN. Acknowledgements are transmitted as reliably as the congestion control mechanism in use requires.

DCCP obviously offers a number of very attractive features but no consideration, again, seems to have been made for the needs of wireless networks. This especially so when the consider when consider the case that the majority of wireless networks do not support ECN.

2.2.4 Ant-Colony Based Routing

Many researcher argue that network goodput, particularly in high bandwidth product networks such as wireless, is limited due to ineffective routing strategies. Ant-Colony Based Routing, although arguably a little irrelevant, does bear some striking resemblance to XCP and has been included for comparison. This is where uses a form a explicit feedback from the network infrastructure to improve on performance.

As a concept it is based on the basic idea of the ant colony optimization meta heuristic (Bouazizi, 2002). This intriguingly is taken from the food searching behaviour of real ants where on they way to search for food, they start from their nest and walk toward the food. When an ant reaches an

intersection, it has to decide which branch to take next. While walking, ants deposit pheromones, which marks the route taken. The concentration of pheromone on a certain path is an indication of its usage (Bouazizi, 2002). Although far removed from the realms of computer networks the link between this and the need for a optimal routing strategy has been made.

This is best seen where a scenario with two routes from the nest to the food place. At the intersection, the first ants randomly select the next branch. If one route is shorter than other, the ants which take this path will reach the food place first. On their way back to the nest, the ants again have to select a path. After a short time the pheromone concentration on the shorter path will be higher than on the longer path, because the ants using the shorter path will increase the pheromone concentration faster. The shortest path will thus be identified and eventually all ants will only use this one. This behaviour of the ants can therefore be used to find the shortest path in network and optimize network traffic.

Although very interesting and to a point effective at routing in mobile networked environments this approach assumes that the de facto transport layer protocol TCP would operate as normal. This is by no means a flaw in the argument for the use of such routing techniques where the ultimate goal is to route traffic around high BDP networks. The other equally valid argument remains however that wireless, so long as radio transmission remains ever changing and therefore unreliable, routing can only do so much. Even outside the scope of Computer Science even the most cleaver of Ants are shown in nature to starve in changing conditions.

2.3 Biology - A possible source of inspiration?

Although Ant-Colony Based Routing is one concept where Biology has inspired it does not seem to be the answer at least where lossy wireless networks are concerned. From this it would seem apparent that there is a lack of crossover between the areas of Computer Science and Biology in the arena of Data Communications and Networking. Granted there is the very successful field of Bioinformatics where the application of Information Technology and Computer Science has been applied the field of molecular biology.

However its primary use since at least the late 1980s has been limited to genetics, particularly applications that involve large-scale DNA sequencing. This is very apparent when it is uniquely focused on the creation and advancement of databases, algorithms, computational and statistical techniques, and theory to solve formal and practical problems arising from the management and analysis of biological data. Despite the work been done is nothing short of revolutionary it's does

seem to be very one traffic between the research field combo Computer Science and Information Technology with that of Biology. From this observation one should be able to ask themselves if this field of Computer Science (with Information Technology) could be applied to Biology with so much success then why could this not be so effective in reverse - Biology applied to Computer Science.

Although you may, quite rightly, argue that comparing Biology with Computer Science is a definitive case of "chulk and cheese". However there a number of specific research quarters where you may, at the very least, find the comparison interesting. One such topic is that *Neuroscience* and in particular, when regrading this the topic dissertation, *Neurotransmission*. When put along side the concepts of "networks" and especially "network transmission" we can see that both Biology and Computer Science are trying to come to the same answer - How to get messages from one node to another, possible passing through several nodes, undamaged and more importantly efficiently. This is discussed further in the following subsection.

2.3.1 Biology and Data Communications - The Link

So it is in the arena of Neuroscience and the foundations of Neurotransmitter where there is a direct comparison with Data Communications within Computer Networks. As already highlighted both are means to an end where they focus through the various issues in trying to get electronic messages across several modes of communication. It can even be regarded that the only real difference is what can be defined nodes in the networks and the way in which they are connected. Conceptually therefore the two fields are similar in both there set-up and execution. Some researchers even go as far as suggesting that the most complicated and extravagant networks are found on the Internet but in the nervous systems of you and me.

Looking deeper toward a technical level and at a biological cellular level we can see that the nervous system, like computer networks, have also come to put where two manners of transmission between nodes have had to be accommodated. In traditional computer networks we know that to be wired and wireless between various network interfaces where convenience has driven the need for users to become mobile. In neuroscience these are the chemical and electrical synapses between the specialised group of cells know.

From this we can determine that there are two types of synapse being chemical and electrical. Synapses are essential to neuronal function: neurons are cells that are specialised to pass signals to individual target cells, and synapses are the means by which they do so. At a synapse, the plasma membrane of the signal-passing neuron comes into close apposition with the membrane of the target post-synaptic cell. These are know as the pre-synaptic and the post-synaptic neuron

Wireless Networking Term	Neurological Term	Unified Conceptual Term
Networked Computer	Neuron	*Node*
Packets	Neurotransmitters	*Transmission Quanta*
Over-the-air Modulation	Synaptic Cleft	*Transmission Medium*
Protocol Stack (Sender)	Axon Terminal	*Sending Device*
Protocol Stack (Receiver)	Dendritic Spine	*Receiving Device*
Acknowledgements	Neurotransmitter Reuptake	*Finalization Action*
Congestion Window (Sender)	Synaptic vesicle	*Indexed Buffer*
Congestion Window (Receiver)	Post-Synaptic receptors	*Buffer Advertisement*

Table 2.1: The relation between Computer Networks and Neurology

respectively. Both the pre-synaptic and post-synaptic sites contain extensive arrays of molecular machinery that link the two membranes together and carry out the signalling process. In many synapses the pre-synaptic part is located on an axon and this gives the name of the protocol described in this dissertation.

It can be expected and it is correct to assume that chemical and electrical synapses have very different mode of operating in sending messages between neurons. In a chemical synapse, the pre-synaptic neuron releases a chemical called a neurotransmitter that binds to receptors located in the post-synaptic cell (Koch and Poggio, 1987). Binding of the neurotransmitter to a receptor can affect the post-synaptic cell in a wide variety of ways. On the other hand in a electrical synapse, the pre-synaptic and post-synaptic cell membranes are connected by channels that are capable of passing electrical current, causing voltage changes in the pre-synaptic cell to induce voltage changes in the post-synaptic cell (Koch and Poggio, 1987). This can be liken to a wired and wireless form network interface where quite conveniently chemical synapses are slower and more reliable than chemical synapses.

All related concepts with in the confines of Wireless Networks are described in tables 2.1 and 2.2 will be used as a forward for the design.

2.3.2 Biology and Data Communications - The How

A summary of the sequence of events that take place in synaptic transmission from a pre-synaptic neuron to a post-synaptic cell through a chemical synapse is described in the following (Osborne, 1996):

1. The process begins with a wave of electrochemical excitation called an action potential travelling along the membrane of the pre-synaptic cell, until it reaches the synapse.

Unified Conceptual Term	Unified Conceptual Definition
Node	Something that processes and transmits information by electronic signalling via connections with others *nodes*
Transmission Quanta	These relay and modulate messages between *nodes*
Transmission Medium	Some through which *nodes* send messages to each other. Often shared any means of transmission (e.g. packets) must then be cleared out efficiently so that it can be ready to used again as soon as possible
Sending Device	Interface where the sending of messages is processed
Receiving Device	Interface where the receiving of messages is processed
Finalization Action	Performed after a *node* has performed the function of transmitting a message. Maybe used to regulate the level of *Transmission Quanta* in the *Transmission Medium*
Indexed Buffer	Stores various neurotransmitters that are about to be sent
Buffer Advertisement	Involved in a wide range of differing reactions from the node receiving the message. This may trigger anything from activation to inhibition of a connection.

Table 2.2: A description of shared concepts between Neurotransmission and Data Communications with descriptions

2. The electrical depolarization of the membrane at the synapse causes channels to open that are permeable to calcium ions.

3. Calcium ions flow through the pre-synaptic membrane, rapidly increasing the calcium concentration in the interior.

4. The high calcium concentration activates a set of calcium-sensitive proteins attached to vesicles that contain a neurotransmitter chemical.

5. These proteins change shape, causing the membranes of some “docked” vesicles to fuse with the membrane of the pre-synaptic cell, thereby opening the vesicles and dumping their neurotransmitter contents into the synaptic cleft, the narrow space between the membranes of the pre- and post-synaptic cells.

6. The neurotransmitter diffuses within the cleft. Some of it escapes, but some of it binds to chemical receptor molecules located on the membrane of the post-synaptic cell.

7. The binding of neurotransmitter causes the receptor molecule to be activated in some way. Several types of activation are possible, as described in more detail below. In any case, this is the key step by which the synaptic process affects the behaviour of the post-synaptic cell.

8. Due to thermal shaking, neurotransmitter molecules eventually break loose from the receptors and drift away.

The entire process may run only a few tenths of a millisecond in the fastest synapses and is a very complicated one. It is therefore very necessary that we focus on a number of specific areas where neurotransmission has attempted to solve some of problems shared with mobile wireless networks. These fall into the much more specific sub-areas of signalling and modulation. That is how messages are sent and how it maybe influenced. Signalling can be broken down further into the concepts of Neurotransmitter release, receptor binding and termination. Modulation, in confines of this dissertation, is best used to refer to desensitization, homosynaptic plasticity and heterosynaptic plasticity.

All these apprehensions will be used as elevation for the design found in the next section.

2.4 Contributions

It seems to be apparent that numerous research problems in data transport protocols and in particular those that are involved in Wireless Networks in general. This is an important aspect to consider where it seems to be case that many of problems experience in Wireless Ad-Hoc environment are shared with a managed Wireless environment. To solve the problems with Ad-Hoc we should attempt to solve the fundamental shared issues where more will benefit. This dissertation will therefore strive to make the following specific contributions:

- A new protocol that provides a practical solution to the problem of effective data transfer in high-speed wireless networks. Importantly it more be easily deployable where we should consider that only four versions of TCP have been widely deployed in the past three decades. This due to the long time required for the standardization, implementation, and deployment of kernel space protocols. This means that although there are numerous TCP variants proposed these should not expected to be deployed widely in the near future.

- A protocol where bandwidth estimation techniques are not used in any form of congestion control mechanism. This removes the need for manual tuning of the control parameters for optimal performance as it necessary in TCP.

- To systematically investigate the design and implementation issues of high performance data transport protocol at the application level. This appears to have been neglected in many cases

of transport protocol design despite it seems to be the fact that the design and implementation does have a significant impact on efficiency. As address previously in this State-of-the-Art this is directly related to the overhead arising from acknowledgements, loss processing, threading, and memory copy. One appropriate goal in this dissertation is to therefore to propose solutions to such issues that are applicable to all transport layer solutions.

- To suggest a congestion control algorithm that addresses both the objectives of efficiency and fairness.

- A protocol whose flows are fair to each other, even if they have different round trip times as is typically the case in a Wireless Networks. While a protocol that is highly efficient is a very desirable one in today's bandwidth rich society, it should not be aggressive. This is so that it is friendly to TCP flows which in the age of the Internet we simply cannot expect to go away due to TCP forming a de facto standard.

- The protocol that uses an appropriate algorithm to solve the loss synchronisation problem. The loss synchronisation problem is the situation when all multiple flows increase and decrease their sending rate at the same time, thus the aggregate throughout has a very large oscillation and leads to a low average utilization of the bandwidth. This does not seem have been addressed in any of the protocols examined and should be considered where it is a very narrow minded to assume that users will be limited to a single flow.

- Finally but arguably most importantly a protocol that can also handle non-congestion packet losses. This should not be read as a requirement for the protocol to have explicit knowledge of when network congestion is occurring but to react more cautiously when packet loss occurs.

3

Design

This section proposes Axon as an alternative data transfer protocol in wireless networks where TCP does not always work well. One of the most common cases, and also the original motivation of Axon, is to overcome TCP's inefficiency in wireless high bandwidth-delay product networks. We focus on wireless networks as a whole rather than the Ad-Hoc environment specifically. As stated in the previous chapter this is necessary where to solve the issues Ad-Hoc wireless environments we must first deal the problems that are also apparent with wireless network in general.

The design therefore particularly focuses on the challenges that wireless networking encourages us to overcome. This includes the challenge of how we may go about uniquely address the issue of congestion from connection errors or if such an outcome is overspecialised.

From an architectural perspective Axon sits at the application layer and is therefore reliant on UDP. This brings an important advantage as regards implementation where this makes a relatively easy undertaking when compared to the alternatives. For example a modification to TCP would involve implementation around the inner workings of the Operating System where the TCP/IP would normally be located. A connection oriented, unicast, and duplex protocol, Axon is feature rich where it enables reliable data streaming and implements a neurologically inspired congestion control algorithm.

The congestion control algorithm is based on a MIMD (Multiplicative Increase Multiplicative Decrease) rate control technique specially devised for the high bandwidth delay wireless links. A troublesome mode of data communication stunted by the running of transport layer protocols ill equipped to be used optimally.

3.1 Introduction

Make no mistake the Transmission Control Protocol (TCP) is very successful and greatly contributes to the popularity of today's Internet. TCP, as of the present day, still contributes the majority of the traffic on the internet and should be expected to do so in some form for many years to come. However TCP is not perfect by any means and due to it's maturity has found use in networks which it was never designed to be used in. This is apparent where we have seen a rapid advance of wireless networks and media rich Internet applications. This where TCP has been found to be inefficient as the network bandwidth delay product increases (Henderson et al., 1998) as highlighted in the State of the Art.

On closer inspection we see that current research suggests that the choice of a AIMD (additive increase multiplicative decrease) algorithm for congestion control is not a well informed one in the specific case of Wireless Networks. This is as TCP reduces the congestion window drastically in the event of network issues but fails to recover it to the available bandwidth in the time required to use the available bandwidth optimally. Further more theoretical flow level analysis has shown that TCP becomes increasing susceptible, to packet loss as the bandwidth increases (Henderson et al., 1998). This is bad news in the case of wireless networks where packet loss, due to unpredictable nature of radio communications happens more often than we, as developers, would like.

To overcome the TCP's inefficiency problem over the high speed wireless links and hopefully mobile ad-hoc networks is the primary motivation of Axon. The term high speed is a valid one even if this displease the marketeers of optical fibre networks and similar technologies. TCP was designed in the infancy of data communications between computers and at a time where speeds approaching 400b/s were becoming commonplace. Although there are new TCP variants deployed today such as BiC TCP on Linux and Compound TCP on Windows better suited to today's faster networks certain problems still exist. The main issues that stand out and that will remain the focus of this dissertation is that none of the new TCP variants address network latency unfairness or that packet loss might not be a result of congestion.

Network latency bias proves to be a big issue in the wireless networking world where this causes connections with shorter network latency to unfairly consume more bandwidth (Henderson et al., 1998). A varying network latency is a typical occurrence wireless network as concluded by (Balakrishnan et al., 1995). This is frequently caused by transmission error and the inevitable requirement retransmission at the underlying MAC layer. This becomes much more probable in the case of mobile networks where the state of the network is fluid as nodes are not static. This is seen particularly

in the case of Ad-Hoc wireless networks where this maybe caused by the varying number of base stations that data packets must travel across before they reach the required destination. What is important to remember though is that in either case the varying network latency problem is likely to be more frequent than in traditional wired networks.

The real motive behind this design is the hope that they will be situations when Axon proves a more helpful choice than TCP. For example where the congestion control and reliability control in TCP may not desirable in emerging applications such as High-Definition Telepresence, Media streaming and Real-Time Data Backup over wireless networks. To achieve this it is necessary to design a new, well defined data transfer protocol. This is the case where we do not want to restricted by the needs of an existing protocol, such as TCP, that was designed for legacy networking environment and would not be as accommodating to the novel approach required by modelling transmission mechanism that occur between neurons.

3.2 Neurology - An inspiration

Axon gains it's name from the long slender projection of a nerve cell that conducts electrical impulses away from a neuron's cell body. From this it should be inferred that the design of protocol has been embolden by the inner mechanisms of Neurotransmission. Here the mechanism of Neurotransmission will be summarised in addition to some of the more fundamental concepts so that we may gain some insight of how we can improve both the efficiency and effectiveness of the transport layer in the general case of Wireless Networks.

As we have already looked into the similarities of Neurons and Mobile Ad Hoc Networks it is necessary to look at the mechanisms of neurotransmission closer. This is in addition to the fundamental concepts that will aid us in the design of Axon as a effective transport layer protocol for this particular breed of wireless networks.

3.2.1 The Neuron - A technical perspective

Neurons are the basic processing units of the brain. Each neuron receives electrical inputs from many other neurons who then pass the signal on to many more. Impulses arriving simultaneously are added together and, if sufficiently strong, lead to the generation of an electrical discharge, known as an action potential. The action potential then forms the input to the next neuron in the network.

Neuron have a very specialized structure as far as biological cells go. They have a cell body or soma and fine tissues that run from it. These processes are split into two types - dendrites are the highly branched processes that carry the incoming information, in the form of electrical impulses, to the soma; the action potential leaves via the axon. Axons can be very short or very long depending on the location of the neuron which is meant to receive the signal. They can also be highly branched, so neurons not only receive information from many neurons, they pass the result of the processing to many other neurons, forming the dense neural networks that are such a feature of the brain. It is quite easy to relate this arrangement to what occurs in a Mobile Ad Hoc Network.

It is our hope therefore to examine the methods and operations between Neurons so we may better the data transmission between wireless nodes.

3.2.2 The Focus - How neurons communicate between each other?

As already stated in the State of the Art neurons communicate at structures called synapses in a process called synaptic transmission. The synapse consists of the two neurons, one of which is sending information to the other. The sending neuron is known as the pre-synaptic neuron placed before the synapse while the receiving neuron is known as the post-synaptic neuron where it is found after the synapse. Although the flow of information around the brain is achieved by electrical activity, communication between neurons is a chemical process. This is arguably comparable to the differences between a wired and wireless network connection where the nature of process is fundamentally different.

When an action potential reaches a synapse, pores in the cell membrane are opened allow an influx of calcium ions (charged atoms) into the pre-synaptic terminal. This causes a small 'packet' of a chemical neurotransmitter to be released into a small gap between the two cells. This is what we call the synaptic cleft. The neurotransmitter diffuses across the synaptic cleft and interacts with specialized proteins called receptors that are embedded in the post-synaptic membrane. These receptors are ion channels that allow certain types of ions to pass through a pore within their structure. The pore is opened following interaction with the neurotransmitter allowing an influx of ions into the post-synaptic terminal. This is propagated along the dendrite toward the soma.

