OPEN PROTOCOL DESIGN

BOH CHEK LIANG DOMINIC

NATIONAL UNIVERSITY OF SINGAPORE

2003

Founded 1905

OPEN PROTOCOL DESIGN

BY

BOH CHEK LIANG DOMINIC

(B.eng.(Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2003

ACKNOWLEDGEMENTS

I would like to acknowledge my supervisor, Associate Professor Guan Sheng-Uei for his advice and support in my research work. I would also like to thank the National University of Singapore for the research scholarship grant. In addition, my gratitude extends to Ms. Rose Seah, the lab supervisor for her prompt help in equipment and facilities. Finally, I thank God and my loved ones for their support during the past two years.

SUMMARY

This thesis presents an intelligent protocol, Genetic Algorithm Transport Protocol (GATP) based on Genetic Algorithms (GAs), which evolve and adapt to the network environment to achieve a best effort user-configurable Quality of Service (QoS). Surveys on current competitor's work on protocol engineering for configurability, adaptability, and QoS networking are done. However, the greatest feature of GATP is the amalgamation of all the features of configurability, adaptability and best effort QoS orientation combined together. Work also encompasses the study on how low-level packet flow control can similarly achieve best effort QoS. The networking environment is modeled as an evolutionary playground for data packets, which evolve using a fitness level of QoS achievement. The different QoS criteria in jitter, error rate, throughput and round trip time provided multiple objectives from GATP. Different fitness functions of weighted, single objectives, and finally multi-objectives are applied to understand the network problem. Experiments provide performance analysis of GAs in an actual network environment. The solutions obtained from the different fitness functions, exemplifies the dynamic problem area of networking, where best solutions for QoS are changing according to network environment. Experiments also show how GATP is able to achieve best effort QoS compared with Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Although, GATP may lack the efficiency in code compared to TCP and UDP, it possesses potential through virtues of its sensitivity to network environment and fast solution. The nature of networking on the Internet is dynamic and even unpredictable at times and will be better served by such a protocol in GATP. This paper surveys the possible techniques used in Multi-Objective Genetic Algorithm (MOGA) to solve a similar problem in dynamic landscaping. Using such a technique, GATP can likewise enhance the networking performance, to provide solution to this dynamic landscaping cum multiple QoS problem area. An experiment to show the possible benefit of such a measure is studied. A controlled network experiment is also done to demonstrate the effectiveness of GATP to restore QoS in a controlled changing landscape. An additional study of the overheads of GATP is done. This includes various Automatic Repeat Requests (ARQs) Algorithms, which are modified for GATP usage. The efficiency of each ARQ incorporated into GATP, is computed and discussed. This thesis also shows that using less than maximum packetization feature in packet size, it allows GATP to achieve better overall QoS. A greater understanding into the possibility of deploying such a protocol on varying scales is achieved.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Open Protocol

This thesis proposes an open protocol, which as suggested by the word “open”, a willingness to accept new ideas. In this case, a networking protocol is proposed to allow feedback on the performance of various protocol configurations and to use genetic algorithms to produce new configurations. This protocol serves 3 purposes being quality of service, configurability and adaptability which will be discussed in this section.

1.2 Quality of Service

Through greater applications of continuous media (CM) application, there is a demand for meeting QoS requirements instead of simply delivering data of the highest quality. QoS assures that data not just transmitted but to also conforms to a certain standard of networking required for different applications as discussed in [1-2]. QoS is of utmost importance without which, the applications are useless. Two main approaches by the IETF are through integrated services (IntServ) [3-4] with the resource reservation protocol (RSVP) and the differentiated services (DiffServ) [5-7]. Resource reservation and prioritisation are the two main methods of QoS assurances.

Resource reservation and prioritisation are the two main methods of QoS assurances using monitoring on the server and appropriate reactive or pre-emptive measures. Their approach locks up resources and requires changes to routers and networks. A minimum invasion of current systems to provide easily deployable multiple QoS needs to be explored. Best effort protocols like UDP and TCP monopolise the resources at any given opportunity. A low QoS requirement should not aim to get the best that the network can offer. On the contrary, it should only get what it needs, such that the other systems may have a better resource availability. For a real time streaming system with low expectation of voice data integrity, a decrease in voice data would offer a good compromise for a reasonable QoS. Perhaps by modification of certain packetization characteristics or transmission trait, a better QoS is achieved in a congested environment.

An open networking environment presents constant changes and unpredictable situations, contrary to a closed computing environment with unique solutions [8]. Internet traffic is bursty and random, therefore networking should ideally explore a larger solution space and provide intelligent solutions to scale with this environment. Maintenance and achievement of QoS in such an open environment, are important for networking to service applications successfully. This is the greatest challenge for this thesis.

1.3 Configurable and Adaptable Networking Protocol

Protocol configurability is the ability to customise a different set of working protocols while protocol adaptability is the ability of a protocol to respond effectively to changes.

1.3.1 Configurability

Currently, data communication by TCP/IP over the Internet is a default for general uses. Protocols like TCP/IP, UDP and RTP prevent users from specifying and receiving the exact quality of service it requires of during networking. A nondiscriminated best effort in error-free throughput approach in TCP is an inflexible solution to users. Network video streaming, file download and telephony all require different QoS. For example, high data integrity is important in file transfer but telephony requires short round trip time. MPEG video also has different data rates, where a much higher data rate is incurred in fast action scenes and vice versa. As new applications may require different QoS at different stages of networking, specification of QoS should be on user end rather than protocol, for maximum relevance in ability of protocol to satisfy users' purpose in networking.

1.3.2 Adaptability

Network environment is not always static and bandwidth may not be consistent or predictable. A previous set of protocols may not be relevant in a different environment, and therefore protocol adaptation to the environment should be actively pursued to ensure that QoS is adhered to and the purpose of networking is achieved. TCP/IP currently avoids congestion with windowing technique, slow start algorithm and MTU discovery. However, TCP only ensures maximum error-free throughput in networking, while other aspects of QoS in jitter and round trip delay are neglected. In addition, TCP could only change the transmission window for throughput manipulation of networking as an adaptation method. This is a limited measure as opposed to GATP's method of manipulating multiple packet parameters. Window size in TCP is change stepwise to discover the best throughput whereas GATP uses GA to derive and test for the best solutions. This thesis offers a finer grained solution in customised QoS suite of round trip time, jitter, error rate and throughput. The performance feedback from multiple QoS criteria, allows networking to adapt and achieve satisfaction of multiple QoS in open environment, by making a greater effort through changes in finer grains of protocol in data integrity, retries limit, packet size and interpacket length. These changes at the low packet levels allow changes in all the QoS criteria, not just the throughput and error rate.

From the above sections 1.3.1 and 1.3.2, TCP's approach to networking was contrasted to GATP. The main purpose of TCP was mentioned to be networking with the best-effort reasonable throughput with full sequenced data integrity. This may be adequate for some applications like web browsing, but for applications with greater need for multiple QoS like video conferencing, more can be done. By adopting the approach of GATP, multiple QoS may be achieved better. The finer grain approach of GATP also allows a greater change to be exacted by the sender to control the results of networking.

The paper will first discuss the related research by others on improve the networking protocol through configurability or adaptability. Subsequently, the design goals and implementation details of Genetic Algorithm Transport Protocol (GATP) will be discussed. Treating the network domain as a problem area for GA, the solution methodology of using weighted fitness, single fitness and finally multi objectiveness shall explore. Experiments conducted using various schemes provide a very effective means of studying the workings of genetic algorithm in this specific problem domain as well as the effectiveness of GATP. This also answers how similar problem spaces with dynamic solutions and good overall population fitness can be tackled.

1.4 Networking landscape as Evolutionary Background

The playground of genetic evolution is found naturally in all habitats and biological systems. The processes of selection, mutation, crossover and survival all exist to find the best-fitted individuals for the systems. Computer networking where data packets are sent across the physical network can be an evolutionary playground. Data packets are individuals or genes bearing individual traits like packet parameters, with a survival need for QoS achievement. Individuals compete in the network for resources. In TCP/IP flow control, the windowing technique changes the window size of transmission, while maintaining a maximal reasonable throughput and yet prevent buffer overflow. However, the adaptation assumes a maximal QoS criterion of only error rate and throughput, which may not necessarily be the user's choice. On the contrary, GATP evolves to all aspects of QoS according to user's QoS specification. Benefits of evolving to user's QoS specification was discussed in section 1.3.1. Buffer overflow is also discouraged through poor QoS achievement of packets with larger throughput.

1.5 Dynamic Landscaping in Networking

GATP was designed and implemented for the purpose of achieving adaptability, configurability and QoS satisfaction. However, GATP is proposed to solve a problem that's changing dynamically. In an actual networking environment [8] there are constant changes and unpredictable situations, quite contrary to a close computing environment with unique solutions. This is a challenge to networking to provide a greater dynamic solution space for scaling this environment intelligently. However, all these must take place with QoS achievement as a primary goal of solution. This is the greatest challenge for GATP. The Internet is not only dynamic, it also lacks real time predictability. The performance of routers, switches, hubs and Internet traffic may be random, bursty or fluctuating.

GATP uses the NSGA techniques of MOGA to solve the multiple objectives in QoS for networking. However, the techniques in MOGA were traditionally applied to static solution search, and may not be so effective in a dynamic landscaping environment. Greffenstein in [2] and Mark in [3-4] answered these issues of dynamic landscaping in GA. Using the shifting balance technique of dynamic landscaping, combined with the NSGA MOGA techniques, GATP was able to solve multi objectives problems in a dynamic Internet environment. The NSGA technique was shown to be more effective in GATP by using an elitist selection scheme instead of a tournament scheme, while the subcolony in the shifting balance technique produced a better result when more genes are different from the main colony. These results will be elaborated in subsequent sections.

GATP will contribute to the area of networking protocol, through the usage of the abovementioned techniques to provide a multiple objectives as well as a greater effectiveness in reacting to dynamic changes in networking environment. GATP taps into a large resource of genes, and searches for a heuristic and fast solution. However, a traditional protocol like TCP avoids congestion by the slow and progressive windowing technique. This is a stepwise reaction that attempts to slow down throughput in a stepwise fashion, and may therefore be less efficient in a dynamic changing landscape.

This report will confirm the nature of networking and the capability of GATP to return the system to QoS satisfaction in the event of dynamic changes. In addition, an overhead study of this protocol will be studied for possible scalability.

Chapter 2

Related Work

GATP is a protocol that is easy to implement and deploy with configuration and evolvability intelligence. Similar works on protocol configuration have been done by others in [11], [12], [13], [14], [15], [16], [17], [18] with limited adaptability and insufficient QoS orientation. The evolvability of GATP with intelligence from genetic algorithm provides a multiple QoS objectives orientation. In addition, Chapter 5 will show the performance of traditional weighted GA applied to a networking protocol. More advanced GA techniques employed will be discussed in Chapter 6. Some existing work on configurable and adaptive protocol will first be discussed in this chapter.

2.1 Self Modifying Protocol

Firstly, Guan & Jiang in [9-10] provides Self Modifying Protocol (SMP), which is an initial design of the engine for evolution of transport protocols. The simulation results are favorable and explore the possibility of GA to solve networking issues. However, implementation details are insufficient. This report aims to provide solutions to actual design and implementation issues, which we shall explore in an actual network environment. GATP has a focus in three main areas of adaptability, configurability and QoS orientation. The amalgamation of all three issues motivated the design and development of GATP.

GATP needs to harness a robust protocol to carry its packets and yet provide the full flexibility in configuration. IP is the best candidate due to its backward and upward compatibility from Asynchronous Transfer Mode (ATM) to Wavelength Division Multiplexing WDM). Full flexibility in sending customized packets is also possible. Redesigning of IP to take on the entire transport mechanism is also feasible. However, GATP runs on IP in this report. Even after GATP is built on IP, it runs alongside all existing technologies, and can be upgraded to optical networks, and other newer technologies. The overheads in terms of header size will be discussed in Chapter 9.

GATP is modeled into two specialized engines; transport and intelligence. These two engines will provide a framework to achieve configured protocol as well as adaptation intelligence. Its essential to have two separate and yet integrated engines for a full realization of the three motivations. This is based on the usage of object-oriented programming, to allow instantiation of networking protocol. The configurability of protocol is intact, as the full suite of transport mechanism is made available. The adaptability is strong, since the transport engine will compute the genetic evolution of the next generation protocol. This intelligence is running based on a robust genetic algorithm engine not limited to fixed congestion avoidance strategies. This differs from conventional protocols that see a tight integration of intelligence and packetization activities, like TCP, UDP and IP. Changes in conventional IP need to be made to allow for full IP control such that GA can exercise its intelligence.

GATP has innovated strongly in terms of transport engine intelligence. Firstly, is the application domain innovation, which sees dynamic GA issues being transposed to the networking domain. The issue of appropriate population size, and degree of mutation, which is although not new to dynamic GA, is an innovation in the networking domain. Chapter 9 will show the cost analysis that reveals a resulting QoS satisfaction in GATP using less than efficient packet size.

2.2 Programming Language Constructs

In [11], programming language constructs are used to support run time software adaptation. An adaptive middleware is used but with an explicit issue that the degree of adaptation could result in undesirable effects versus a greater survival in adverse conditions. A three component interface in Java was used with meta socket to create a dynamic observation and change effecting protocol. An achievement in transformation of components at run time to adapt to different dimension was made. Expert knowledge was use for the intelligence to adapt by employing forward error correction or noise detection algorithm. Java is used which may be rather sluggish and slow especially in events of rapid and frequent adaptations. Intelligence in adaptation is also limited by expert knowledge. The meta socket used makes the implementation less portable.

GATP however offers a solution based on existing IP using a conventional socket and C programming. Its immense portability in Operating systems and ease of implementation is extremely desirable. The efficiency of the C program is beneficial for rapid and frequent adaptations. Using intelligence from GA, a fast optimization can be achieved with little or no expert knowledge. Such intelligence is extremely suited for dynamic environments, which may require a solution outside of implanted existing expert knowledge.

The work on GATP also demonstrates clearly, the best performance comes from different protocol characteristic. Using QoS achievement as a basis of protocol adaptation allows a high percentage of QoS achievement. This approach vastly differs from a best effort performance. It offers a self-restrained usage of networks to only use as much resources as possible to achieve its targeted QoS.

2.3 DROPS

DROPS [12] use a configurable protocol that persists during runtime for adaptability. Benefits of adaptability to a changing network environment were mentioned in the paper. Persistent configurability and adaptability was a key issue in their work. Their work was on the Operating System (OS) and differed from GATP, that uses socket programming. Many intelligent schemes were suggested like lookup tables, Boolean logic, and Fuzzy logic. But further study into intelligence was not provided.

2.4 Configurable Transport Protocol

Configurable Transport Protocol (CTP) in [13], is a user configurable protocol, which gives users the flexibility of building up a protocol in x-kernel process level. Performance efforts are limited to best effort or simply reserving resources. CTP doesn’t discuss much on the adaptation ability.

The limitations of CTP is over-reliance on the x-kernel push-pop for interacting with upper levels as well a need for modification of socket API to support the transport properties is considerable. Interoperability is low as custom headers are required. Intelligence is also very much limited.

2.5 Adaptive Software

Adaptive software in [14], provides an adaptation framework on Cactus [13], [15] and [16]. Run time adaptation in [14] is achieved in 3 phases of change detection, agreement and adaptive action. A global system state is concluded and a consensus reached on an adaptive action. Their approach uses Component Adaptor Module (CAM) that calculates the fitness of the different algorithms and switches to the algorithm that has the best fitness. Their work focused on the gracefulness of adaptation. [14] is actually a reactive solution such that an event will trigger adaptation through theoretical calculations. The best-fit function for determining the best protocol for adaptation could be difficult. GATP actually evolves the protocol and test for its actual fitness using an evolutionary process that’s based on fitness of each gene. GATP uses an experimental fitness evolutionary method where practical solutions could remove expert or unpredicted judgment errors.

2.6 Other Protocols

Ensemble [20] may provide a framework for new protocol stacks but there is a disadvantage of a runtime disengagement of services for the new protocol to take effect.

Fuzzy control [15] was used for adaptation on the application layer. A hybrid adaptation was used where linear behavior was solved with Task Control and non-linear problems were solved with Fuzzy control. Application-specific choices can be used in Fuzzy control with a rule base.

The systems discussed lack intelligence in adaptability. Heuristic knowledge is required and at best a complex fuzzy knowledge [15] is employed. Evolving protocol is a possible candidate to offer the intelligent adaptation required.

The evolution of protocol engineering from static protocol to a runtime configurable and adaptive protocol progresses to the next stage in GATP. The full suite of intelligent adaptability, run time configurability, and QoS orientation makes GATP the next evolution of protocol.

2.7 Dynamic Landscaping in Genetic Algorithms

In Genetic Algorithms (GAs), dynamic landscape problems take on different models and require different measures. However, solving stationary problems in GA has always been the norm. Lately GA has been applied to solving dynamic landscape problems. [24-29]

Grefenstette in [25] discussed several mutation schemes and their respective performances in varying dynamic landscapes. Models of Evolvability are Fixed Mutation(FM), Genetic Mutation(GM), Fixed Hypermutation(FH), and Genetic Hypermutation(GH). In FM, all individuals have a fixed random probability of bits changed. GM however, puts the mutation rate under genetic control. The FH mandates a fixed fraction of population for random mutation while the remaining population undergoes baseline mutation or FM. The GH model has hypermutation rate under genetics control, where individual will either hypermutate or baseline mutate. Landscapes are primarily 2 types; Gradual and Abrupt. The experiments conducted by Greffenstette found that Fixed Hypermutation Strategies perform well in gradual changing landscape. GH Strategies perform well in both landscapes. Controlling of hypermutation rate genetically, allows GA to climb well even after abrupt change and as hypermutation rate decreases in stable landscape.

Mark Wineberg & Franz Oppacher proposed the technique Shifting Balance Genetic Algorithm (SBGA) in [26-27] as strategies to outperform traditional GA in difficult dynamic environment. Firstly, colonies are forced away from the core. Secondly, migrants enter core for integration and exploitation. Colonies are forced away from core using cluster analysis. Bi-Objectives are derived from following of landscape, and yet moving away from core. The Selection involves two populations using Objective fitness and distance from core. Mating restricted to within sub population is also enforced. Effective migration of colonies towards the best fitness is usually pioneered by diverse small colonies. Integration of Migrants is achieved by replacing current population with migrants and to enlarge population to cover migrants. This technique was shown to outperform traditional GA in dynamic environments.

Karez-Duleba in [22] presented the work on performance of the population using uni and bimodal fitness functions and and demonstrated that under certain conditions, the equilibrium of traits can be multi modal.

HDEA [25] reinforced the work on using GA to solve non-stationary environment through the usage of specie adaptation, species memory and microevolution within species.

GATP adopts the SBGA techniques to solve the dynamic landscaping problem in networking. This problem is also a multi-objectives problem, such that GATP shall combine the techniques of both multi-objectives GA and dynamic landscaping GA. This combinational approach is used in a networking protocol to allow a fast adaptation of the protocol to fast changing networking conditions. Traditional protocol like TCP uses a slow and cautious stepwise discovery of appropriated throughput, and its focus on multiple QoS apart from throughput, is weak. Work on configurable and adaptive network protocols may deliver in terms of configuration and adaptability. However, the solution of GATP is one of multiple QoS and the use of heuristic intelligence in GA for adaptation to a dynamic landscaping network.

Chapter 3

Genetic Algorithm Transport Protocol (GATP)

GATP is proposed as an evolutionary transport protocol that will adapt to the network environment using the intelligence from genetic algorithms (GAs). This protocol allows a customized packetization, data integrity, sequencing, and QoS specification even at run time. The evolvable characteristics include packet size, inter-packet length, throughput and retries limit. The number of evolvable parameters can be more, but only these few are used here as a prototype. However, GATP can only achieve best effort QoS according to the fitness function used. Best-effort QoS is the utilization of available resources to provide a QoS as close as possible to the predefined QoS.

Optimizing a network that may have different reasons for data loss other than congestion could be found in a general optimization algorithm like Genetic algorithm. These will allow for seamless transport across different media with different reasons for data loss. Possibly network routers could be smart enough to implement heuristic weightages into the algorithm as transition into a different medium occurs.

Intelligence is implemented through genetic algorithm. Fitness level combined with heuristic knowledge as well as pure optimization methodology enables the transport engine to determine the best configuration for the current networking needs. This ensures that heuristic knowledge that may provide solutions are complemented by the optimization of genetic algorithm. The fitness level weightage would most likely be influenced by heuristic knowledge.

3.1 Design Goals

GATP aims to achieve a thorough QoS achievement through protocol reconfiguration according to fitness function. Genetic algorithms are used to provide adaptability and configurability. The protocol shall have a configurable transport mechanism for transportation of data and this shall be controlled by an intelligence embedded in the transport engine. The server and client model is used in this work for simplicity although it can be extended to peer-to-peer, where a networking entity can be both server and client. The client will execute the evolved protocol and upload data to the server that uses GAs for protocol evolution. Intermediate routers treat GATP packets as IP packets and thus require no special reconfiguration.

1. The Transport Mechanism shall achieve configurability through controls in micro protocols shown below.

   a. Packetisation factor: Interpacket length, packet length, maximum retries limit, maximum round rime trips time, different data integrity.

   b. QoS values: Jitter, error rate, throughput and round time trips.

2. The transport engine based on genetic algorithm shall adapt the transport mechanism to network environment through fitness level monitoring. Evolutionary process can be tracked and studied.

3.2 User Level Configurable Protocol

User QoS requirements in jitter, round trip time, error rate and throughput are directed to the transport engine. Configuration is achieved by sending a packet with a preferred set of QoS values from the client to the server, using the header fields for specified throughput, specified round trip time, specified jitter, and specified error rate as shown in Section 3.3 Figure 3.3.2. Reconfiguration on server is achieved by sending a packet with configuration derived from GAs. Traditional protocols like TCP only evolve to best effort throughput and error rate and are unable to provide all rounded QoS satisfaction as opposed to GATP's adaptability to network changes and QoS achievements. The configurable networking features in GATP are packet size, Interpacket length, and retries limit, which are discussed below. Details of exact configurations are discussed later in Section 4.3.1.

3.2.1 Packet Size

The protocol shall be able to send out datagrams of different sizes according to Genetic Algorithms. To minimize header size, two bits are chosen to represent each GA parameter which has 4 predefined levels. The maximum packet size is chosen to be 1024 bytes which is a non-fragmented size for Ethernet networks shown later in Table 9.2.1. The minimum size was set at a minimum of the GATP header and IP header. This will cover the 2 possible size limits of the GATP packets.

3.2.2 Interpacket Length

Inter-packet length is a major factor contributing to the value of jitter, and it can be controlled by GAs. A few predefined levels, represented by two bits in the header are used for minimum overhead. The minimum time to send subsequent packets is immediate while the maximum is set at single-trip time. Timeout is set at twice the round trip time like TCP.

3.2.3 Maximum Retries

This will allow GATP to consider user's specification for maximum attempts at resending packets. There are a few predefined levels, which are determined by 2 bits in the header. The transport protocol will ensure that the limit is not exceeded before retransmission. Other wise, the next sequence will be transmitted.

