A GENETIC ALGORITHM FOR NETWORK TRANSPORT PROTOCOL PARAMETER OPTIMIZATION

by

Adrián Granados Murillo

B.Sc., Instituto Tecnológico de Costa Rica, 2002

A thesis submitted to the Department of Computer Science
College of Arts and Sciences
The University of West Florida
In partial fulfillment of the requirements for the degree of
Master of Science

2009

The thesis of Adrián Granados Murillo is approved:

Marco M. Carvalho, Ph.D., Committee Member	Date
Dennis L. Edwards, Ph.D., Committee Member	Date
Sharon J. Simmons, Ph.D., Committee Chair	Date

Accepted for the Department/Division:

Leonard W. ter Haar, Ph.D., Chair	Date

Accepted for the University:

Richard S. Podemski, Ph.D., Dean of Graduate Studies	Date

DEDICATION

I dedicate this thesis to my family, who always supports me from miles away. If it was not for their sacrifice and their love, I could have never been the person I am today. And to Veronica, whose encouragement and sacrifice enabled me to accomplish this goal.

TABLE OF CONTENTS

DEDICATION iv

LIST OF FIGURES vii

ABSTRACT viii

INTRODUCTION 1

CHAPTER I. LITERATURE REVIEW 5
A. Transport Protocol Performance 5
B. Related Work 9
1. Offline Optimization 10
2. Online Optimization 11

CHAPTER II. GENETIC ALGORITHM FOR TRANSPORT OPTIMIZATION 19

CHAPTER III. TECHNICAL APPROACH 21
A. Introduction to Genetic Algorithms 21
B. Optimization Algorithm 24
1. Representation 25
2. Genetic Operators 26
3. Fitness Function 28
4. Stopping Criteria 28
C. Case Study 28

CHAPTER IV. EXPERIMENTS AND ANALYSIS OF RESULTS 34
A. Experimental Setup 34
B. Preliminary Analysis 36
C. Simulation Results and Discussion 40
D. Improving the Efficiency of the Optimization Algorithm 42
E. Online Adaptation to Network Conditions 47

CHAPTER V. CONCLUSIONS 51

REFERENCES 54

LIST OF FIGURES

1. A network topology with a multi-hop communication path. 7

2. The hidden node problem scenario. Node C is hidden from node A, causing collisions at node B if node A and node C transmit at the same time. 8

3. The exposed node problem scenario. Node B's transmission to node A prevents node C from sending packets to node D. 8

4. The crossover operator combines genetic code at randomly chosen points of the parents' genome. 22

5. The mutation operator randomly alters sections of the individual's genome. 23

6. Pseudocode of a simple genetic algorithm (Russell & Norvig, 2002, p. 119). 23

7. Pseudocode of the genetic algorithm for transport optimization. 25

8. Emulated wireless network. There is a maximum of 8 hops (from node 1 to node 9). Circles indicate the transmission range of a node. 34

9. Maximum throughput achieved by the Reliable User Datagram Protocol from 1 to 8 hops. 37

10. Maximum throughput achieved by the Reliable User Datagram Protocol over 2 hops when using the parameter values found by the Genetic Algorithm for Transport Optimization. The horizontal line indicates the average maximum throughput when using the optimal configuration. 39

11. Standard deviation of the throughput achieved by the Reliable User Datagram Protocol over 2 hops when using the parameter values found by the Genetic Algorithm for Transport Optimization. The horizontal line indicates the standard deviation of the maximum throughput when using the optimal configuration. 40

12. Online optimization of the Reliable User Datagram Protocol using the Genetic Algorithm for Transport Optimization over 4 and 8 hops. The vertical line indicates the time at which the algorithm converged. 41

13. Performance comparison of the Genetic Algorithm for Transport Optimization (GATO) and its adaptive version (Adaptive-GATO) over a 2-hop path, using population sizes of 10, 30, 60, and 120 individuals. 44

14. Online optimization of a flow using the Adaptive Genetic Algorithm for Transport Optimization over 4 and 8 hops. The vertical line indicates the time at which the algorithm converged. 45

15. Online optimization comparison of the Genetic Algorithm for Transport Optimization (GATO) and its adaptive version (Adaptive-GATO) over a 4-hop path (using averaged data points to facilitate readability). 46

16. Another emulated wireless network with two additional nodes. A secondary connection interferes with the existing flow going through the 8-hop path. 47

17. Dynamic online optimization of a flow over an 8-hop path. At time 780 (vertical line), a secondary flow is started from node A to node B, interfering with the optimized flow and triggering a new optimization task. 48

18. Dynamic online optimization of a flow over an 8-hop path using a predefined initial population. At time 700 (vertical line), a secondary flow is started from node A to node B, interfering with the optimized flow and triggering a new optimization task. The elite group of individuals from the previous run was inserted into the initial population of the next run of the optimization algorithm. 49

19. Comparison of a dynamic online optimization of a flow over an 8-hop path (using averaged data points to facilitate readability). At each restart of the algorithm, the initial population was comprised of either purely random individuals (Random) or a combination of random individuals and the elite members of the previous run of the algorithm (Elite + Random). 50

ABSTRACT

A GENETIC ALGORITHM FOR NETWORK TRANSPORT PROTOCOL PARAMETER OPTIMIZATION

Adrián Granados Murillo

The fields of wireless communications and mobile networking are rapidly growing and changing. In a mobile ad hoc network, the variation of link characteristics, and frequent and unexpected changes in topology, decreases the performance of commonly used transport protocols, which assume that packet drops occur only in the event of network congestion. Link failure and higher bit error rates may also induce a sudden increase of packet losses, triggering the activation of congestion avoidance mechanisms that reduce the overall transmission rate.