Quite interestingly Neurotransmission can be either excitatory where it increases the possibility of the post-synaptic neuron firing an action potential, or inhibitory. In this case, the inhibitory signal reduces the likelihood of an action potential being generated following excitation. At a cellular and chemical level this is hard to comprehend where the true complexity of the nervous system becomes all too apparent.

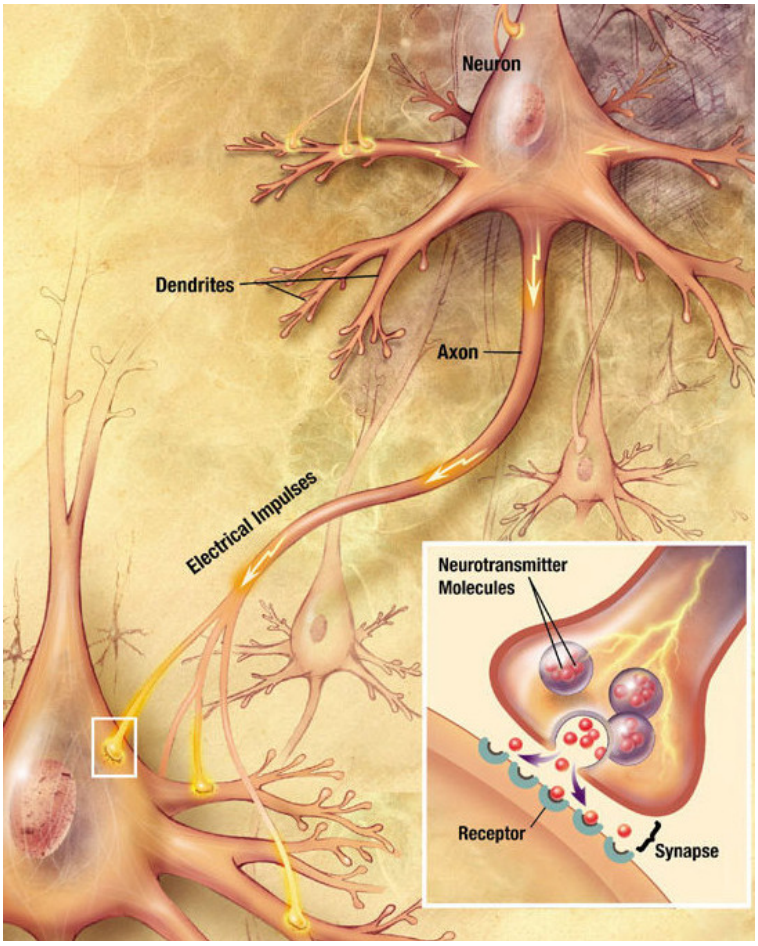


Figure 3.1: A diagram illustrating Neurotransmission

We have already seen that action potential is propagated by the leading edge of a depolarisation wave activating sodium channels further down the axon. We have also seen that the activation of these sodium channels is achieved by a small depolarisation of the neuronal membrane.

We have not considered the case when membrane potential is stabilised, The depolarisation inside the neuronal axon would dissipate and the action potential would not be able to propagate any further so inhibiting it. This stabilisation of the membrane potential is achieved by an influx of negatively charged chloride ions that are unaffected by the depolarisation wave coming down the axon. Formerly, this is equivalent to an efflux of positively charged sodium ions. Put in layman's speak this is like punching a hole in a hose so that water will leak out through the puncture and not get to the sprinkler! Put in the language of a Computer Scientist this is like a node halting the sending of data packets when required.

Although very intrinsic this does identity the fact that neurotransmission is a controlled process this is where this dissertation will focus it's efforts in determining how neuroscience could lead us to developing something that can be applied effectively on a Mobile Ad Hoc Network. How are these small 'packets' of chemical neurotransmitters released into the synapse between the two cells controlled? What are causes and effects of such control measures? Both very valid questions that we shall attempt to answer in the following subsection.

3.2.3 An in-depth analysis

To fully understand how control in built into the process of neurotransmitter it is necessary to understand how the signal output of a neuron can either cause excitation or inhibition in the neuron it is connected to. This is what we inner by the term *Action Potential*.

When we take an arguably well deserved step from biological jargon and processes this is quite to easy to understand. a neuron sends an excitatory signal to another neuron, then this signal will be added to all of the other inputs of that neuron. If it exceeds a given threshold then it will cause the target neuron to fire an action potential, if it is below the threshold then no action potential occurs. This is illustrated in Figure 3.2

By definition an action potential is an electric pulse that travels down the axon until it reaches the synapse where it then causes the release of neurotransmitters. The synapses are extremely close to the dendrites of the target neuron. This allows the neurotransmitters to diffuse across the intervening space and fit into the receptors that are located on the target neuron.

This causes some action to take place in that neuron that will either decrease or increase the membrane potential of the neuron. If it increases the membrane potential then it is exciting the

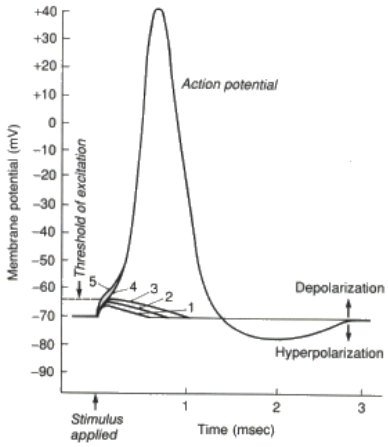


Figure 3.2: Graph showing the relation between Action Potential and Stimulus

neuron, and if it decreases the membrane potential it is inhibiting the neuron. If it causes the membrane potential to pass the firing threshold then it will activate an action potential in the target neuron and send it down its axon.

The action potential travels down the length of the axon as a voltage spike. It does this using the steps outlined above. As a section of the axon undergoes the above process it increases the membrane potential of the neighbouring section and causes it to spike. This is like a mini chain reaction that proceeds down the length of the axon until it reaches the synapse. An important thing to keep in mind about the action potential is that it is one way, and all or nothing. The action potential starts at the top of the axon and goes down it.

Also, if a neuron fires then the action potential is the same regardless of the amount of excitation received from the inputs. What is important in neurons is the rate of fire. Figure 3.3 demonstrates this principal. A weak stimulus will cause a lower rate of fire than a strong stimulus. This shows it is not the amplitude of the action potential that is important in terms of control, but the number of times a neuron fires for a given time period. The extent at which neurons exchange “data” in the broadest sense is therefore rate based.

Simply put you keep the “packets” that enable transmission of “data” of a static size but just vary how often you send them depending on the given stimulus. In Neurons this maybe anything from the concept of reward to pain. In Ad Hoc Networks we need to decide what stimulus we should choose to react on.

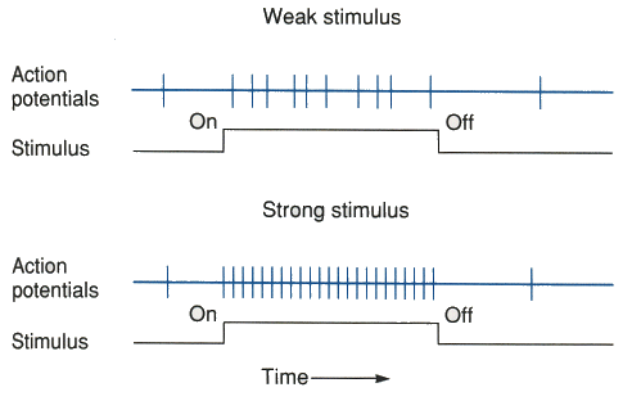


Figure 3.3: The relation between Stimulus and Neuron Firing rate

3.2.4 The borrowed concepts

Like neurons Axon will follow the sender and receiver paradigm where one node sends something to another. It will feature a rate based mechanism that will determine the rate at which data is sent depending on a number of variables or stimulus. In the Synapse these take the form of chemical neurotransmitters but in the case of Axon this will be packets where the field of application changes from Biology to Computer Science. This will form part of a hybrid rate-window congestion control determined by the amount of available bandwidth detected as well as receiving rate of packets at a particular node. These have been chosen as they simple but effective measures of the state of the network. It is hoped that this will enable Axon to overcome the problems faced by TCP which arguably takes too simplistic a view of network.

Another borrowed concept related to bandwidth is that of a limited number of neurotransmitters or rather the degradation and elimination of the chemical that forms them. Neurotransmitter must be broken down once it reaches the post-synaptic cell to prevent further excitatory or inhibitory signal transduction. For example, acetylcholine, an excitatory neurotransmitter, is broken down by acetylcholinesterase. Other neurotransmitters such as dopamine are able to diffuse away from their targeted synaptic junctions and are eliminated from the body via the kidneys, or destroyed in the liver.

As we have already determined we will take the simplistic view that neurotransmitters are analogous to packets in a network. Again stepping back from the biological jargon we can make a number of important observations of potential features that should be modelled in the Axon protocol.

The first is that neurotransmitters are produced and consumed like packets on network. Crucially this means we do not have to depart or better abstract away from the way we currently quantised data prior to transmission over a network. More substantially is the concept that number of neurotransmitters or packets "in play" should have a effect on transmission. Unlike in neurons where this determined through potential differences we use a combination of acknowledgements and loss reports to determine the amount of packets that have been received.

A full and accurate explanation of how these will be used will feature in the following sections but it is important to look at loss reports a little closer. This is as their derivation from the mechanism is less simple. Loss reports simply informs the sender of what the receiver has not received and requires to ensure the accurate receiving of information. This is necessary in Axon being network protocol as neurotransmitters do not carry sequenced and packet specific data. If a neurotransmitter goes missing the neuron cell just makes another one depending on the state of excitation at that time. What is ultimately important is that the signal is past on.

Acknowledgements are closer to what happens in the synapse however where this replaces the concept of polarisation. We specific acknowledgements as excitatory where in neurons it increases the possibility of the post-synaptic neuron firing an action potential. In the more plain world of wireless networks this means that we should consider this a stimulus, either weak or strong, for the sender.

3.3 Implementation Structure

Axon is built solely on the top of UDP and like between neurons in biology the mechanisms of connection management (action potential) and data transfer (neurotranmission) are kept separate. This means that both data and control packets are regarded as separate entities over UDP. Axon is therefore connection-oriented where this allow us to easily maintain congestion control, reliability, and security. Again like the nervous system where communication occurs from neuron to neuron through a shared synapse a unicast protocol is proposed where data may be transferred in duplex. The protocol also supports reliable data streaming where for simplicity the data streaming semantics is similar to that of TCP.

From a more technical perspective Axon adapts itself into the layered network protocol architecture 3.4 as the application layer. Axon uses UDP through the socket interface provided by operating systems. Meanwhile, it provides a Axon socket interface to applications. Applications can call the Axon socket API in the same way they call the system socket API.

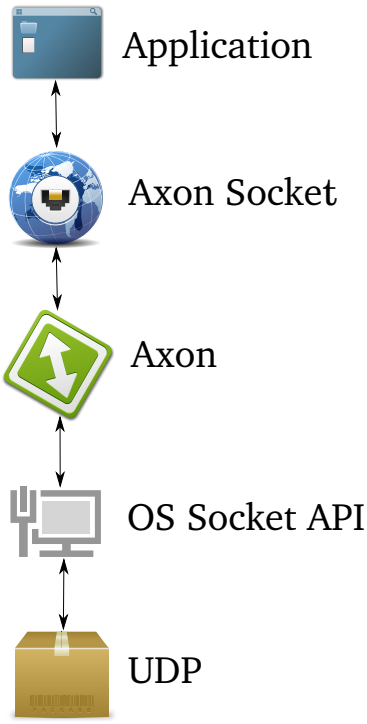


Figure 3.4: Axon in the layered network protocol architecture

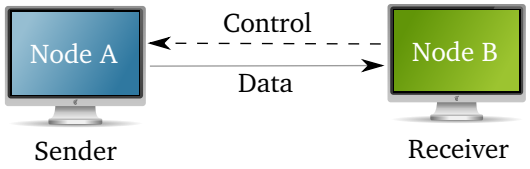


Figure 3.5: Relationship between Axon sender and receiver

As already stated Axon entity should include two logical parts: the sender and the receiver. This is analogous to what occurs between neurons where you have pre and post-synaptic connections. The sender sends application data according to flow control and rate control and retransmits whenever this proves necessary. This is dependent on the state of the network at that particular time depending on aspects such as packet loss. The receiver forms a similar but more simplistic entity where it listens for both data packets and control packets, and sends out control packets depending on packets received.

This relationship between the sender and the receiver in Axon is described in Figure 3.5. Here one Axon entity sends application data to another Axon entity. Conceptually data is therefore sent from one sender to another receiver, whereas the control flow is exchanged between the two receivers.

The receiver is also responsible for triggering and processing all control events, including congestion control and reliability control, as well as any related mechanisms. Axon uses rate-based congestion control (rate control) and window-based flow control to regulate the outgoing data traffic. Rate control updates the packet-sending period every constant interval, whereas flow control updates the flow window size each time an acknowledgement packet is received.

Respecting the fact that neurotransmitters are quantised Axon always tries to pack application data into fixed size packets, unless there is not enough data to be sent which may occur toward the end of stream. It is important to note however since Axon is designed to be used in modern day bandwidth heavy streams it is safe to presume that there is only a very small portion of irregular sized packets in a Axon session. The fixed size may be set up by applications and the optimal value is the maximum transmissible unit across that path so to avoid the media access layer taking issue with anything we are trying to do at the transport layer such as large scale fragmentation. The actual size of a Axon packet should therefore be inferred from the UDP header where this may vary from network to network.

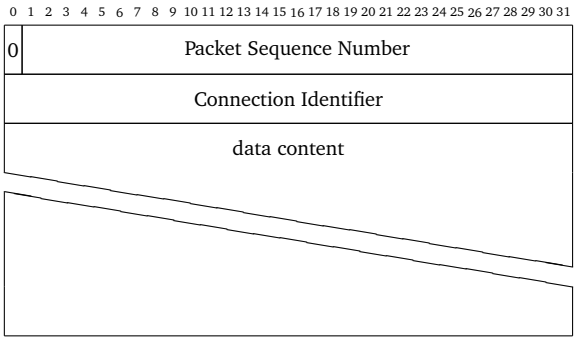


Figure 3.6: Axon Data Packet Format

3.4 Data Sending and Receiving

In terms of the sending and receiving data the sender sends and retransmits application data as is needed. This is similar to how the receiver listens for both data packets and control packets and sends out control packets according to what specific packets are processed. This is in addition to the monitoring of timers used to determine the state of the connection. These are necessary as the receiver is responsible for triggering and processing all control events, including congestion control and reliability control and any mechanisms related.

Again in keeping with what occurs between neurons a multiplexer should be implemented where this allows a single UDP port being shared. This has the complication where the sender and the receiver should always be called from the multiplexer. This emulated what fundamentally occurs in the dendrites; the branched projections of a neuron that act to conduct the electrochemical stimulation received from other neural cells to the cell body. Shying away from the biological terms again we again take the simplistic view where connections should be shared in a single entity. This should bring benefits in terms of usability where this make network management an easier task as only one port need be open or forwarded.

3.5 Packet Structures

As well as having a separate sender and receiver component Axon also has two kinds of packets – data packets and the control packets. This emulates the fact that control and transmission are two very independent mechanism even though one brings about the other. To keep things simple from the perspective of implementation this should be distinguished by the first flag bit of the packet header. The data packet header structure is shown in Figure 3.6.

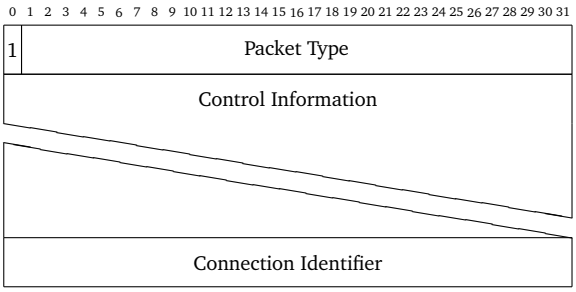


Figure 3.7: Axon Control Packet Format

The data packet header always starts with a 0 to define it as such. Then follows the packet sequence number that occupies the following 63 bits after the flag bit. Axon uses packet based sequencing and this requires for the sequence number to increased by 1 for each and in each order of data packet sent. Should the sequence number even be increased to the maximum number $(2^{32} - 1)$ it will be wrapped around.

The Connection Identifier is used for the UDP multiplexer that forms a crucial component of Axon so enabling multiple Axon sockets to be bound on the same UDP port much like many neurotransmitters share a single synapse. The Connection Identifier is thus used to differentiate between multiple Axon connections. This is another decision that has taken inspiration from the inner workings of the nervous system. At the terminus of the Neurological Axon, that enables communication between neurons possible, includes many transmitters and receptors of the neurotransmitter, serotonin, over a shared synapse.

If the flag bit of a Axon packet is *1* however then it should be inferred that is a control packet and parsed according to the structure outlined in Figure 3.7. In the draft version of Axon outlined in this design there are four types of control packets in Axon. The differential type information is put in bit field 1-15 of the header. Such a large field has been chosen to offer substantial flexibility that may occur in future developments of the protocol. The contents of the following fields will depend on the packet type where some are naturally more complicated then other. The first 32 bits and final 32-bits should exist in the packet header whereas there may be an empty *Control Information* field depending on the packet type. What is contained in the *Control Information* field will depend on varying types of control packets are detailed in the following sections.

3.6 Timers

Axon uses four timers to trigger different the periodical events that maybe necessary to keep a connection open and working optimally. Each event has its own period and are all independent in their operation. They use the system time as origins and should process wrapping if the system time wraps. This is the one of the many areas where artistic license has been applied to the inner workings of the neuron. The neuron according to present research has no concept of timers but computer networks fall short in manner where the connections between neurons do not. Transmission been neurons is reliable where the axon is protected by a myelin sheath.

The main purpose of a myelin sheath is to increase the speed at which impulses propagate along the myelinated fibre. It sheath increases electrical resistance across the cell membrane and decreases capacitance (McNeal, 1976). Thus, myelination helps prevent the electrical current from leaving the axon and makes communication between neurons reliable. It is through the use of timers that we will able to make such bold assumptions of a connection across a wireless network. This is as well as using such events as a stimulus, either excitatory or inhibitory, for the sending of packets.