3.2.4 Other Configurations

The configurations are not restricted to only these few. In fact, more degree and variation of configurations can be used according to requirements and header size limit. For example, are number of acknowledgements, time-out time, and transmission window size. Networking will benefit through higher security in successful transmission and faster transmission in environment of lesser congestion. Checksum ensures a level of integrity of the packet. Based on the error rate and integrity required by user, GA shall decide the types between 1's complement, 2's complement, Cyclic Redundancies Check (CRC), Fletcher 16 checksum or other choices. These simple CRCs are selected for ease of implementation. The support for different checksum types is to cater to different needs of data integrity.

3.3 Protocol Communication Overhead

GATP uses information embedded in packet headers to execute different protocol configurations. The header structures will be explained first, followed by the different configurations.

IP Header (20 Bytes)
GATP Header (40 bytes)
DATA (0 to 900 bytes)

Figure 3.3.1: Header of Typical Packet for GATP

Figure 3.3.1 above shows the total header structure of a GATP packet. The Internet Protocol (IP) header allows GATP packets to flow through the network like any other IP packets. Actual GATP header contains protocol statistic used by GATP and DATA is the actual user data transmission. The inter-packet length, packet length, retries limit contain protocol

Checksum value (8 bytes)
Request (1 byte)
Sequence Number (4 bytes)
Interpacket Length (1 bytes)
Packet Length (1 byte)
Maximum Retries Size (1 byte)
Number of Retransmission (1 byte)
Time (8 bytes)
Specified Round Time Trip (1 byte)
Specified Jitter (1 byte)
Specified Error rate (1 byte)
Specified Throughput (1 byte)
Options (11 bytes)

Figure 3.3.2: GATP Header

Figure 3.3.2 above shows the format of the GATP header. These fields affect protocol configuration. Below are the different packet configurations embedded in the GATP header. These configurations are minimal, to ensure small overhead and yet adequate information for protocol execution. Reconfiguration of packet size, inter-packet delay, and retries limit are done based on GA evolution of the configuration for QoS achievement. Request type is used to identify nature of transaction. A value of 1 will indicate a new request by client, while 2 means an acknowledgment reply and 3 means a useful data transmission. Sequence number allowed a control of ordering in transmissions. The inter-packet length, packet length, retries limit contain protocol configurations. Number of retransmission and time are useful statistics to be used by GA for computation of fitness. Specified values displayed the intended QoS of the packet. The options field in the header allow for future expansion of header data. For example, the bits for window control can be implemented in this optional header field. However, there is only implementation work based on stop and wait automatic acknowledgement request for simplicity. In Chapter 9 on studies of overhead of GATP, a further exploration of the efficiency of other types of automatic repeat request types is done. Processing of header field starts with the request field, where 1 indicates a new request, 2 indicates a reply from the server and 3 indicates an download data packet. Firstly, the client will send a packet with request set to 1. Then the server will start by sending the first download packet to the client according to the QoS specified by the client. Replies packet from client will allow server to compute the new configurations using GAs. The semantics of the header will be discussed in section 4.3.

Checksum value: The checksum value using the default checksum of Fletcher 16.

Request: This contains the type of services below.

1: Request for new data stream

2: Acknowledgement of Packet reception at end point

3: Data packet

Sequence number: Sequence of packet in the transmission stream

Interpacket Length: The intermittent time delay of the sending of packets

Packet Length: Length of packet

Maximum Retries Size: The maximum times that this particular packet can be resent.

Number of Retransmission: The current number of attempts to send this packet

Time: Time packet was sent

Specified Round Time Trip: User specified round time trip in milliseconds

Specified Jitter: User specified Jitter in milliseconds

Specified Error Rate: User specified error rate

Specified Throughput: User specified error rate in bit per seconds

3.4 Intelligent Transport Engine

Transport Engine manages the GAs to evolve the protocol to changing network environment and user needs. The GA engine used in Guan & Jiang [5] has been reconfigured for an actual implementation. Based on user specifications and packet performance encapsulated in the GATP header, the transport mechanism uses GAs to evolve subsequent transmission. A Genetic Algorithm based engine [5] is used by the transport mechanism. Firstly, the engine will initialize a population of random genes with different packet length, inter-packet length, and retries limit which will be passed to the transport mechanism for transmission. When acknowledgement packets for the population returns, computation of QoS is done. Evolution occurs where the genes are ranked according to a specified fitness function. Mutation and crossover occurs to evolve a new population, for subsequent transmissions.

3.4.1 Fitness Level

The basis of performance measure shall be a fitness function shown below [4]. This is a method in GA for combining multiple fitness objectives with its own relative importance.

$$\boxed{\begin{array}{l} \text{Fitness} = W1 * \dfrac{Q1}{\text{RTT Specified}} + W2 * \dfrac{Q2}{\text{Jitter Specified}} + \\ \qquad W3 * \dfrac{Q3}{\text{Throughput\_Specified}} + W4 * \dfrac{Q4}{\text{Error Rate\_Specified}} \\ \quad \text{where } Q1 = \text{RTT achieved-RTT Specified,} \\ \qquad Q2 = \text{Jitter Achieved-Jitter Specified,} \\ \qquad Q3 = \text{throughput achieved–Throughput Specified,} \\ \qquad Q4 = \text{Error Rate Achieved–Error Rate Specified.} \end{array}}$$

(3.4.1)

Q1 to Q4 if negative, are set to zero to favor QoS achievement towards specified levels and not beyond so that a fitness better than the specified QoS does not gain any advantages compared to the specified QoS. Fitness level thus range from best value of zero and above.

3.4.2 Jitter and Throughput

Throughput is calculated as below:

Throughput = (Packet Size/round trip time)(1.0/sqrt(p_err))* (3.4.2)

Probability of error (P_err) was taken to be the packet loss rate while packet size is the size of the packet used in GATP. The round time trip is the time taken for a packet to travel from the sender to the receiver and the acknowledgement back to the sender. Sliding window can be achieved by acknowledging a transmission window of packets. This Send and Wait implementation was chosen for simplicity.

The jitter computation follows Guan & Jiang [3] as below where $D_{i,i-1}$ is the difference in arrival rate between the most recent packet and the previous packet..

$$J_i = J_{i-1} + (|D_{i-1,i}| - J_{i-1})/16 = 15/16 * J_{i-1} + 1/16 * |D_{i-1,i}| \qquad (3.4.3)$$

Instead of the server controlling QoS degradation through preemption and remedy, the client can take on a more active role. Packetization and evolvable statistics can be conveyed to the client for appropriate data transmission. In this case, negotiation of client’s transmission to achieve the same QoS is done with a major load removed from the server. For example, a client will first request for a certain QoS configuration, and the server will send a packet to approve the request. When successful, the client will immediately download data from the server. Computation of the achievement of QoS is done on the server, which also uses GA to derive the next generation of packet configuration. This new configuration will be used to send data from the server to the client.

Chapter 4

Implementation

The GATP protocol is made up of two parts. One part is the transport mechanism to provide configurable data transmission. The other part is the transport engine that provides the intelligence and instructions to the transport mechanism.

4.1 Transport Mechanism

The implementation of the predefined levels in jitter, throughput, round-trip time, and error rate were set at 4. This was done for simplicity. Implementation was done on raw socket through the raw IP protocol for flexibility and control. Redesigning of IP for GATP is not necessary in this socket implementation as protocol execution is done by user-level programs. A connectionless approach was used without reservation of resources. Below in Figure 4.1.1 shows the pseudo code of the transport mechanism. The transport mechanism is a typical socket program with two concurrent processes; Send_data and Listener. The function Send_data sends out data according to the configurations given by transport engine, in terms of packet size, interpacket delay, and retries limit. A timer is used for interpacket delay countdown. Inter-packet delay is implemented by checking the timer to ensure that the delay is enforced between sending of consecutive packets. The function Listener, waits for incoming data, and forwards this data to another function called display for processing of incoming packets. Function display processes the data to ascertain the integrity of the packet and the request type. This information is passed to the transport engine for processing.

Pseudocode of Transport Mechanism

```
//Create Socket
if ((sockfd = socket(AF_INET, SOCK_RAW, proto->p_proto)) == -1) {
perror("socket");
exit(1);
}

//SET THE HOST ADDRESS
//  printf("sockfd after raw socket creation:%d", sockfd);
my_addr.sin_family = AF_INET;         /* host byte order */
my_addr.sin_port = htons(MYPORT);     /* short, network byte order */
my_addr.sin_addr.s_addr = INADDR_ANY; /* automatically fill with my IP */
bzero(&(my_addr.sin_zero), 8);        /* zero the rest of the struct */

//BIND THE HOST ADDRESS TO THE SOCKET

if (bind(sockfd, (struct sockaddr *)&my_addr, sizeof(struct sockaddr)) == -1){
perror("bind");
exit(1);
}

//FORK LISTENING PROCESS
fork();
listening(sockfd);

//FORK SEND_DATA PROCESS TO SEND PACKETS
fork()
Send_data(sockfd, (struct sockaddr*)&myhost_addr, myhost_addr);

Function Listener(int sockfd){
//CHECK FOR MESSAGES
recvfrom(sockfd, buf, sizeof(buf),0, addr, &len);
//if message arrives, send for processing
If(buffer!=0)
Display(buf, bytes, sockfd)
}

Function Send_Data(int sockfd, char ipaddr[14], int intersize, int pack_size, int retries_size, int
request, int actual_retries){
//Check Timer and send packet
if(timer is up){
//Format packet according to genes
sendto(sockfd, &fullheader1, sizeof(fullheader1), 0, toclientadd, sizeof(struct sockaddr));
}
}
Function Display(void *buf, int bytes, int sockfd){
//Process packet
Calculate checksum
Check Request Type
//Call transport engine for next evolution
Ga(ipaddr, chkpacket.pack_info.inter_size, chkpacket.pack_info.pack_size,
chkpacket.pack_info.retries_size, chkpacket.pack_info.actual_retries, rtt, Q1, Q2, Q3, Q4)
```

Figure 4.1.1 Pseudo Code of Transport Mechanism

4.2 Transport Engine

A GA based engine [5] shown below is used by the transport mechanism. Figure 4.1.2 below shows the pseudo code of the transport engine. Firstly, the engine will initialize a population of random genes with different packet length, inter-packet length, and retries limit which will be passed to the transport mechanism for transmission. Acknowledgment packets contain information of transactions and are sent by the client to the server upon receiving a data packet. When acknowledgement packets for the population returns to the server, computation of QoS will be done. Round-trip time is derived using timestamp of packet, error rates are tabulated from the history of packet transmissions, while fitness, jitter and throughput are derived from above Equations 3.4.1-3.4.3. The population round trip times, throughput, and fitness are obtained from the average values of all packets in the population. Evolution occurs where the genes are ranked according to fitness function. Mutation and crossover occurs to evolve a new population, for subsequent transmissions.

Pseudo Code of Transport Engine

```
GA(ipaddr, chkpacket.pack_info.inter_size, chkpacket.pack_info.pack_size,
chkpacket.pack_info.retries_size, chkpacket.pack_info.actual_retries, rtt, Q1, Q2, Q3, Q4);

//INITIALIZE A RANDOM POPULATION OF INDIVIDUALS
init(); // create random genes
for(sample1 to SampleN){
// whole population
//evaluate fitness of gene samples
calculatefit(gene);            }
if(population completed){ // check that QoS results from the entire generation has arrived
evolution(gene_pool); // evolution of fit genes to create a unique subcolony}
}
evolution(gene_pool){
Fitness-Rankin(gene_pool);// genes are ranked
Mutate(gene_pool) // mutation between fit genes
Crossover(gene_pool) //cross between fit genes
Create-new-population// new evolved population
}
```

Figure 4.1.2 Pseudo Code of Transport Engine

Actual transmission was conducted between two computers located in the Intranet. Both terminals used Linux. The server is a Pentium 4 1.2 GHz with 128 MB RAM while client is a Pentium 667MHz with 128 MB RAM. Transmission was done across the actual campus 100Mbps Ethernet LAN, to ensure that it is a typical LAN environment. Experiments to investigate the effects of GATP in an actual network environment can be studied.

This experiment aims to study the effects of GATP in an actual network environment. However, an optimum performance of GATP needs to be explored through the variation of the workings of the GATP. The parameters to be adjusted and studied are: sampling size, number of mutated genes used, and performance of GATP. Using the weighted fitness function discussed earlier, the round trip time and throughput will be assigned weights of 3 while Jitter and Error Rate has weights of 1. These experiments were conducted by allowing the user to specify the QoS requirements and subsequent transmission of 10, 000 to 20, 000 packets were monitored.

4.3 Genetic Makeup

The gene format used in packet features is presented below in Table 4.3.1. However, only the packet parameter chromosome will be embedded in the header and communicated across the network. The gene used in packet features are represented below in Table 4.3.1. The packet genotype of packet size, interpacket delay and number of maximum retries will be concatenated to form a single gene. For example, a genotype of 0 would be 000000 in binary, meaning a packet size of 64, zero interpacket delay and no retries are allowed.

Table 4.3.1: Chromosome of GATP Packet

Chromosome	Packet Size	Inter-packet Delay	Retries Limit
AABBCC	AA	BB	CC

4.3.1 Packet Level Parameters

Shown below in Table 4.3.2 are the values of packet parameters for each genotype. Packet sizes range from 64 to 964 bytes so that a good spread of size is employed which are not too large to be fragmented by routers or terminals. Interpacket delay was also set to be less than 80 ms, to ensure throughput does not become too low. Number of retries was set to less than 4 to prevent excessive delay.

Table 4.3.2: Packet Level Parameters

Genotype	Packet size/ bytes	Interpacket Delay/ ms	Number of Retries
00	64	0	0
01	364	1	1
10	664	50	2
11	964	80	3

4.3.2 QoS Parameters

Below in Table 4.3.3 is the chart of the predefined levels of QoS parameters. These few levels of possible QoS objectives are implemented for simplicity and convenience. For consistency in experiments, the specified RTT, Jitter, Error and throughput was set to genotype 10, 11, 11, 01. This set of specified QoS shall be used consistently in all experiments to ensure a fair comparison of performance, as they will all have the same QoS objectives.

Table 4.3.3: QoS Parameters

Genotype	RTT/ ms	Jitter/ ms	Error Rate/ %	Throughput/ Mbps
00	5	1.5	0.25	0.8
01	6	2	0.33	3.2
10	7	2.5	0.5	6.4
11	8	3	1	12

4.3.3. Mutation and Crossover

Although mutation rate used is 75%, it only applies to the genes made available for mutation. Crossover is based on 80% where 2-point crossover is used. Figure 4.3.1 below shows the process of evolution. Firstly, 10 genes are randomly initialized. These genes are implemented and run on the network. The performance of the genes are derived and genes are ranked. In this example, the best 7 out of 10 genes according to fitness function will survive to the next population. The last 3 genes from mutation and crossover make up the remaining population. The process of sending data into the network and evolution continues.

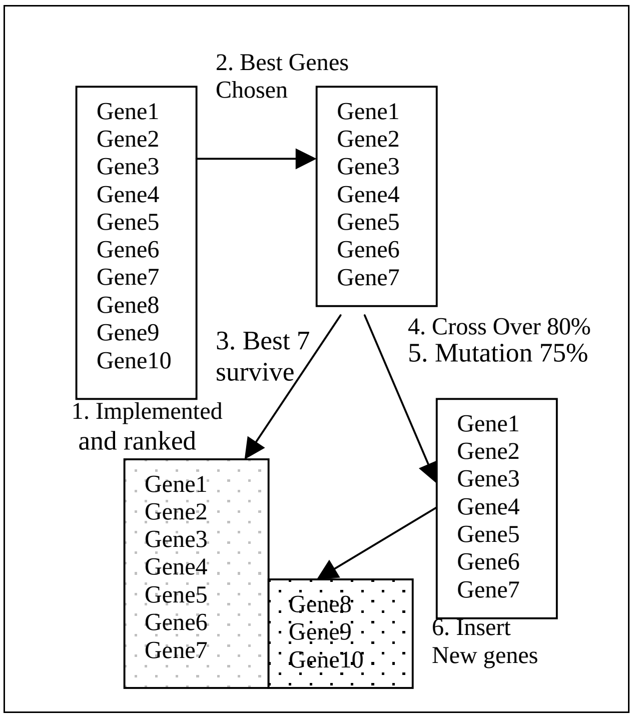
2. Best Genes Chosen
Gene1 Gene2 Gene3 Gene4 Gene5 Gene6 Gene7 Gene8 Gene9 Gene10
1. Implemented and ranked
Gene1 Gene2 Gene3 Gene4 Gene5 Gene6 Gene7
3. Best 7 survive
4. Cross Over 80%
5. Mutation 75%
Gene1 Gene2 Gene3 Gene4 Gene5 Gene6 Gene7
6. Insert New genes
Gene1 Gene2 Gene3 Gene4 Gene5 Gene6 Gene7
Gene8 Gene9 Gene10

Figure 4.3.1: Process of Selection of Next Generation of Genes

4.4 GATP Dynamic Landscape Strategies

Simple strategy is adopted using SBGA [23] principles. A main colony is created for coexisting with a sub colony of possible migrants. Migrants are created from mutation and crossover of the fittest main colony. Migrants and random genes form the subcolony. This subcolony shall initiate shifting balance [24]. Randomness and differentiation from main colony is achieved in sub colony to present migrants to effectively track dynamic landscape changes. Shifting balance for four objectives of QoS are round trip time, jitter, throughput and error rate. Introducing reproduced diverse and random genes into the subcolony, the gene pool is diversified for multiple objectives balance shifting. Integration of migrants to the main colony shall be based on fitness levels. Studies on the diversity and population of the sub colony will be discussed later.

4.5 Stopping Criteria in GATP

The stopping criterion as it is traditionally known in GAs, is a measurement of optimisation. However, for GATP to persistently adapt to the environment, there is no stopping of evolution. The determination of the stopping criteria is only for performance comparison. In Chou [22], several stopping criteria suggested were number of generations, computing time, and fitness convergence. Such an approach assumes a static environment and is unsuitable for GATP. In the approach of Jiang in [23], the stopping criterion was used for fitness evaluation. If a cycle yielded a greater or lesser score than previously, the counter is decreased or increased respectively. Achieving a certain predefined value for the counter, signifies stabilisation. The fitness score is the best fitness of the generation and the predefined stability indication counter value is 3. This training of GA must be done sufficiently but not to produce an overly specialised solution. But in the case of GATP we use the attainment of counter value as an indication of the speed of GATP in solution finding. To prevent solutions from being trapped in a local minimum, or over specialisation of the solutions, the use of new mutated genes at every generation ensures that sufficient gene pool is available This can be taken as a gauge for GATP completing its evolution temporarily till another drastic network change occurs.

For GATP, the stopping criteria can be taken as the instance that the counter reaches a predefined value. Subsequently, fitness can deteriorate when the network environment changes and GATP has to evolve again to achieve the stopping criteria. The determination of stopping criteria is when there are at least 6 genes ranked highest 7 in the same generation being repeatedly selected for the next 2 consecutive generations. This consecutive generation condition is used to ensure that if 6 genes do not produce good fitted results, it will not survive to the next generations. This stopping criterion is used primarily in experiments as a guide to investigate the effects of sampling size and mutation rates in GATP. For example, if gene 1, 4, 6, 63, 23, 43 are seen in generations 4, 5, and 6, then generation 6 has met the stopping criteria.

4.6 Congestion Avoidance Strategy in GATP.

GATP employs a rate adjustment adaptation to the network environment. This protocol monitors the fitness of the packet transmission. The changing of transmission rates is achieved by using a different interpacket delay, packet size and maximum retries limit, with the fitness level as a monitor. This fitness will undergo a survival tournament where the best gene of the generations will be retained for subsequent generations. In order for new variants to be injected into the new generation, a crossing over and mutation process will occur to allow for new genes to participate. During congestion, the fitness of genes will change and the favored genes are those that perform well in the congestion and will be promoted and retained while poor genes will die. Likewise in a more free environment, the fitness changes accordingly and evolution will produce the best performers.

Although GATP can be adapted with the fastest response to produce the fittest gene for the current network environment, this adaptability will produce a slower response in a less busy network environment. Since adaptability is introduced by increasing the search space through injection of new genes, this extra search space may be unnecessary when the network environment is less busy and less dynamic and solutions may be found faster. Therefore, to overcome network congestion, suitable adaptability is suggested. This is done by ensuring that appropriate sub population size of new genes are inserted into the subsequent generations to check if its a better solution.

Chapter 5

Weighted Fitness Experiment

This experiment will show the behavior of the transport protocol in a robust networking environment. The fitness computation in Equation 3.4.1 is used. Using a different number of reproduced genes and sampling sizes, the protocol performance is observed in different networking environments. Crossover and mutation used is as shown in Section 4.3.3. The traffic conditions for all subsequent experiments in this thesis are defined as follows. Heavy-traffic is defined as average traffic on networking being 90% of full load which is 90 Mbps. Likewise, moderate traffic is 80% and light traffic is 50%.

5.1 Number of genes employed in sub colony

The experiments were done on the intranet using a gene pool of 10 and using the GATP features as described in section 4.3 above. The fitness function used is based on Equation 3.4.1. Weightings *W1-W4* are chosen to be 3, 1, 3, 1 respectively. The main colony consists of genes which are retained for subsequent generations while sub colony consists of new genes that didn’t appear in the previous generation. The main colony when 7, has a sub colony of 3. This sub colony consists of reproduced genes. The sample size, which is the number of packets of the same genes sent into the network, is 10. Therefore each gene is sent out using 10 packets, making 1 generation of 10 genes being 100 packets. The effect of varying the number of new genes was studied. Figure 5.1.1 shows that the fastest speed was achieved in using only 1 new gene in a light network environment. The protocol is able to achieve the fastest speed of optimization when a very small number of genes are allowed to undergo mutation.

For a busy network environment as shown in the Figure 5.1.2, the greatest speed of optimization was towards a larger number of permissible mutant genes.