In general, a transport protocol has numerous configuration parameters (e.g. window size, retransmission timeout, etc.) that can be adjusted to compensate for these environment effects. Thus, the challenge is to identify what is the best configuration (i.e. combination of parameter values) for a given scenario.

In this work, an optimization approach based on genetic algorithms is used to automatically tune, in real time, the parameter values of a transport protocol so that it can adapt itself to different operating conditions. When compared to an exhaustive search over a reduced problem space, the experimental results show that the proposed algorithm can identify the optimal configuration settings to maximize throughput for end-to-end wireless communication over multiple hops.

INTRODUCTION

The fields of wireless communications and mobile networking are rapidly growing and changing. Mobile ad hoc networks (MANETs) are of great interest for military and rescue applications where an easy and quick deployment of a communication infrastructure is required to support the operations in the battlefield or disaster area. More recently, advances in wireless technologies have led to an increasing popularity of MANETs in other areas of application such as home and office networking, industry, and person-to-person communication. In a MANET, wireless nodes can freely move to organize arbitrary, and temporary network topologies, building an ad hoc communication infrastructure. Compared to wired networks, where nodes have fixed locations and topologies are static, MANET topologies are quite dynamic; therefore, operating conditions cannot be predicted during the network design stage. Sudden and rapid changes in topology can occur as a results of the movement of one or more nodes, signal interference, power failure, and other important factors (Aggelou, 2005). As a result, frequent and unexpected node disconnections and path disruptions are common in MANETs.

In general, the use of protocols for wired networks has been adopted for mobile wireless networking. However, it has been demonstrated that the unpredictable movement of nodes and the nature of wireless communication technology greatly decreases the performance of commonly used network protocols (Calagaz, Chatam, Eoff, & Hamilton, 2004; Elaarag, 2002; Li, Qiu, Zhang, Mahajan, & Rozner, 2008). The lack of adaptability and a design based on assumptions that are only valid for fixed network topologies prevent these protocols from having the same performance that is achieved in wired networks. In

particular, the Transmission Control Protocol (TCP) does not perform as well in MANETs as it does in fixed networks (Calagaz et al., 2004; Elaarag, 2002; Gurtov & Floyd, 2004; Mulabegovic, Schonfeld, & Ansari, 2002).

Normally, protocols for fixed networks are configured and evaluated for generic environments. In addition, once protocols are deployed, they require few or no adjustments because operating conditions in this type of network rarely vary. Moreover, the main assumption in the operation of transport protocols is that adverse conditions, if any, will occur only in the event of network congestion (Gurtov & Floyd, 2004). When congestion occurs, packet drops increase, and transport protocols immediately activate flow control mechanisms to significantly reduce the transmission rate to revert the congestion condition.

In the case of wireless networks, link failure or higher bit error rates produce an increase in the number of packet losses, activating the same mechanisms for congestion avoidance. As a result, protocols unnecessarily and immediately decrease the transmission rate. For this reason, frequent and significant variations in link characteristics have a great impact on the performance of transport protocols in MANETs.

Transport protocols for ad hoc networking need to continuously adapt in order to satisfy the Quality of Service (QoS) requirements of higher-level applications and improve the performance in wireless networks. This continuous adaptation can be accomplished by (a) modifying existing protocols to make them aware of the changes in network conditions that are not related to congestion, or (b) designing protocols that are optimized for wireless networks. These two approaches usually result in protocols that are characterized for having a number of modifiable parameters that still need to be tuned to satisfy the QoS requirements of higher level applications, which often have a need for a network that is optimized along certain characteristics such as reliability, delay, power consumption, or

overhead. However, the task of manually tuning and identifying sets of optimal (or near-optimal) protocol parameter values is combinatorial in nature and not practical.

In this work, a genetic algorithm (GA) is used to automatically tune the parameter values of a transport protocol. The main objective of the proposed algorithm is to optimize the protocol to improve its performance based on one of several potential QoS metrics such as number of retransmissions, transmission delay, and throughput.

A GA is a heuristic search method inspired in Darwin's theory of evolution where populations of individuals compete and only the fittest survive. GAs incorporate the biological principle of natural evolution and genetics to artificial systems and have been successfully applied to numerous problems, including automatic programming, machine learning, economics, social systems, population genetics, and others (Mitchell, 1996).

More importantly, GAs are considered to have a surprising potential as adaptive search techniques to solve optimization problems because GAs can quickly explore an extensive search space, and are highly effective in the presence of noise (Russell & Norvig, 2002). Additionally, GAs have been shown to be successful at finding solutions or optimizations for complex and difficult problems for which traditional search methods are less effective. However, in addition to advantages, GAs also have disadvantages. Their main drawback is their tendency to prematurely converge to a local optimum. This premature convergence occurs when an individual, who is more fit than others at early stages, dominates on the reproduction process and leads to a local optimum convergence, preventing the algorithm to perform a more systematic search that could lead to a global optimum. Another disadvantage of GAs is the time they may take to find a possible solution, which depends on the computational complexity of the function that evaluates the fitness of an individual. Nevertheless, different strategies for genetic code manipulation and approaches for parallel execution have been proposed to avoid the premature convergence problem and other disadvantages of GAs (Mitchell, 1996).