The way these timers work is a matter best explained mathematically so we may keep things clear and concise. So for a certain periodical event E in Axon, suppose the time variable is E_t and its period is p. If E is set or reset at system time $t_0 \because E_t = t_0$, then at any time $t_1 \because t1 - ET \geq p$ is the condition to check if E should be triggered.

The four timers are Acknowledge, Loss, Expiry, Send. Send stands out where it is only used in the sender component for rate-based packet sending whereas the other three are used in the receiver component.

The granularity of their periods is in microseconds where this is most realistic value we should apply to the wireless networks we see today. Although greater accuracy may prove useful in pushing the capabilities of network connections this is not always practical where accurate clocks are difficult to implement, particularly in a platform independent manner.

The timers work where the system time is queried after each time bounded UDP receiving to check if any of the Acknowledge, Loss, or Expiry event should be triggered.

3.6.1 Acknowledge

Acknowledge is used to trigger an acknowledgement and its period is set by what is specific in congestion control. However, Axon will send an acknowledge at a frequency no greater than every

0.01 second, even though the congestion control does not need timer-based acknowledgement. Here, 0.01 second is defined as the Synchronisation Time ($Sync$) where it affects the other timers used in Axon.

3.6.2 Loss

The loss timer is used to trigger a negative acknowledgement (Loss Report). Its period is dynamically updated to $4 * RTT + RTT_{var} + Sync$, where RTT is the network latency, measured in milliseconds, RTT_{var} is the network latency variance also in milliseconds and Computed using the variance of network latency samples. $Sync$ is the value assigned the $Sync$ timer.

3.6.3 Expiry

The Expiry is used to trigger data packets retransmission and maintain connection status. Similarly to the loss timer its period is updated to $4 * RTT + RTT_{var} + Sync$ when required.

3.6.4 Send

Send is the timer used to time the sending of packets by. As with neurons this will vary depending on state. The logic follows if we need to send data faster we will send data more frequently as we are dealing with a fixed packet size.

3.7 Connection Management

Again this is a departure from what maybe occurs between neurons but is a necessary requirement in the case of the transport layer. In neurons once a connection is made it is quite permanent until it is cut through injury. Network connections are on the other hand are quite spontaneous by comparison and we need a method so that new connections can be made both efficiently and securely.

From a practical perspective Axon should support two different connection set-up methods, the traditional client/server mode and the rendezvous mode. In the latter mode, both Axon sockets connect to each other at around the same time. The rendezvous connection is useful when both peers are behind firewall where we use UDP hole punching (Ford, 2004). This is often necessary in modern day networks that need to employ stringent security policies to protect both data and users.

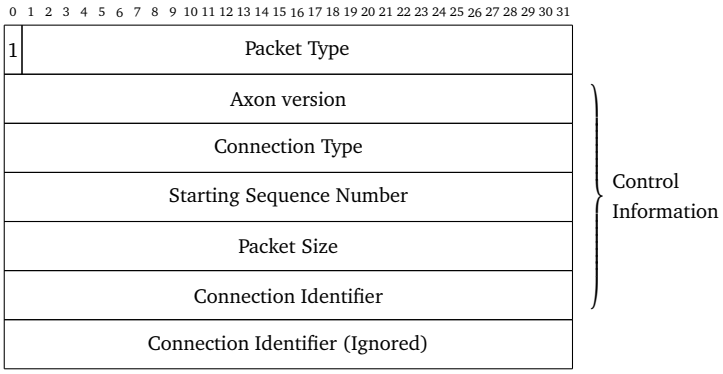


Figure 3.8: Packet Type 0 - Connection Handshake

3.7.1 The Handshake

The Axon client sends a handshake request control packet to the server or the peer side. The handshake packet has the following information. If this is preformed in rendezvous mode both peer are clients:

1. Axon version: this value is for compatibility purposes where we should expect the protocol, if used, to change over time. The current version is 1 and this should be updated in later versions to ensure compatibility.

2. Connection Type: This information is used as a differential between the connection configuration modes and request/response.

3. Starting Sequence Number: It is the starting data packet sequence number that the Axon entity that sends this handshake will use to send out data packets. This should be a random value.

4. Packet Size: the maximum size of a data packet including headers where This should be the value equal to the Maximum Transmission Unit.

5. Connection Identifier: A differential between concurrent Axon connections that maybe active.

This is illustrated appropriately in Figure 3.8.

3.7.2 The Client/Server Connection Process

One Axon entity starts first as the server which acts like listener. The server accepts and processes incoming connection request and creates new Axon socket for each new connection as needed.

A client that wants to connect to the server will send a handshake packet first. The client should keep on sending the handshake packet every constant interval until it receives a response handshake from the server or a time-out timer expires.

The server, on receiving a handshake packet, should compare the packet size and maximum window size with its own values and set its own values as the smaller ones if necessary. This will include attributes such as version, packet size and window size. The result values are also sent back to the client by a response handshake packet, together with the server's version and initial sequence number.

The server, after this step is finished, should be ready for both sending and receiving data immediately. However the server must send back a response packet as long as it receives any further handshakes from the same client. This may have to occur in the instance of a faulty connection.

The client can start sending/receiving data once it gets a response handshake packet from the server. Further response handshake messages, if they are received, should be omitted. The connection type from the client should be set to 1 and the response from the server should be set to -1. The client should also check if the response is from the server that the original request was sent to for both sanity and security reasons.

3.7.3 Rendezvous Connection Setup

In this mode, both clients send a connect request to each other at the same time. The initial connection type is set to 0. Once a peer receives a connection request, it sends back a response. If the connection type is 0, then the response sends back -1; if the connection type is -1, then the response sends back -2. No response should be sent for -2 request.

The rendezvous peer does the same check on the handshake messages as described in the case for a client/server connection. In addition, the peer only process the connection request from the address it has sent a connection request to. Finally, rendezvous connection will be rejected by a regular Axon server listening for incoming connection requests.

A peer initializes the connection when it receives -1 response.

3.7.4 Acknowledgements

This packet consists of a acknowledgement sequence number that occupies 32-bits in the control packet header as, one would expect, ranges from 0 to $(2^{32} - 1)$. The structure of the control packet carrying the Acknowledgement is shown in Figure 3.9.

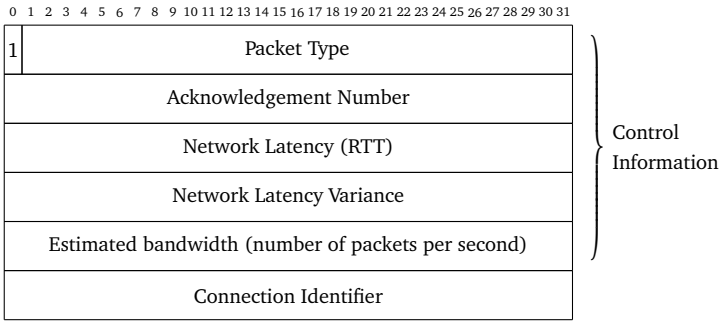


Figure 3.9: Packet Type 1 - Acknowledgement

These acknowledgements are selective by nature and are generated at every constant interval to send back largest continuously received sequence number of data packets. So that we gain crucial information about the network state this packet also carries the measured Network Latency, the variance in Network Latency as well as packet arrival speed, and estimated link capacity in respective 32-bit fields. Acknowledgements in Axon crucially form of a feedback indicating the current situation of network through a form of “Network Status Update”.

3.8 Loss Reports

Loss reports are best described as a “negative-acknowledgement” where by it is explicitly generated as soon as packet loss is detected. Loss information may be resent if receiver has not received the retransmission after an increasing interval. These measures emulate the role of the meylin sheath around the axon of a neuron where by reliable data transmission can be assured.

A useful side effect of including Loss Reports is that we can use this to determine periods of network congestion where often in Wireless Networks packets do not regularly get lost in flight as IEEE 802.11 and other standards insist on a retransmission mechanism whereby packets that are not successfully received and acknowledged are resent. This mechanism generally serves to reduce the packet loss rate to less than 0.1 percent (Aguayo et al., 2004). This makes loss reports a very versatile tool in determining if a wireless link is congested or just subject to radio transmission errors.

The packet structure for the loss report is shown in figure 3.10.

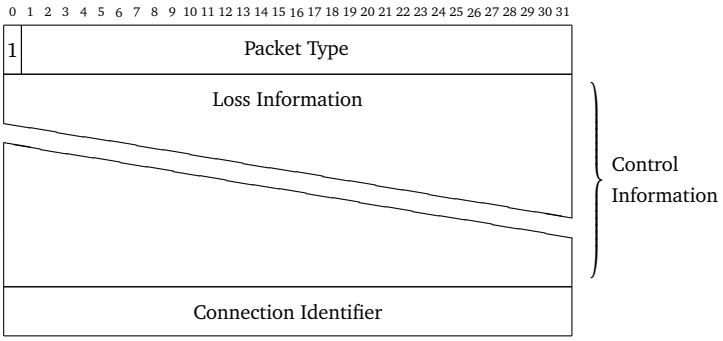


Figure 3.10: Packet Type 2 - Loss Report

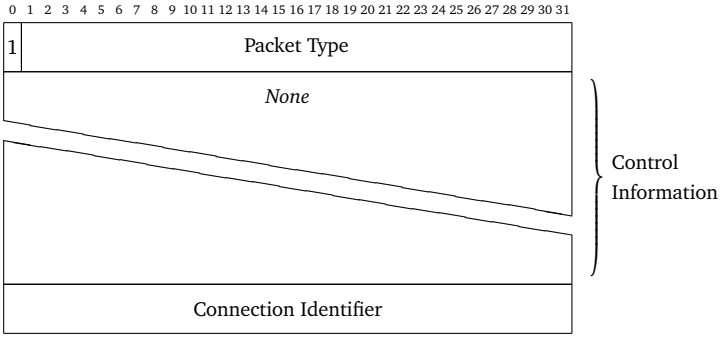


Figure 3.11: Packet Type 3 - Stay Alive

3.9 Connection Maintenance

Again departing from the theory we understand about neurons we need a mechanism to determine whether or not a node remains accessible on a wireless link. This is a necessity where Axon should be classed as a Streaming Protocol where we make every effort to ensure the correct delivery of each bit presented. TCP accomplishes this with a system of time outs and retries and similarly so does Axon albeit with less complexity. In Axon a connection should remain open between a sender and receiver so long as *Stay Alive* packets are periodically exchanged between the two. Notably if the message is not received, the peer side should be closed after 16 continuous time outs of the Expiry timer.

The structure for the Connection Maintenance packet is shown in figure 3.11.

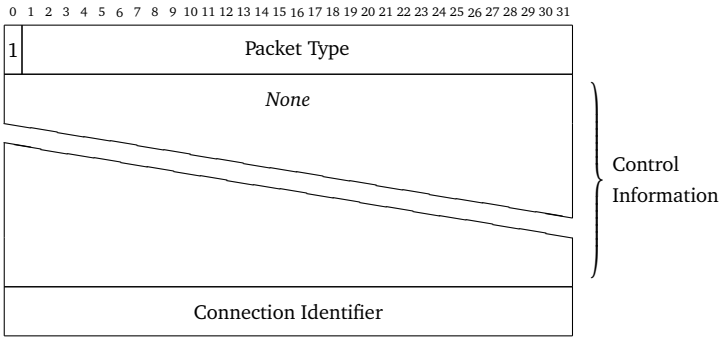


Figure 3.12: Packet Type 4 - Shut Down

3.10 Connection Shutdown

If one of the connected Axon entities is being closed, it will send a shut down message to the peer side. The peer side, after received this message should, as a consequence, also be closed. This shut down message is only sent once and not guaranteed to be received due to the nature of UDP. If it is case that the message is not received, the peer side should be closed after 16 continuous time outs of Expiry timer. This similar to the case in Stay Alive packets. The total time out value should therefore be in region between 3 seconds and 30 seconds for the most standard wireless network connections.

The structure for the Connection Maintenance packet is shown in figure 3.12.

3.10.1 The Sender's Algorithm

The Sender's Algorithm is simple where it focuses around a single data structure, the *Sender's Loss List*. The *Sender's Loss List* is used to store the sequence numbers of the lost packets fed back by the receiver through Loss Report packets. The numbers are stored in increasing order where they should increase as packets are sent. Please refer to Algorithm 7 for further details.

3.10.2 The Receiver's Algorithm

The Receiver's Algorithm is more complex where it deals with six separate data structures and variables. The first is the *Receiver's Loss List* a list of tuples whose values including the sequence numbers of detected lost data packets, the latest feedback time of each tuple, and a parameter k that is the number of times each one has been fed back in the loss report. These values are stored in the increasing order of packet sequence numbers for manageability.

Algorithm 7 - The Sender's Algorithm

while An Axon Connection is open **do**
  if The sender's loss list is not empty **then**
    Retransmit the first packet in the list
    Remove this packet from the list
  end if
  Wait until there is application data to be sent
  if If the number of unacknowledged packets exceeds the flow/congestion window size **then**
    Wait until an acknowledgement arrives from the receiver
    Continue
  else
    Send a new data packet
  end if
  if the sequence number of the current packet is 8n, where n is an integer **then**
    Continue
  end if
  Wait $(Send - t)$ time, where $Send$ is the inter-packet interval updated by congestion control and t is the total time used by step 1 to step 5
end while

Next is the *Acknowledgement History Window*, a circular array of each sent acknowledgement and the time it is sent out. The most recent value will overwrite the oldest one if there is no more free space in the array. This is closely followed by two further circular arrays: the *Packet History Window* and the *Packet Pair Window*. These store the arrival time of each data packet and time interval between each probing packet pair respectively.

Finally are the two integers *Largest Received Sequence Number* and *Expiry Count*. *Largest Received Sequence Number* is a variable that records the largest received data packet sequence number and initialised to the sequence number -1. The *Expiry Count* is a variable to record number of continuous Expiry time-out events.

3.10.3 Acknowledgement Event Processing

The primary purpose of this algorithm find the sequence number prior to which all the packets have been received by the receiver. This maybe referred to as the Acknowledgement Number.

Notably the pseudocode outlined in Algorithm 11 makes uses of the $LargestReceivedSequenceNumber$ that records the largest received data packet sequence number and two useful subroutines find in Algorithms 9 and 10.

Algorithm 8 - The Receiver's Algorithm

while Time bounded UDP receiving **do**
  if A Acknowledgement, Loss, or Expiry timer has expired **then**
    if This is a acknowledgement timer **then**
      Check the acknowledgement packet interval
    end if
    Process this event accordingly (see subsections 3.10.3, 3.10.4, and 3.10.5)
    Reset the associated time variables
  end if
  if no Packet arrives **then**
    continue
  end if
  Reset the Expiry Count to 1.
  if there is no unacknowledged data packet **then**
    reset the Expiry timer.
  end if
  if if this is an acknowledgement **then**
    reset the Expiry timer.
  end if
  if Loss Report control packet **then**
    reset the Expiry timer.
  end if
  if This is a control packet **then**
    Process it accordingly to packet's type
    continue
  end if
  if the sequence number of the current data packet is $8n+1$, where n is an integer **then**
    Compute the time interval between this packet and last data packet in the Packet Pair Window
  end if
  Compute the packet arrival time in Packet History Window.
  if the sequence number of the current data packet is greater than Largest Received Sequence Number + 1 **then**
    Put all the sequence numbers between and excluding these two values into the receiver's loss list
    Send the receiver's loss list to the sender in an Loss Report packet
  end if
  if the sequence number is less than Largest Received Sequence Number **then**
    Remove this sequence number from the receiver's loss list
  end if
  Update Largest Received Sequence Number
end while

Algorithm 9 - Calculation of the packet arrival speed

Compute the median value of the last 32 packet arrival intervals (x_{ai}) using the values stored in Packet History Window.
In these 32 packet arrival intervals, remove those either greater than $x_{ai} \times 8$ or less than $x_{ai}/8$
if more than 8 packet arrival intervals are left **then**
  Compute the average of the left packet arrival intervals x'_{ai}, and the packet arrival speed $1/x'_{ai}$ (number of packets per second)
else
  Assume that $x'_{ai} = 0$ AND $1/x_{ai} = 0$
end if

Algorithm 10 - Calculation of the Link Capacity

Compute the median value of the last 32 packet pair intervals (x_{pi}) using the values in Packet Pair Window
Compute the link capacity as $1/x_{pi}$

Algorithm 11 - acknowledgement Event Processing Algorithm

if the receiver's loss list is empty **then**
The acknowledgement number is $LargestReceivedSequenceNumber + 1$
else
The acknowledgement number is the smallest sequence number in the receiver's loss list
end if
if the acknowledgement number is equal to the acknowledgement number in the last acknowledgement and the time interval between this two acknowledgement packets is less than twice the network latency **then**
DO NOT send this acknowledgement
end if
Assign this acknowledgement a unique increasing acknowledgement sequence number. Pack the acknowledgement packet with network latency, network latency Variance, and flow window size
Compute the packet arrival speed using Algorithm 9
Compute the estimated link capacity according to Algorithm 10
Pack the packet arrival speed and estimated link capacity into the acknowledgement packet and send it out.
Record the acknowledgement sequence number, acknowledgement number and the departure time of this acknowledgement in the acknowledgement History Window.

3.10.4 Loss Report Event Processing

Algorithm 12 details of what should occur if a Loss timer should expire and we should assume packets have gone missing.

Algorithm 12 - Loss Report Timer Expiration Processing

$k \leftarrow 2$
for those sequence numbers whose last feedback time is $k \times networklatency$ before where k is initialized as 2 and increased by 1 each time the number is fed back **do**
$k \leftarrow k + 1$
Compress these numbers back to the sender in an Loss Report packet
end for

3.10.5 Expiry Event Processing

Algorithm 13 details what should occur in the event of Expiry event being triggered by the appropriate timer. This is an important mechanism where it decides where a link should remain open or not. This is quite crucial in a wireless network and in particular a streaming protocol where data may not always be exchanged on a constant basis.

Algorithm 13 - Expiry Event Processing Algorithm

Put all the unacknowledged packets into the sender's loss list
if $ExpCount > 16$ OR 3 minutes has elapsed **then**
  Close the Axon connection
end if
if the sender's loss list is empty **then**
  Send a keep-alive packet to the peer side
end if
$ExpCount \leftarrow ExpCout + 1$

3.10.6 On receiving an Acknowledgement Packet

Algorithm 14 shows the importance of acknowledgements where they give us the variables and means to apply the neurology inspired transmission approach to wireless links. These will act as stimulus and can be used in Axon's rate based approach to congestion control. Axon could be regarded as driven by acknowledgements in this manner where entails the most computational complexity.