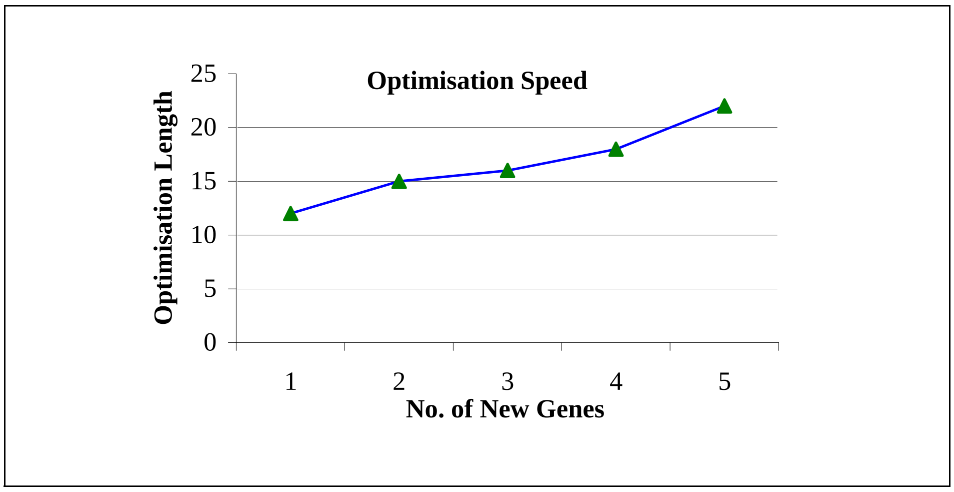

Figure 5.1.1: Number of Generations to Satisfy Stopping Criteria in a Light-Traffic Network Environment

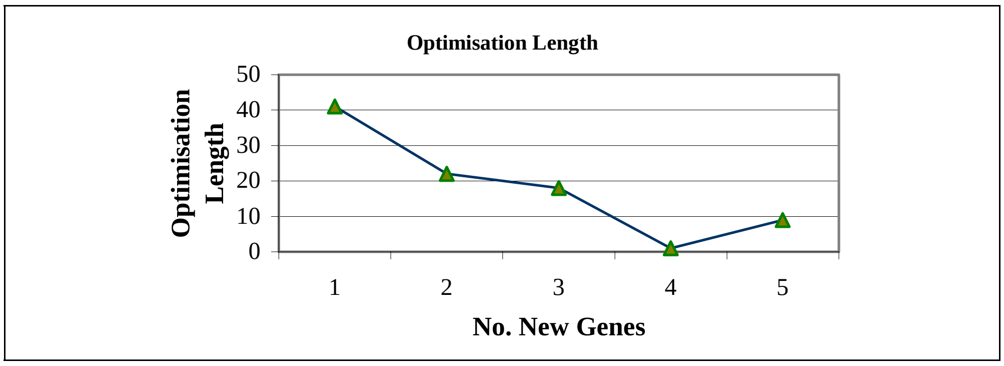

Figure 5.1.2: Number of Generations to Satisfy Stopping Criteria in a Heavier-Traffic Network Environment

5.2 Sampling Size

The experiments were done on an intranet using a gene pool of 10 and using the GATP features as described in section 4.3 above. The fitness function used is based on weighted fitness function in Equation 3.4.1. Weighting *W1*-*W4* are chosen to be 3, 1, 3, 1 respectively. Thus if the main colony is 7, then the sub colony is 3. The experiments were conducted by sending each gene into the network using *n* packets. Therefore each generation uses 10*n* packets. The effect of varying the sampling sizes, n are studied in the experiments below. Performance is done in terms of optimization speed. The environments were in both light and heavy network environment for better contrast.

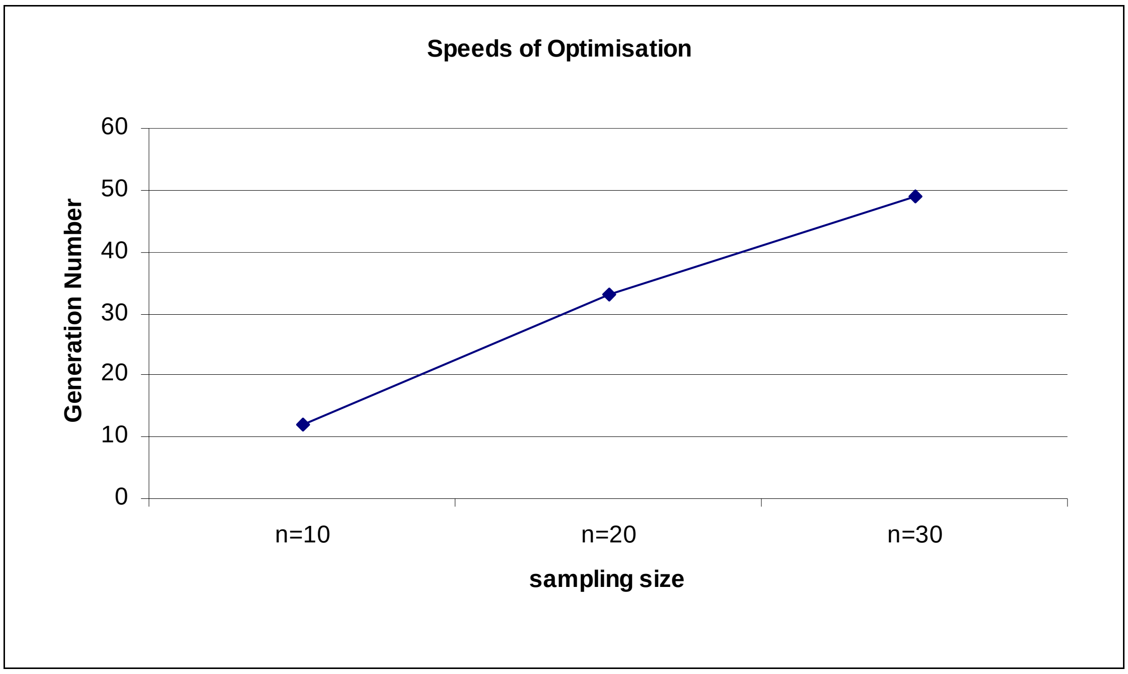

Figure 5.2.1: Number of Generations to Satisfy Stopping Criteria in Varying Sampling Sizes in Light Network Environment

Figure 5.2.1 above shows the performance of GATP in a relatively light network condition. The fastest speed is achieved by using a sampling size 10 where only 12 generations is required to achieve optimisation. Only in a very congested environment can there be benefits in using sampling size smaller than 10 as shown in Figure 5.2.2 below. Likewise, only in a very light congestion environment, benefits are seen in using a sampling size greater than 10.

To better confirm the impact of sampling sizes, a finer calibration of the experiment was done and depicted in figures 5.2.2 and 5.2.3. Figure 5.2.3 shows that in a very light networking environment, sampling size of 18 actually performed the best while figure 5.2.2 shows that in a heavy network environment a smaller sampling size of 10 or below yielded a better performance. Through a smaller sampling size, a greater reaction to the network environment is enabled and thus a better adaptability and hence subsequent optimization of GATP is achieved.

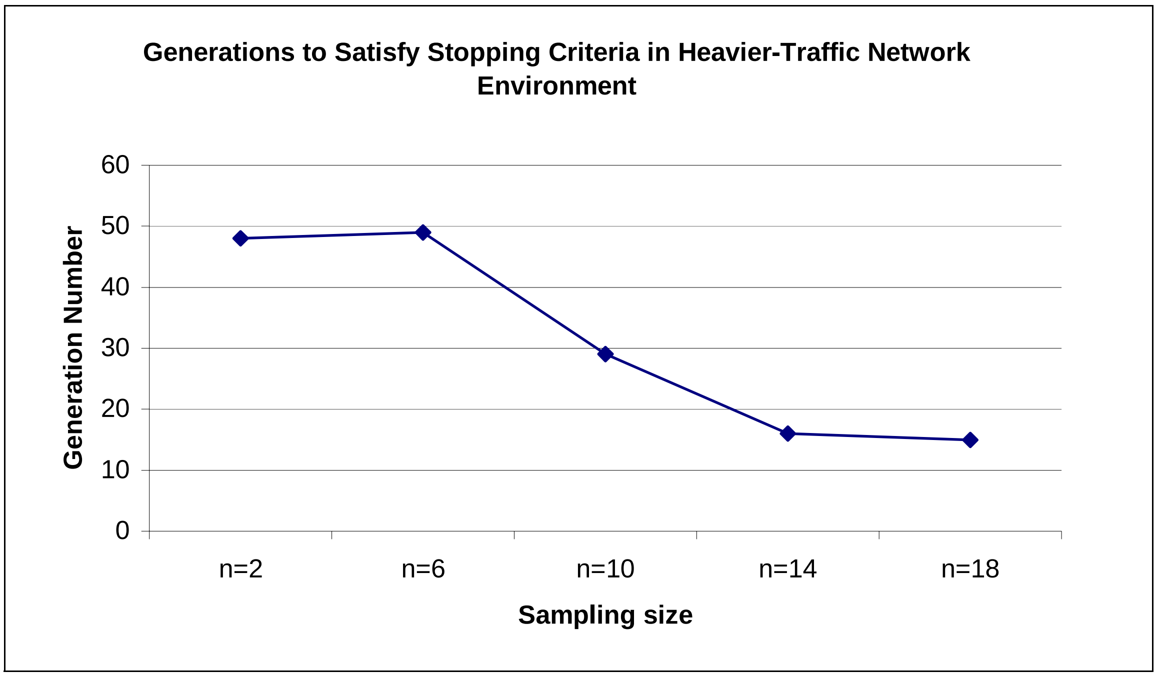

Figure 5.2.2: Number of Generations to Satisfy Stopping Criteria in Heavier - Traffic Network Environment

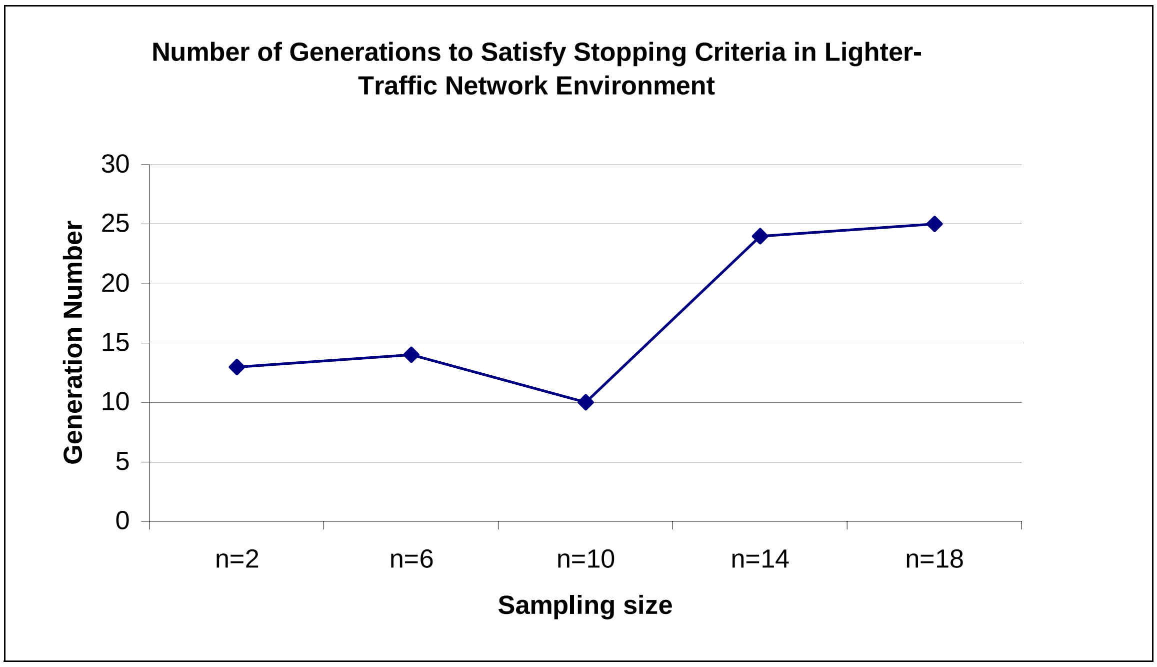

Figure 5.2.3: Number of Generations to Satisfy Stopping Criteria in Lighter-Traffic Network Environment

5.3 Study of Efficiency

GATP will evolve the packet size according to the fitness and networking environment. However, the efficiency will be studied in this section and the QoS will be done in later sections. Figure 5.3.1 shows the graph of total data sent against the iterations of generations. Trial 1, 2 and 3 shows the experiments using sampling size of 10, 20 and 30 respectively and a sub colony size of 3 with all other experiment parameters being similar to Section 5. Shown Below in Table 5.3.1, is the size of data sent at the respective generations of 325 and 556 for the 3 sampling sizes.

It can be seen that trial 2 being of sampling size 20 is twice the sampling size of trial 1. Trial 2 achieved twice the data sent in trial 1. For trial 3 its is 3 times the sampling size of trial 1 and it achieved less than an exact multiple of 3 times the data sent in trial 1 for generation 325. However, at generation 556, trial 3 shows a better performance then expected. However, for larger sampling size, there are relatively little benefits in terms of sending of data. The efficiency achieved from iterating large evolutionary generations for large sampling sizes is too much of an overhead.

The conventional approach in most protocols employs the strategy of having a larger sampling size and therefore a greater efficiency. This may not be relevant in a evolvable protocol. As the protocol aims to adapt to a changing landscape, a small sampling size will be sufficient. In fact too large a sampling size removes much adaptability and sensitivity to the landscape. Thus GATP cannot adopt a strategy of aiming for the largest sampling size for efficiency.

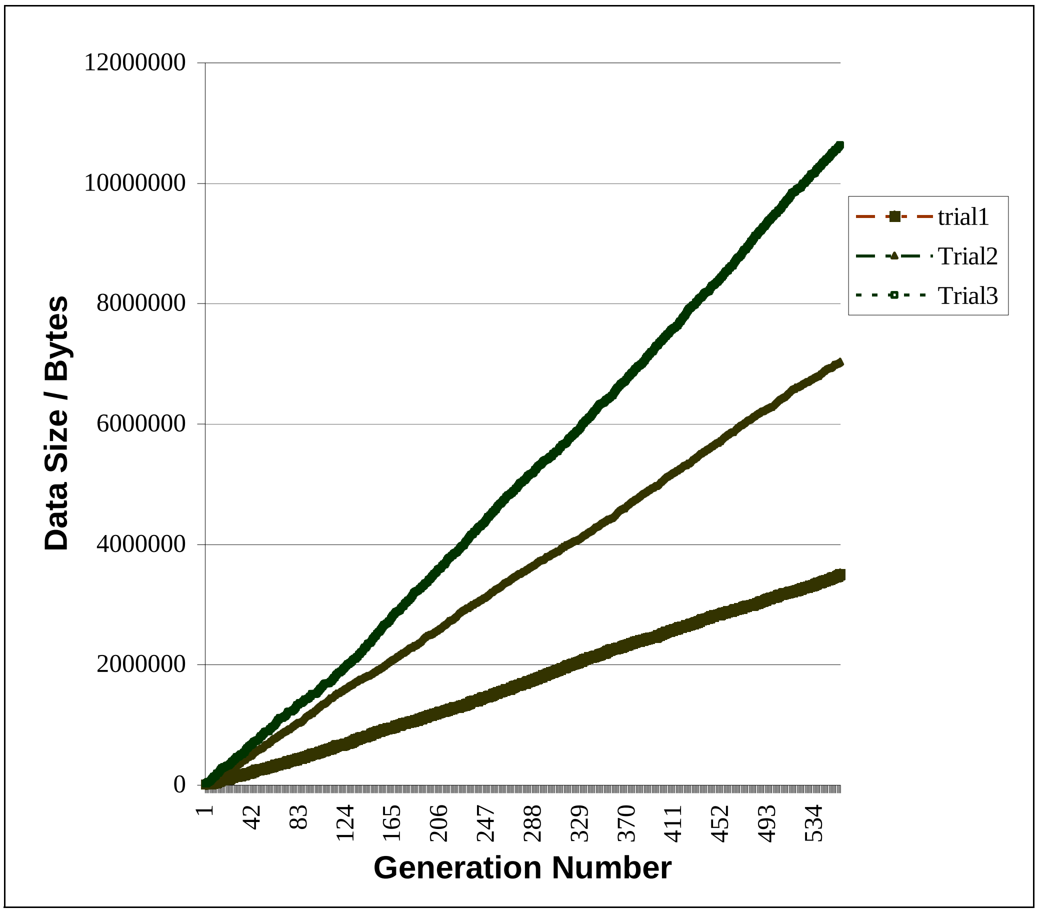

Figure 5.3.1: Size of Data Sent against Generations

Table 5.3.1: Analysis of Efficiency of Sampling Size in terms of Total Data Sent

	Trial1 Total Data	**Trial2 Total Data**	**Trial3 Total Data**	**Trial 3 Efficiency Against Trial 1**
325 Generations	2.035M	4.07M	5.87M	-3.8%
556 Generations	3.5M	7.0M	10.6M	+1%

5.4 Fitness Phenomenon

The experiments followed Section 5 using a sub colony size of 3 and sampling size of 10. Other experiment criteria remained the same. Iterations in the experiment results refer to the results of each gene that was sent out, while generations, refer to the result of each generation consisting of all the genes in the same generation. The actual fitness of each genes employed in GATP is shown below in Figure 5.4.1. The poor fitness is due to unsuitable poorly performing genes. It can be seen that GATP maintains a best fitness of 0.8. Further explanation will be given in the subsequent section.

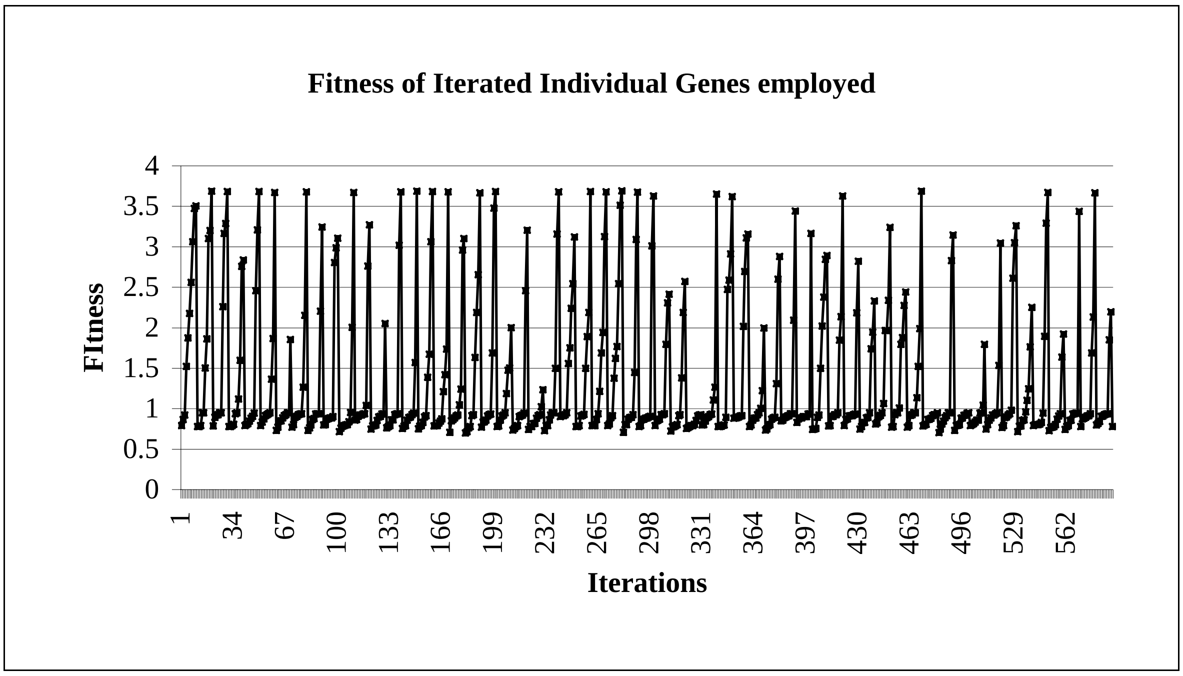

Figure 5.4.1: Weighted Fitness against Iterations

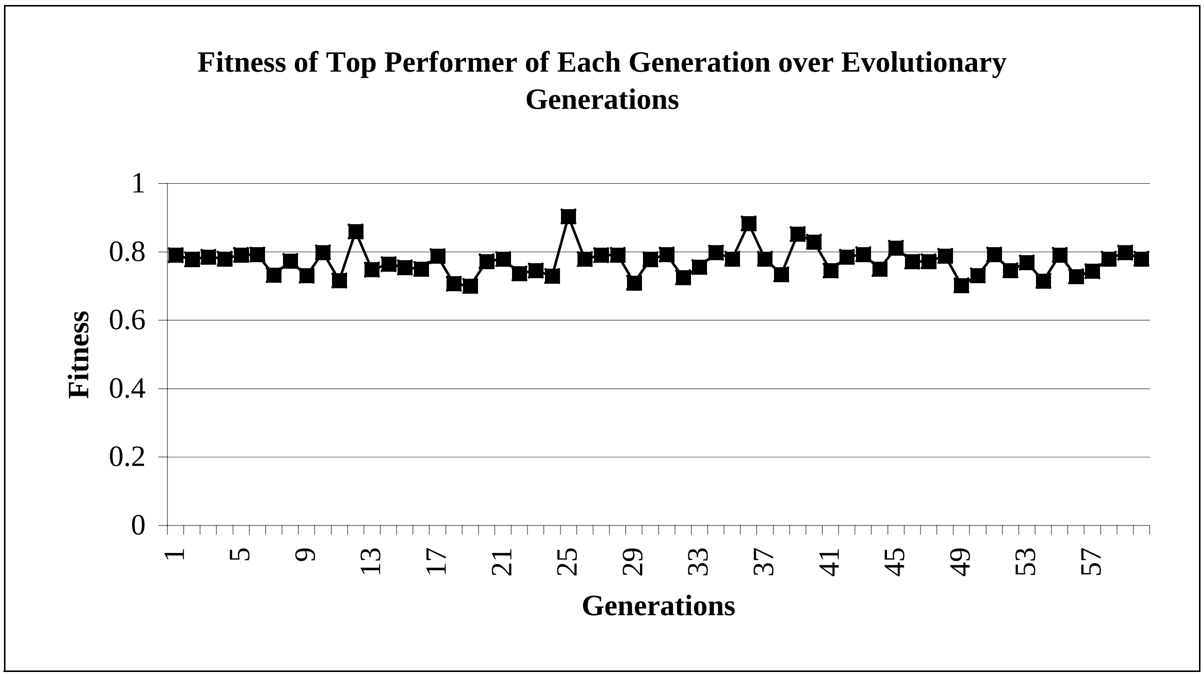

Figure 5.4.2: Weighted Fitness of Best Performer of Population against Iterations

5.5 Genotype Phenomenon

Earlier, the fitness phenomenon was explored, and the fittest gene of each population achieved an average fitness of 0.8. However, Figure 5.5.1 below shows a diversity of genes exist in the population to support a discovery of the best genes in Figure 5.5.2. A best gene not necessarily similar in each population is derived through evolutions. GATP uses different genes to maintain good fitness of 0.8 for a dynamic networking environment.