Despite their shortcomings, GAs have been successfully applied to optimization problems in MANETs by exploiting their ability to quickly provide good solutions in highly dynamic contexts (Barolli, Koyama, & Shiratori, 2003; Montana & Redi, 2005; Turgut, Das, Elmasri, & Turgut, 2002), demonstrating that GAs can be used effectively for complex problems in the field of wireless networks, a field where finding estimate solutions would be otherwise impossible or difficult to achieve.

This document is organized as follows. Chapter 1 characterizes the effects that wireless communications have on network transport protocol performance and the different proposals that can been found in the literature that aim to improve the performance of transport protocols in wireless networks. Chapters 2 and 3 describe the thesis proposal, technical approach, and optimization algorithm. Chapter 4 presents the experimental setup and the analysis of results that demonstrate the feasibility of transport protocol parameter optimization using GAs. Finally, Chapter 5 discusses the advantages, disadvantages, and limitations of the proposed optimization algorithm.

CHAPTER I

LITERATURE REVIEW

The literature review can be classified into two distinct categories. The first category consists of research work that illustrates the effects that wireless communications have on the performance of network transport protocols. The second category consists of proposals that aim to improve the performance of these protocols over MANETs. A complete survey of this large body of work is beyond the scope of this chapter; results that directly impact this research work are summarized.

Transport Protocol Performance

Various authors have investigated the causes of performance degradation that is experienced by transport protocols in wireless networks. Much of the previous research studies have been focused in the analysis of the effects that wireless environments have on TCP, a protocol vastly used in wired networks and the Internet. TCP is a connection-oriented protocol that provides reliable and in-order packet delivery (Information Sciences Institute, 1981; Tanenbaum, 2002). This protocol can adapt itself to different network conditions using mechanisms for flow control that are triggered when packet drops are detected. The flow control algorithms found in several TCP implementations were designed under the assumption that packet drops occur only in the event of network congestion and not as a result of damaged packets, given that bit errors in wired networks were and are still practically negligible.

Nevertheless, packet drops in MANETs not only occur from network congestion but also from higher bit error rates, link failures, limited and variable bandwidth, and node mobility. In wireless networks, higher bit error rates produce a higher number of damaged

data and acknowledgment packets, whereas node movement and power failure may cause periodic disconnections of mobile nodes. According to researchers, the most important cause for TCP performance degradation in MANETs is the TCP sender's inability to distinguish packet losses that are due to congestion from those that occur when the link fails (Aggelou, 2005; Calagaz et al., 2004; Elaarag, 2002; Holland & Vaidya, 2002; Xylomenos, Polyzos, Mahonen, & Saaranen, 2001).

In general, TCP activates its congestion control mechanisms when packet losses are detected, reducing the number of outgoing packets with the goal to stabilize the state of the flow. However, TCP also activates these mechanisms when a link or route fails and packets are dropped, unnecessarily reducing the flow of packets even after links and routes are reestablished (Lochert, Scheuermann, & Mauve, 2007). This behavior results in immediate and undesired performance degradation in the presence of node mobility, bit errors, power failure, and network handoff (Gurtov & Floyd, 2004; Xylomenos et al., 2001).

Mechanisms to provide the TCP sender with information about link and route failures can significantly improve the performance of TCP (Holland & Vaidya, 2002). For example, by monitoring explicit congestion notification (ECN) or ICMP (Internet Control Message Protocol) destination unreachable messages, TCP can determine that detected packet drops are caused by link failures and are not a result of network congestion.

Interference is also an important cause of performance degradation in wireless networks (Al Hanbali, Altman, & Nain, 2005; ElRakabawy, Klemm, & Lindemann, 2005; Fu et al., 2005; Jain, Padhye, Padmanabhan, & Qiu, 2003). Even in the presence of a single TCP flow, the successive transmission of packets at intermediate nodes, through a multi-hop path, interferes with each other as packets move toward the destination (Figure 1). Protocol performance degrades as the number of hops traversed increases

because communication happens over a shared medium and is affected by the induced interference of hidden (and exposed) nodes.

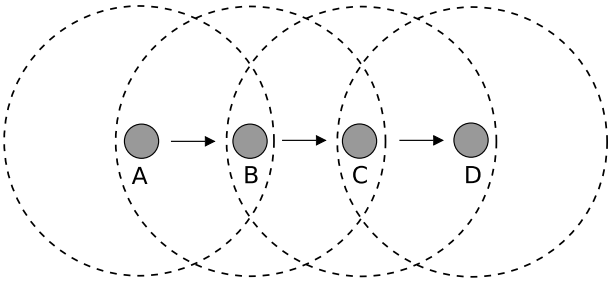


Figure 1. A network topology with a multi-hop communication path.

In wireless networks, the hidden node problem (Figure 2) refers to a scenario where a node is out of range from other nodes. For example, consider a scenario with three nodes. Node A and node C have a packet to transmit to node B. Because node C is outside the transmission range of node A, node C cannot detect node A's transmission to node B; node C is hidden from node A. If node A and node C transmit at the same time, there will be packet collisions at node B. The hidden node problem is partially solved not at the transport level, but usually at the Medium Access Control (MAC) level by a two-way handshake that precedes the transmission called virtual carrier sensing (Tanenbaum, 2002). However, the problem may persist in ad hoc networks because the range at which a MAC frame can be decoded and successfully received is always shorter than the range at which a transmission can be detected, hence interrupting the reception of virtual carrier sensing messages (Cordeiro & Agrawal, 2006; Fu et al., 2005).