Algorithm 14 - Acknowledgement Processing Algorithm

Update the largest acknowledged sequence number
Send back an Acknowledgement of Acknowledgement (AoA) with the same acknowledgement sequence number in this acknowledgement with the next data packet
Update network latency and network latency variance
Update both acknowledgement and loss report period to $4 * RTT + RTT_{var} + Sync$, where RTT is the network latency and RTT_{var} the network latency variance.
Update flow window size
Update packet arrival rate, A to $(A * 7 + a)/8$, where a is the value carried in the acknowledgement
Update estimated link capacity, B to $(B * 7 + b)/8$, where b is the value carried in the acknowledgement
Update sender's buffer (removing elements in the buffer that have been acknowledged)
Update sender's loss list (removing all those elements that have been acknowledged)

3.10.7 On receiving a Loss Report packet

The processing of explicit Loss Reports is detailed in Algorithm 15. As already suggested this be regarded as a sign of congestion where wireless networks do not drop packets as we should expect due to retransmission measures deployed within the MAC Layer. This causes the Packet Sending Period to be adjusted and slowed on receiving such a packet. The topic Rate Control is dealt with in relevant following subsection detailing Congestion Control.

Note that the use of explicit loss reports is very different to the use of duplicate acknowledgements in TCP. With duplicate acknowledgements, the sender may not know all the loss events in one congestion event, and usually only the first loss event is detected. In fact most TCP implemen-

tations will not drop the sending rate more than once in a time period equal to the network latency. However, in Axon, all loss events will be reported by a Loss Report so we have added clarity.

Algorithm 15 - Loss Report Processing Algorithm

Add all sequence numbers carried in the Loss Report into the sender's loss list
Update the Packet Sending period using the *Rate Control Method*
Reset the Expiry time variable

3.10.8 On Acknowledgement of Acknowledgement (AoA) packet received

Initially the use of AoA appears to be a bit overkill but it is necessary if we to keep the task of data transmission and control separate as in the neuron model. AoA packets where the sender is required to send one back to the receiver for each ACK. For the small amount of bandwidth they do arguably waste bandwidth but we must not disregard the fact that they could prove useful where they allow us to gain an accurate measure of network latency and crucially network latency variance. This gives a fair idea of what situation of the network is at that time. Network latency variance or jitter is important metric in wireless networks, in particular, where the round trip time for packets will vary due to problems brought by wireless transmission. On receiving a AoA packet Algorithm 16 should be followed.

Algorithm 16 - Acknowledgement of Acknowledgement (AoA) Processing Algorithm

Locate the related acknowledgement in the acknowledgement History Window according to the acknowledgement sequence number in this AoA
Update the largest acknowledgement number ever to be acknowledged so far
Compute new network latency according to the AoA arrival time and the acknowledgement departure time
Update the network latency variable
Update network latency variance
Update both acknowledgement and loss period to $4 \times RTT + RTT_{var} + Sync$, where RTT is the network latency and RTT_{var} the network latency variance.

3.10.9 On Keep-alive packet received

This is nice and easy, just do the necessary sending of the control packet to keep the connection alive and therefore open.

3.11 Flow Control

Current research shows that Window control sends data in bursts, and may have sent a large number of packets by the time when the sender learns that there is congestion along the link. In

addition, the bursting traffic requires that routers have a buffer as large as the bandwidth delay product. This is however unrealistic on lossy wireless links whose delay is significant, especially when compared to wired equalising.

It has been proposed that packets be sent within the congestion window at average intervals to alleviate this problem in TCP. However using the congestion control algorithm in TCP to determine the packet-sending period in this manner often decreases the throughput and works especially poorly when coexisting with standard TCP (Aggarwal et al., 2000).

In such wireless links neurons may show us a the better solution where by we tune the packet-sending period directly with an efficient rate control mechanism depending on a number of variables. However, we must consider that rate control can also lead to another situation of continuous loss: when congestion occurs and the loss report packets get lost, the data source may continue to send out data before it receives a loss report or a time-out event. Therefore, a supportive window control should be used together with rate control to specify a threshold on the number of unacknowledged packets. This is yet another departure of the inspiration model of neuron but we should respect the many differences between what goes on in the nervous system and computer networks.

Axon therefore combines these two mechanisms. Rate control is the major mechanism used to tune the packet-sending period, whereas window control is a supportive mechanism used to specify a dynamic threshold that limits the number of unacknowledged packets. The window control is refereed to flow control in this protocol as it incorporates a simple flow control mechanism by feeding back the minimum value between the congestion window size and the current available receiver buffer size. This works where Axon's flow control computation is done at the receiver side and is simple in operation where on receiving acknowledgement packet received the flow window size is updated to the receiver's available buffer size. The flow control window size is 32 initially.

3.12 Congestion Control

We explain the rationale of the Axon congestion control algorithm and determine its performance in this chapter. We generalize a new class of MIMD algorithms in which the size of the additive increases will decrease as the data sending rate increases. We call these algorithms MIMD algorithms with decreasing increases and increasing decreases or DIID algorithms. We will show that DIID algorithms are stable, fair, and efficient. The Axon algorithm is a special case of the DIID

algorithm where the increase and decreasing parameter is proportional to the available bandwidth that is estimated by a bandwidth estimation technique.

3.12.0.1 MIMD with decreasing increases and increasing decreases (DIID)

We consider a general class of the following AIMD rate control algorithm:

For every rate control interval, if there is no negative feedback from the receiver due to loss, but there are positive feedbacks (acknowledgements), then the packet-sending rate (x) is increased by a constant factor $\alpha(x)$ ($0 < \alpha < 1$).

$$x \leftarrow \alpha.x \tag{3.1}$$

$\alpha(x)$ is non-increasing and it approaches 0 as x increases

For any negative feedback, the sending rate is decreased by a constant factor β ($0 < \beta < 1$):

$$x \leftarrow (1-\beta)x \tag{3.2}$$

By varying $\alpha(x)$, we gain a class of rate control algorithm that we name the DIID algorithm where decreasing increases and increasing decreases.

If we use the rate control interval as a unit of time, then from time t to $t+1$, the increase to the sending rate would be at follows:

$$x(t+1) = \alpha^n x(t) \tag{3.3}$$

and the decrease from is Computed by:

$$x(t+1) = (1-\beta)^n x(t) \tag{3.4}$$

where n is the number of negative feedbacks.

There is one fundamental difference between DIID and some TCP variants that use loss as a congestion signal: as the window size becomes larger HighSpeed TCP, for example, increases faster, whereas the increase of BiC TCP may be independent of the absolute sending rate but it is determined by the distance between the current sending rate and a target rate.

One fact remains is that the function of $\alpha.x$ has to be large to be efficient and it has to decrease quickly to reduce oscillations. The first stage in the function should be used to decide how quickly a DIID flow can probe the available bandwidth at the beginning, and the length of the stage determines how aggressive is observed to be. The longer the stage is, the more aggressive it will be. Each

later stage has a smaller increment as the flow approaches available bandwidth. This will reduce the oscillations at the equilibrium.

Specifically, to achieve efficiency, the increment at each stage should be proportional to the available bandwidth and this is what is aimed for in the DIID algorithm.

3.12.0.2 MIMD (DIID) in Axon

Axon adopts this efficiency idea and specifies a piecewise $\alpha.x$ that is related to the link capacity.

The Axon rate control directly tunes the packet-sending period t which indirectly determines the packet-sending rate x:

$$t_x = 1 \tag{3.5}$$

We therefore can write the rate control formula in the form of the sending rate. The fixed rate control interval of Axon is SYN, or the synchronization time interval, which is 10 milliseconds. Axon rate control is a special DIID algorithm by specifying $\alpha(x)$ in Equation 3.6 where S is the current packet sending rate, B_{MAX} is the maximum detected bandwidth, B_n is the amount of bandwidth currently being used and t is the time since the last increase or decrease. The unit in all cases is packets per second where this removes the need to worry about a varying packet size depending on implementation.

$$S = \alpha(x) = B_{MAX}.(1 - e^{\frac{t}{\log_{10}(B_n)}}) \tag{3.6}$$

Conversely Equation 3.7 shows the $\beta(x)$ and the manner in which the packet sending rate should be decreased in the event of a Loss Report being received.

$$S = \beta(x) = B_{MAX}.e^{\frac{t}{\log_{10}(B_n)}} \tag{3.7}$$

As you should expect this is similar the mathematical model applied to an action potential in a neuron. If we assume the time constant τ is defined as $\tau = rc$ where where r is the resistance across the membrane and c is the capacitance of the membrane.

The time constant is used to describe the rise and fall of the action potential, where the rise is described by Equation 3.8.

$$V(t) = V_{max}.(1 - e^{\frac{-t}{\tau}}), \tag{3.8}$$

The fall of action potential is conversely defined in Equation 3.9.

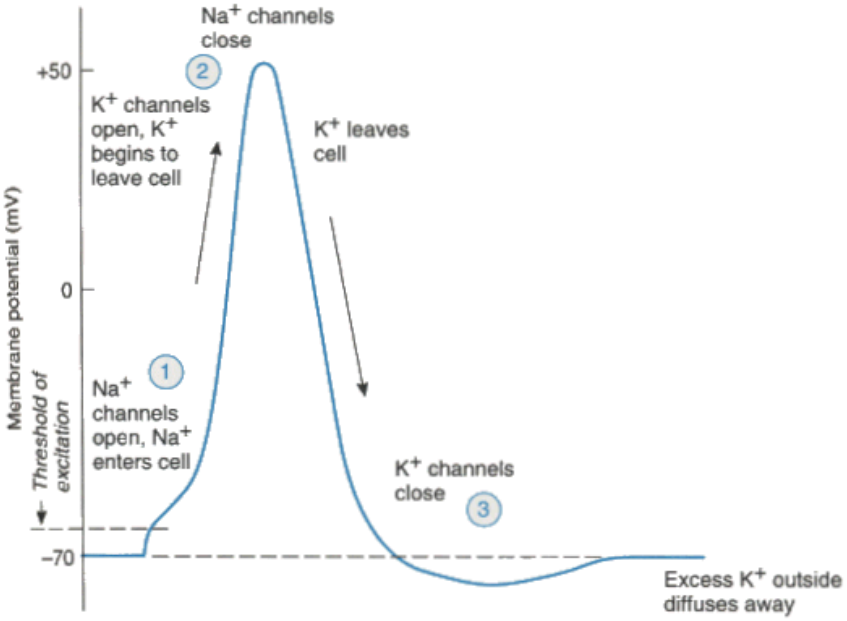


Figure 3.13: Ion flow in action potential down the length a Neural Axon

$$V(t) = V_{max}.e^{\frac{-t}{\tau}}, \tag{3.9}$$

Where V is the potential difference (voltage) measured in millivolts, the time, t, is in seconds, and τ is in seconds. The behaviour of these two equations is illustrated in Figure 3.13.

In the case of Axon, the transport layer protocol, we have substituted the base 10 logarithm of current bandwidth B_n for τ, the time constant. $\log_{10}(B_n)$ therefore specifies congestion step response to reach $1 - 1/e \approx 63.2$ of the maximum sending rate in the case of a required increase. We use the logarithm of the bandwidth where we assume that step response should be logarithmic. In the case where a decrease is a required, for example when a loss report is received by the sender, this is step response to reach $1/e \approx 63.2$ of the maximum sending rate.

It should be noted that is loss reports are received and no decrease in bandwidth is estimated the packet sending rate will be reduced to eventually $1/e$ of the available bandwidth until an Acknowledgement is received.

3.12.1 An outline of the benefits DIID brings

DIID follows Multiplicative-Increase Multiplicative-Decrease law (MIMD), which brings it's own advantages where this increases the traffic rate proportionally to the spare bandwidth in the system. This is instead of increasing by one packet as is done in many of the popular TCP implementations. This allows Axon to quickly acquire the positive spare band- width when it may become available and this makes it competitive. A quality that any high performance protocol should have but with some reservations. This is where the Decreasing Increases and Increasing Decreases comes into it's own.

Axon increases its sending rate quickly at the beginning and slows down as it is approaches the detected and estimated available bandwidth. Similarly the protocol decreases its send rate slowly initially and speeds up as further decreases are deemed necessary. It is hoped that this nature will make Axon a fair competitor when used with other protocols such as TCP. It should have stability where the sending rate is pegged to the current rate rate and estimated bandwidth. It should also be fair where we should expect the detected bandwidth to decrease as Axon is make to share a single wireless link. This is due to the multiplexing that should occur at the IP layer by the Operating System.

Finally oscillations should be avoided where developers should be able to expect a constant amount of bandwidth when needed. This has been achieved where the sending rate is scaled back quickly to allow any network congestion to pass. This hence allows us to also resume an optimal packet sending speed expeditiously.

3.12.2 Rate implied Congestion Control

Our primary goal to ensure that Axon has a congestion control algorithm designed for the data intensive application we presently see on high bandwidth delay product wireless networks. As we have defined in the previous subsections this is best achieve through a type of a hybrid congestion control algorithm. One that allows us to adjust both the congestion window size and the inter-packet interval to gain a high degree of throughput. This is achieved through the use of time sensitive acknowledgements where we assume initially that the acknowledgement interval is equal to $Sync$.

Axon dense uses a modified MIMD algorithm where on the passing of every $Sync$ time, if there is no Loss Reports, but there are acknowledgements received in the past $Sync$ time, the number of packets to be increased in the next $Sync$ time is Computed using Equation 3.6.

If however a loss report is received by the sender the number of packets to be decreased in the next $Sync$ time is Computed using Equation 3.7. A slight exception to this is where to help clear the congestion, the sender should stops sending packets in the next $Sync$ time if the largest sequence number a Loss Report is received which contains a greater value than the largest sequence number sent when the last decrease occurred.

The Axon congestion control described above is not enabled until the first Loss Report is received or the flow window size has reached the maximum flow window size. This is the slow start period of the Axon congestion control. During this time the inter-packet time is kept as zero. The initial flow window size is 1 and it is doubled each time an acknowledgement is received. The slow start only happens at the beginning of a Axon connection, and once the above congestion control scheme is enabled, it will not happen again. We require such a feature when negotiation new connections as we can not infer any state until an acknowledgement is received. This main reason for this is so Axon do not add load to an already congested network.

The initial congestion window size is 32 packets and the initial inter-packet interval is 10 milliseconds. The algorithm starts with slow start phase until the first acknowledgement or loss report arrives.

3.12.3 The Determination of Bandwidth

The rate inferred congestion control depends much on a method of estimating the amount of bandwidth that is available on a wireless link. Such information will prove invaluable where although congestion can be presupposed from the presence of Loss Reports this is too simplistic. A measure a bandwidth enables Axon, as a protocol, to be aware of the networking resources it has to it's disposal and to use them in optimal manner. Arguably the most precious information we gain from this is limit of the network described as the number of packets we may every second.

Axon uses receiver-based packet pairs to estimate the available bandwidth (Min and Dai Loguinov, 2004). This works where by a sender in Axon sends out a packet pair every 8 data packets omitting the inter-packet waiting time. The receiver then records the inter-arrival time of each packet pair or train and uses a median filter on them to compute an accurate measurement.

The process is straightforward where we suppose the median inter-arrival time is t and the average packet size in the measure period is s, then the link capacity can be estimated by calculating $fracst$. Although simple there are a number of concerns in using packet pairs to estimate bandwidth. One is the impact of cross traffic where this may cause the capacity be under estimated.

Packet pairs may also lead to a value referred to as Asymptotic Dispersion Rate, which is a value between available bandwidth and link capacity.

3.12.4 Being sensible about Packet Loss

While most loss-based congestion control work effectively, packet loss is regarded as a simple congestion indication, few of them have investigated the loss pattern in real networks. As one single loss may cause a multiplicative rate decrease in Axon, dealing with packet loss is very important.

In wireless network there are particular kinds of situations related to packet loss that need to be addressed including loss synchronization and link error occurrence. Loss synchronization is a condition in which all concurrent flows experience packet loss at almost the same time. Most familiar is the concept of link error that can give transport protocols false signals of network congestion.

The phenomenon of "loss synchronization" is less familiar where this deals with the event when a finite number of concurrent flows increase and decrease their sending rate at the same time, thus the aggregate throughout undergoes a very large oscillation and leads to low aggregate utilization of the bandwidth. This is due to the fact that almost all the flows will experience packet drops when congestion occurs and have to drop their sending rates. Similarly when there is no congestion, they all increase the sending rate. This causes problem in application wanting to make use of more than one Axon connection limiting the usefulness of the protocol.

We use a randomization method to alleviate both these problem. As we already know a Axon sender can detect a loss event when it receives a loss report. We thus define congestion event is a particular loss event only when the largest sequence number of the lost packets is greater than the largest sequence number that has been sent when the last rate decrease occurred. The period between two continuous congestion events should be referred to as a congestion epoch and should be the only only time when the sending rate is decreased.

3.13 An Overview

In this chapter, we described a new general type of MIMD congestion control algorithm, named DIID, whose increment decreases as the sending rate increases. This is different from other MIMD-based algorithms recently proposed to improve the performance of TCP in Wireless Ad Hoc environment. We also described how to estimate the parameter of link capacity in Axon and discussed the impact of estimation error. Further we discuss the robustness of our method on bandwidth estimation error.

Finally, we proposed two methods to deal with the global loss synchronization problem and the non-congestion packet loss problem, respectively. These loss handling schemes have great impacts on the transport protocol performance in today's Wireless Networks. The following chapters will deal with the implementation and the success of these approach in the Axon protocol.

4

Materials and Methods

In this chapter we will go into how the performance of Axon will be conducted using real networks. The performance characteristics to be examined include efficiency in terms of throughput, intra-protocol fairness, TCP friendliness, and stability. These performance characteristics and their measurement are listed and explained below. Finally we will describe the simple test bed used to gain the results that will be presented in later chapters.

4.1 Efficiency and Throughput

The efficiency of Axon is defined as the aggregate throughput of all concurrent Axon flows that may exist. Efficiency is one of the major motivations of Axon, which is supposed to utilize the high bandwidth efficiently, that is, utilize as much bandwidth as possible. Notably a single Axon flow should reach high efficiency regardless of what we should expect out of concurrent flows.