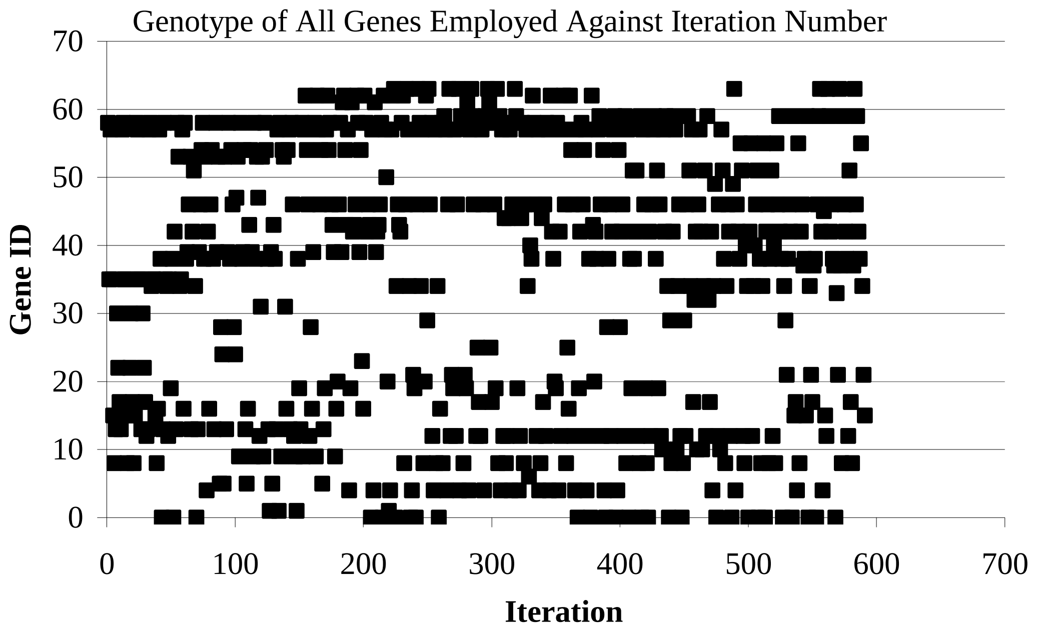

Figure 5.5.1: Gene ID of All Genes Employed against Iteration Number

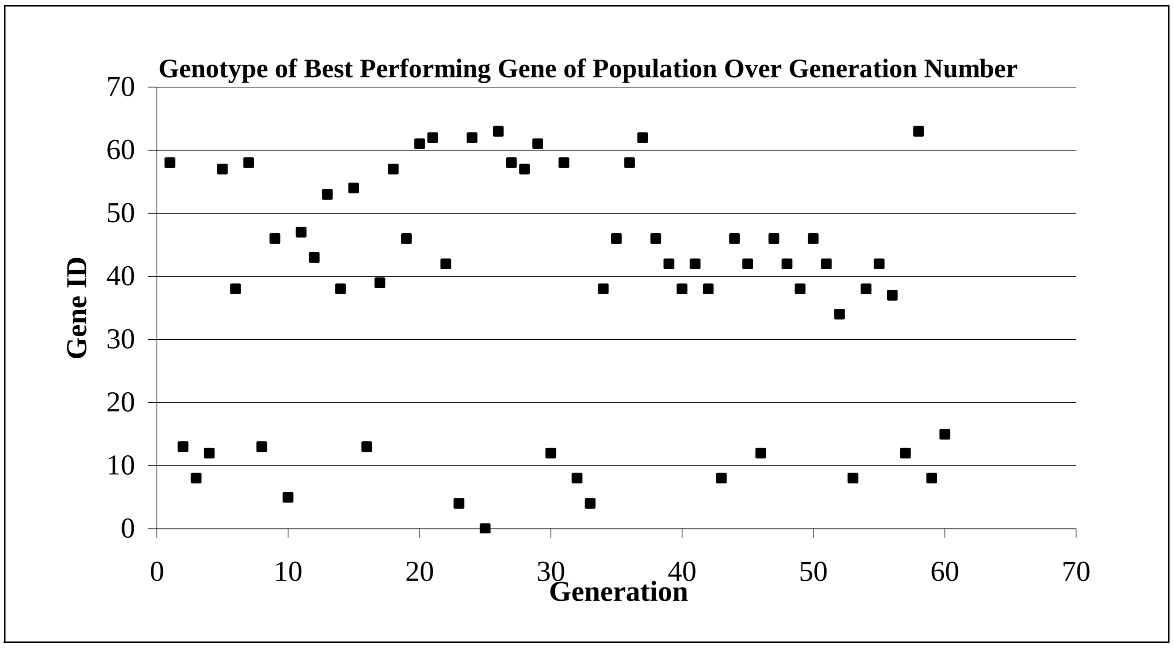

Figure 5.5.2: Gene ID of Best Performing Gene against Generation Number

5.6 QoS Satisfaction in Weighted Fitness Function

Shown below in Table 5.6.1 are the actual QoS achieved from the experiments using the weighted fitness function in Equation 4.3.1. QoS specification followed section 4.3.2. The subcolony size is 2, while sampling size is 10. The average results over 10 iterations were used to compute the results. The population performance is the rate of successful achievement of specified QOS within the entire data transmission. The weighted fitness method actually achieved a reasonable satisfaction of QoS of at least above 55.7% for all 4 QoS measurements.

Table 5.6.1: Performance of Weighted Fitness

Performance	Error Rate/%	Jitter/%	Throughput/ %	Round Trip Time/%
Population Performance	55.7	92.1	87.5	88.7

5.6.1 Non Persistent Best solution

The observation in Section 5.6 has clearly shown that the previous best solution gene actually performs poorer than the entire population. Therefore, a best performer in any generation does not guarantee a best performance in subsequent generations. In fact, GATP has successfully achieved a best performer with a fitness level of 0.8 with the entire population achieving below 3.75. This could be a problem from attempting to solve 4 different objectives with a single fitness function. A possible confirmation of the inappropriateness of the weighted function can be confirmed with GATP functioning on a fitness function based on a single objective in the next section.

5.7 Single Fitness Function

Conflict of fitness function in multiple objectives is eliminated by experiments with only a single fitness objective. Experiments of weighted fitness and single fitness were iterated to allow for a good sampling distribution and objectivity. The experiment was conducted across the intranet with sample size of 10 and sub colony size of 3. The fitness function was a modification of Equation 3.4.1 to only contain the objective under study. The single throughput fitness chart of the best performer of each generation against the evolutionary generations in Figure 5.7.1 shows that good fitness can be achieved when best gene are found in the gene pool. At generation 19, when an extremely good solution gene was found, the fitness immediately drop to a fittest value of zero. The same result was obtained in experiments conducted based on other single objectives: Round trip time, jitter, and error rate shown in Figure 5.7.2. to 5.7.4. Finding solutions in multiple objectives is more difficult and explains for the poorer fitness.

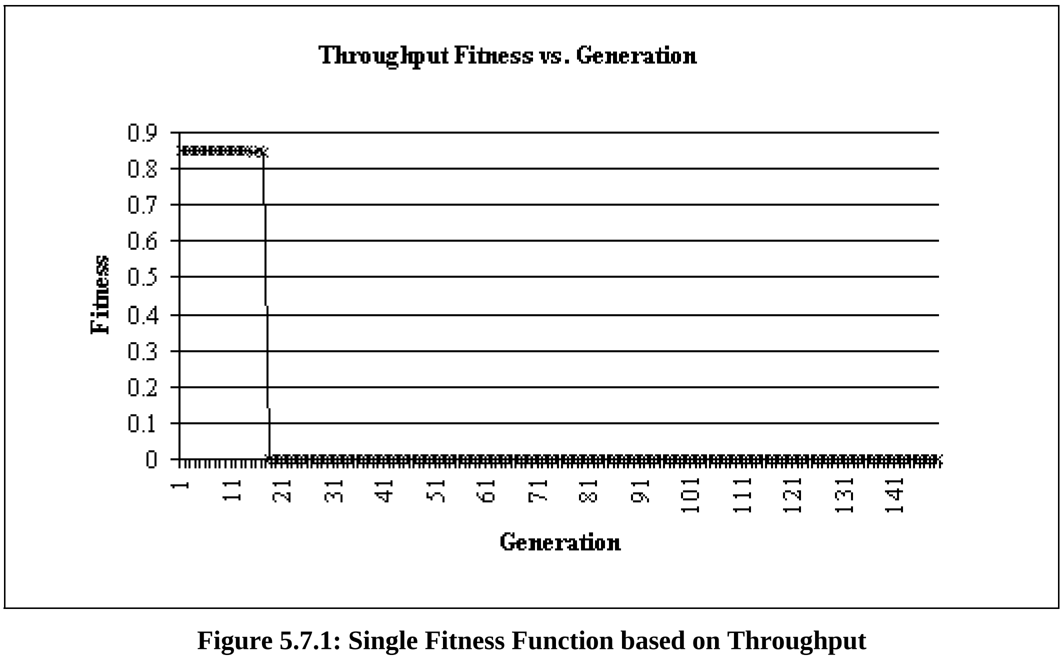

Figure 5.7.1: Single Fitness Function based on Throughput

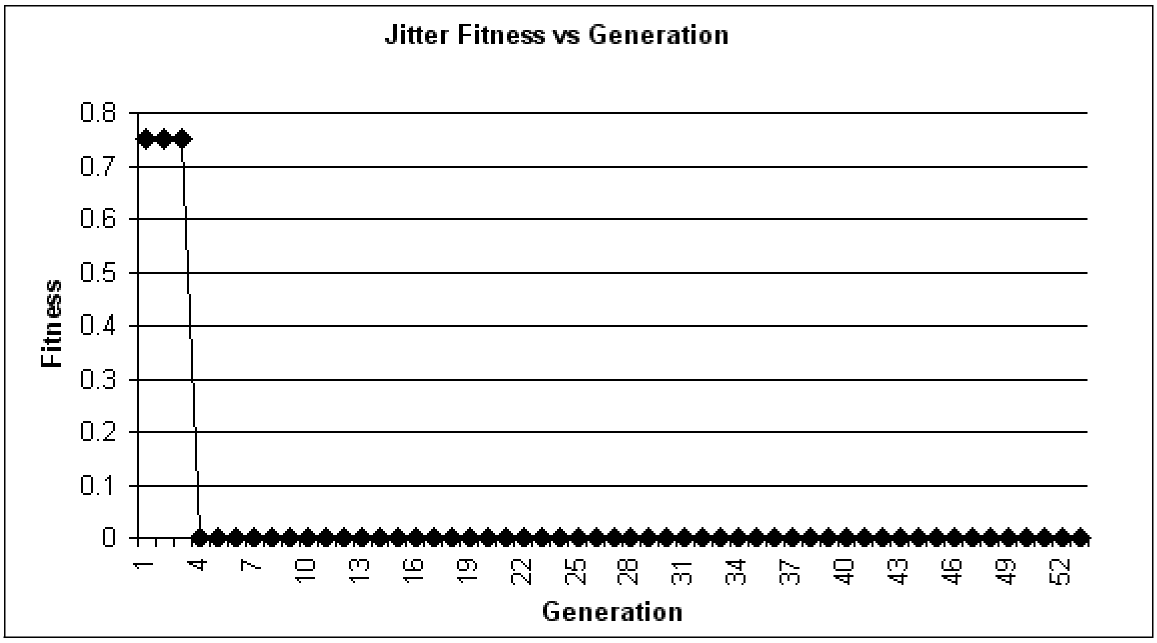

Figure 5.7.2: Single Fitness Function based on Jitter

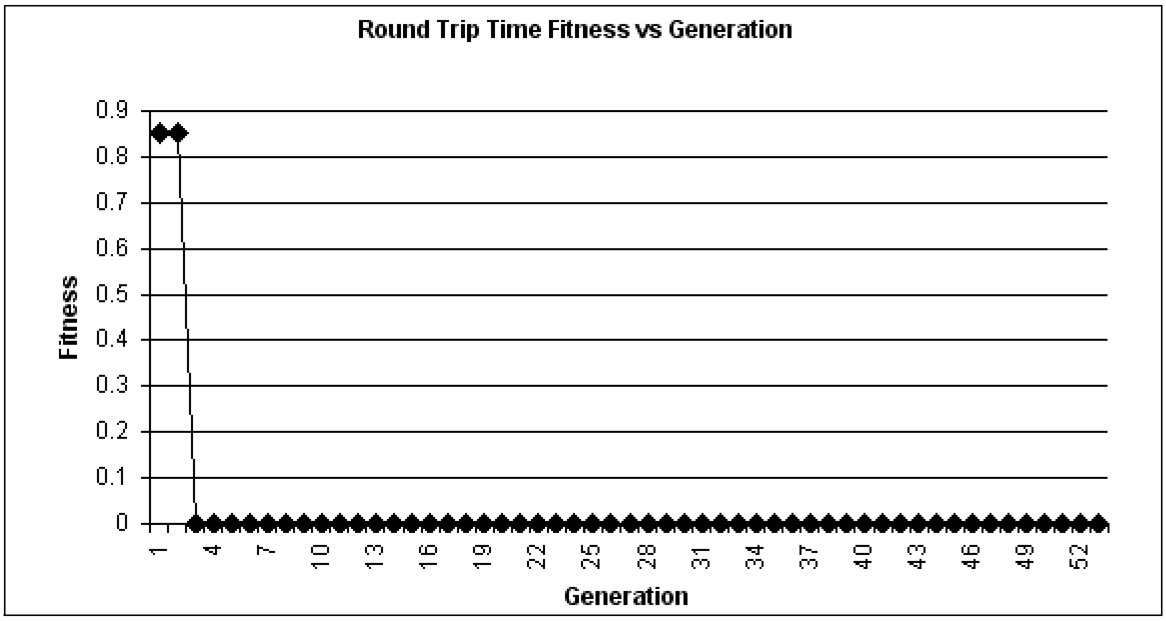

Figure 5.7.3: Single Fitness Function based on Round Trip Time

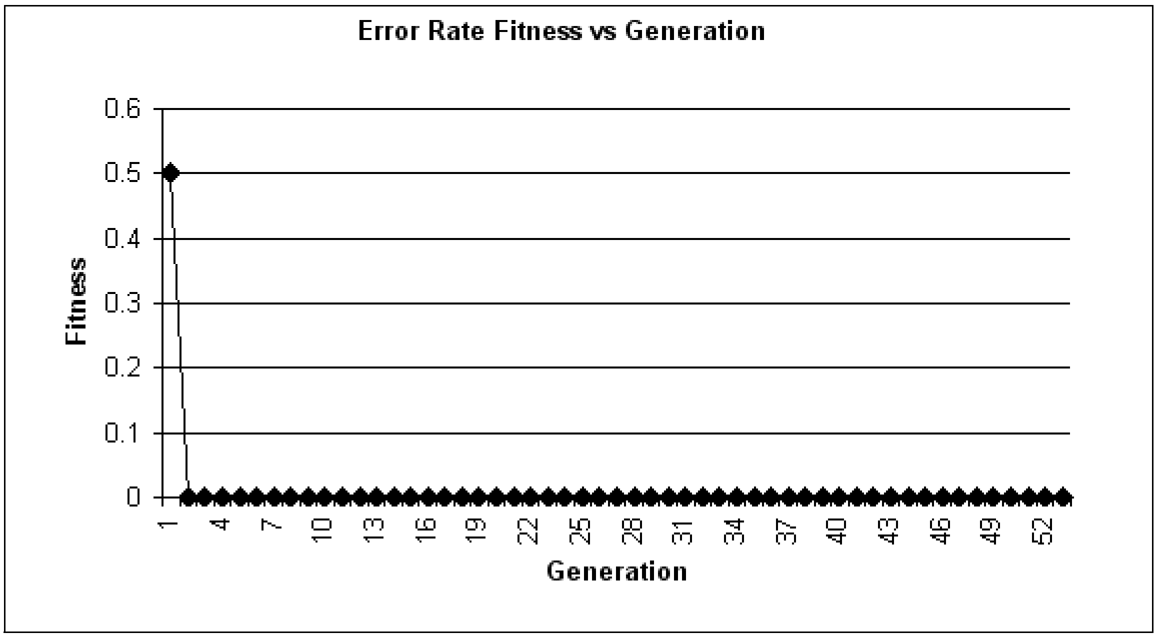

Figure 5.7.4: Single Fitness Function based on Error Rate

5.7.1 Genotype Observation

A great diversity of gene pool was available as shown below in Figure 5.7.5. However, from Figure 5.7.6, the best genotype employed actually changes according to the landscape changes, to achieve the best fitness. This further impresses the existence of a dynamic landscape that GA is called upon to climb even in the absence of multiple contending objectives.

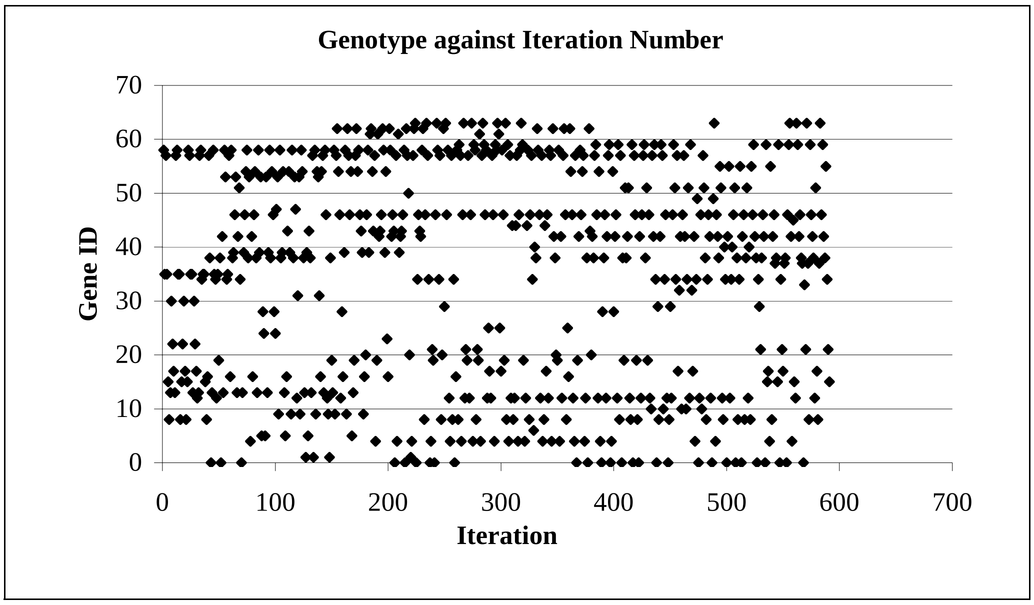

Figure 5.7.5: All Genotype against Iteration Number

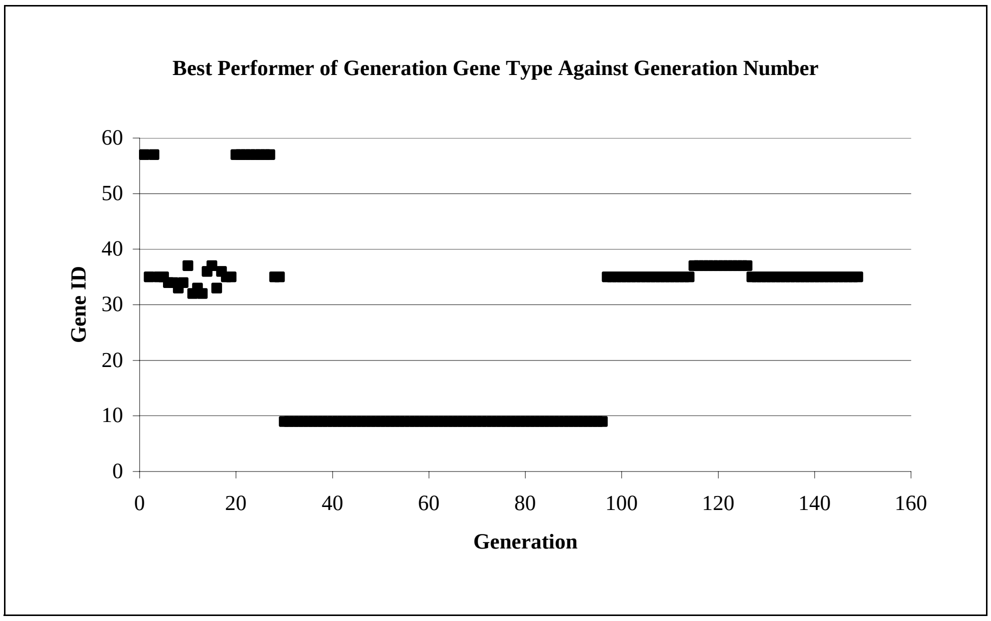

Figure 5.7.6: Best Gene ID against Generation Number

5.8 WAN vs. LAN

The experiments for weighted fitness function were repeated on a Wide Area Network (WAN). The results are shown below in Table 5.8.1. QoS satisfaction in WAN was much poorer due to poorer efficiency in GATP. However, reasonable achievement in QoS was achieved.

Table 5.8.1: Performance of WAN vs. LAN

Performance	Error Rate/%	Jitter/%	Throughput /%	Round Trip Time/%
LAN Performance	55.7	92.1	87.5	88.7
WAN Performance	50.2	63.0	79.1	60.1

Chapter 6

Multi Objective Genetic Algorithm

With 4 contending fitness objectives, Multi-objective genetic algorithm (MOGA) is proposed as a more appropriate transport engine for GATP. The algorithm used will be modeled after Dias [30] and Deb [31].

6.1 Assigning fitness level to the genes

The fitness assignment is similar to NSGA [30, 31]. There are four performance objectives. The four fitness computations were shown earlier in Section 3 for jitter, throughput, round trip time and error rate. A fitness rank will be given for each objective according to the actual objective performance. The most undominated gene in a single objective will be assigned the fittest rank followed by the next most undominated gene. This ranking starts from one and increases towards less fit genes. However, genes with equal fitness will have the same ranking. The final ranking is a summation of the four fitness ranking derived from the different objectives.

6.2 Tournament

This is primarily the tournament stage. A random assignment is done to assign 3 different groups with extra care taken to prevent any particular order of the groups. 10 genes will fall into 3 groups with members of 4, 3, 3 in each group. In the event of similar fitness ranking for all competitions, a matter of chance will decide the competitors' fate. A complete tournament is held within each group and a distinctive ranking is achieved. The top 2 players of each group will survive to the next round of competition-semifinals, while the rest of the losers will fight it out to produce a distinctive list of winners also classified as winnersC. The top player will fight the second best winner from another group. The winner shall proceed to the final stage while the losers form winnersB will fight it out again. A complete tournament is held within winnersA, winnersB and winnersC to produce their own distinctive winners. Then the 3 groups shall form the final winners list. The aim of this tournament is to produce a supreme winner and to eliminate individuals sitting on the same fitness front.

6.3 Elitist method with complete competition

This is an alternative to the tournament process shown in section 6.2. In fact, this is the process used by the weighted fitness function. The objective is to have a death match of all the individuals in the population. By allowing every individual to have competed with all others in the population, the true ability of the individual is obtained and ranked. Individuals sitting on the same front will no longer destroy each other by a matter of chance.

6.4 Crossover and mutation

Crossing over is done using two point and one point crossover. Flipping the bits of the genes does mutation. A random selection of the new genes will be done.

6.5 Experiment Results

Experiments were conducted by iterating all experiments with different processes changed or omitted for a fair and accurate analysis. The network environment was that of an Intranet LAN with moderate congestion defined here as having a TCP flow rate of 40 Mbps between the 2 machines being tested. Below in Figures 6.5.1 and 6.5.2, are results of using MOGA with tournament. The best performing gene in Figure 6.5.2 persisted shortly, due to rapidly changing network environment, even though a vast diversity of genes were available in Figure 6.5.1.