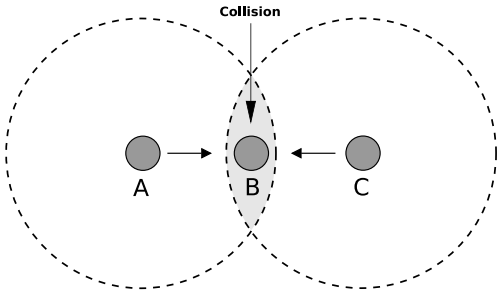


Figure 2. The hidden node problem scenario. Node C is hidden from node A, causing collisions at node B if node A and node C transmit at the same time.

Conversely, another effect of interference in wireless ad hoc networks is defined as the exposed node problem (Figure 3). In this case, node A and node C are within node B's transmission range, and node A is outside the transmission range of node C. If node B has packet to transmit to node A and node C also has a packet to transmit to node D, node B's transmission would prevent node C from sending a packet to node D, although this transmission would not cause interference at node A.

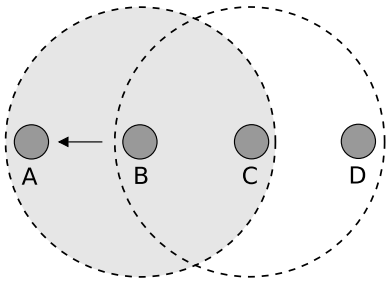


Figure 3. The exposed node problem scenario. Node B's transmission to node A prevents node C from sending packets to node D.

The problems caused by interference are usually addressed by link-layer mechanisms such as virtual carrier sensing and retransmission of damaged packets. However, the exchange of messages to perform virtual carrier sensing in the presence of

multiple nodes and the excessive number of retransmissions may indirectly degrade the performance of higher layer protocols, including transport protocols like TCP.

Related Work

Various proposals to improve the performance of transport protocols in wireless networks can be found throughout the literature. The characterization of these proposals made by Al Hanbali et al. (2005) suggests that they can be classified into (a) layered proposals and (b) cross layer proposals. In layered proposals, the performance of a transport protocol is improved, directly or indirectly, by the adaptation of one of the layers in the network stack. Conversely, cross layer proposals involve two or more layers that share information through a predefined set of interfaces so that transport protocols can adapt accordingly to specific network operating conditions.

Many of the proposals found in the literature have focused on adaptations to improve the performance of the TCP protocol (Calagaz et al., 2004; Elaarag, 2002; El Khayat, Geurts, & Leduc, 2005; ElRakabawy et al., 2005), although new transport protocols have also been designed specifically for wireless networks (Akan & Akyildiz, 2004; Mulabegovic et al., 2002; Navaratnam, Cruickshank, & Tafazolli, 2007; Wu & Rao, 2005).

In general, adapting preexisting protocols allows for seamlessly inter-networking between the wired and the wireless worlds. On the other hand, cross layer based strategies and transport protocols specifically designed for MANETs are more difficult to implement. However, they may achieve better results since underlying protocols can work jointly, allowing the transport protocol to monitor changes in the environment and adapt itself accordingly.

Each of these proposals attempts to overcome the main causes that degrade the performance of TCP in wireless environments, such as high bit error rates, link breakage,

limited and variable bandwidth, and interference. However, it is usually the case that these adaptations do not consider all causes at the same time because of the complexity of the design and implementation. Therefore, they offer better performance only under very specific conditions of the network, where environments often lack frequent and sudden topology changes, multi-hop paths, and induced interference. Also, most of these adaptations lack mechanisms to dynamically tune protocol parameters in order to optimize the performance of individual connections based on the current network conditions.

Given the impossibility of transport protocols to globally address all the challenges arising from the nature of wireless networks, some proposed adaptations focus on the optimization of protocol parameters to improve the performance under a variety of scenarios. In general, optimization approaches can be categorized into (a) offline optimization and (b) online optimization.

Offline Optimization

In offline optimizations, network conditions are known (or assumed), and protocol parameter tuning is achieved through the simulation of several network configurations for which one or more optimization algorithms are executed. The goal of offline optimization is to find suitable protocol parameters that can maximize the performance of the network when operating under certain predetermined conditions. In his work, Suydam (2004) designed an algorithm based on adaptive simulated annealing to optimize the TCP segment and receiver window size in a simulated environment. The goal of this proposal is to maximize the throughput of TCP in real wireless ad-hoc networks. Adaptive simulated annealing is a variation of simulated annealing, which is a search heuristic used for large-scale optimization problems (Russell & Norvig, 2002). It is adaptive because the parameters of the algorithm are automatically adjusted as it progresses, making it more efficient and less sensitive to user defined parameters. The results obtained in this work

demonstrated that the proposed offline optimization algorithm provided a significant increase of the average throughput in ad hoc wireless networks.

El Khayat et al. (2005) proposed an algorithm to classify packet loss causes to improve the performance of TCP with the objective to achieve a better throughput over wireless links. In their work, the authors applied a supervised learning algorithm to automatically infer a packet loss cause classifier from a database of 25,000 losses observed in a thousand simulated random topologies with simulated packet flows. Supervised learning is a machine learning technique which focuses on identifying a mapping from some input variables to some output variables from a sample of observations of these variables (Russell & Norvig, 2002). More precisely, the main goal of supervised learning is to generalize from training data so as to predict the value of the output variables when presented to any valid input set. The classifier constructed by the learning algorithm provided good accuracy at classifying losses for all the simulated network topologies that were considered in the experimental analysis and in actual wireless networks.