This is calculated supposing there are n Axon flows in the network and the i-th flow has an average throughout of x_i, the efficiency index is defined in Equation 4.1.

$$E = \sum_{i=1}^{n} \bar{x_i} \tag{4.1}$$

4.2 Inter-protocol Fairness

The fairness characteristic measures how fairly the concurrent Axon flows share the bandwidth. The most frequently used fairness rule is the max-min fairness, which maximizes the throughput of the poorest flows. If there is only one bottleneck in the system, then all the concurrent flows should

share the bandwidth equally according to the max-min rule. In this case, we can use a fairness index (Jain et al., 1999) to quantitatively measure the fairness characteristics of Axon.

$$F = \frac{(\sum_{i=1}^{n} \bar{x}_i)}{n \sum_{i=1}^{n} \bar{x}_i^2} \tag{4.2}$$

Equation 4.2 where n is the number of concurrent flows and x_i is the average throughput of the i-th flow. F should always be less than or equal to 1. A larger value of F means a better rating of fairness. $F = 1$ should be considered the best which means all flows have equivalent throughput.

4.3 TCP Friendliness

TCP friendliness is a more obscure measurement than the others, because it is almost impossible for a protocol with different control algorithms to reach the same performance as TCP when used on the same link. It is not reasonable to limit the throughput of a new protocol in a high bandwidth product wireless environment for the sake of the throughput of TCP. Such an approach is pessimistic and, as a consequence, inefficient.

We consider the TCP friendliness separately in different situations, which are related to two factors: the network bandwidth delay product and the TCP flow lifetime. When consider the low bandwidth delay product environments where TCP may utilize the bandwidth efficiently, we should expect that Axon to share the bandwidth with TCP both equally and fairly. In high bandwidth delay product networks however we should expect Axon to make use of the bandwidth that TCP fails to use where TCP cannot efficiently use the bandwidth.

TCP's behaviour may though be very different for long-lived bulk flows and short-lived flows where we must take into account the impact of TCP slow start at the beginning of a connection. We therefore consider the situation of short-lived TCP separately because a majority of TCP traffic over the Internet are short-lived flows as seen with general web traffic.

For a TCP flow we may suppose there are n_{Axon} Axon and n_{TCP} TCP flows coexisting in the network. When using the same network configuration, we should start n and m flows separately. The average throughput for the i-th TCP flow in each run is $\bar{x}_i$ and $\bar{y}_i$,respectively. The definition the TCP friendliness index is shown in Equation 4.3. Here the denominator is the fair share of TCP. Here $T = 1$ is the ideal value we should apply to friendliness; $T > 1$ means Axon is too friendly; and $T < 1$ means Axon overruns TCP. For short-lived flows, we should compare the aggregate throughput of a large number of small TCP flows under different numbers of background bulk Axon flows.

$$T = \frac{\frac{1}{n}\sum_{i=1}^{n}\bar{x_i}}{\frac{1}{m+n}\sum_{i=1}^{m+n}\bar{y_i}} \tag{4.3}$$

4.4 Stability (Oscillations)

We use the term of "stability" in this chapter to describe the oscillation characteristic of a data flow. A smooth flow is regarded as desirable behaviour for most applications where it often leads to better throughput. This is a different concept from the meaning of "stable" in control theory where this required the convergence to a unique equilibrium from any start point.

To measure oscillations, we have to consider the average throughput in each unit time interval as a sample. We use standard deviation of the sample values of the throughput of each flow to express its oscillation in Equation 4.4 where n is the number of concurrent flows, m is the number of throughput samples for each flow, $x_i(j)$ is the jth sample value of flow i, and x_i is the average throughput of flow i.

$$S = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{\bar{x_i}}\sqrt{\frac{1}{m-1}\sum_{j=1}^{m}(x_i(j-\bar{x_i})^2)}\right) \tag{4.4}$$

4.5 CPU Usage

CPU usage is usually measured in terms of a percentage of use and is important to consider where this indicates the efficiency of the implementation. However we sometimes need to consider the data throughput when comparing CPU utilisations where the CPU may be used to process different sizes of data per a given unit time. We should also take note of the fact a percentage of CPU usage does not form a generic measurement. That is, these values are only comparable against those values obtained on the same system, or at least systems with the same configuration. This bring onto the subject of test beds where we need a uniform hardware configuration that will be used to determine the performance of the Axon protocol once it is implemented.

4.6 Test bed

As this forms a first attempt on our part to solve some of the problems shared between managed Wireless and Wireless Ad-Hoc networks we keep the test bed very simple. This is necessary where any tests conducted will form a "dry run" of the design and implementation of the neurologically

inspired Axon protocol. This allows us to determine any of the more fundamental problems that may exist and possible be overlooked. It would be arguably counter-productive if we focused too much on the small issues that exist in the complex Ad-Hoc networking scenarios than the large issues that exist in the simpler scenarios. Typically if we can not solve such large issues the small issues become very trivial and this limits usefulness of the protocol.

In keeping with this idea all Axon tests featured in this dissertation are preformed on a elementary 802.11g Ad-Hoc link between two nodes. The nodes will be formed of two Samsung NC10s running the 2.6.33 Linux Kernel in a real world network environment. By a real world environment we mean one where other Wireless Networks exist and we may observe periodic traffic. This is important where if we test in a “clean room” where no other wireless network exist we limit the change of observing how Axon reacts in some very common situations observed on real world wireless networks. These crucially includes, among others, bit errors and radio interference which typically results in packet loss.

5

Implementation

5.1 Software Architecture

Figure 5.1 depicts the Axon software architecture. The Axon layer has five function components: the API module, the sender, the receiver, the listener, and the UDP channel, as well as four data components: sender's protocol buffer, receiver's protocol buffer, sender's loss list, and receiver's loss list.

Because Axon is bi-directional, all Axon entities have the same structure. Noticeably the sender and receiver in Figure 5.1 have the same relationship as that in Figure 3.5. The solid line represents the data flow and the dashed line represents the control flow. The dotted line specially represents congestion control. The blocks outlined in a dashed line form the four data components whereas the solid lined blocks are function components.

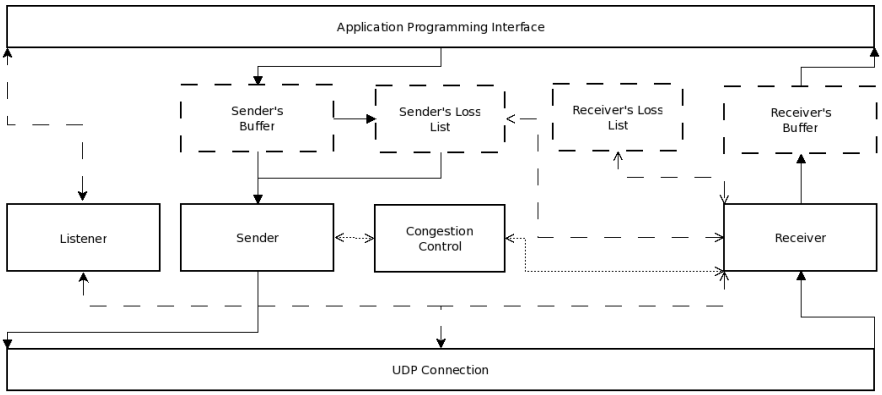


Figure 5.1: Diagram of the Axon Protocol Architecture

The Application Programming Interface (API) module is responsible for interacting with applications that wish to link with it. The data to be sent is passed to the sender's buffer and sent out by the sender into the UDP channel. At the other side of the connection, the receiver of the same architecture reads data from the UDP channel into the receiver's buffer, reorders the data, and checks packet losses. Applications can then read the received data from the receiver's buffer when and if required.

The receiver also processes any control information received during the time a connection has been open. The receiver should then cause an update the loss list at both the sender and receiver end when a loss report is received. Crucially certain control events will trigger the receiver to update the congestion control module that controls the sending of packets where this enables us to react any change in the network situation.

5.2 Even Distribution of Processing

One of the most common problems of lossy, relatively high-speed data transfer is that the generation of control information and application data reading at the receiver side can take a considerably long time when compared to the high packet arrival rate. Poor implementation of any transport protocol can potentially cause frequent packet drops, time outs, and even packet loss avalanches where the processing of loss report may cause even more packet loss. In a Wireless Networking environment where packet loss is already an issue this is something we should make every effort aim to avoid.

A very good example comes from the Linux implementation of SACK TCP in kernel 2.4.18 (Leith and Clifford, 2005). In this case the TCP implementation used a linked list to record unacknowledged packets, which is scanned upon receiving SACK notifications. In high bandwidth product networks this list is can become so long that the scanning can cause unnecessary time outs. In Axon however we should not dismiss the accomplishments of SACK but rather learn from it's shortfalls. As the protocol only uses explicit loss reports and this requires the receiver to maintain a loss list to record the loss information. In the worse case accessing the loss list may take such a long time that the arriving packets overflow the UDP buffer. This is potentially a very real issue where due to the small packets used in Axon, hopefully matching the transmission unit at the MAC layer, may contain up to tens of thousands of packets in a given lossy link.

To handle this type of problem we have attempted to evenly distribute the processing into small pieces in the implementation even if this leads to higher aggregate processing time. This is an acceptable trade off where processing resources are relatively inexpensive at the time of writing.

We will therefore attempt to describe how Axon manages the information of “in-flight” packets and handles the necessary memory copy when applications attempt to receiving packets.

5.3 Loss Information Management

Lost packets are generally represented as holes in a sliding window using some form of bit array. However as we have already considered in wireless networks where a high bandwidth product should be expected this window may become very large and may take a considerable amount of time to scan. Adding to this issue is the fact that the number of lost packets during congestion can also be very large, especially where packets are small. Notably to report such a loss will take several packets. Finally we must also take into account the fact that, the insert, delete, and query operations to the loss storage need substantial time in a simple array or list data structure.

The noticeable perplexity here is great but the reference implementation includes a simple solution. If we consider the fact that loss is often continuous during high congestion, we may use two values to represent a loss event, instead of using all the sequence numbers. For example, if packets from sequence number 100 to 200 are lost, the pair of (100, 200) could be used to record the loss, rather than inefficiently using a hundred and one numbers.

Furthermore, with each loss event, loss information is stored as one entity. The main access operations are therefore required to split and combine the entities. The practical scanning complexity is much smaller than that of scanning a regular array or list, because there are much fewer loss events than lost packets. Meanwhile, each access takes a similar amount of time.

The loss information carried in the loss report is compressed as follows where the flag bit of the first 32-bit packet header has been reused: If the flag bit of a sequence number is 1, then all the numbers from the current one to the next one are lost; otherwise, the sequence number itself is a lost sequence number. For example, in the following segment of a loss list where we represent the loss information using hexadecimal values: $0x00000001$, $0x80000006$, $0x0000000F$, $0x00000024$ the lost sequence numbers are 1, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 36.

The loss list in Axon is a static list as show in Figure 5.2 where each node should contain two values: the start and the end sequence numbers. This means that all numbers between and including these two numbers should be considered as lost. If there is only one single loss, the end number is -1 the location of a node would therefore equal to the position of the head node plus the distance between the start numbers of the two nodes. The list is logically circular where it should act as a single, fixed-size buffer, connected end-to-end.

	Head			Tail
Link	4	...	...	-1
Start	1	...	...	10
End	3	...	...	-1

Figure 5.2: The figure shows a loss list with loss 1, 2, 3, and 10. Each node on the list has a start value and an end value. The list uses a static list data structure

The major operations on the loss list are insert, delete, and query. Here, in Algorithm 17, we only explain how an insert is done to this data structure as the other two related algorithms dealing delete and query are derived from this. Theoretically, the complexity of this algorithm is $O(n)$, where n is the number of nodes in the data structure, and the time is mostly consumed through the need to search the prior node. However, according to the locality phenomenon, most searches can be finished in several steps around the near neighbours, so in practice it should prove to be quite fast. As you should expect delete and query operations have the same complexity as insert.

Algorithm 17 - Insert new loss sequence of n (start, end) to loss list L

if L is empty **then**
  insert (start, end) at position 0
  break
end if
Find the position of n.start in L (i). If L is empty, insert (start, end) at node 0; break.
Find the position of n.start in L (i) and the offset from the list head ($offset$);
if $offset < 0$ **then**
  insert (start, end) at i where i becomes new head of L
end if
if $offset > 0$ **then**
  if L[i].start = n.start AND n.end > L[i].end **then**
    modify L[i].end to n.end
  else if if prior node p and n overlaps or are continuous **then**
    search the prior node p
  else if n.end > L[p].end **then**
    modify L[p].end to n.end
  else
    Insert n at i.
  end if
end if
if the new modified node overlap **then**
  Combine them
end if
if $offset = 0$ AND n.end > L[head].end **then**
  modify L[head].end to n.end
end if

In Axon, the operations in the sender and the receiver are slightly different from the above algorithm because more information can be used to simplify them. For example, at the receiver side,

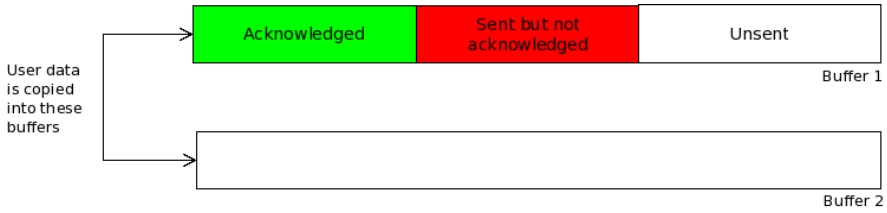


Figure 5.3: The sender's buffer as a list of both protocol allocated buffers

Figure 5.4: The receiver's buffer as a protocol buffer

insert only happens at the end of the list. It is also import to note that with this loss information storage, no other bit array or map is needed for data reliability.

5.4 Memory Copy Avoidance

The buffer management modules are responsible for temporally storing the outgoing or incoming data.

The sender's buffer shown in Figure 5.4 is a list of outgoing data blocks. When a application sends data, it should then be copied into a newly allocated memory block and linked on the list in the order of the data sends. After all the data in a block is sent and acknowledged, it should then be removed from the list and released. Insertion of the new block always occurs at the tail of the list and removal always happens at the head of the list. Note that since the buffer blocks are allocated dynamically, the sender's buffer is economical with memory.

Critically the Axon sender reads and sends out data in the order that the buffers are linked on the list. There are three regions in the sender's buffer in an attempt to keep the implementation reasonably organised. These are acknowledged data, sent but unacknowledged data, and unsent data.

The receiver's buffer shown in Figure 5.4 consists of a list containing user (application allocated) buffers and a block of protocol buffers. Incoming data is always written into the protocol buffer and read into the user buffer when a packet is received. The protocol buffer is dynamically resized according to usage. Again there are three regions in the receiver's buffer: the acknowledged data, the unacknowledged data, and the free buffer space

Due to the increasing amount of data being transferred in today's network due to influx of bandwidth hungry applications, copy avoidance in high performance transport protocols is becoming more critical. Here the motivations include reducing the delay, jitter, timeout, and packet loss when the CPU or the system bus is busy copying a large data block. This in addition to saving CPU time from any unnecessary copying that maybe taking place. This keeps the implementation light weight as well as effective at sending data between nodes in a wireless networking environment.

5.5 Assuring accurate Rate Control

As we saw in the design Axon, due to it being inspired by the way in which neurons communicate, rate control forms the major mechanism that ultimately manages efficiency and fairness. This is in contrast to the window control where this only plays a supportive role to specify a dynamic threshold that limits the number of unacknowledged packets. Although this looks good in theory, there is a potential hazard in the implementation that window control could become the dominating mechanism and rate control becomes impaired. This situation must be avoided in order that the protocol behaves are we expect. This may be caused when the packet-sending period decreases to below the actual packet sending time. Once this happens, the packet sending is completely controlled by the flow window and the packet-sending period may continue to drop.

Although most theoretical work evidently assumes an instant packet sending time, this assumption does not hold at high transfer rates. For example, to send a 1500-byte packet out from a 802.11g Network Interface will take in the region of 65 microseconds at 54Mb/s. This all too comparable to the packet-sending period as the transfer speed approaches the maximum in such networks. To avoid this problem we make sure that the value assigned to the current packet sending period is corrected using the real sending rate before updating the packet sending period.

5.5.1 High Precision Timer

Related to rate control is the requirement of very high precision timers are not available in most general-purpose operating systems. This becomes more of an issue when you consider that to support rate control at high speeds becoming available along with technologies such 802.11n and MIMO, the timing precision should be at least at the microsecond level.

We should take understand that very high precision timers required to be accurate to the microsecond are not available in most general-purpose operating systems. However to support rate

control at the speeds available in today's, and more importantly future, wireless networks the timing precision should be at least at the microsecond level. It is even better at the CPU frequency level where a simple implementation could the use busy waiting approach of querying CPU clock cycles using the RDTSC instruction.

The Time Stamp Counter is a 64-bit register present on all x86 processors since the Pentium and simply counts the number of ticks since reset. Although this does limit portability it provides an excellent high-resolution, low-overhead way of getting CPU timing information (Intel, 1997) and this enables us to plan the rate at which we send packets very accurately. Busy waiting, although it may consume 100 percent of the time on one CPU, may be scheduled to a lower priority so that other jobs are allowed to continue. Due to the blocking manner of UDP sending, higher speeds need less CPU time for the busy waiting implementation.

An alternative implementation could use an additional variable of burst to control the number of packets that can be sent out continuously. The sender then sleeps for a longer time, hoping that it is long enough to meet the minimum sleep interval that operating systems can provide. The Axon implementation uses the second option where the sender will sleep for the minimum time allowed by the host system after it sends out a burst of data packets. To reduce the burst size, the Axon sender will be awakened by any possible Axon events, such as the arrival of data or control packets or an application call.

5.6 Speculation of Next Packet

Another optimisation that was included in the reference implementation becomes obvious if we consider if the incoming data can be placed directly into its destination buffer position. If we receive or send data that are stored in different memory locations together the need for a temporary buffer can be eliminated and memory copy is further reduced. The key problem to achieving is how we should guess the sequence number of the next arrival packet where this will decide where we should put the incoming data.

If most packets arrive in order, speculation is easy when no loss occurs. However during periods of congestion where we assume the receiver to received retransmitted packets, it is very likely that the next incoming packet will have the lowest sequence number greater than the current one among those that have not yet been received.