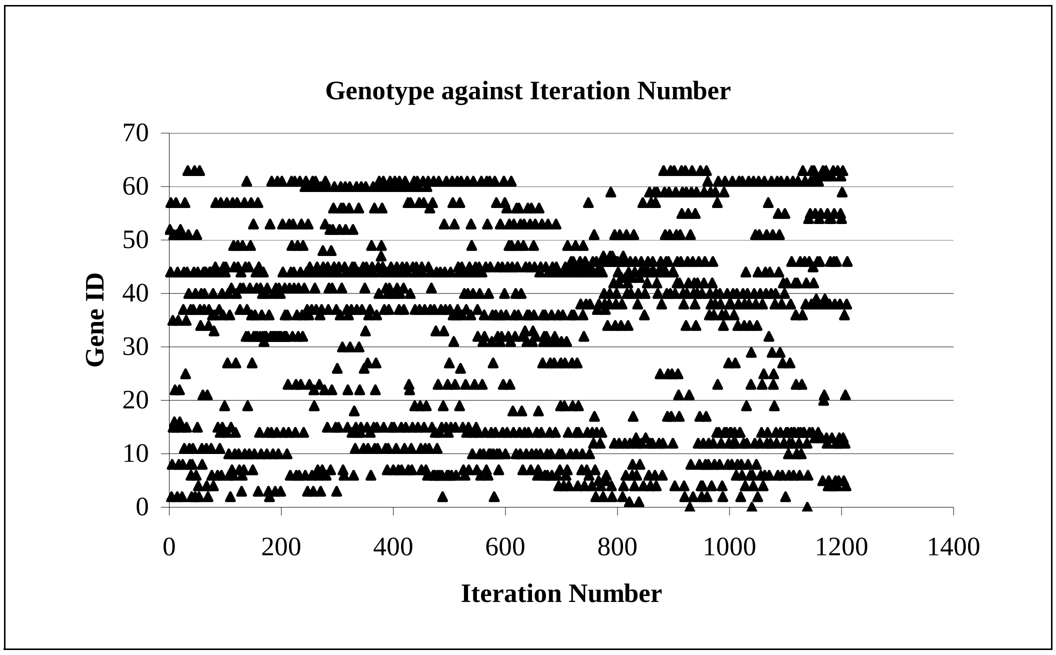

Figure 6.5.1: Graph of All Genotype employed against Iteration Number

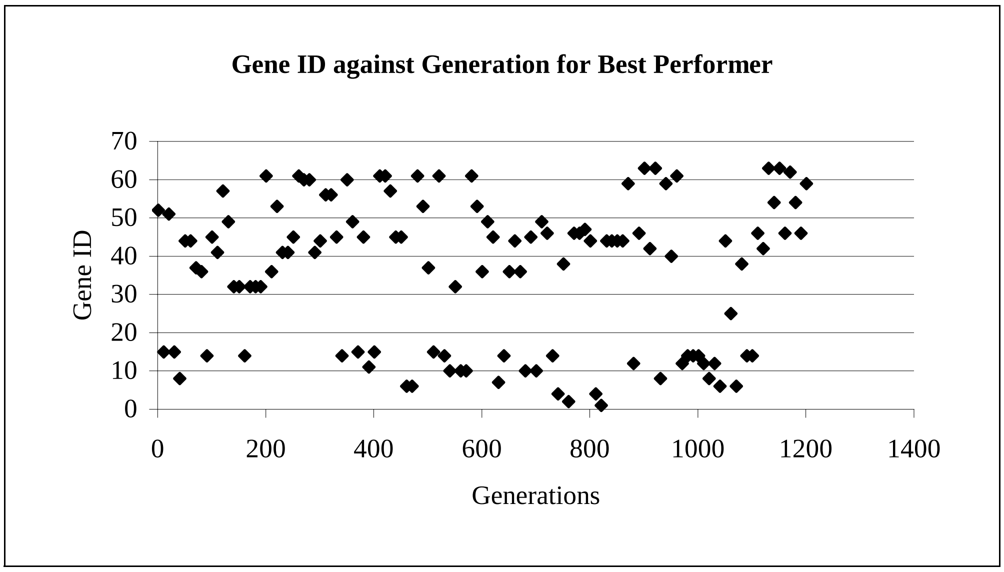

Figure 6.5.2: Graph of Top Performing Genotype against Generation Number

6.6 QoS Satisfaction in MOGA

The experiments were iterated between the different schemes on the Intranet LAN, with sample sizes of ten, sub colony size of three, and all other experiment parameters being the same as in Section 5. The average of ten results was tabulated in Table 6.6.1. The Elitist was shown to produce better results than tournament.

Table 6.6.1: Results MOGA using Tournament vs. Elitist Schemes

	Tournament				**Elitist**			
Performance	**Jitter/%**	**Error/%**	**Tp/%**	**RTT/%**	**Jitter/%**	**Error/%**	**Tp/%**	**RTT/%**
Population	89.8	42.9	67.2	67.5	91.7	43.2	66	74.5

Legends: Tp Represents Throughput, and RTT represents Round Trip Time.

6.7 Tournament Inadequacy

The above results in Tables 6 shows the lacklustre performance of tournament, good for finding the absolute best, but inadequate in achieving good overall population performance.

6.8 QoS performance of Weighted Fitness function

The weighted fitness method was repeated in the same experiment with MOGA for consistency. Below in Table 6.8.1 are the results. Since the experiments were iterated within the same batch, a fair comparison could be made.

Table 6.8.1: Results of Repetition of Weighted Fitness

	Weighted fitness with absolute uniqueness in new genes			
Performance	**Jitter/%**	**Error/%**	**Tp/%**	**RTT/%**
Population	83.5	37.6	82.0	77.9

Legends: Tp Represents Throughput, and RTT represents Round Trip Time.

6.9 Comparison of MOGA with Weighted Fitness

Shown below in Table 6.9.1 are the results of implementation of the MOGA compared to the weighted fitness function method. It is seen that MOGA method produced a higher QoS achievement compared to the weighted fitness function. MOGA is a more ideal technique for GATP to adapt and react to the networking environment, through its multi-objectiveness.

Table 6.9.1: Comparison of MOGA Elitist with Weighted Fitness Method

	MOGA with Elitist				**Weighted fitness**			
Performance	**Jitter/%**	**Error/%**	**Tp/%**	**RTT/%**	**Jitter/%**	**Error/%**	**Tp/%**	**RTT/%**
Population	94.8	67.8	65.1	80.3	88.4	40.5	89.5	78.4

Legends: Tp Represents Throughput, and RTT represents Round Trip Time.

6.10 Sub Colony effects on GATP in dynamic Landscape

The experiments were done using using MOGA with elitist selection. The sample size used was 10 and sub colony size of 3 with all other experiment parameters being the same as in section 3. 1 set of experiment has the side colony being reduced such that only small mutation rate and crossover of parents being 25%. Random genes were not allowed into the sub colony. The other set contained a side colony with random genes. Shown in Figures 2 and 3 are the results of the experiments. Using a side colony allows GATP to perform better. A faster reaction to network changes can be seen and a better QoS achievement in Table 6.10.1 below.

The experiments were done using using MOGA with elitist selection. The sample size used was 10 and sub colony size of 3 with all other experiment parameters being the same as in section 3. One set of experiment has the side colony being reduced such that only small mutation rate and crossover of parents being 25%. Random genes were not allowed into the side colony. The other set contained a side colony with random genes. Shown in Figures 6.10.1 and 6.10.2 are the results of the experiments. Using a side colony allows GATP to perform better. A faster reaction to network changes can be seen and better QoS achievement in Table 6.10.1 below. It can be seen that GATP achieves greater QoS satisfaction when a higher degree of mutation and crossover occurs in the sub colony for a stronger shifting balance. This occurs in an environment of higher and more dynamic traffic.

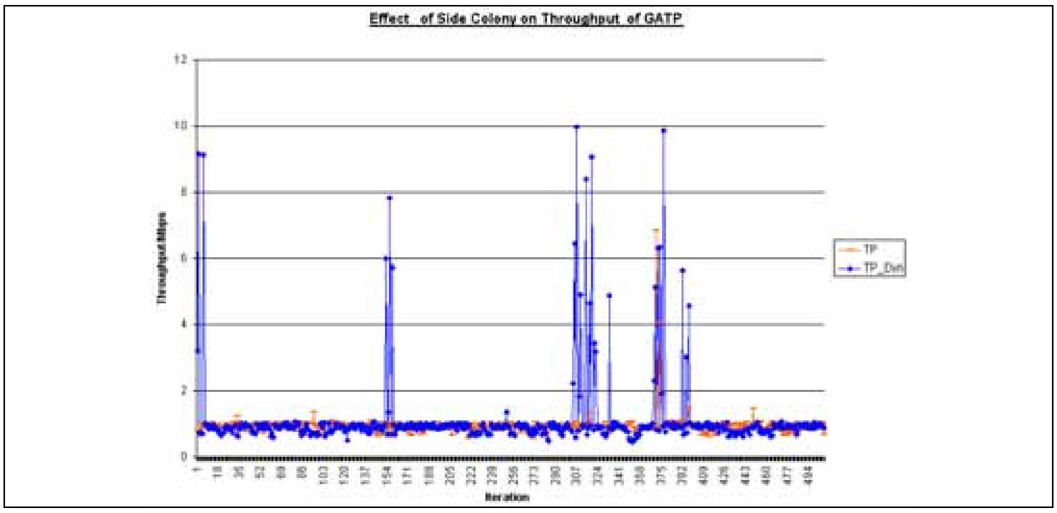

Figure 6.10.1 Effect of Sub Colony on Throughput Performance in GATP
Notes: TP represents Throughput without sub colony while TP_Dyn represents throughput using sub colony.

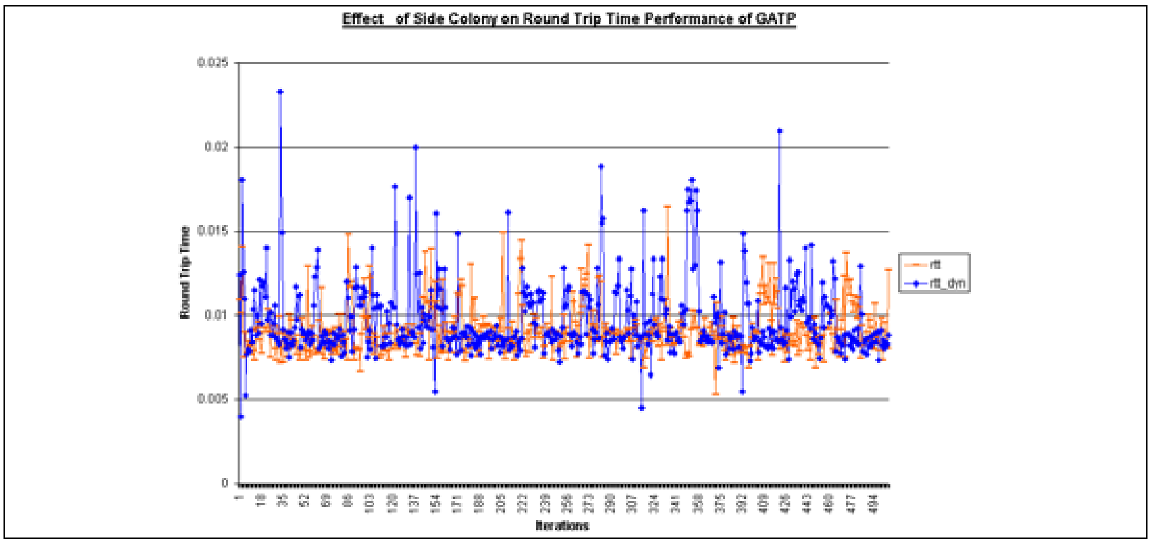

Figure 6.10.2 Effects of Sub Colony on Round Trip Time Performance in GATP
Notes: rtt represents round trip time without sub colony while rtt_dyn represents round trip time using sub colony.

Table 6.10.1: Percentage QoS Satisfaction

	Round Trip Time Satisfaction/ %	**Jitter Satisfaction / %**	**Throughput Satisfaction / %**
25% Mutation 25% Crossover	9.86	50.30	12.82
75% Mutation 80% Crossover	22.49	67.46	22.49

Chapter 7

Comparison Experiments with TCP and UDP

Data transmission using TCP and UDP were also implemented on the socket client and server. The maximum packet size was 1024 bytes, similar to GATP. GATP used MOGA scheme and elitist selection, with QoS; Jitter, Round Trip Time, Error Rate and Throughput of 1.5ms, 6ms, 0.05 and 2Mbps respectively. These values were selected, as they were relatively attainable but not too easy. The sub colony size used was three. Other parameters were same as section 4.3.1 and 4.3.2. The results are shown below in Section 7.1.

7.1 Jitter

GATP achieved good jitter performance as compared to UDP and TCP. This is due GATP adopting a QoS objective orientation towards jitter. Shown below in Figure 7.1.1, is the jitter performance of the 3 protocols under study. Although GATP has the lowest jitter value, there were also instances of poor jitter. UDP and TCP achieved a lesser range of jitter values but with a poorer performance bound.

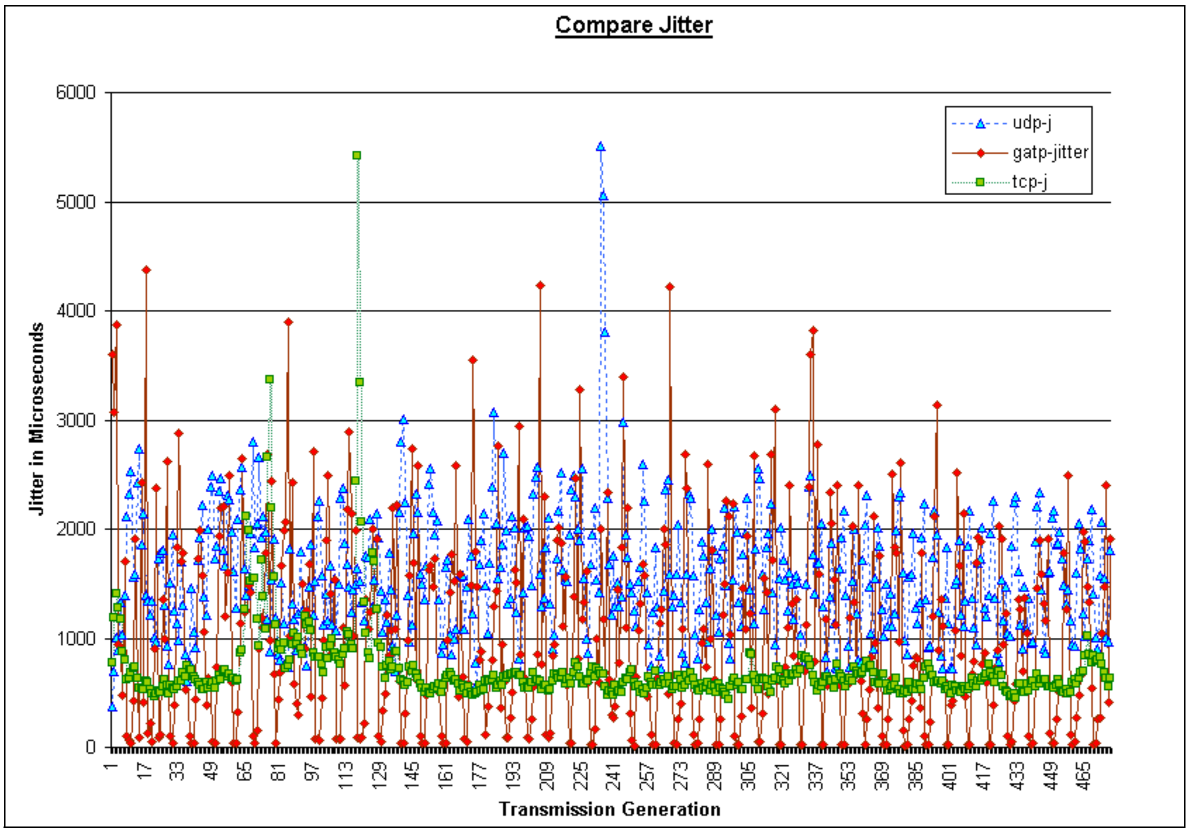

Figure 7.1.1: Jitter Performances of UDP, GATP and TCP against Transmission Generation

Notes: udp-j represents jitter performance of udp, gatp-jitter represents jitter performance of GATP and tcp-j represents jitter performance of TCP.

7.2 Throughput

Shown below in Figure 7.2.1 are the Throughput performances of GATP, UDP and TCP. The throughput of GATP was worse than TCP and UDP on the whole. GATP was able to achieve a greater range of performance by exploiting the best packet configuration according to the network conditions. Poorer throughput of GATP is due to efficiency issues of different Automatic Repeat Request (ARQ) schemes, which will be discussed, later in the overhead studies on GATP.

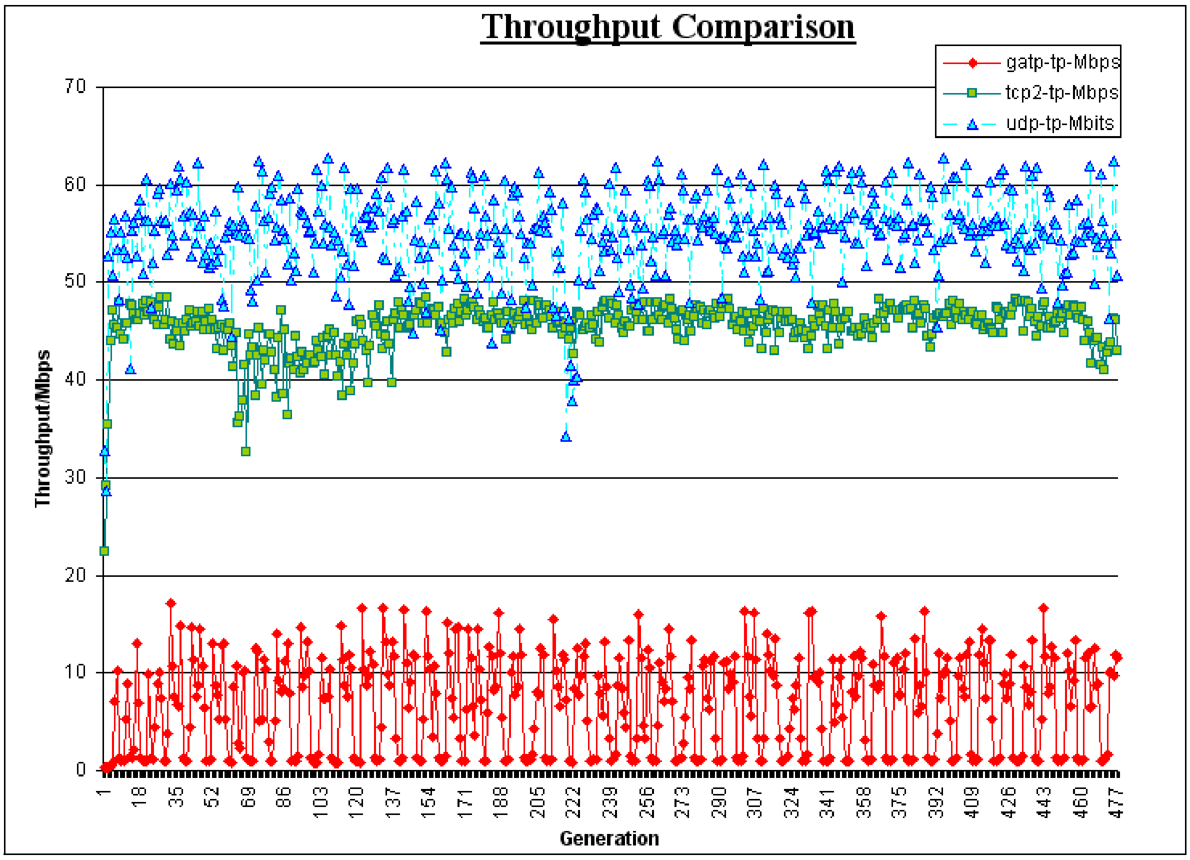

Figure 7.2.1: Throughput Performances of UDP, GATP and TCP against Transmission Generation

Notes: udp-tp represents throughput performance of udp, gatp-tp represents throughput performance of GATP and tcp-tp represents throughput performance of TCP.

7.3 Round Trip Time

Shown below in Figure 7.3.1 are the performance of UDP TCP and GATP for Round Trip Time. The Round Trip Time of GATP was much poorer than TCP and UDP. This is mainly because GATP is not able to compete with TCP and UDP, which were optimised greatly over many years. The packetisation delay was a main factor as much intelligence was required at each packetisation process. There was also a micro management of packets. The overheads of GATP will be explained later in Chapter 9. However the QoS orientation of GATP towards a Round trip time of 6ms was fulfilled by at least half the transmission.

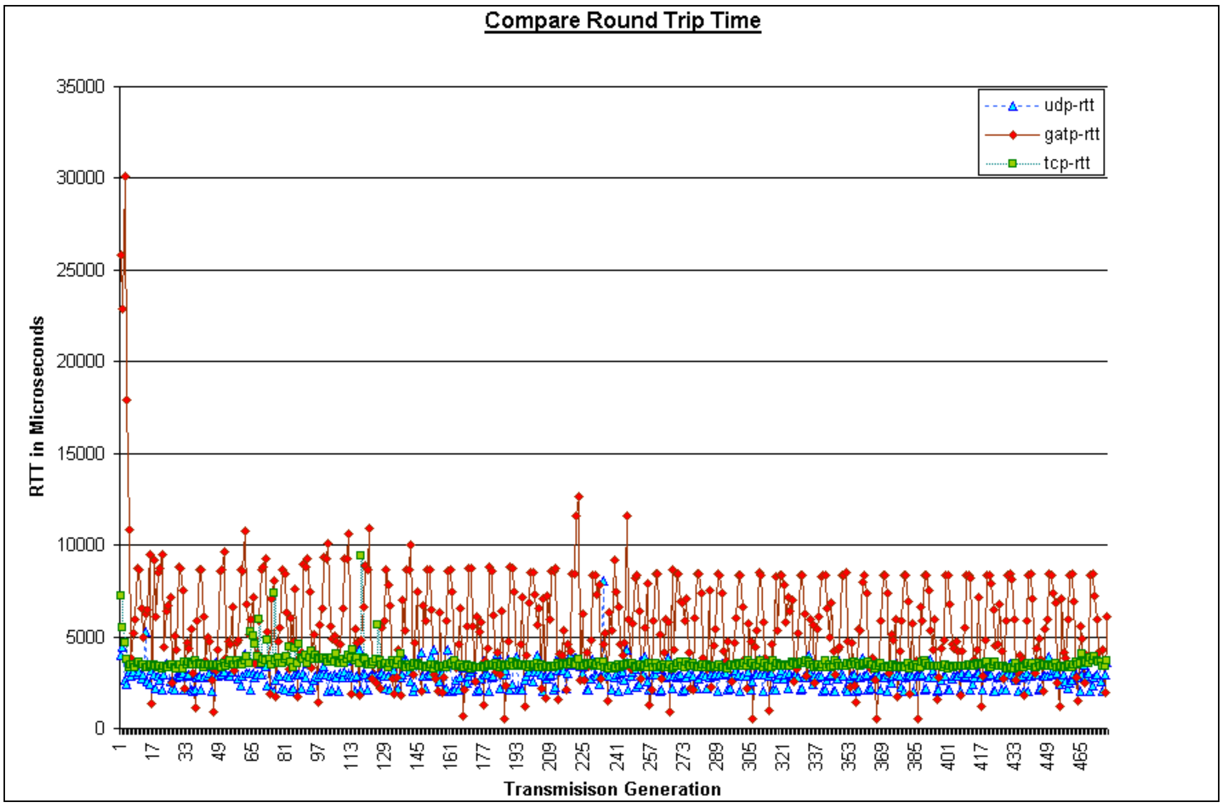

Figure 7.3.1: Round Trip Time Performances of UDP, GATP and TCP against Transmission Generation

Notes: udp-rtt represents round trip time performance of udp, gatp-rtt represents round trip time performance of GATP and tcp-rtt represents round trip time performance of TCP.

7.4 Error Rate

Shown below in Figure 7.4.1 are the error rates of UDP TCP and. The error rate of GATP ranged the greatest from the lowest error rate to the highest error rate equivalent to the UDP. This is mainly because GATP is able to perform better than TCP and UDP, through a faster search mechanism which may also result in a worse error rate. The overheads of GATP will be explained later in Chapter 9. However the QoS orientation of GATP towards a error rate of 0.05 was fulfilled by at least half the transmission.