It is important to note that the main limitation encountered with offline optimization is that optimal protocol settings may be found only for the network conditions that were considered in the simulations. In addition, once optimal settings are found, they are fixed for all possible scenarios. As a result, a protocol may perform poorly if network conditions significantly vary from those that were considered during the simulation.

Online Optimization

In online optimization the network conditions are unknown, and protocol parameter tuning is achieved through optimizations that are performed dynamically as conditions change. This type of approach requires a constant observation of the environment and monitoring of the current protocol performance, which makes online

optimizations more difficult to implement than offline approaches. Moreover, the online optimization of protocol settings in MANETs should be made as quickly as possible as network conditions may change rapidly.

Some online optimizations of network transport protocol parameters based on heuristic search algorithms can be found in the literature. Rao and Iyengar (2004) proposed an algorithm to dynamically optimize the size of the window for window-based transport protocols. The algorithm was designed to monitor the network conditions and adjust the sending rate in order to achieve a target throughput rate. It uses stochastic approximation methods to stabilize the throughput of the transport stream by adjusting the size of the window if the estimated throughput is above or below a certain threshold. Stochastic approximation algorithms are heuristic-based optimization algorithms that incorporate probabilistic elements to attempt to estimate the solution of a problem from noisy observations. The randomness introduced to the search process aims to speed up the convergence of the algorithm and to make it less sensitive to the noise introduced by modeling or simulation errors (Russell & Norvig, 2002; Ye & Kalyanaraman, 2004).

Wu and Rao (2005) proposed a similar approach called Reliable UDP-based Network Adaptive Transport (RUNAT). In this approach, a protocol based on the User Datagram Protocol (UDP) was designed to dynamically control the transmission rate of the sender based on the statistical behavior of individual connections. The authors defined three different congestion states for a given connection: underutilized, saturated, and overloaded. The main goal of RUNAT is to avoid congestion states above or below the saturated level using a rate control mechanism for interleaving delays between packets. Once the connection congestion is at the saturated level, the sending rate is statistically stabilized using stochastic approximation methods to allow the protocol to achieve maximum throughput.

The results obtained by Rao and Iyengar (2004) and Wu and Rao (2005) demonstrated the effectiveness of statistical control methods for throughput stabilization over wide-area networks under various traffic conditions. However, experiments were not conducted to verify the effectiveness in throughput stabilization in MANETs, although these networks seem to be an ideal ground for the proposed algorithms because of the noisy conditions under which communications are performed.

Akan and Akyildiz (2004) proposed another redesign of a transport protocol capable of performing online optimization of the protocol parameters. In this case, the authors designed a new transport layer that incorporated an adaptive congestion control mechanism to dynamically adjust the AIMD (Additive-Increase, Multiplicative-Decrease) parameters of the transport protocol. The algorithm uses a guided random search strategy based on the wireless-related packet loss probability and the one-way wireless link delay, which are calculated with the help of an underlying adaptive MAC layer. This layer is capable of detecting packet losses resulting from access failure, bit-errors, fading and signal loss caused by network handoff (switching over a different wireless access point). Thus, the protocol can distinguish correctly between wireless-related and congestion-related packet losses used to calculate the best AIMD settings given the conditions of the network. The results obtained in this work show that the new transport layer was capable of achieving high performance for heterogeneous wireless networks under a wide range of packet losses and link delays. However, the implementation of this algorithm is complex because of the requirements for cross-layer interaction in order to monitor the conditions of the wireless link.

A different approach for online optimization of network protocol parameters was suggested by Ye and Kalyanaraman (2001). In their work, they proposed a hybrid optimization strategy that uses a simulation component to calculate and dynamically adjust the settings of the transport protocol. The search component of the algorithm is

based on random sampling, the simplest and one of the most commonly used randomized search algorithms. In random sampling, each state in the search space has an equal chance of being chosen as a possible solution to the problem. The emphasis of the algorithm is not to find the optimal value for each protocol parameter, but to find better operating points within a limited time frame to quickly adapt the protocol to new network conditions. It explores the whole parameter space, but focuses on the optimization of only those points that fall within a certain region. In order to perform the optimization in a real-time fashion, the algorithm uses a simulation component that collects real-time data from the network and runs a simulation based on the current conditions of the network. The simulation component uses stochastic approximation methods to find better parameter settings which, once found, are applied back into the network. The test results of the algorithm indicated that it was efficient and robust to noises on real optimization problems for TCP flows in wired networks.