Being optimistic Axon always speculates that the next packet will be the next consecutive number after the largest sequence number already received. This scheme has the advantage of computa-

tion and storage simplicity. The accuracy of the speculation is in approximately reverse proportional to the packet loss rate, because one loss can cause two speculation errors. This will occur when it a packet is lost and when the retransmission arrives. An admittedly less than ideal optimisation but the benefits should become clear as transmission across wireless networks becomes more reliable.

5.7 Threading

Three threads are started in a particular Axon entity. These are the sending thread, the receiving thread, and the listening thread. All the threads are not started at the same time where the listening thread is only started in a listening Axon entity in which the sender and the receiver are not started. A regular Axon entity will create a receiving thread at the beginning of the connection and the sending thread is only ever started after the first data is initially sent. This one area where laziness really does pay where we can save system resources since many connections are typically only used for one-way data transfer; from server to client.

An alternative and possibly simpler approach could be to use one single thread for both sending and receiving. However with two separate threads we can take advantage of today's multiple core processors that were design for concurrency. The benefits of this should become more apparent as we see the data transfer speeds inevitably increase over time.

The sender and the receiver implement the sending and the receiving algorithms detailed in Algorithms 7 and 8. The sender should only send a data packet from the sender's loss list if it is not empty or from the sender's buffer if there is user data to be sent. This is subject to congestion control though. The receiver, on the hand, should receive both data and control packets from the UDP channel in addition to also sending out control information.

The in-order data should be put into the receiver's buffer and any detected loss be recorded in the receiver's loss list. It should also update the sender's loss list when a loss report is received. Finally, the receiver thread will remain responsible for triggering congestion control events and updating control parameters as required.

The listener thread will accept connection requests and maintains the accepted sockets in the globally shared Axon descriptors queue. Axon identifies different connections by the random initial sequence number and IP address.

5.8 Related Work

Researchers also have continually worked to improve TCP in both wired and wireless networks. TCP SACK (Fall and Floyd, 1996), which is currently the most supported TCP version, uses selective acknowledgement to alleviate the TCP performance degradation from multiple continuous losses. TCP New Reno (Fall and Floyd, 1996) is another example designed to recover quickly from packet loss by using bandwidth estimation techniques on the acknowledgement packets. TCP Vegas and FAST TCP (Wei et al., 2006) use delay instead of loss as the main indication of congestion. In particular, FAST TCP provides an equation-based control algorithm designed to react to network situations more quickly and with higher stability. HighSpeed TCP (TOKUDA et al., 2003), Scalable TCP (Kelly, 2003), and BiC TCP (Xu et al., 2004) are focusing on fast probing of available bandwidth.

Improvements to TCP variants are often limited by the compatibility requirement with standard TCP where in order to communicate with original TCP we are limited to only modifying the TCP sender. Some important deficiencies there remain, particularly in fairness and automatic parameter tuning.

People in the past have been looking for application level solutions to overcome the limitations presented by TCP. One of the common solutions is to use parallel TCP connections and tune the TCP parameters, such as window size and number of flows. However, parallel TCP is inflexible because it needs to be tuned on each particular network scenario. Moreover, parallel TCP does not address fairness issues and the problems caused by packet loss on Wireless Networks

When attempting high data transfer experiences in this area have shown that implementation is critical to performance and this is a view we have taken on board when implementing Axon. Ideas take on board can be found in some of the basic implementation guidelines addressing performance. These include Application Level Framing and Integrated Layer Processing where the basic idea behind these two guidelines is to ensure efficiency by breaking down the explicit layered architecture..

Arguably the biggest problem we hope to over is that of memory copy that often costs the most in terms of CPU time for high-speed data transfer (Rodrigues et al., 1997). Solutions do however exist that focus around the avoiding of data replication between kernel space and user space. There is also literature that describes the overall implementation issues of specified transport protocols. One of more relevant examples includes a description of an implementation of a user level TCP (Thekkath et al., 1993)

5.9 Concluding Remarks

As we seen high-speed and effective data transfer brings with it many challenges for the design and implementation of a transport layer protocol. This becomes more of a concern when we make use of small 1500 byte packets where to achieve 54Mb/s data transfer speed an end host needs to process 45,000 regular sized packets per second. This means that any additional action on per packet processing could lead to a significant increase in CPU usage, whereas a bursting of CPU usage can further lead to packet loss. Moreover, on wireless links with a comparative large network latency, the number of on flight packets is also huge and requires large data storage to temporally record their information. The access to such data storages is thus also critical.

This chapter dealt with implementation issues related to the design of Axon. We also presented the details of the Axon protocol and in particular those related to high performance data transfer.

We hope that the experiences described here and the reference implementation of Axon may be useful for future work on transport protocols. Some ideas that previously appeared mainly in theory and simulations have been tested in Axon, such as the use of bandwidth estimation in transport protocols. By testing the design and any trade-offs necessary may prove to be useful in other experimental high performance network transport protocols in the future. Most importantly the Axon implementation has been designed so that the described methods of loss list processing, congestion control algorithms, and bandwidth estimation techniques can be overridden. These can even be reused where both allow a reduction in the implementation work required for any related research or development even if protocol fails to meet the requirements of a future wireless application.

6

Discussion

While the simulations, such as those conducted in applications such as NS-2 cover the majority of network situations, they do not simulate the real world perfectly. This not ideal in case of Axon where the aim of this dissertation is to produce a neurologically inspired transport layer protocol that is practical in the real world. Experiments in real world settings give us more insight into the performance Axon. Hence there is a need to determine the performance of Axon in this section through number of experiments deployed on real world wireless ad-hoc networks.

We will discuss throughput and efficiency, intra-protocol fairness, and stability in section 6.1. TCP friendliness will be covered in section 6.2. Section 6.3 examines the implementation efficiency (CPU usage) and we will examine the performance of Axon in real applications in section 6.3.

All experiments were conducted on the test bed introduced in the Materials and Methods chapter and the results were plotted from a random sample for fairness. It should be noted any results gained from the implemented protocol should only be considered preliminary where Axon fits into the description of a prototype. It is hoped that the library could be improved in time from the results gained in this chapter. This forms a realistic approach where the protocol is only a recent development and should be open for improvements.

6.1 Efficiency, Fairness and Stability

We performed two groups of experiments in different network settings to examine efficiency, intra-protocol fairness, and stability property in the Axon protocol.

In the first group of experiments, we started a single Axon flow between two wireless nodes in close proximity at a distance of 1 metre (3 feet). This should serve as a useful result set we can

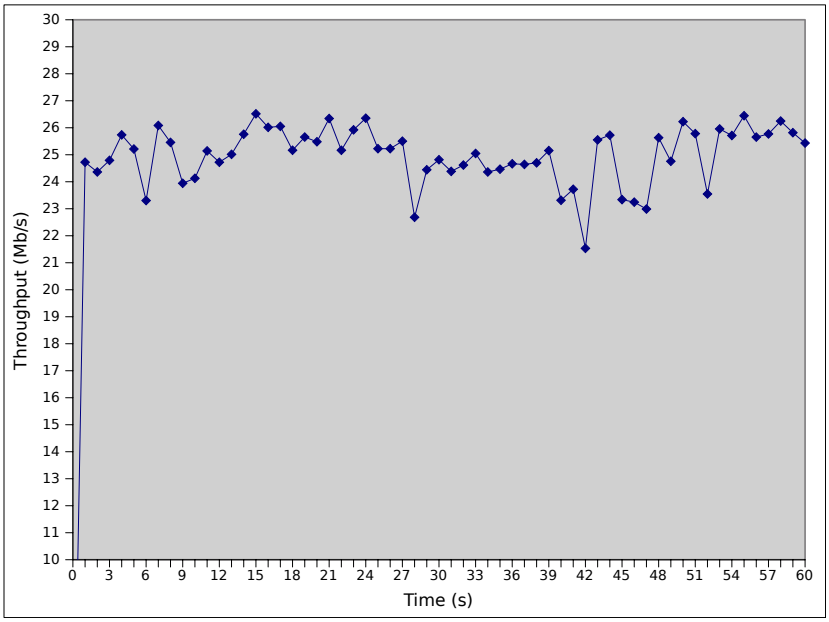


Figure 6.1: The variation of throughput in Axon over time

refer to where this shows us the behaviour of Axon on the highest quality of wireless links.

Figure 6.1 shows the throughout of the single Axon flow over each link when the one flows is started. As is apparent a single Axon flow is capable of 26.5Mb/s over a 802.11g 54Mb/s connection with the varying network latency we would expect on such a wireless connection. On closer examination of the results we can determine a good average of 24.99Mb/s. In contrast TCP reaches 25.65Mb/s as a peak value but only achieves an average of 19.51Mb/s making Axon the better performer. This may be taken as an indication that the tuning of the packet sending period, illustrated in Figure 6.1, is effective in importantly promoting throughput stability. We also see in Figure 6.1 the relation between throughput and the packet sending period. The longer the period the small the realised throughput.

For comparison the throughput of TCP in the same experimental configuration is illustrated in Figure 6.1.

Figure 6.1 shows the throughput when with the same flow configuration but the nodes were moved to limit of where 24Mb/s could be achieved. 54Mb/s was not possible in this case due to way that 802.11g applies throughput steps in the event of radio interference. This experiment demonstrates the unbiased nature of Axon flows across a wireless link in terms of network latency. The flow measured a peak throughput approaching 12.14Mb/s as shown in Figure 6.1.

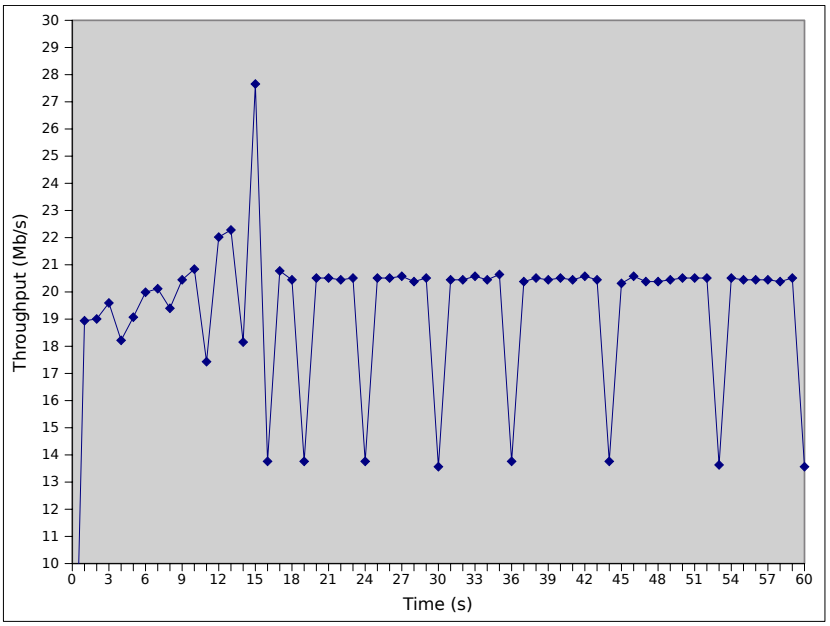


Figure 6.2: The variation of throughput in TCP over time

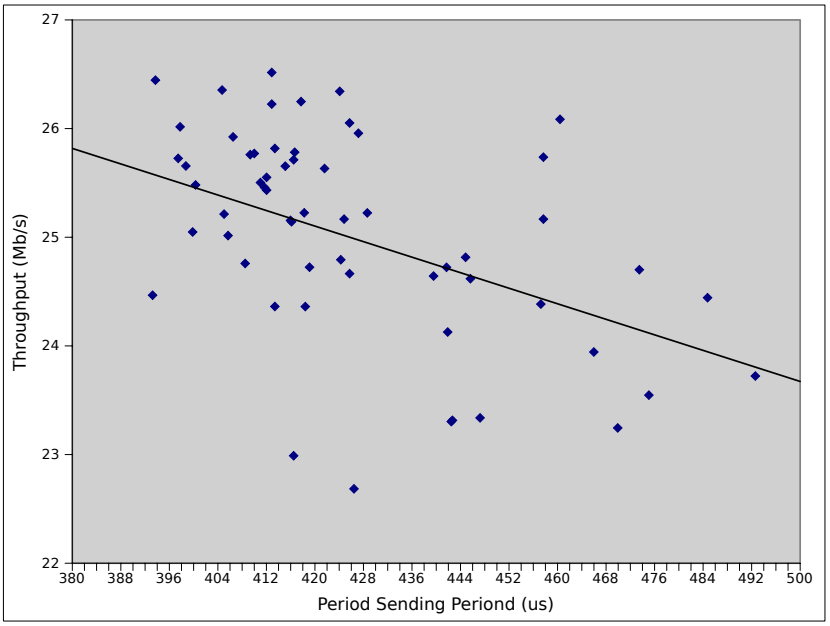


Figure 6.3: The tuning of the packet sending rate in Axon

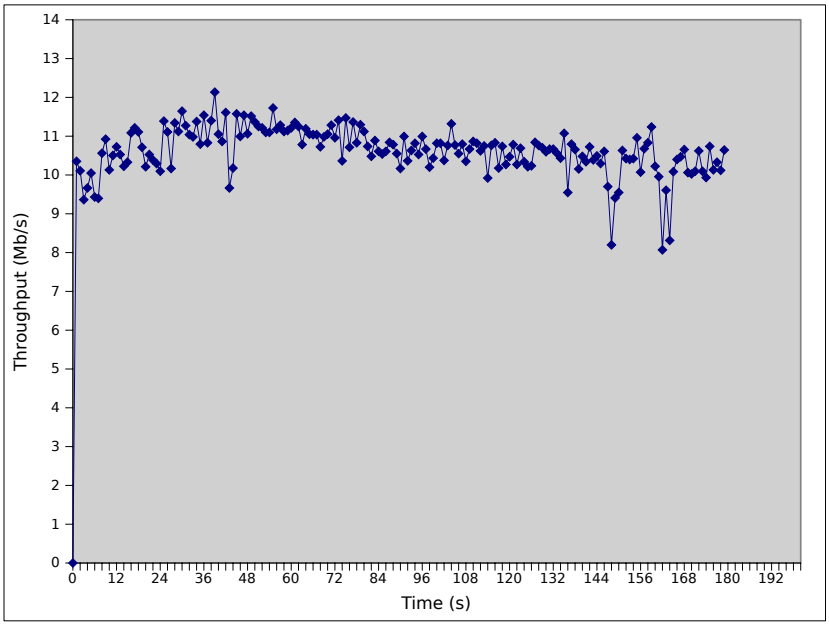


Figure 6.4: The variation of throughput in Axon over time in a high delay environment

This should be considered with Figure 6.1 where we can clearly see Axon does not exhibit bias toward shorter round trip times as we would expect with TCP. Although we are limited to a narrow range of network latencies further evidence we see nothing conclusive to lead us to believe that Axon favours a shorted round trip as is the case with TCP.

To check the efficiency, fairness, and stability performance of concurrent Axon flows we set up another experiment using the same simple network configuration as described. The mean network latency observed between the two sites was in the region of 25ms and the nodes remained in a static in terms of movement.

For the a given pairs of nodes five Axon flow was started, one after each other, every 60 seconds. These were then stopped in the reverse order every 60 seconds. The detailed performance of each flow and the aggregate throughput in this scenario is shown in Figure 6.1. Similarly table 6.1 lists the average throughput of each flow, the average network latency and loss rate at each stage, the efficiency index, the fairness index, and the stability index.

Importantly all stages achieve good and fair bandwidth utilization. As we have seen previously the maximum possible bandwidth is about 26.5Mb/s on the link as measured in the case of a single flow. The fairness among concurrent Axon flows is virtually 1 as shown in Figure 6.1. Furthermore the stability index values was, crucially, very small as illustrated in Figure 6.1, which means the

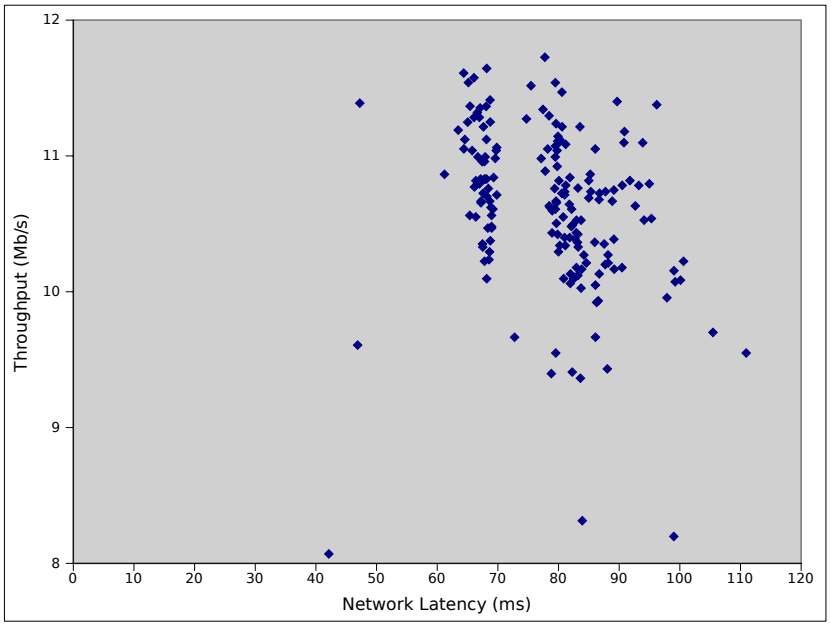


Figure 6.5: The weak correlation between network latency and throughput

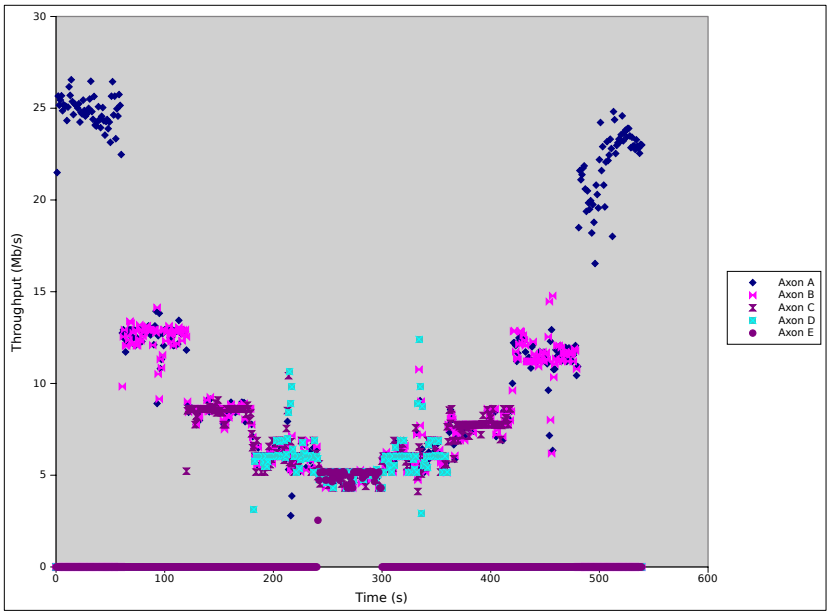


Figure 6.6: The fair and stable sharing of throughput between Axon flows over time

Time (s)	Flow A	Flow B	Flow C	Flow D	Flow E	Average
0	0	0	0	0	0	0
60	0.8923	0	0	0	0	0.89230
120	0.7045	0.812	0	0	0	0.75803
180	0.357	0.359	0.519	0	0	0.4115
240	0.6778	0.4563	0.809	1.0453	0	0.74710
300	0.2871	0.3331	0.3236	0.316	0.427	0.33728
360	0.629	0.918	0.613	1.2633	0	0.85576
420	0.4879	0.455	0.4976	0	0	0.48022
480	1.0634	1.17	0	0	0	1.11671
540	2.3301	0	0	0	0	2.33008

Table 6.1: TThe fairness of multiple Axon flows

sending rate is very stable crucially meaning fewer oscillations. Again these seem to show that the tuning of the packet sending rate is effective where the data for Flow A is plotted in Figure 6.1. Once more we also see again no real conclusive evidence of a round trip time bias in this scenario seen in Figure 6.1.