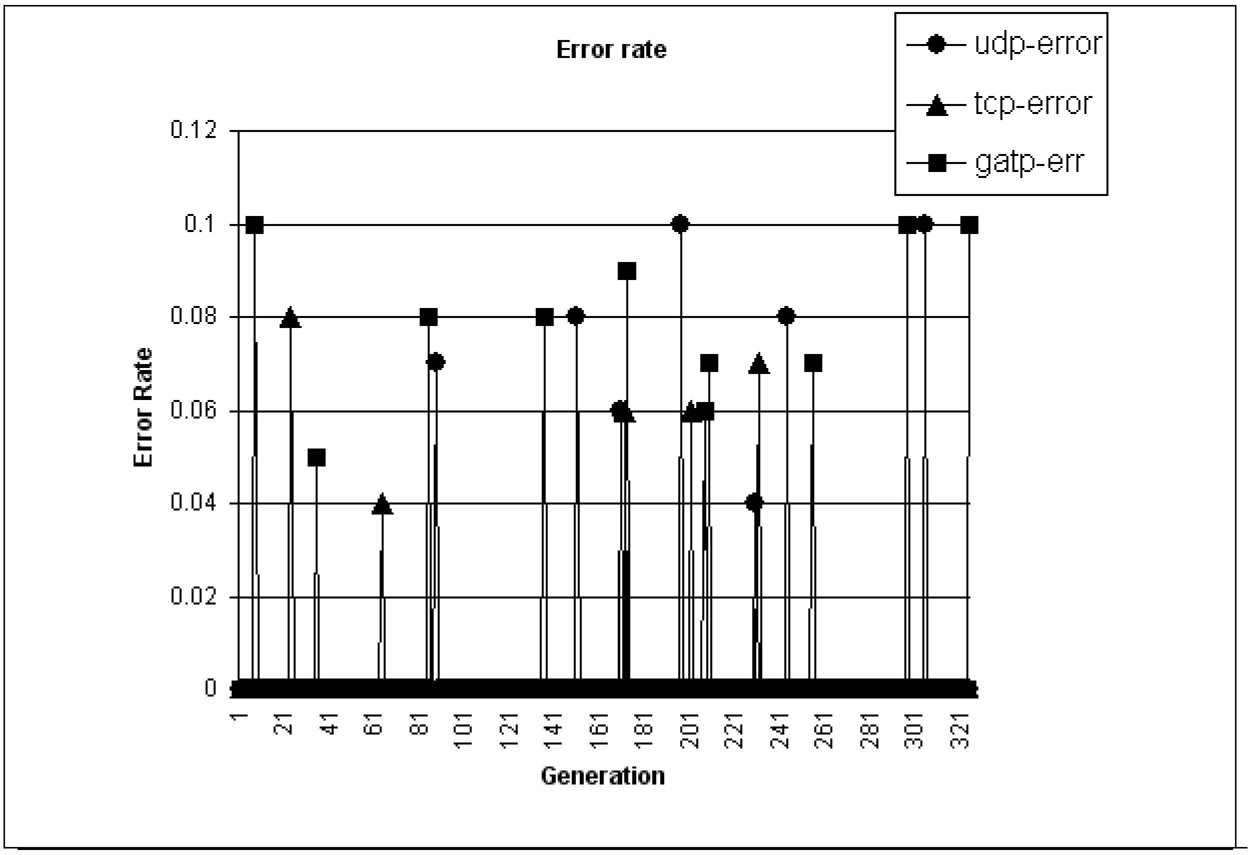

Figure 7.4.1: Error Rate UDP, GATP and TCP against Transmission Generation
Notes: udp-error represents error rate of udp, gatp-err represents error rate of GATP and tcp-err represents error rate of TCP.

Chapter 8

Controlled Network Environment Experiment

To Control the network environment, the server and client computer was isolated from the intranet and networked directly. Experiments were conducted for GATP employing NSGA techniques with elitist and unique diverse sub colony. Single objectives experiments were conducted on throughput, Round Trip Delay and Jitter. Solutions were found to propagate all generation in this non-competitive environment.

To further investigate the performance of GATP with competing traffic, the experiments below were conducted. Firstly, GATP was conducted with elitist selection and unique diverse sub colony with no competing traffic. Other parameters were same as section 4.3.1 and 4.3.2. The results are shown below in Section 8.1. Crossover and Mutation probability is 75% and 80% respectively.

QoS targets for GATP was set to be Round Trip Time was 0.006s, Jitter 0.0015s, Error Rate of less than 0.02%, and Throughput of 3.2 Mbps. The sub colony size of GATP was 3 while the main colony was 7. Each gene was transmitted in sample sizes of 10 packets. Therefore the each generation was 100 packets.

Solution genes were found and propagated. Then at generation 20 of GATP transmission, a UDP traffic generator was started to transmit 1024 bytes sized packets at a speed of 1 Mbps. The size chosen was 1024 to prevent UDP fragmentation which could in term create a bursty competing traffic. This would cause a highly uncontrolled competing traffic quite opposed to the aim of this experiment. The UDP traffic was terminated at generation 28. This allowed GATP to adapt back to it initial non competitive environment. Then at generation 47 the UDP traffic was again restarted till generation 54.

8.1 Jitter and Round Trip Time

Shown below in Figure 8.1.1 is the result for jitter and round trip time, where j represents jitter and r represents round trip time. Competing UDP traffic was introduced at generation 20. The Round Trip Delay worsened as a result and managed a worst value of 0.03s. However, it was restored to original Round Trip Time of 0.0055s at generation 23.3. Likewise for jitter, it managed to achieve 0.002s before the introduction of UDP traffic at generation 20. However after the competing traffic entered the network, jitter took a worst value of 0.0058s. This poor performance was however, remedied by generation 23.3 where jitter was restored to original value. Thus restoration of jitter and Round trip time took less than 4 generations with competing UDP traffic. It can be seen from Figures 8.1.1 and 8.1.2 below that at generation 23.3 and generation 50.8, the jitter and round trip time was restored to the original levels before the introduction of the UDP traffic. The response of GATP to the UDP traffic at generation 20 could be seen by the poorer jitter and round trip time. However by generation 24, GATP has found the solutions to the networking environment. When the UDP traffic was removed at generation 28, GATP was able to continue providing the same level of QoS in terms of jitter and RTT. The next change came at generation 47. GATP again exhibited the same robustness to change and was able to restore QoS by generation 51.

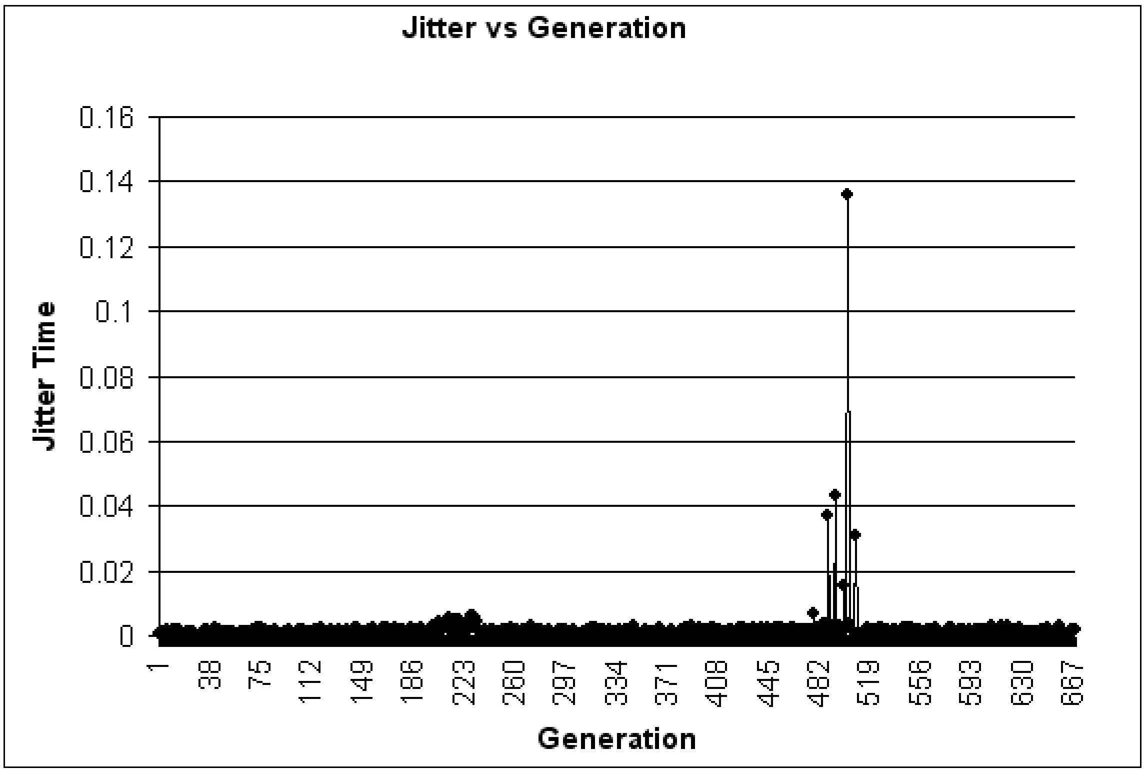

Figure 8.1.1: Jitter Performance of GATP in Controlled Network Environment
Notes: Actual Generation number is X axis value divided by 10

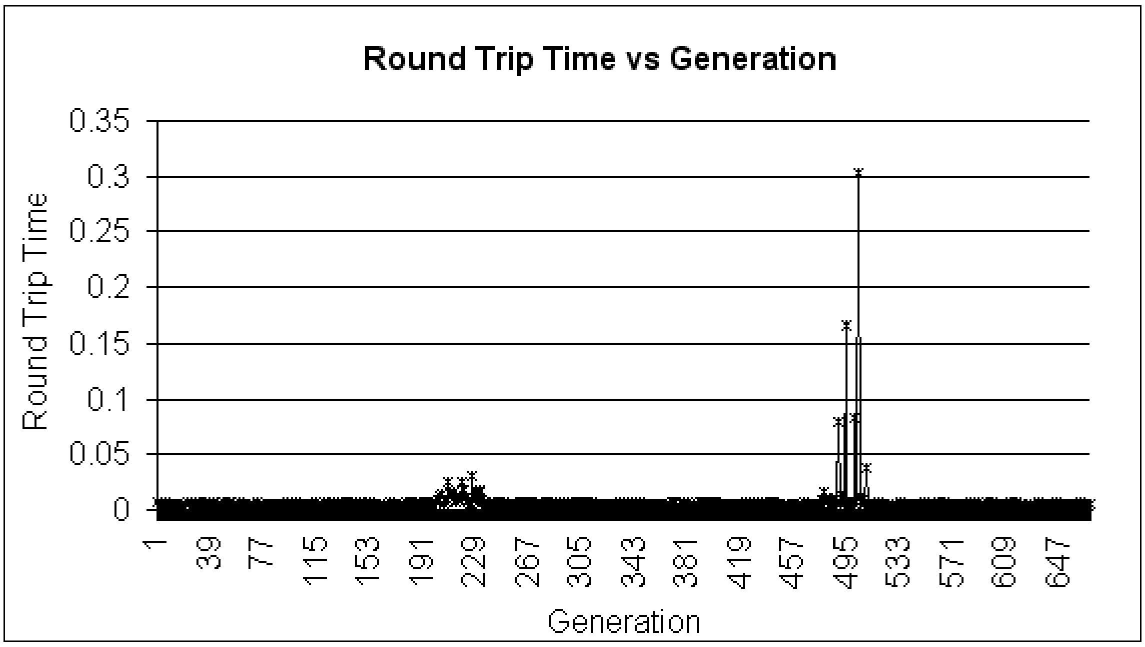

Figure 8.1.2: Round Trip Time of GATP in Controlled Network Environment
Notes: Actual Generation number is X axis value divided by 10

8.2 Throughput

Figure 8.2.1 below demonstrates the throughput QoS of GATP under the same controlled network environment. It can be seen that at generation 20, the introduction of UDP traffic reduced the throughput of GATP traffic, which was restored to initial original level at generation 23. Throughtput has a worst value of 1.11Mbps before generation 20. At generation 20, GATP throughput worsened to a minimum of 0.46Mps at the worst case at generation 22. However, by generation 24, Throughput was restored to the original performance before the competing traffic was introduced. Likewise throughput lessened in generation 47 and was restored at generation 51.

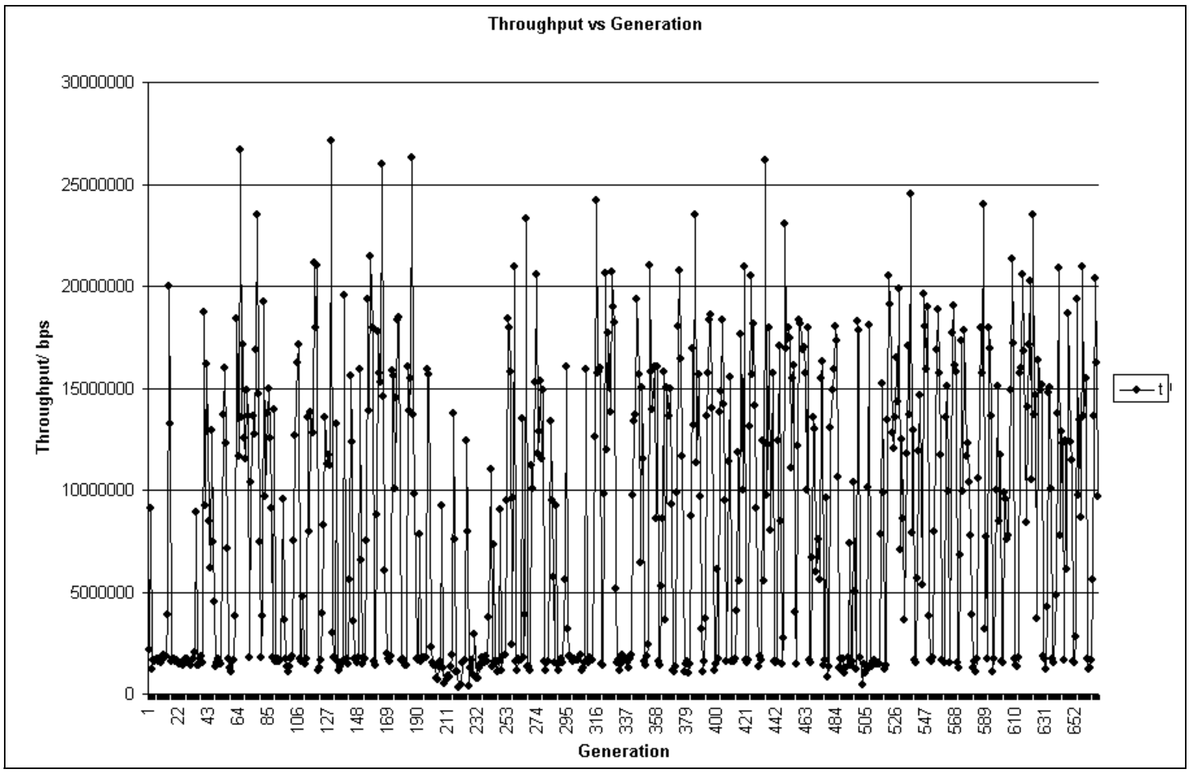

Figure 8.2.1: Throughput Performance of GATP in Controlled Network Environment

Notes: Actual Generation number is X axis value divided by 10. t represents throughput of GATP.

8.3 Error Rate

Below in Figure 8.3.1 are the results for error rate. The error rate behaviour was similar to throughput, jitter and Round Trip Time. The robustness of GATP to a changing network environment was again demonstrated. Error rate was kept close to zero before generation 20. However, after UDP traffic was introduced at generation 20, GATP shown a worst error rate performance of 0.55 at generation 22. This error rate deterioration was remedied by generation 24, where restoration to original healthy error rate was achieved. The restoration of QoS to original levels before introduction of any traffic was seen after GATP adapted to changes in the network. This phenomenon was revisited in generation 47 to 51.

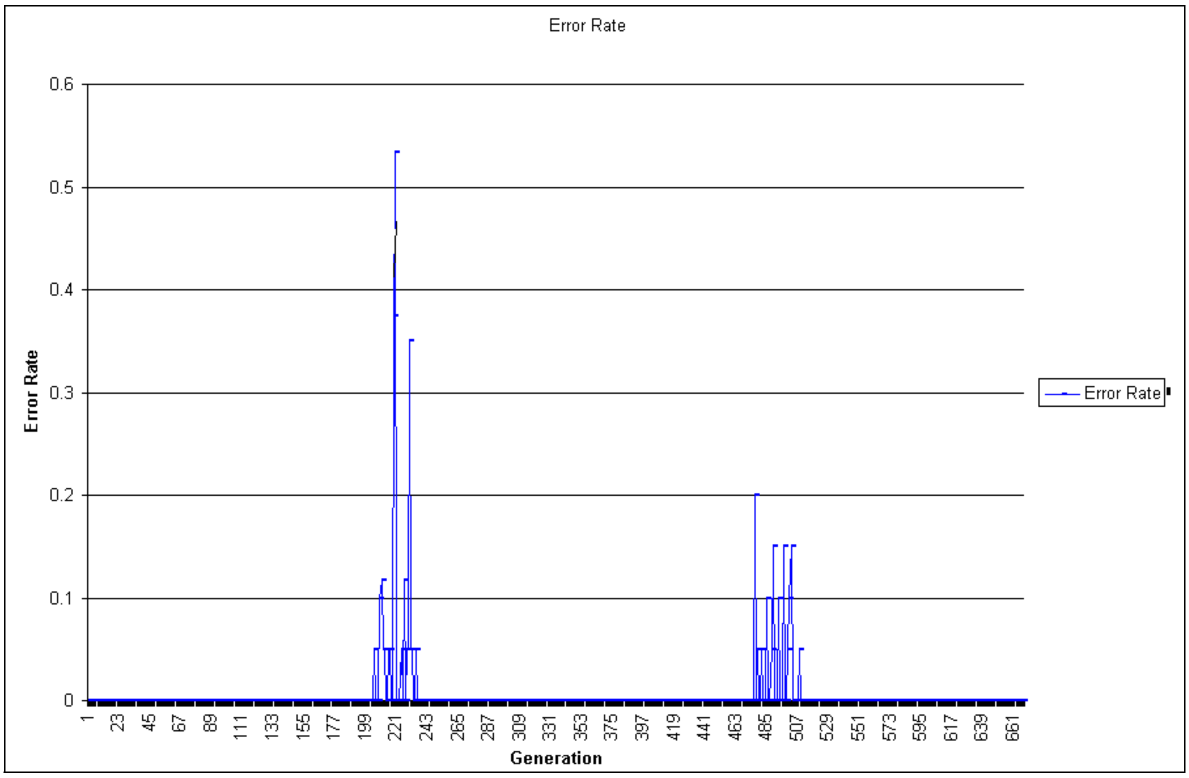

Figure 8.3.1: Error Rate Performance of GATP in Controlled Network Environment

Notes: Actual Generation number is X axis value divided by 10

Chapter 9

GATP Overhead Computation

9.1 Theoretical Analysis of GATP Flow Control Overhead

GATP has implemented several flow control mechanisms combined with genetic algorithm for optimization. The Automatic Repeat Request (ARQ), variable frame size, was implemented for reasons of improving real time data streaming ability and exploring the feasibility of a genetic algorithm approach. Subsequently, this section shall discuss the mechanisms employed in GATP; These are mainly Stop and Wait (SAW) ARQ, Variable Frame sizes and multiplexing. Finally, a proposal for a better ARQ is suggested.

9.1.1 Stop and Wait Automatic Repeat Request (SAW ARQ)

Stallings [40] describes various Automatic Repeat Request (ARQ), schemes. GATP has adopted the most basic scheme, the stop and wait ARQ technique. The overhead of such a technique will be studied and discussed.

Assume that the time to send 1 frame of data is

$$T_f = t_{frame} + 2t_{prop} + t_{ack} + t_{proc} \qquad (9.1.1)$$

Where,

t_{prop} = Propagation time from source to destination

t_{frame} = Transmission time of a frame

t_{proc} = Total processing time at stations for 1 transmission

t_{ack} = Time to transmit an acknowledgement

Nr = Expected number of transmission of a successful frame

$$\alpha = \frac{\text{Propagation Time}}{\text{Transmission Time}} \qquad (9.1.2)$$

Assume that transmission time is normalized to 1 and therefore propagation time is α.

p is the probability that a single frame is in error

$$\text{Time taken to send 1 frame} = N_r(2t_{prop} + t_{frame} + t_{ack} + t_{proc}) \qquad (9.1.3)$$

9.1.2 GATP and Stop & Wait (SAW)

The total time taken to send a successful frame of data is

$$T = Nr\,(2t_{prop} + t_{frame} + t_{ack} + t_{proc}) \qquad (9.1.4)$$

The utilization or efficiency is,

$$U= \frac{(h/h+d)^{*}\ t_{frame}}{N_r(2t_{prop} + t_{frame} + t_{ack} + t_{proc})} \qquad (9.1.5)$$

Where $N_r = 1/(1\text{-}p)$, h=header size, d=useful data size

Using

$$U= \frac{(h/h+d)^{*}(1-p)}{2\,\alpha + 1 + (t_{ack} + t_{proc})/\ t_{frame}} \qquad (9.1.6)$$

$$U= \frac{(h/h+d)^{*}(1-p)}{2\,\alpha + 1 + (t_{proc})/\ t_{frame}} \qquad (9.1.6)$$

** Notes: t_{ack} is the processing time of acknowledgement packet which is neglected since its 100 times smaller that t_{proc} as shown below in Section 9.1.3*

9.1.3 Processing Overhead at Stations

An estimated 30123 and 1000 lines of machine instructions from the GATP programs are processed at the server and client stations respectively. A processor of Pentium 1.5Ghz with a capability of processing $1.5\text{x}10^9$ instructions in a single clock cycle is used. Therefore the processing time is obtained for the stations is as shown in the Table 4 below. The average processing time, t_{proc} is $1.38x10^{-14}$s. The time to process an acknowledgement packet which is smaller than a frame since it only contains no data at the client stations would be $4.44x10^{-16}$.

Table 9.1.1: Processing Time Computation

Station	Processor /hz	Clock Period/s	Processor Instructions per cycle	Total Instructions	Time taken/s
Server	1.50E+09	6.67E-10	1.50E+09	30123	1.34E-14
Client	1.50E+09	6.67E-10	1.50E+09	1000	4.44E-16

9.1.4 GATP Overhead in LAN with SAW ARQ

LAN is considered in this study Utilization in GATP using SAW. In LANs distances typically range from 0.1 to 10km and data rates range from 10 to 100Mbps. Link speed for copper medium is approximately 0.67 times the speed of light while link speed is speed of light for optic fiber. Shown below in Tables 9.1.2 and 9.1.3 are the calculated utilization or efficiency of Stop and Wait ARQ on LAN for minimum and maximum frame size respectively. A shorter distance of 0.1 km can be seen in both tables to produce a better utilization than longer distances. Best utilization is achieved with a shorter distance between stations. A slightly higher utilization is achieved when the data rate is slower at 10Mbps compared to 100Mbps. Maximum and minimum frame sizes used in Tables 9.1.2-9.1.5 are obtained from Section 4.3.1, which are 7712 bits and 512 bits respectively. Computation for actual useful data are derived from Equation 9.1.7 in Tables 9.1.2–9.1.5.