In a similar path, Ye and Kalyanaraman (2004) designed a heuristic for an online network parameter optimization algorithm named Recursive Random Search (RSS). The main goal of the algorithm is to overcome the inefficiencies of random sampling, which guarantees the convergence to global optima if the space is finite, but becomes inefficient with many sampling steps and large search spaces (Russell & Norvig, 2002; Ye & Kalyanaraman, 2004). In order to keep the initial high-efficiency property of random sampling, the RSS algorithm was designed to constantly restart the sampling over a new reduced sample space to increase its robustness. Initially, RSS samples the whole parameter space to inspect the general structure of the optimization problem. Then, it moves or resizes the sample space accordingly to previous evaluations of the optimization function and restarts the random sampling to gradually converge to a local optimum. The exploitation (local) phase of the algorithm is separated into two different sub-phases: realignment and shrinking. In the realignment sub-phase, the algorithm performs random

sampling in the neighborhood of a promising point *x*, with the assumption that the algorithm will find a better solution close to *x*. If the algorithm fails to find a better point, the shrinking sub-phase takes over to reduce the size of the sample space. This process continues until the size of the sample space falls below a certain threshold previously determined by the user and dictated by the requirements of the optimization problem. Even though tests were not conducted on any network transport protocol, the simulation results show a substantial improvement in network performance when the algorithm was applied to different protocols for route and queue management such as OSPF (Open Shortest Path First) and RED (Random Early Detection).

Most of the proposed offline and online optimization algorithms discussed so far are known as stochastic search algorithms. A GA is also a stochastic search method that has been previously applied to various optimization problems in ad hoc networks (Barolli et al., 2003; Roy & Das, 2004; Turgut et al., 2002), particularly motivated by the fact that GAs perform much better than other stochastic methods in rugged landscapes (where many local optima is located far from the global optimum) as the population-based approach allows GA to efficiently explore more extensive regions of the search space (Mitchell, 1996).

More precisely, a GA is a search and optimization technique that was invented by John Holland in the 1960s and inspired by the theory of evolutionary computation. Evolutionary computation aims to develop artificial systems based on the principles of the Darwinian evolutionary system, where one or more dynamic populations of individuals compete for limited resources (De Jong, 2006). A simple evolutionary algorithm consists of a population of constant size that evolves over time, where each individual in the population represents a possible solution to a problem. Just like humans, individuals in the population reproduce to generate children that resemble their parents and diversify the population. These new individuals can then reproduce themselves, and as the generations

pass by, the population evolves. To maintain the constant size of the population, the resulting expanded group of individuals (parents and children) is then reduced to its original size by following a replacement strategy that selects and discards individuals based on certain criteria. In general, this cycle repeats until a predefined evolution state is achieved for one or more individuals, or until a fixed number of iterations have passed.

The central notion of a GA follows the same idea of evolutionary algorithms. In a GA, states are treated as individuals in a population, each one represented by a finite string of symbols, known as the *genome*, encoding a possible solution in a given problem space. The evolution process is guided by a fitness function that rates each individual state and determines its probability of survival and reproduction, simulating Darwin's theory of the survival of the fittest during the evolution of species. Consequently, high-fitness individuals stand a better chance of passing to the next generation and reproducing, while low-fitness ones are more likely to disappear, obtaining approximate solutions that are closer to the problem's optimal solution (Russell & Norvig, 2002; Sipper, 1996).

The success of a GA depends on the problem representation (encoding) and the genetic operators: (a) selection, (b) crossover, and (c) mutation (Michalewicz, Eiben, & Hinterding, 1999; Vasconcelos, Ramirez, Takahashi, & Saldanha, 2001). In the general case, the selection operation dictates which pairs of individuals are chosen to participate in the reproduction process, whereas crossover combines genetic code from two individuals to generate children that hold part of the genome from both parents. On the other hand, mutation consists of randomly modifying the genetic code of an individual by altering sections of the genome based on some (small) probability. Mutation allows the algorithm to randomly sample new points in the search space to prevent a premature convergence to local optima (Mitchell, 1996; Sipper, 1996).

In the field of transport protocols there is not much literature regarding the use of GAs in MANETs, except for some research studies that have been focused on improving

the performance of mechanisms that are closely related to transport, such as congestion control and routing protocols. Galily, Roudsari, and Riazi (2005) made an interesting contribution to congestion avoidance, for which they proposed a controller for an Active Queue Management system that applies GA to find the optimal parameters to efficiently regulate the link utilization and delay experienced by data packets. The simulation results show that the system was responsive and tolerant to the dynamics and noise of the network environment, although only fixed networks were considered in the experimental analysis.

Montana and Redi (2005) made another important contribution that illustrates the promising use of GAs to perform optimization of network protocols in MANETs. Performance of routing protocols in MANETs, especially reactive protocols which maintain a global view of the topology, have a great impact on the overall performance of the network infrastructure (Huang, Bhatti, & Parker, 2006; Montana & Redi, 2005). Routing protocol parameters such as refresh and update intervals guide the ability of adaptation to changes. Short intervals provide a more accurate view of the topology at any given time, but increase the overhead incurred by the exchange of link state messages. Therefore, parameter values need to be tuned so as to improve the performance of the routing protocol under different scenarios. However, the task of manually tuning and identifying sets of optimal (or near-optimal) parameter values is combinatorial in nature and not practical, so the proposed algorithm was designed to tune multiple parameters of a routing protocol through offline optimization using a single objective classical GA.

The optimization algorithm randomly generates an initial population with states that consist of a list of values for the parameters to be optimized. The algorithm uses a selection scheme with a steady-state replacement policy that adds a new individual and discards the worst member of the population at each generation. It evaluates the fitness of an individual by running a simulation of the network, collecting statistics about the network performance given the current parameter values. The fitness value for an

individual is a combined score based on the percentage of dropped packets and average delay that were observed during the simulation.

The results of this work show that automated parameter optimization produced significantly better parameter values than hand tuning. Although the algorithm was considered to optimize only the parameters of a routing protocol, results also indicate that automated parameter tuning through a GA may be successfully applied to improve the performance of transport protocols in MANETs.