6.2 TCP Friendliness

When designing a new transport layer protocol we should always consider the short lived TCP flows that make up the web traffic and certain control messages which comprises a substantial part of data traffic on real world networks. To examine the TCP friendliness property against such TCP flows, we set up a TCP connection that transfers data from one node to another as a distance of 1 metre (3 feet). In addition a similar single bandwidth consuming Axon flows was initiated as background traffic as an attempt to determine the behaviour of Axon during a period of congestion. The results gained through preliminary testing are summarised in Figure 6.2.

From this there no denying that TCP does not under-perform significantly in the case of a simple 54Mb/s 802.11g Ad Hoc wireless link capacity with a overall low average network latency in the region of 25ms. The TCP flow utilized significantly more bandwidth than the Axon flow and this is further highlighted in Figure 6.2. This could be a result of an overly large buffer size in the implementation of flow control and shortfalls in the way we estimate bandwidth. Both factors do seem to cause the protocol to be uncompetitive but very quick to react in the instance of congestion where the TCP flow is designed to push the link to and, in the process, beyond capacity. Importantly

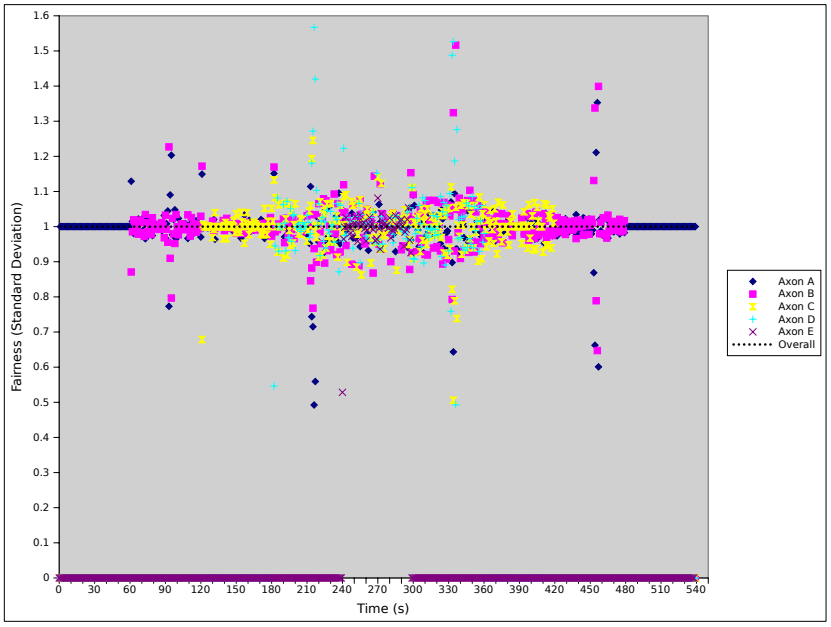


Figure 6.7: The fairness of concurrent Axon flows

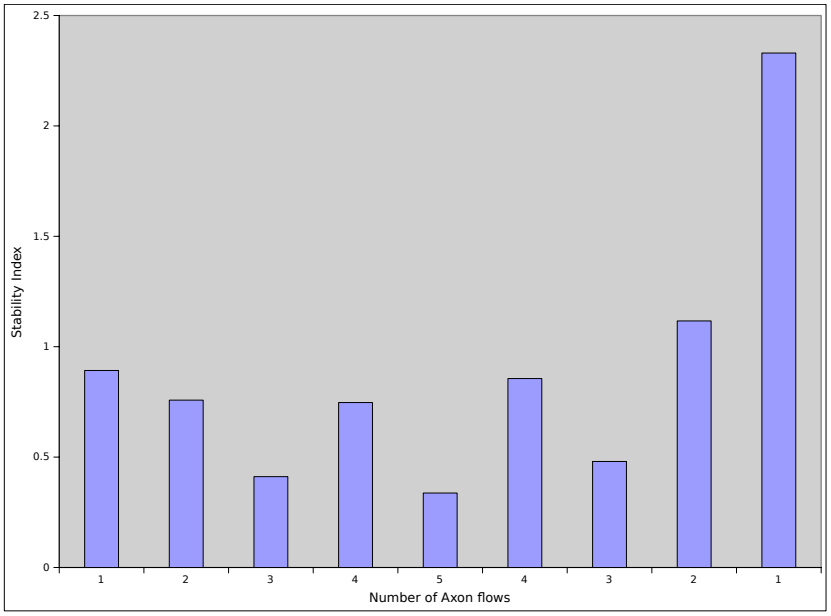


Figure 6.8: The stability of concurrent Axon flows

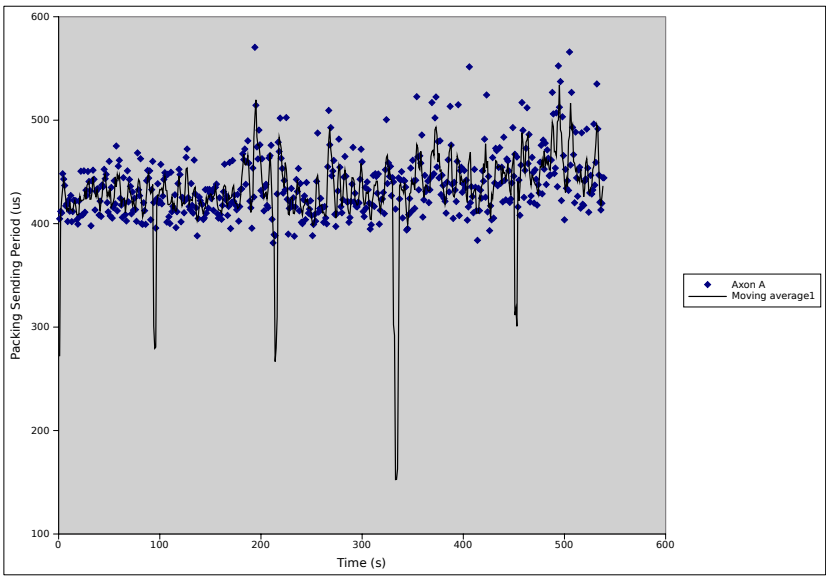


Figure 6.9: The tuning of the Packet Sending Period in concurrent Axon flows

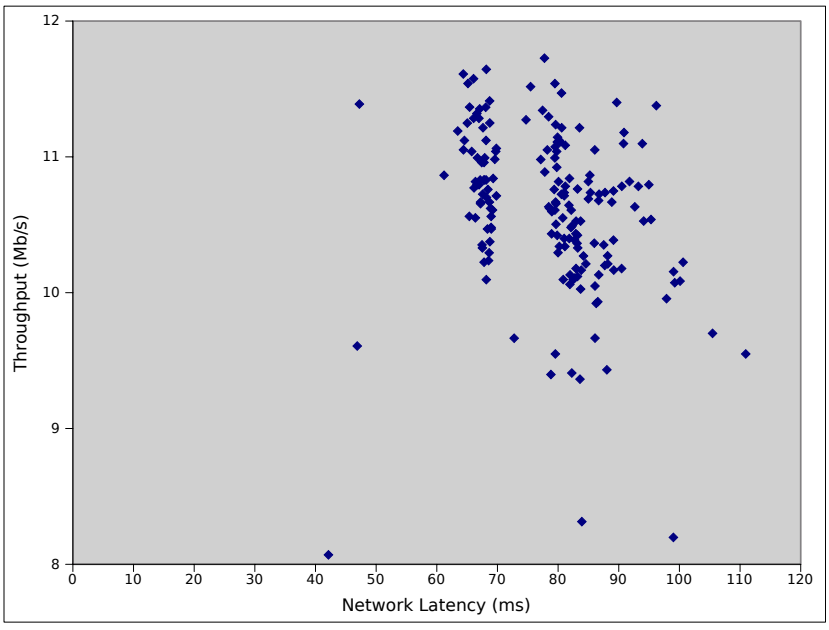


Figure 6.10: The weak correlation between Network Latency and Throughput in concurrent Axon flows

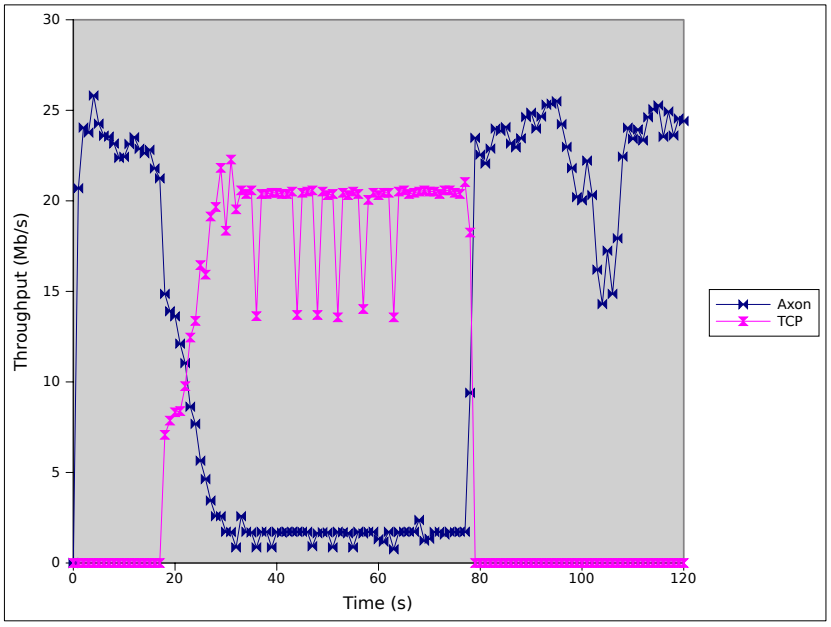


Figure 6.11: The throughput of concurrent Axon and TCP flows

though this does show Axon to be not overly aggressive despite the MIMD nature of it's congestion control algorithm.

6.3 Implementation Efficiency

In this sub-section we examine Axon's implementation efficiency through the CPU usage. This is important due to the method in which have employed to time the sending of packets by - a derivative of busy waiting. CPU usage is also crucial when you consider the efficiency of TCP. The efficiency of TCP is this respect is likely to be difficult to beat where you consider that the implementation of the protocol is found in the kernel space rather than at the user level. Moreover we should consider the perspective of the developer where they are unlikely to use a library that make inefficient use of system resources regardless of any technical advantage.

Figure 6.3 shows the CPU utilization of a single Axon flow and TCP flow, sending and receiving for the purposes of a bulk data transfer. As is apparently the CPU utilization of Axon is slightly higher than that of TCP. Axon averaged 7.5% when sending and 6.5% when receiving). TCP is notably a fraction of this but when we consider that Axon is implemented at the user level and performance available in modern computers this difference in efficiency should be acceptable. This

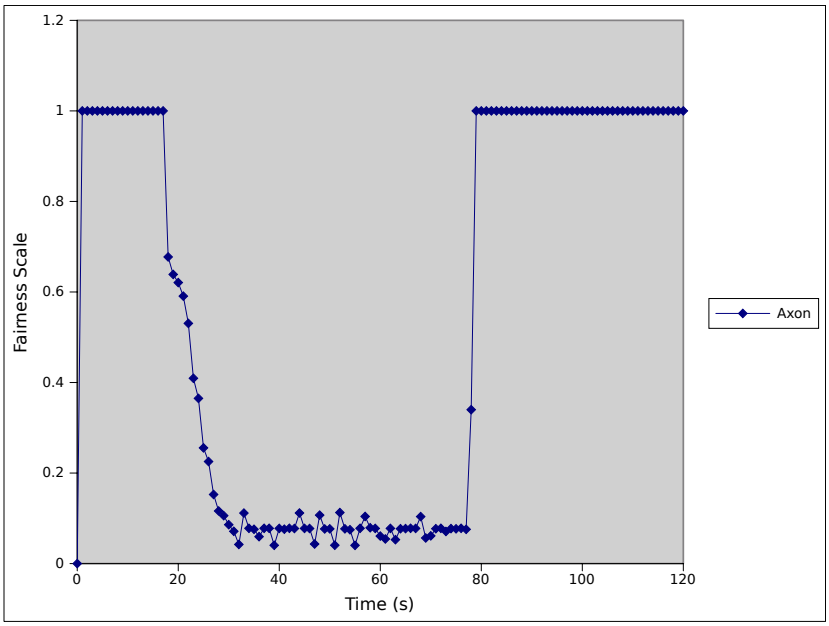


Figure 6.12: The fairness of concurrent Axon and TCP flows

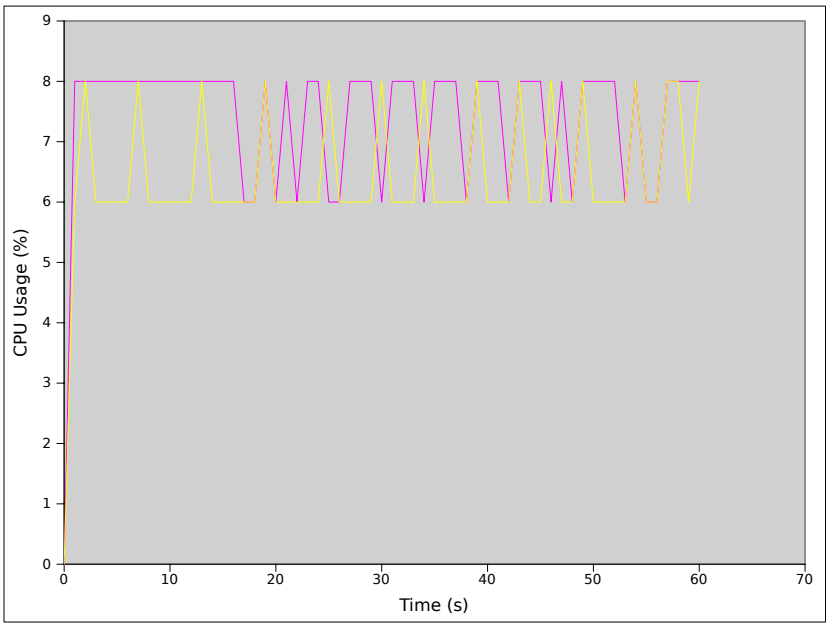


Figure 6.13: CPU usage of the reference Axon Implementation

maybe even considered an achievement when we consider the method we have employed to time packets by accurately.

This data was obtained using the Linux System Monitor and should be considered as the average for the modules of buffer management, loss processing, congestion/flow control, and UDP I/O that are called from the sending and receiving threads. We should therefore consider that these CPU times are overlapped in Axon where we have both a sending thread for sending packets and a receiving thread for receiving. Other aspects that should be contemplated are the measures undertaken in the implementation that were designed to ensure that the sender and receiver threads should consume the most CPU time. In contrast the UDP I/O time should only be that consumed in the Axon layer. This would include function calls to UDP socket and necessary preparation of the calls but excluding the data I/O time inside the Linux kernel.

Moreover the congestion control module, the necessary timing and performance monitoring to make the neurological inspired rate control possible are the most likely candidates of the most major CPU consumers. This forms one definitive area that should be marked for improved should the reference implementation be revised.

6.4 Performance in Real World Applications

While we have demonstrated Axon's performance with both extensive simulations and experiments, the real goal of Axon, however, is to benefit real world applications with high performance data transfer support over wireless links. Therefore, it is very important to examine Axon's performance in real applications. We went about this is by reproducing a scenario that is very familiar to even the most inexperienced of users; the transfer of a file from one networked computer to another. This was done using the testing configuration used to measure the throughput of the single axon link as seen in Section 6.1.

Figure 6.4 therefore shows the average throughput when both sending and receiving a file off over a lossy wireless link. This experience in the region of 2% packet loss, a average network latency of 25ms. Again this compares well with TCP where on sending 1GB (1024 MB) file took it over 2 minutes (123.76 seconds) longer to complete. TCP also limped behind Axon in terms of throughput where average through was measured at significantly below the 25Mb/s provided by Axon. This is likely to have been partially caused by the greater throughput stability offered by the neurologically inspired congestion control mechanism featured in the Axon protocol.