Utilization is derived from Equation 9.1.6.

$$\text{Actual Useful Data} = \frac{\text{Frame Size} - \text{Header Size}}{\text{Frame Size}} \qquad (9.1.7)$$

**Notes Header size used is 480 bits from Section 3.3 Figure 3.3.1.*

For example in Table 9.1.2 first row,

For a Distance 0.1km, Link Speed 2.00E+08/ms^{-1}, Frame Size 512 / bits, DataRate 10 /Mbps,

Min Actual Useful data ratio = Frame Size –Header Size / Frame Size
= (512-480)/512
= 0.0625

α =Propagation Time / Transmission Time

= (Distance/Link Speed) / (FrameSize/Data Rate)

= (0.1*1000)/2x10^8 / 512/10*10^6)
= 0.009766

$$\text{Min Utilization, U} = \text{Useful Data Ratio} * \frac{1}{2\,\alpha + 1 + (t_{proc})/\,t_{frame}}$$

$$= 0.0625 * 1/\,[2\text{x}0.009766 + 1 + 1.38\text{x}10^{-14}/(512/10\text{x}10^{6})]$$

$$= 0.0613$$

Table 9.1.2 Study of GATP Efficiency on LAN using Minimum Frame Size Traversing Copper Media

Distance /km	Link Speed /ms^{-1}	Min Frame Size/ bits	Data Rate/ Mbps	Min Actual Useful data ratio	α	Min Utilization
0.1	2.00E+08	512	10	0.0625	0.009766	0.06130
10	2.00E+08	512	10	0.0625	0.976563	0.02116
0.1	2.00E+08	512	100	0.0625	0.097656	0.05228
10	2.00E+08	512	100	0.0625	**9.765625**	0.00304

Utilization increases, as the useful data size is maximal reducing the percentage overhead of header data in a packet, as shown in Tables 9.1.3 and 9.1.4. The utilization would range from 0.004 to 0.937, depending on the strategy of GATP to employ varying packet sizes. However, the SAW ARQ adopted, would see a better utilization when the distance is shorter and data rate not as fast.

Table 9.1.3: Study of GATP Efficiency on LAN using maximum frame size traversing copper media

Distance /km	Link Speed /ms^{-1}	Max Frame Size/ bits	Data Rate/ Mbps	Max Actual Useful data ratio	α	Max Utilization
0.1	2.00E+08	7712	10	0.9378	0.000648	0.93654
10	2.00E+08	7712	10	0.9378	0.064800	0.83011
0.1	2.00E+08	7712	100	0.9378	0.006483	0.92575
10	2.00E+08	7712	100	0.9378	**0.648340**	0.40831

Table 9.1.4: Study of GATP Efficiency using Minimum Frame Size on LAN Traversing Optic Fibre Media

Distance /km	Link Speed /ms^{-1}	Min Frame Size/ bits	Data Rate/ Mbps	Min Actual Useful data ratio	α	Min Utilization
0.1	3.00E+08	512	10	0.0625	0.006510	0.06169
10	3.00E+08	512	10	0.0625	0.651047	0.02714
0.1	3.00E+08	512	100	0.0625	0.065104	0.05529
10	3.00E+08	512	100	0.0625	6.510417	0.00445

Table 9.1.5: Study of GATP Efficiency using Maximum Frame Size on LAN Traversing Optic Fibre Media

Distance /km	Link Speed /ms^{-1}	Max Frame Size/ bits	Data Rate/ Mbps	Max Actual Useful data ratio	α	Max Utilization
0.1	3.00E+08	7712	10	0.9378	0.0004322	0.93694
10	3.00E+08	7712	10	0.9378	0.0432226	0.86314
0.1	3.00E+08	7712	100	0.9378	0.0043222	0.92972
10	3.00E+08	7712	100	0.9378	**0.4322268**	0.50296

The above computation of overhead is that of actual data transmission. An alternative approach is to scout for network conditions. For a LAN copper media network, the utilization is between the range of 0.003 and 0.937 from Tables 9.1.2 and 9.1.3. Thus a periodic sending of data packets could be a feasible means of measuring the best genetic performance. The accuracy of results can be peg to the sampling size. A utilization cost of 0.003 and 0.937 could be used to find the best genes for actual data transmission. Rather than finding a single optimum solution, the entire main gene colony of GATP can be retained and used for subsequent data transmission.

9.1.5 GATP Utilization under Varying Error Rates

Show below in Figure 9.1.1 is the efficiency of SAW in different values of α. It can be seen that the greatest efficiency is achieved when α is low. This only occurs for propagation time being very much shorter than the transmission time. This can be achieved through faster transport media like optic fiber or satellite medium. Figure 9.1.1 actually show the efficiency as error rate is increased. Efficiency as expected decrease with increasing error rate due to retransmission overhead and acknowledgement waiting time. The main overhead of such an approach is attributed mainly to only one frame being in flight as the sender transits into idle state, to wait for acknowledge. This overhead increases as flight time increases or when data rate increases as seen in above Equation 9.1.6.

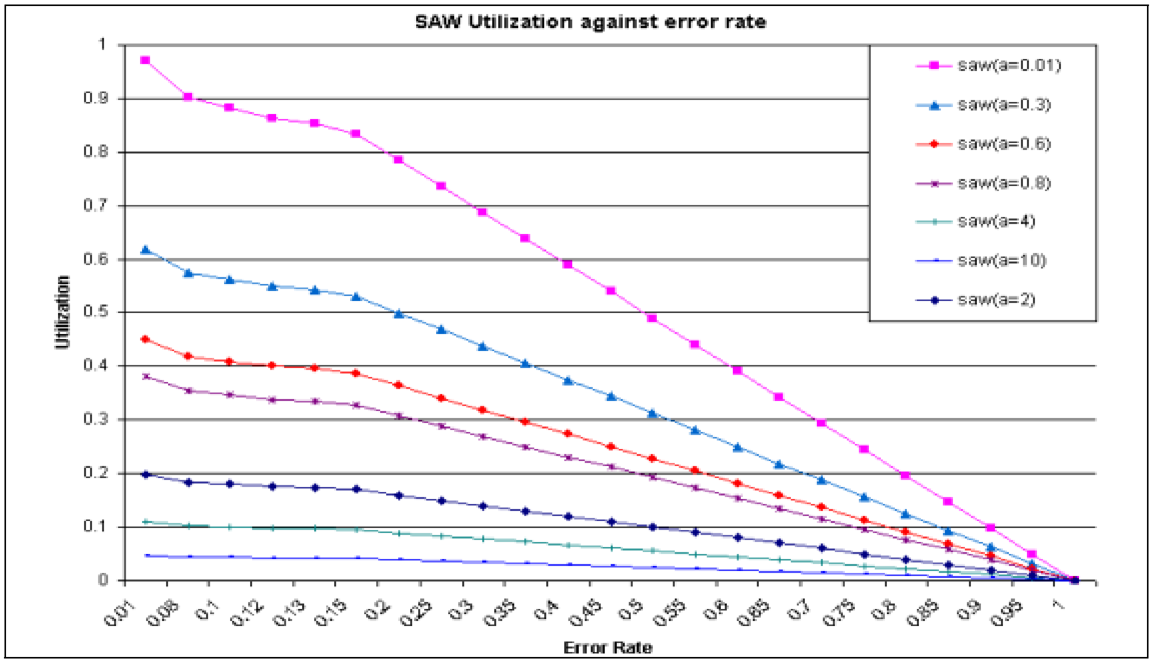

Figure 9.1.1: Effects of α Value on the Performance of SAW ARQ in terms of Utilization under Different Error Rate

9.2 Variable Frame sizes

9.2.1. Maximum Transmission Unit (MTU)

Different network transmission architectures have different physical limit for the number of data bytes in a given frame, which is referred to as the MTU of the network. RFC 1191 has specified the MTU for several architectures as shown in Table 9.2.1 below. IEEE 802.3 Ethernet has a MTU of 1500 bytes.

Table 9.2.1: Maximum Frame Sizes

Network Architecture	MTU/ Bytes
802.3 Ethernet	1500
4 Mb Token Ring	4464
16 Mb Token Ring	17914
FDDI	4352
X.25	576

9.2.2. Throughput Computation for GATP

GATP has employed an IP delivery method with encapsulated GATP control Headers. These do however introduce overheads. Earlier in Section 9.1.2, utilization of GATP was discussed. This section will discuss the throughput of GATP.

Causes of Network Delay:

1. Transmission Delay
2. Propagation Delay
3. Queuing Delay
4. Processing Time

Network delay is caused by transmission delay, propagation delay, queuing delay, and processing time at stations. The queuing time is taken to be very much smaller than the propagation time and transmission time and can be neglected for simplicity of computation. The transmission time for one packet is t_f. When no error occurs, acknowledgement packet arrives at the transmitter site, t_{ack} or βt_f seconds after transmission. The t_{ack} used in this section is the time that the acknowledgement packet takes to reach the sending station. Thus it includes the processing, and propagation time. Retransmission-Time-Out, t_{out} is set as $2*t_{ack}$, so that its twice the computed time, t_{ack} similar to TCP

Packet Error Probability is,

$$p = 1 - (1-e)^{d+h} \qquad (9.2.1)$$

Where h= Header size in bits, d=Data size in bits, e= Error probability of link with error rate of each bit being independently.

Retransmission will begin after time t_{ret},

$$t_{ret} = t_f + t_{out} \qquad (9.2.2)$$

$$= t_f(1 + 2\beta)$$

Total Time, for a single packet transmission including retransmission,

$$T = t_f + R(1 + 2\beta)\, t_f \qquad (9.2.3)$$

Where R is the number of retransmission.

AverageTotal Time, for a single successful packet including retransmission time,

$$\check{T} = t_f + \check{R}(1 + 2\beta)\, t_f \qquad (9.2.4)$$

Where $\check{R}$ is the number of retransmission.

Probability of packet retransmitting k times,

$$r_k = p^k(1-p) \qquad (9.2.5)$$

Number of retransmission,

$$\check{R} = \sum_{K=0}^{\infty} kp^k(1-p) \qquad (9.2.6)$$

$$= p / (1-p)$$

AverageTotal Time, for a single successful packet including retransmission time,

$$\check{T} = \frac{p(1 + 2\beta) + (1-p)}{1-p} \frac{(d+h)}{C} \qquad (9.2.7)$$

$$= \frac{(2p\beta+1)}{1-p} \frac{(d+h)}{C}$$

Maximum packet rate,

$$\lambda_{max} = 1 / \check{T} \qquad (9.2.8)$$

$$= \frac{1-p}{(2p\beta+1)} \frac{C}{(d+h)}$$

$$= \frac{C\,(1-e)^{d+h}}{(2\beta(1-(1-e)^{d+h})+1)\,(d+h)}$$

t_{ack}=2βt_f

$t_{ack} = 2_{tf} + t_{prop} + t_{proc}$

β=(2t_f + t_{prog} + t_{proc}) / 2t_f (9.2.9)

9.2.3. Throughput of GATP with varying Packet Sizes

Using values of *c*= 100Mbps, *β*=2, *e*=0.00001, *h*=480. The table below is obtained. It can be seen that with a smaller d value, a greater inefficiency is incurred as opposed to a larger *d* value. However, In spite of these inefficiencies, the main strength comes in a reduced error probability shown in Equation 6. Average total time, Ť for a single successful packet including retransmission time, is shown in Table 9.2.2 below to be fastest for a smaller sized packet. A higher success and faster transmission is compromised with a larger overhead.

Table 9.2.2: Throughput of Different Packet Sizes

d/bits	p	β	λ_{max} / bits per sec	Useful Data rate/ bits per sec	Useful data/ %	Ť/ s
32	0.00001	1	193327.7	12083.0	6.3	512.0
2432	0.00001	1	32424.5	27079.9	83.5	2912.0
4832	0.00001	1	16973.3	15439.6	91.0	5312.0
7232	0.00001	1	11175.0	10479.4	93.8	7712.0

9.2.4 Analysis of Throughput performance against packet sizes

Shown below in Figure 9.2.1 is a graph of useful throughput data. The throughput response to the varying size of useful data embedded in each packet with header size of 480 bits. Increasing the packet size does increase the throughput on to a certain point of about 800bits as shown in the graph, which correspond to a total packet size of about 1180 bits. Increasing of packet size beyond this value actually reduces throughput. This is due to the higher loss probability of a larger packet. Increasing packet size that could earlier increase throughput, has on the contrary decreased throughput. The higher loss probability of a large packet has become an overhead to throughput.

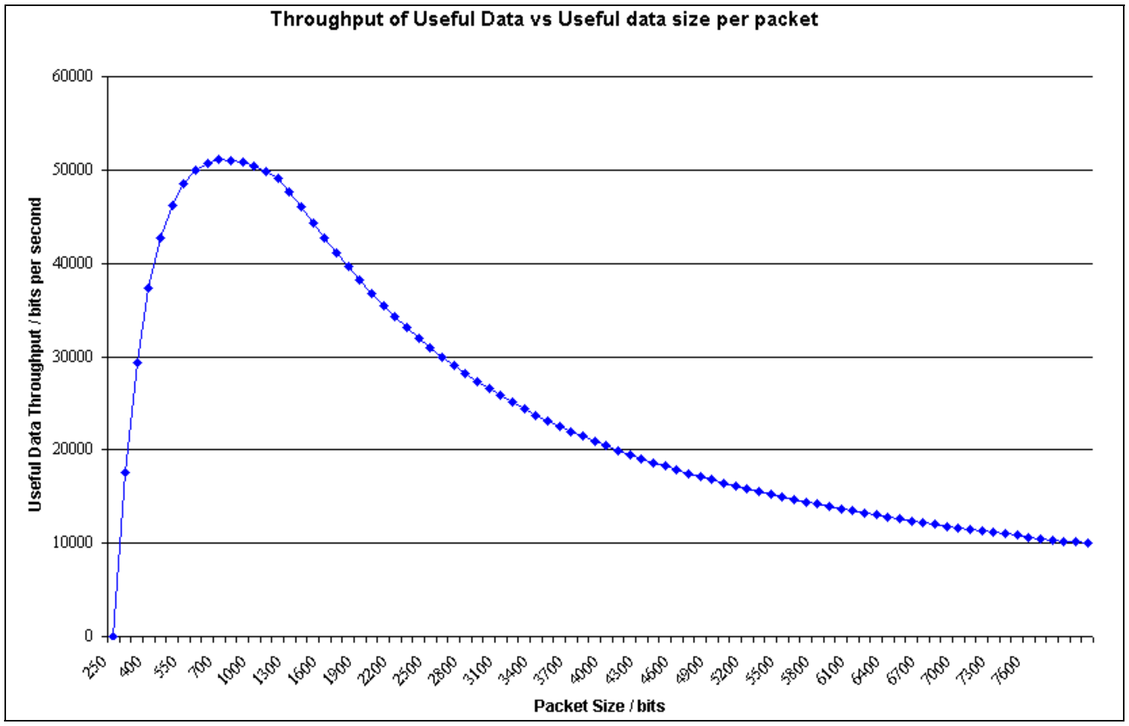

Figure 9.2.1: Throughput of Useful Data against Size of Useful data in each Packet

Figure 9.2.2 below shows the average total time of 1 successful packet transmission inclusive of all retransmission according to the equations explained earlier. It can be seen that the time taken for a successful packet increases steadily as the packet size increases. The speed of transmission is in fact an advantage to round trip time. The rate of increase of time in figure 3 can be seen to be higher after packet useful data size 800 bits, where error rate of the larger packet has diminished the benefits of increasing packet size.

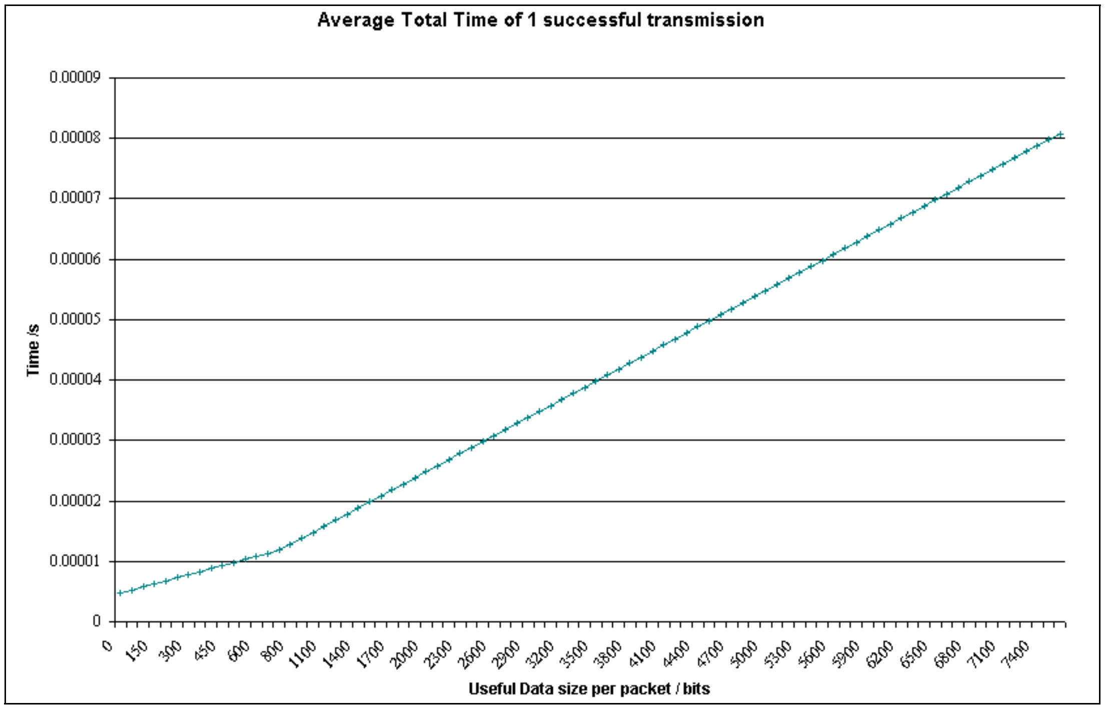

Figure 9.2.2: Average Total Time to Send a Successful Packet against Size of Useful Data in each Packet

The header size of 480 bits consisted of IP header of 160 bits and GATP header of 320 bits. Compression of GATP header is very feasible, since only IP header is necessary for transport. Such compression will allow a larger header size with more QoS factors to be implemented. According to Moore's Law, processing capability is ever increasing. These factors of improving compression and processing power can certainly reduce processing time and header size in time to come.

9.3 Automatic Repeat Request (ARQ) Schemes

There are basically three types of ARQ being, Stop & Wait(SAW), Selective Reject(SR), and Go-Back-N(GbN). The current implementation of GATP has chosen to adopt SAW for its simplicity.

9.3.1 SAW ARQ

This was explained earlier in section 9.1.

$$U = \frac{(h/h+d)*(1-p)}{2\alpha + 1 + (t_{ack} + t_{proc})/\ t_{frame}} \qquad \textbf{(9.3.1)}$$

$$= \frac{(h/h+d)*(1-p)}{2\alpha + 1 + (t_{ack} + t_{proc})/\ t_{frame}}$$

$$= \frac{(h/h+d)*(1-p)}{2\alpha + 1 + (t_{proc})/\ t_{frame}}$$

**Notes: t_{ack} is negligible as discussed earlier in section 9.1.*

9.3.2 SR ARQ

This method retains the channel utilization efficiency of Go-Back-N ARQ and yet improves on the retransmission method where retransmission of single error frame is allowed rather than a mandatory entire window retransmitting. Out of order frames are also retained in the buffer. The maximum window size is 2^{k-1} due to the overlapped sender and receiver windows.

When packets acknowledgement of first frame arrives at source before the sending of N packets,

$$U = (h/h+d)(1-p) \qquad N > 2\alpha + 1 \qquad \textbf{(9.3.2)}$$

When acknowledgment of first packet arrive at source after the sending of N packets,

$$U = \frac{N*(h/h+d)*(1-p)}{2\alpha + 1 + (t_{ack} + t_{proc})/\ t_{frame}} \qquad N < 2\alpha + 1 \qquad \textbf{(9.3.3)}$$

$$= \frac{N*(h/h+d)*(1-p)}{2\alpha + 1 + (\ t_{proc})/\ t_{frame}}$$

**Notes: t_{ack} is negligible as discussed earlier in section 9.1.*

It can be seen from Equations 9.1.6, 9.3.2 and 9.3.3, that SR ARQ has a significantly higher utilization than SAW. The waiting of a single packet in flight for SAW ARQ wastes utilization. SR ARQ gains utilization advantage by allowing packets to be sent while waiting for acknowledgement. In fact, when the acknowledgment packet returns before *N* packets are sent, SR ARW is at least double the efficiency of SAW ARQ. If acknowledgment takes very long, much longer than the sending of *N* packets, and this delay is acceptable, efficiency is at best, *N/2* times faster than SAW ARQ.

9.3.3. GbN ARQ

Go-Back-N ARQ is designed to allow continuous frames to utilize channel while waiting for acknowledgement. This will cut down the idle time of source. A limited number of frames, which are collectively referred to as the window size is used in this technique. The source will transmit the entire window of frames without waiting for the acknowledgement of each frame. A time out at the end of the window transmission will see the source re-transmitting the entire window. An acknowledgement from the destination can also allow the source to retransmit the frames starting from the first lost frame. This allows efficient utilization of the channel. The maximum window size is 2^n-1.

For Go back N ARQ, the *K* frames are retransmitted upon an error. Using *Nr* as the expected number of transmission of a frame, below is the calculation of utilization.

$$Nr = \sum_{i=1}^{\infty} f(i)p^{i-1}(1-p) \qquad (9.3.4)$$

Using f(i), total number of frames transmitted if original frames takes *i* transmission,

$$f(i)= 1+(i-1)K$$

$$Nr = \frac{1-p+Kp}{1-p} \qquad (9.3.5)$$

K is approximated as $(2\alpha +1)$ for $N>2\alpha +1$ and $K =N$ for $N<(2\alpha +1)$.

$$Nr = \frac{1+2\alpha p}{1-p} \qquad K=2\alpha+1 \qquad (9.3.6)$$

$$Nr = \frac{1-p+Np}{1-p} \qquad K=N \qquad (9.3.7)$$

$$U = \frac{(h/h+d)*(1-p)}{1+2\alpha p} \qquad N>2\alpha+1 \qquad (9.3.8)$$

$$U = \frac{N*(h/h+d)(1-p)}{(2\alpha + 1+ (t_{proc})/ t_{frame})(1-p+Np)} \qquad N<2\alpha+1 \qquad (9.3.9)$$

9.3.4 Comparison of ARQs

A comparison of the utilization of the different ARQ schemes is done in this section to show the efficiency of each scheme.