In a summary, the optimization of protocol parameters has several advantages over particular adaptations of existing and new transport protocols. First, rather than implementing one or more specialized mechanisms to deal with mobility, interference, and others factors that have an impact on the protocol performance, an optimization algorithm can improve the performance of the protocol under different scenarios with little or no protocol modifications. Second, because parameter tuning should require little or no protocol adaptation, protocols can still interoperate between the wired and the wireless worlds in a seamless manner. Third, optimization algorithms provide a more general solution for end-to-end transport optimization over multi-hop paths, where noisy conditions on intermediate links may significantly affect the overall performance of the protocol.

Given the results and conclusions of the related work, the proposed optimization algorithm aims to take advantage of the capabilities of GAs to provide a general solution for protocol parameter optimization, so as to improve the end-to-end transport performance under different mobility and interference conditions in MANETs.

CHAPTER II

GENETIC ALGORITHM FOR TRANSPORT OPTIMIZATION

The proposal for protocol parameter optimization aims to improve the performance of a network transport protocol for different classes of operating conditions. Based on previous work, it has been demonstrated that mobility and interference have a great impact on the performance of routing and transport protocols in multi-hop wireless networks (Fu et al., 2005; Jain et al., 2003).

Modifiable parameters of transport protocols can be adjusted to adapt the protocol to different operating conditions, improving the protocol performance and allowing higher level applications to satisfy their QoS requirements. The proposed algorithm, called Genetic Algorithm for Transport Optimization (GATO), is a GA-based search algorithm that enables automatic parameter tuning for transport protocols.

The main goal of GATO is to find a set of parameter values that improves the protocol performance given the conditions of the network. If conditions change, the algorithm automatically searches for a new configuration of parameters that allows the protocol to continue operating at maximum performance.

Several metrics can be used to quantify the performance of a transport protocol, for example, transmission delay, number of retransmissions, number of bytes received per unit of time, and others. For more complex scenarios and QoS requirements, a combination of various metrics can also be used to determine the performance of a transport protocol.

In this research study, protocol performance will be measured in terms of throughput, which can be defined as the amount of user data transferred from the sender to the receiver over a given period of time. Other performance metrics or objective functions

combining multiple metrics could be chosen, but for simplicity, this work will focus on a single metric.

The maximum throughput that can be achieved by a transport protocol depends on the majority of modifiable parameters that can be found in transport protocols, in particular, parameters that are used for flow control. For example, a sender may stop sending packets until an acknowledgment of the last sent packet is received. If the previous sent packet is lost and the retransmission timeout is too large, the flow of packets is reduced and the throughput decreases. By adjusting the protocol parameters, flow control mechanisms can react better to adverse conditions, such as higher packet drops, and increase the flow of packets if necessary.

Hence, the fitness function of GATO computes the throughput achieved by the protocol when configured with the parameter values that are encoded in the individual. Each individual encodes a possible protocol configuration as a list of parameter values. The genetic operators of GATO combine and randomly change the genetic information of individuals as a mean to explore different regions of the search space, which is comprised of a finite, but usually large, number of possible protocol configurations.

In this work, a reduced search space is used to validate the effectiveness GATO. By having a reduced search space, the optimal parameter values of the protocol for a specific scenario can be found in a reasonable amount of time using an exhaustive search. The throughput achieved by the protocol when configured with the optimal settings can be used as a baseline to compare the performance of the protocol when configured with the parameter values found by GATO. This comparison can be made as long as network conditions are replicated over different runs of the search algorithms to assure that differences in protocol performance are due only to different parameters values.

CHAPTER III

TECHNICAL APPROACH

The proposed algorithm for transport optimization is based on GAs, which are stochastic search algorithms inspired in the theory of evolutionary computation and genetics. GAs are commonly used to solve optimization problems as they can quickly explore an extensive search space (Russell & Norvig, 2002) and are highly effective in the presence of noise, where fitness evaluations may be subject to inconsistent or error-prone measurements (Mitchell, 1996). Additionally, they have been shown to be successful at finding solutions or optimizations for complex and difficult problems for which traditional search methods are slower or less effective. This chapter describes how GAs work and the proposed algorithm for transport optimization.

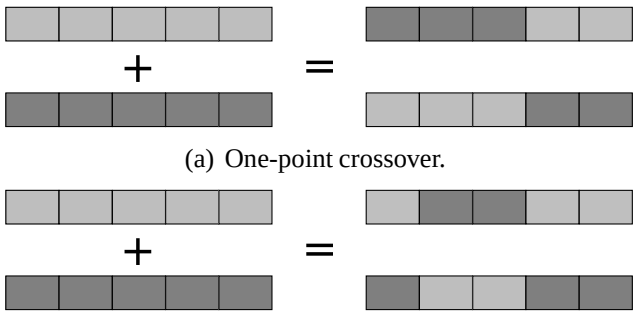
Introduction to Genetic Algorithms

GAs follow the same idea of evolutionary algorithms where a population of individuals evolves over time. Each individual in the population encodes a possible solution to a problem as a finite string of symbols. During the evolutionary process, individuals undergo a selection phase that places the fittest individuals into a mating pool, from which they are paired in the presence of genetic operators that combine the genetic information from both parents to generate offspring.