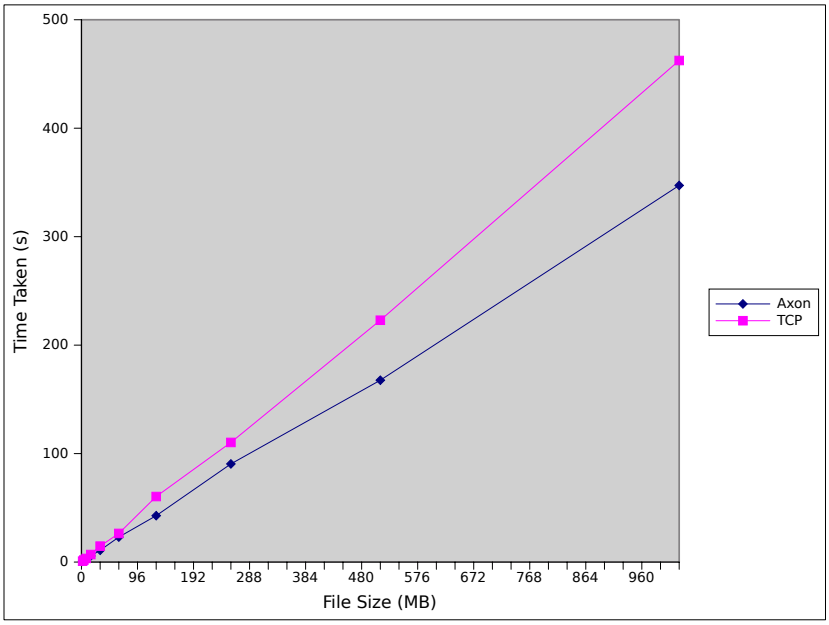


Figure 6.14: Real world performance of Axon in file transfer

Although not conclusive this does prove that Axon could be potentially used in many of today's applications, such as file transfer, remote data replication, distributed data mining, and distributed file servers in both centralised and ad-hoc Wireless Networks.

6.5 Concluding Remarks

In this section we have detailed the several experiments that have been conducted to preliminary examine Axon's performance and any areas for potential improvement. These experimental studies demonstrate in a quantitative way that Axon is efficient, fair, stable, and friendly to TCP. They also show that our Axon implementation is efficient and correct over the wireless link it was designed to accommodate.

In particular we have used experiments to ensure that protocol design of Axon and the related control algorithms in a controlled network environment are both correct and effective. This was quite apparent where we focused on the effect of network latency, number of concurrent flows and the issue of sharing bandwidth with TCP. Although we have unfortunately neither the time or hardware to gain definite results it was only the behaviour of Axon when a bandwidth consuming TCP flow was introduced that let the side down. Although some result sets were not ideals we may have hoped for initally we have seen Axon to be fairer, more stable and have a better efficiency

than TCP. This is both in instances of individual or multiple flows over a wireless link. Even though we must admit have fallen short where that the protocol is far too friendly (fair) to TCP flows this is but one area for improvement. Most importantly this we have demonstrated arguably the biggest benefit in using Axon in a future applications; the ability to react quickly to congestion and changes to available bandwidth but not to transmission errors.

However, the real world experiment plays a more important role in the Axon project. The fundamental objective of Axon is to develop a transport protocol that can be used in real applications that need to send data efficiently over high-speed wireless linls, and the best way to examine this performance is to use it in real applications and in large scale ad hoc networks. This forms a step that should perhaps be taken in the near future.

7

Conclusions

This dissertation addresses the solution to bridge high-speed wireless ad hoc networks and the many data intensive applications we see in present day. Today's emerging high performance applications requires large amounts of data to be transferred at relative high speed among geographically distributed machines. However, although the maturity of wireless network technology can provide this ability, the current data transport protocols on today's networks simply fail to effectively utilize the network resources in this situation.

In response we have presented Axon, a high performance data transport protocol to meet these requirements. We divided the problem into two orthogonal parts where both were areas we could draw inspiration from what is observed in neurons:

- Protocol design and implementation
- Congestion control.

The Axon protocol appears to work well for fast data transfer where we aimed to introduce minimum overheads to the network and the end host. We also designed the Axon congestion control algorithm such that the Axon data traffic can utilize the bandwidth efficiently and fairly. Axon's native congestion algorithm integrates a specific class of rate based MIMD congestion algorithm, bandwidth estimation technique, and several packet loss processing schemes all modelled from the inner process that occur in a Chemical Synapse. As we have seen in the preceding chapter experimental studies have been conducted to examine the performance and the outcome is initially promising.

This final chapter begins with a summary of the contributions of this dissertation, followed by an analysis of some of the manifesting limitations as well as the guidelines for future research work. Finally we offer some final remarks on the Axon project as a whole.

7.1 Contributions

In the following we attempt to highlight the contributions and successes of this dissertation.

7.1.1 A High Performance Data Transport Protocol and a Implementation of

Axon provides a timely and practical solution to the problem of transferring bulk data in high-speed wireless ad hoc networks. The Axon protocol is both efficient and fair for effective data transfer in such high bandwidth delay product network environments.

Our work systematically investigated the design and implementation issues of a such high performance data transport protocol at the application level. The Axon project identified the overhead arising from acknowledgements, loss processing, threading, and memory copy, and we have suggested appropriate solutions that have been modelled in the reference implementation. Although these could have easily been neglected, protocol design and implementation have prove these to be a significant impact on efficiency.

Axon is also easily deployable where it will work, in theory, on platform that supports UDP. This is important as there are only four versions of TCP that have been widely deployed in the past three decades because of the long time lag of standardization, implementation, and deployment of kernel space protocols. Although there were numerous TCP variants proposed at the same time as Axon was developed, these protocols are not expected to be deployed in the near future due to the development climate.

Finally, as a result of our research and development work, we developed an reference Axon library for both research and application use. The Axon library is near the productivity phase and should have the potential to be used in many real world applications.

7.2 An Efficient and Fair Congestion Control Algorithm

We summarized a class of MIMD-based control algorithms named DIID, whose increase parameter is decreasing as the sending rate increases and decrease parameter is increasing as the sending rate decreases. We have shown that DIID may be fair and stable, and can be efficient as well given the

proper tuning. Notably Axon's control algorithm is a specific case of DIID where is it related to the mathematical model applied to Action Potentials that are sent from neuron to neuron.

We believe that this form of rate based congestion control algorithm addresses both the issues of efficiency and fairness in transport layer protocols. This is apparent where the algorithm takes a near constant time to converge to what available bandwidth maybe. Axon flows are also fair to each other, even if they we are in the frequent situation where we have a varying network latency over a wireless link. While Axon is highly efficient though that is not evidence that it is necessarily aggressive. This is seen in our experiments where it is friendly to concurrent TCP flows.

Other more practical benefits have been gained through the implementation of Axon. The bandwidth estimation techniques used in the congestion control mechanism means that there is no longer a need for the manual tuning of the control parameters necessary in some TCP varients. Also, it is evident that the Axon has arguably solved solve the loss synchronization problem using a random decreasing method. Finally and arguably most importantly we have demonstrated that Axon can also handle that limited non-congestion packet losses that are observed wireless links.

Axon is quite unique amount transport layer protocols where it attempts to use bandwidth estimation techniques in determining control parameters; a technique that has origins where neurons "fire" at varying rates depending on stimulus. Despite there is quite a gulf between Neuroscience and Computer Science the rationale that the increase of the sending rate should be proportional to the available bandwidth is a sound. We see this in XCP where this uses bandwidth utilization information obtained from the intermediate routers. However since end-to-end protocols cannot get explicit information from networking hardware such as routers this estimation technique does come across as an attractive choice.

7.3 Limitations and Future Research Direction

Here we discuss both the apparent and realised limitations of the Axon protocol.

7.3.1 Bandwidth Estimation in the Context of Transport Protocol

Current research all too often has hinted that end-to-end transport protocols such as TCP usually suffer from the lack of explicit network information; especially in the case of a wireless ad hoc environment where the situation is very fluid. Looking again at the example of XCP we have seen how such information can significantly promote the efficiency of congestion control protocols.

However, a different neurologically inspired approach is to estimate this information at the end hosts is now available in the bandwidth estimation techniques used in Axon.

Axon's approach is effective in data intensive applications where there are a small number of concurrent flows, but it is not suitable in a high concurrency environment, because the bandwidth estimation technique currently used in Axon may overestimate the available bandwidth in high concurrency environments. Once again this is forms another area for improvement.

We should not however disregard the fact that future work can further investigate bandwidth estimation techniques in the context of transport protocols. A bandwidth estimation scheme within transport protocols may even have greater advantages than using regular bandwidth estimation tools but the evidence is not clear as yet.

7.3.2 Implementation Optimization

Profiling the Axon implementation shows that there is still space to improve, in particular, the CPU usage. A efficient implementation is especially important to user space protocol stacks because they cost more CPU time on memory copies and context switches. We believe therefore that further investigation of the efficient implementation of Axon will also benefit many other transport protocol implementations that may be conducted at the user level.

7.3.3 Tuning and Configuration

The useful side effect of developing Axon in C++, an object oriented language, is that is does have limited support for the implementation of alternative and innovative congestion control mechanisms. A more advanced version could though enable many more extensions to network protocols, similar to what network simulators such as NS-2 offer us, but working on real IP networks.

A particularly interesting extension as regards wireless link could be issues the data reliability and timeliness control, which often has different requirements depending on applications. For example, multimedia applications may only require a "best effort" delivery service where data quickly becomes useless once it is out of date. Furthermore, more complex extensions could potentially make the Axon an example of gateway software for protocols that focus on, amongst others, gateway algorithms and for overlay networks.

8

Concluding Remarks

Although throughput and network latency bias has been the major concern in the development of Axon much of the protocols acheivements can be summerised through the concept of Scalability. This is quite an achievement when we consider this was not factored into the design. Scalability has been one of the major research problems of the Internet community ever since the emergence of the internet. The insufficient number of IP addresses may be the most commonly known scalability problem but in many wireless networks researchers have also found that as a network's bandwidth-delay product increases TCP, the major Internet data transport protocol, does not scale well either. This is a worry when we consider the mass potential growth of wireless internet connection in future over 4G candidates such as LTE and Wi-Max.

As an effective, timely, and practical solution to this scalability problem, we designed and implemented a protocol that can utilize abundant bandwidth efficiently and fairly in distributed data intensive applications inspired from the mode of transmission between neurons.

Axon's approach appears highly scalable. Given that there is enough CPU power, Axon can support up to unlimited bandwidth within terrestrial areas. The timer-based selective acknowledgment generates a constant number of Acknowledgements no matter how fast or slow the data transfer rate is. The congestion control algorithm and the bandwidth estimation technique allow Axon to increase to very near the available bandwidth no matter how limited it is. Finally and most importantly the constant rate control interval removes the impact of a varying network latency; a issues that adversely affect the performance of TCP over Wireless Ad Hoc Networks.

We have done extensive perliminary experimental studies to verify Axon's performance characteristics. Axon can utilize wireless bandwidth very efficiently and fairly. The intra-protocol fairness is maintained even between flows regardless of any difference in network latency. This is very im-

portant for many applications where a stable throughput makes the the performance of the network predictable.

To benefit a broader set of network developers and researchers, Axon has been implemented in a manner so to accommodate alternative congestion control algorithms if required.

In the short term though Axon forms a practical solution to the data transfer problem in the emerging data intensive applications that our being used more and more on wireless links. In the long term, because of the long time lag in deployment of in-kernel protocols but the fast speed with which new applications are emerging, Axon could prove be a very useful tool in both application development and network research.

References

A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of TCP pacing. In *IEEE INFOCOM*, volume 3, pages 1157–1165, 2000. 58

D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11 b mesh network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 121–132. ACM, 2004. 50

M. Allman. On the generation and use of TCP acknowledgments. *ACM SIGCOMM Computer Communication Review*, 28(5):4–21, 1998. 13

A.V. Bakre and BR Badrinath. Implementation and performance evaluation of indirect TCP. *IEEE Transactions on Computers*, 46(3):260–278, 1997. 4

H. Balakrishnan, S. Seshan, and R.H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks*, 1(4):469–481, 1995. 4, 34

H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking (TON)*, 5 (6):756–769, 1997. 4, 16

D. Barman and I. Matta. Effectiveness of loss labeling in improving TCP performance in wired/wireless networks. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings*, pages 2–11, 2002. 11

I. Bouazizi. ARA-the ant-colony based routing algorithm for MANETs. In *Proceedings of the 2002 International Conference on Parallel Processing Workshops*, page 79. IEEE Computer Society, 2002. 26, 27

T. Bu, Y. Liu, and D. Towsley. On the TCP-Friendliness of VoIP Traffic. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, 2006. 15

F. et. al. Cali. IEEE 802.11 protocol: design and performance evaluation of anadaptive backoff mechanism. *IEEE Selected areas in communications*, 18(9):1774–1786, 2000. 10

K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *ACM SIGCOMM Computer Communication Review*, 26(3):21, 1996. 11, 12, 79

S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 7(4):458–472, 1999. 10

B. Ford. Unmanaged Internet Protocol: taming the edge network management crisis. *ACM SIGCOMM Computer Communication Review*, 34(1):98, 2004. 47

E.W. Fulp and D.S. Reeves. Bandwidth provisioning and pricing for networks with multiple classes of service. *Computer Networks*, 46(1):41–52, 2004. 2

J. Gao and N.S.V. Rao. TCP AIMD dynamics over Internet connections. *IEEE Communications Letters*, 9(1):4–6, 2005. 11

S. Garg and M. Kappes. An experimental study of throughput for UDP and VoIP traffic in IEEE 802.11 b networks. *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003*, 3: 1748–1753, 2003. 4

S. Ha, I. Rhee, and L. Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008. 11, 14

E.L. Hahne. Round-robin scheduling for max-min fairness in data networks. *IEEE Journal on Selected Areas in communications*, 9(7), 1991. 9

T.R. Henderson, E. Sahouria, S. McCanne, and R.H. Katz. On improving the fairness of TCP congestion avoidance. *GLOBECOM NEW YORK*, 1:539–544, 1998. 34

G. Holland and N. Vaidya. Analysis of TCP performance over mobile ad hoc networks. *Wireless Networks*, 8(2/3):275–288, 2002. 12

Intel. Using the RDTSC Instruction for Performance Monitoring. Technical report, Intel Coorporation, 1997. 77

V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review*, 25(1):187, 1995. 14

R. Jain, A. Durresi, and G. Babic. Throughput fairness index: an explanation. In *ATM Forum Contribution 99*, volume 45, 1999. 68

D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 89–102, 2002. 15

T. Kelly. Scalable TCP: Improving performance in highspeed wide area networks. *ACM SIGCOMM Computer Communication Review*, 33(2):91, 2003. 17, 19, 79

C. Koch and T. Poggio. Biophysics of computation: Neurons, synapses, and membranes. *Synaptic function*, pages 637–697, 1987. 29

E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion control without reliability. *ACM SIGCOMM Computer Communication Review*, 36(4):38, 2006. 9, 25

W. Lehr and L.W. McKnight. Wireless internet access: 3G vs. WiFi? *Telecommunications Policy*, 27 (5-6):351–370, 2003. 1

DJ Leith and P. Clifford. TCP Fairness in 802.11 WLANs. In *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, pages 649–654, 2005. 72

T.D.C. Little and D. Venkatesh. Prospects for interactive video-on-demand. *IEEE multimedia*, 1(3): 14, 1994. 1

M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, 1997. 13

D.R. McNeal. Analysis of a model for excitation of myelinated nerve. *IEEE Transactions on Biomedical Engineering*, pages 329–337, 1976. 46

S.K.X.L. Min and D. Dai Loguinov. Packet-pair bandwidth estimation: Stochastic analysis of a single congested node. In *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004*, pages 316–325, 2004. 63

J. Mo, R.J. La, V. Anantharam, and J. Walrand. Analysis and comparison of TCP Reno and Vegas. In *IEEE INFOCOM*, volume 3, pages 1556–1563, 1999. 11

G. Motwani and K. Gopinath. Evaluation of advanced TCP stacks in the iSCSI environment using simulation model. In *Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST 2005)*, pages 210–217, 2005. 2

D.C. Mowery and T. Simcoe. Is the Internet a US invention?–an economic and technological history of computer networking. *Research Policy*, 31(8-9):1369–1387, 2002. 11

M. Nabeshima. Performance Evaluation of MulTCP in High-Speed Wide Area Networks. *IEICE Transactions on Communications*, pages 392–396, 2005. 22, 23

K. Nahm, A. Helmy, and C.C. Jay Kuo. TCP over multihop 802.11 networks: issues and performance enhancement. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, page 287, 2005. 3

R.H. Osborne. Insect neurotransmission: neurotransmitters and their receptors. *Pharmacology and Therapeutics*, 69(2):117–142, 1996. 29

J. Padhye, V. Firoiu, D.F. Towsley, and J.F. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking (ToN)*, 8(2):133–145, 2000. 15

S.H. Rodrigues, T.E. Anderson, and D.E. Culler. High-performance local area communication with fast sockets. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, page 20, 1997. 79

S. Ruthfield. The Internet's history and development. *Crossroads*, 2(1):2–4, 1995. 6

H. Samueli. The Broadband Revolution. *IEEE Micro*, 20(2):26, 2000. 1

P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A reliable transport protocol for wireless wide-area networks. *Wireless Networks*, 8(2):301–316, 2002. 20

R. Stewart and C. Metz. SCTP: New transport protocol for TCP/IP. *IEEE Internet Computing*, pages 64–69, 2001. 9

C.A. Thekkath, T.D. Nguyen, E. Moy, and E.D. Lazowska. Implementing network protocols at user level. *IEEE/ACM Transactions on Networking (TON)*, 1(5):565, 1993. 79

K. TOKUDA, GO HASEGAWA, and M. MURATA. Performance Analysis of HighSpeed TCP and its Improvement for High Throughput and Fairness against TCP Reno Connections. *IEIC Technical Report (Institute of Electronics, Information and Communication Engineers)*, 102(694):213–218, 2003. 17, 18, 79

M.J.K. Van Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, 1988. 11

M. Vojnovic, J.Y. Le Boudec, and C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times. In *IEEE INFOCOM*, volume 3, pages 1303–1312, 2000. 16

D.X. Wei, C. Jin, S.H. Low, and S. Hegde. FAST TCP: motivation, architecture, algorithms, performance. *IEEE/ACM Transactions on Networking (ToN)*, 14(6):1246–1259, 2006. 21, 79

L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (BIC) for fast long-distance networks. In *IEEE INFOCOM*, volume 4, pages 2514–2524, 2004. 11, 13, 79

Y.R. Yang, M.S. Kim, and S.S. Lam. Transient behaviors of TCP-friendly congestion control protocols. *Computer Networks*, 41(2):193–210, 2003. 10

R. Yavatkar and N. Bhagawat. Improving end-to-end performance of TCP over mobile internetworks. In *Mobile Computing Systems and Applications, 1994. WMCSA'08. First Workshop on*, pages 146–152, 1994. 4

Y. Zhang and T.R. Henderson. An implementation and experimental study of the explicit control protocol (XCP). In *IEEE INFOCOM*, volume 2, page 1037, 2005. 24, 25