9.3.4.1 SAW & GbN

This section will compare the efficiency of SAW over GbN. The equations below show that SAW ARQ is less efficient that GbN when error rate is not zero.

$$\frac{U_{saw}}{U_{GbN}} = \frac{1+2\alpha p}{2\alpha + 1+ (t_{proc})/ t_{frame}} \qquad N>2\alpha+1 \qquad (9.3.10)$$

$$\frac{U_{saw}}{U_{GbN}} < p$$

$$\frac{U_{saw}}{U_{GbN}} = \frac{(1-p+Np)}{N} \qquad N<2\alpha+1 \qquad (9.3.11)$$

$$\frac{1}{N} \quad <= \quad \frac{U_{saw}}{U_{GbN}} \quad <= 1$$

The utilization of GbN ARQ seen in Equations 9.3.10 and 9.3.11, are better than SAW ARQ shown in Equation 9.1.6. If the acknowledgment packets take very much longer than the sending of *N* packets, then efficiency of GbN is slightly better than SAW. However, when acknowledgment is slow very much slower than then sending of *N* packets, then GbN has a at best *N* times the efficiency of SAW. However, GbN when compared to SR ARQ, has slightly worse efficiency than SR ARQ when acknowledgment is faster than the sending of *N* packets. However, when acknowledgement is slower than sending of *N* packets, and error rate is very low, then GbN and SR have similar efficiency. This efficiency will favour SR as error rate increases. SAW seems to have poor efficiency and is only almost as good as SR and GbN when the acknowledgment is fast.

9.3.4.2. SAW & SR

This section will compare the efficiency of SAW over SR. Selected Reject Method is more efficient than SAW in all circumstances of error conditions. It can be seen from the below equations that as *N* and *α* increases, the efficiency of SR outperforms SAW proportionally**.**

$$\frac{U_{saw}}{U_{GbN}} = \frac{1}{2\alpha + 1 + (t_{proc})/t_{frame}} \qquad N > 2\alpha + 1 \qquad (9.3.12)$$

**Notes: t_{ack} is negligible as discussed earlier in section 9.1.*

$$\frac{1}{2} <= \frac{U_{saw}}{U_{GbN}} <= \frac{1}{2\alpha + 1}$$

$$\frac{U_{saw}}{U_{GbN}} = \frac{1}{N} \qquad N < 2\alpha + 1 \qquad (9.3.13)$$

9.3.5. Performance of ARQ

The performance of the ARQ shall be studied in terms of utilization. The scalability of GATP to extend beyond a smaller LAN network shall be studied in this section.

9.3.5.1 Short Distance Performance

Figure 8 below shows the utilization for close proximity travel of 100m, with link speed of $2.x10^{8}ms^{-1}$ and transmission speed 100Mbps. Performance of SAW in short distance is not as efficient as other ARQs. SAW ARQ achieved a utilization of 0.47. SR and GbN ARQs achieved maximum efficiency constrained by the header size up to an error rate of 0.47. The performance of GbN is optimum at window size 100, but SR ARQ increases utilization with larger window size.

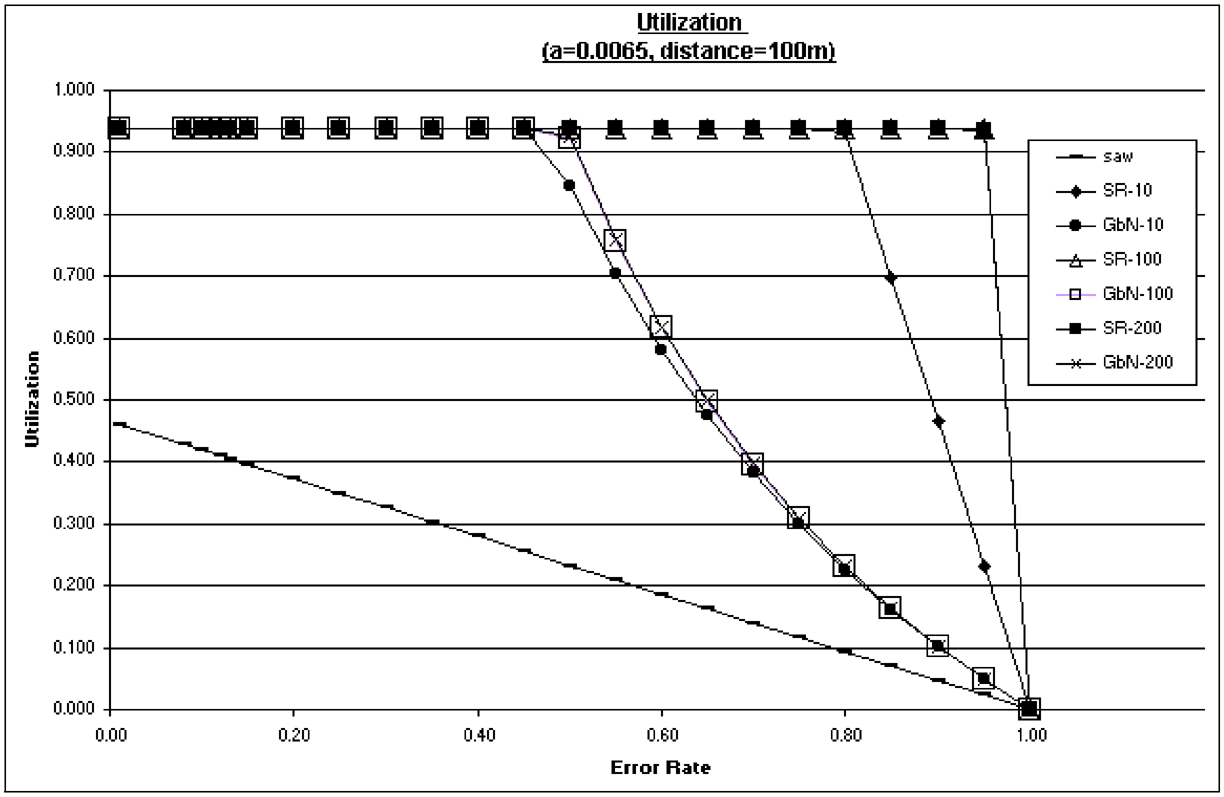

Figure 9.3.1: ARQ Utilization under Different Error Rates with α =0.0065, distance=100m

9.3.5.2 Long Distance Performance

However in longer distances, ∂ increases due to longer propagation time. Figure 7 below shows the utilization of maximum packet sizes. SAW ARQ achieved a zero error rate utilization of 0.0007 as compared to 0.47 in shorter distance earlier. GbN ARQ with window size 10 achieved a zero error rate utilization of 0.0057. GbN ARQ's utilization improved when window size increased from 10 to 100. SR ARQ with largest window size 200 outperforms all ARQs with an optimum utilization of 0.0093 up to error rate of 0.97.

The effects of increasing window sizes can be seen to increase utilization up to a certain point for GbN ARQ but proportionally throughout for Selective Reject ARQ. It can be seen that, SR scheme actually improves utilization to maximum as window sizes increases. GbN was only able to increase efficiency to maximum when error rate is low by increasing window size. This has displayed the effects of improving efficiency through increasing window size for the two schemes; Gbn and SR.

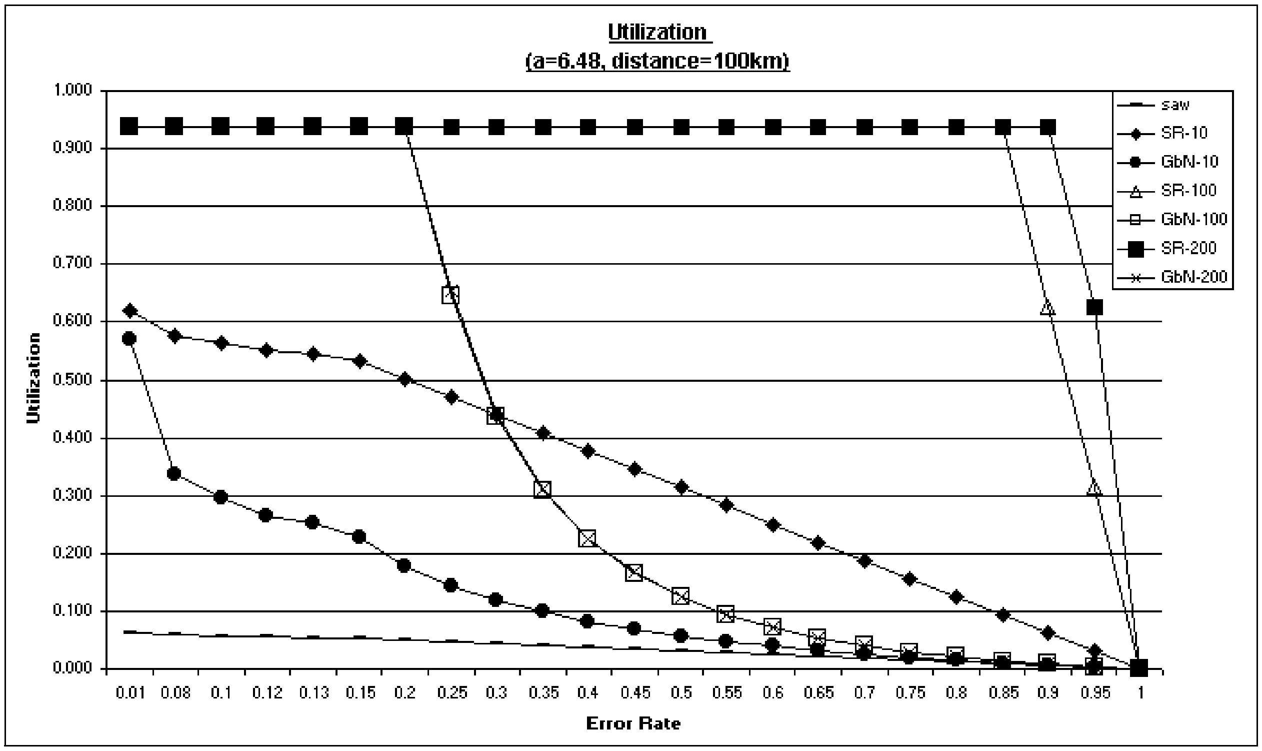

Figure 9.3.2: ARQ Utilization under Different Error Rates with α =6.48, distance=100km

9.3.5.3 Scalability of GATP

Thus by adopting an acknowledgement of larger window size, the utilization of SAW ARQ in GATP can be improved. This implementation of GATP is only for SAW ARQ. However, window sizing can be implemented in the option header field of GATP as discussed earlier. Adopting the best performing ARQ, SR ARQ scheme was shown to increase efficiency. Using the largest window size possible, 2^{k-1} where k is the number of bits available in the header field, further improves SR ARQ. Complexity is the only drawback, although such an approach combined with header compression, will improve optimization to maximal even at high error rates. Alternatively, utilization of GATP can be improved through a multiple streaming technique to upscale the number of concurrent streams to increase utilization.

Chapter 10

Conclusion

GATP allows reconfiguration and evolution of networking protocols to adapt to network environment. Adaptation was achieved by GAs through selection of the best network protocol configuration at run time. The introduction of sub colony and different sampling sizes were shown to provide adaptation of varying speed. A smaller sampling size and larger sub colony provide the fastest adaptation and vice versa.

Through weighted and single fitness functions for GATP, the networking environment is shown to be a dynamic landscape and multi objective problem. The issues of dynamic landscaping were also explored.

GATP employed a MOGA technique to allow a better satisfaction of QoS in networking. Although the MOGA tournament process is useful for discovering the best-fit solution, the overall population performance suffers. GATP has been shown to provide QOS satisfaction, configurability and adaptability. GATP is able to better network performance in jitter, round trip time and even throughput. However, due to the inefficiency of GATP in terms of overheads, it cannot reach beyond certain high levels of QoS. Although GATP lags UDP and TCP due to its inefficiency, it has potential in achieving QoS and adaptability.

Work in this thesis also addresses the unique dynamic landscape of the networking environment, and suggests solutions using traditional MOGA which was not built specifically for networking, but is suitable for multiple network QoS achievements. For example, is a scenario when the network environment is assumed to have only 2 distinctive environment of static and dynamic. In a static network environment, GATP adopts a less frequent injection of migrant population, as the main colony need not traverse elsewhere. On the contrary, in a dynamic network environment, frequent injection of migrant population is necessary to detect changes and shift the main colony towards best performance. Strategies to enhance a greater sensitivity and faster shifting of colony towards best performance would require a more frequent, diverse migrant population. Alternatively, GATP can be left unchanged with a fixed main and migrant colony size as a less optimal approach, due to the lack of shifting signals in dynamic environment. This approach creates too much noise in static environment. This dynamic landscaping problem in networking is exemplified in GATP as a result of the studies on the low-level packet behavior. GATP uses low-level packet changes to achieve QoS and efficiency, opposed to TCP's windowing technique. Other protocols like TCP and UDP, follow the size of MTU according to different network architectures.

GA is used to find the best solution in a pool of 64 candidate solutions. The algorithm may not be fully optimized in such a small solution space and will be more effective in a larger solution space. However, there are some issues involving solution space. A smaller search space decreases the search time for the best effort QoS solution which may be necessary in a dynamic environment where the time step to the next change may be extremely short. This is a compromise of search time and best solution. For example assuming that the gene pool is increased to 4096. Each gene of the original 64 candidates solutions now represent a group of another 64 solutions. There is a greater chance of solutions being churned out within the same group as compared to the earlier case of having only 64 candidates. This may be unnecessary if all solutions within the same group provide the same fitness. A possible improvement to the gene pool would be to offer a finer calibration through a greater gene pool only after solutions in the higher groups have been fully explored. A small gene pool accomplishes a breadth first search and upon exhaustion of solutions, a further depth search can be done. This will reduce the overheads of GATP to only carry a larger header when depth search is launched.

GATP can be deployed in multicasting operations. GATP when configured for the entire routed data transmission may only be as good as the weakest link. Grouping networks in multicasting prevents replicated data transmission. At routers where data is replicated to different destinations, different GATP configurations can be used to achieve optimal performance. This will allow QoS oriented multicasting session where the quality of data transmission is based on what an end user would like to have. In fact, routers, which disseminate replicated multicasting information, can configure GATP packets to achieve adaptation for these networks.

References

[1] G. Ghinea, J. P. Thomas, R. S. Fish, “Multimedia, Network Protocols and Users-Bridging the Gap**”,** Proceedings of the Seventh ACM International Conference on Multimedia (Part 1), pp. 473-476, October 1999.

[2] Rob Procter, Mark Hartswood, Andy McKinlay , Scott Gallacher, “An Investigation of the Influence of Network Quality of Service on the Effectiveness of Multimedia Communication”, Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work, pp. 160-168, November 1999.

[3] Hong, D.W.K. Yoo, J.H. Kim, W.S. Hong, C.S., “An Integrated Service and Network Management System for Point-to-Multipoint Reservation Service in an ATM Network”, Network Operations and Management Symposium, 2002 (NOMS 2002), IEEE/IFIP, pp. 545 –558, 2002.

[4] Jian-Hao Hu, Yeung, K.L., “IRED: A Router Algorithm for Supporting Integrated Services in the Internet”, International Conference on Communications, 2002 (ICC 2002), IEEE, Vol. 4, pp. 2358 –2362, 2002.

[5] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, Resource, “ReSerVation Protocol (RSVP)”, Version 1, Functional Specification RFC 2205, September 1997.

[6] Salles, R.M., Barria, J.A., “Utility-Based Scheduling Disciplines for Adaptive Applications over the Internet”, IEEE Communications Letters, Vol. 6, Issue 5, pp. 217 – 219, May 2002.

[7] Fang Hao; Zegura, E.W.; Ammar, M.H., “QoS routing for Anycast Communications: Motivation and an Architecture for DiffServ Networks”, IEEE Communications Magazine, Vol. 40, Issue 6, pp. 48 –56, June 2002.

[8] C. Williamson, “Internet Traffic Management”, IEEE Internet Computing Vol. 15, Issue 4, pp. 70-74, Nov-Dec 2001.

[9] Sheng-Uei Guan and Zhiqiang Jiang, “A New Approach to Implement Self-modifying Protocols”, IEEE International Symposium on Intelligent Signal Processing and Communication Systems, pp. 539-544, Nov. 2000.

[10] Sheng-Uei Guan and Zhiqiang Jiang, “An Adaptable QoS-centric Transport Protocol

Based on Genetic Algorithms" Proceedings of the IASTED, International Conference on Advances in Communications, Anaheim, CA92804: ACTA Press, pp. 13-18, 3 July 2001.

[11] PK McKinley, E.P. Kasten, S.M. Sadjadi and Z.Zhou, "Realizing Multi-Dimensional Software Adaptation," 16th Annual ACM International Conference on Supercomputing, New York City, NY, June 23rd, 2002.

[12] Fish, R.S.; Graham, J.M.; Loader, R.J., "DRoPS: kernel support for runtime adaptable protocols", Proceedings of the 24th Euromicro Conference, 1998, Vol. 2, pp. 1029 –1036, 1998.

[13] Gary T. Wong, Matti A. Hiltunen, and Richard D. Schlichting, "A Configurable and Extensible Transport Protocol", Proceedings of the 20th Annual Conference of IEEE Communications and Computer Societies (INFOCOM 2001), Anchorage, Alaska, pp. 319-328, April 2001.

[14] Wen-Ke Chen, Matti A. Hiltunen, and Richard D. Schlichting, "Constructing Adaptive Software in Distributed Systems", Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21), Mesa, AZ, pp. 635-643,

April 2001.

[15] B.C. Li and K. Nahrstedt, “A Control Based Middleware Framework for Quality of Service Adaptations”, IEEE Journal on selected areas in Communications, Vol. 17, No.9, September 1999.

[16] B. Stiller, C. Class, M. Waldvogel, G. Caronni, and D.Bauer, “A Flexible Middleware for Multimedia Communication: Design Implementation and Experience,” IEEE Journal of Selected Areas in Communications, Vol. 17, No. 9, pp. 1580-1598, September 1999.

[17] Matti A. Hiltunen, Richard D. Schlichting, Carlos A. Ugarte, and Gary T. Wong , “Survivability through Customization and Adaptability: The Cactus Approach.” DARPA Information Survivability Conference and Exposition (DISCEX 2000), pp. 294-307, January 2000.

[18] Nina T. Bhatti, Matti A. Hiltunen, Richard D. Schlichting, and Wanda Chiu, “Coyote: A System for Constructing Fine-Grain Configurable Communication Services”,. ACM Transactions on Computer Systems, Vol. 16, No. 4, pp. 321-366, November 1998.

[19] D. Schmidt, D. Box, and T. Suda, “ADAPTIVE: A Dynamically Ensembled Protocol Transformation and Integration and Evaluation Environment”, Concurrency: Practice and Experience, Vol 5, No.4, pp 269-286, June 1993.

[20] M. Hayden, “The Ensemble System,” Technical Report, TR98-1662, Department of Computer Science, Cornell University, Jan 1998.

[21] Doll, A.; Verstege, J.F., “ An Evolution Strategy Based Approach for a Congestion Management System”, Power Tech Proceedings, 2001 IEEE Porto, pp. 6, Vol.1, 2001.

[22] Hsinghua Chou, G. Premkumar, Chao-Hsien Chu, “Genetic Algorithms And Network Design: An Analysis Of Factors Influencing GA's Performance,” eBusiness Research Center Working Paper 09-1999.

[23] Jiang, J., “A Fast Competitive Learning Algorithm for Image Compression Neural Networks”, IEE Electronic Letters, Vol. 32, No. 15, pp. 1380-1381, 1996.

[24] I.K. Karez-Duleba, “Dynamics of Infinite Populations Evolving in a Landscape of

Uni- and Bimodal Fitness Landscapes", IEEE Transactions on Evolutionary Computation, Vol. 5, No.4, August 2001.

[25] J.J. Grenfenstette, "Evolvability in Dynamic Landscapes: A Genetic Algorithm Approach", Proceedings of 1999 Congress on Evolutionary Computation, Vol. 3, pp. 2031-2038, 1999.

[26] M. Wineberg and F. Oppacher, " Enhancing the GA's Ability to Cope with Dynamic Environments", Proceedings of the 2nd Conference on Genetic and Ev. Comp. (GECCO-2000), San Francisco, Morgan Kaufmann, pp. 3-10, 2000.

[27] F. Oppacher and M. Wineberg, " Reconstructing the Shifting Balance Theory in a GA: Taking Sewall Wright Seriously", Proceedings of 2000 Congress on Ev. Comp. (CEC2000), IEEE Press, pp. 219-226, 2000.

[28] SK Oh, C.Y. Lee, and J.J. Lee, " A New Distributed Evolutionary Algorithm for Optimization in Non Stationary Environments", Proceedings of the Congress on Evolutionary Computing 2002 (CEC '02), Vol. 1, pp. 279-284, 2002.

[29] Cedeno, W. Vemuri, V.R., "On the Use of Niching for Dynamic Landscapes", IEEE International Conference on Evolutionary Computation, pp. 361 –366, 1997.

[30] Dias, A.H.F.; de Vasconcelos, J.A., "Multiobjective Genetic Algorithms Applied to Solve Optimization Problems", IEEE Transactions on Magnetics, Vol. 38, Issue 2, Part 1, pp. 1133 –1136, March 2002.

[31] Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, Vol. 6 Issue 2, pp. 182 –197, April 2002.

[32] Fonseca, C.M.; Fleming, P.J., "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms. I. A unified formulation", IEEE Transactions on Systems, Man and Cybernetics, Part A, Vol. 28 Issue 1, pp. 26 –37, Jan. 1998.

[33] Rodriguez-Vazquez, K.; Fleming, P.J., "Multi-objective Genetic Programming for Nonlinear System Identification", Electronics Letters, Vol. 34, Issue 9, pp. 930 –931, 30 April 1998.

[34] Ishibuchi, H.; Murata, T., “A Multi-Objective Genetic Local Search Algorithm and its Application to Flowshop Scheduling”, IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, Vol. 28, Issue 3, pp. 392 –403, Aug. 1998.

[35] Bingul, Z., Sekmen, A.S., Palaniappan, S.; Zein-Sabatto, S., “Genetic Algorithms Applied to Real Time Multiobjective Optimization Problems,” Proceedings of the IEEE Southeastcon 2000, pp. 95 –103, 2000.

[36] Obayashi, S., Sasaki, D., Takeguchi, Y., Hirose N., “Multiobjective Evolutionary Computation for Supersonic Wing-Shape Optimization”, IEEE Transactions on Evolutionary Computation, Vol. 4, Issue 2, pp. 182 –187, July 2000.

[37] Haiming Lu Yen, G.G., “Multiobjective Optimization Design via Genetic Algorithm,” Proceedings of the 2001 IEEE International Conference on Control Applications (CCA '01), pp. 1190 –1195, 2001.

[38] Costa, L.; Oliveira, P., “An Evolution Strategy for Multiobjective Optimization”, Proceedings of the Congress on Evolutionary Computation 2002 (CEC '02), Vol. 1, pp. 97

–102, 2002.

[39] Farina, M.; Sykulski, J.K., “Comparative Study of Evolution Strategies Combined with Approximation Techniques for Practical Electromagnetic Optimization Problems”, IEEE Transactions on Magnetics, Vol 37, Issue 5, Part 1 , pp. 3216 –3220, Sept. 2001.

[40] William Stallings. “Data and Computer Communications”, Prentice Hall Inc., Fifth Edition, 1997, pp. 158-197.

Appendix A

<u>**Publications**</u>

1. "**Intelligent Protocol Design Using Genetic Algorithm**", International Conference in Networks 27-30 Aug 2002, Singapore, Chek Liang Dominic Boh, SU Guan.

2. **"An Intelligent Transport Protocol Based on Genetic Algorithm",** Submitted to Journal, SU Guan, Chek Liang Dominic Boh.

3. **"Genetic-Algorithm Based Transport Protocol for Communicating under a Dynamic Networking Environment**", Submitted to Journal, SU Guan, Chek Liang Dominic Boh.