The resulting expanded population is then reduced to its original size by following a replacement strategy for the individuals based on a fitness function that guides how the population evolves over time. In general, high-fitness individuals stand a better chance of passing to the next generation and reproducing, while low-fitness ones are more likely to disappear. This process is repeated until an individual who satisfies a desired fitness level

is obtained. However, in some cases, the stopping criteria can also take into consideration the maximum number of generations, the maximum amount of consumed resources (time, memory), or the inability to find better solutions after a certain number of consecutive generations have passed.

More precisely, a GA incorporates a set of genetic operators that simulate the evolution process of survival of the fittest. These genetic operators are (a) selection, (b) mutation, and (c) crossover. The selection operator is used to populate a mating pool of individuals that can participate in the reproduction process. Conversely, the crossover operator (Figure 4) combines genetic code from two individuals to generate children who resemble their parents, holding one or more sections of their parents' genome.



(a) One-point crossover.

(b) Two-point crossover.

Figure 4. The crossover operator combines genetic code at randomly chosen points of the parents' genome.

On the other hand, the mutation operator (Figure 5) modifies the genetic code of an individual by altering sections of the genome based on some (small) probability. This operator is intended to diversify the population so as to avoid a rapid convergence of the algorithm to local optima. The mutation operation is equivalent to performing random sampling of the search space.

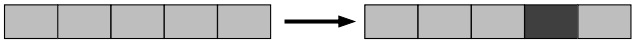


Figure 5. The mutation operator randomly alters sections of the individual's genome.

The standard version of a GA (Figure 6) starts by randomly generating an initial population of individuals. At each iteration step, the individuals in the current generation are evaluated using a fitness function. Then, individuals go through a selection process from which two parents are selected for the reproduction phase of the algorithm.

```
input: the initial population p and the fitness function f
output: the best individual in p, according to f
  repeat
    q ⇐ empty set
    for i = 1 to SIZE(p) do
      x ⇐ RANDOM-SELECTION(p, f)
      y ⇐ RANDOM-SELECTION(p, f)
      child ⇐ REPRODUCE(x, y)
      if some random probability then
        child ⇐ MUTATE(child)
      end if
      add child to q
    end for
    p ⇐ q
  until some individual is fit enough, or enough time has elapsed
```

Figure 6. Pseudocode of a simple genetic algorithm (Russell & Norvig, 2002, p. 119).

The offspring that results from the combination of both parents' genome is introduced to a new population. Then, the new generation replaces the current population and the process is repeated until some individual is fit enough or until enough time has passed. The algorithm returns the best individual in the current population according to

the fitness function. Alghough this individual might not be the optimal solution, the best individual of the population always encodes a partial optimal solution that could still be useful depending on the characteristics of the problem.

Optimization Algorithm

GATO aims to provide the means for automatic protocol parameter tuning by optimizing the parameter values of a network transport protocol for different operating conditions in a MANET. The goal of GATO is to improve the performance of the protocol in terms of throughput. By adjusting the modifiable parameters of a transport protocol, the flow control mechanism of the protocol can react better to different network conditions, increasing the overall throughput of the protocol.

GATO (Figure 7) starts by determining the fitness of each individual in the initial population. At each iteration step, the algorithm computes a group of elite members from the best individuals in the population. Then, for each individual in the remaining population, a new child is generated from two parent individuals who are chosen following a selection scheme called tournament selection. This new individual undergoes a mutation phase based on a variable mutation probability. Then, following a steady-state replacement policy, the new child replaces the worst individual in the population. This process is repeated until the maximum number of generations is reached or until the elite members of the population have not changed for enough consecutive generations. The algorithm returns the best individual in the population. This individual represents the set of parameters that give the best performance of the protocol, given the operating conditions that were current at the time the optimization task was performed.

```
input: the initial population p and the fitness function f
output: the best individual in p, according to f
   for i = 1 to SIZE(p) do
      evaluate p_i, f(p_i)
   end for
   repeat
      elite ⇐ the best n individuals in p, according to f
      for i = 1 to (SIZE(p) - SIZE(elite)) do
         x ⇐ TOURNAMENT-SELECTION(p)
         y ⇐ TOURNAMENT-SELECTION(p)
         child ⇐ REPRODUCE(x, y)
         child ⇐ MUTATE(child, x, y)
         evaluate child, f(child)
         remove the worst individual in p, according to f
         add child to p
      end for
      add elite to p
   until n generations have passed,
         or the group of elite members is the same after m consecutive generations
```

Figure 7. Pseudocode of the genetic algorithm for transport optimization.

More precisely, the design of GATO considers four different aspects: (a) representation, (b) genetic operators, (c) evaluation, and (d) stopping criteria. Each of them plays an important role in the success of the optimization task as they define the algorithm's behavior and how quickly it converges to a solution under different scenarios.

Representation

The genome of individuals is represented as a list of parameter values to be optimized. This type of representation is often referred to as a real-valued representation, and it is commonly used for parameter optimization problems where it is more natural to use real numbers than a binary encoding to form the individual's genome. Although there are not rigorous guidelines for selecting a specific encoding given the problem of interest

(Mitchell, 1996), Wright (1991) demonstrated that real-coded GAs can outperform binary-coded GAs since real-valued representations increase efficiency and precision.

When a real-valued representation is chosen, it is necessary to define a minimum and a maximum value for each of the protocol's parameters to be optimized, as well as a step size that defines a discrete set of possible values within the allowable range of the parameter. Defining a step size is important for parameters that can only take integer values, such as maximum segment size, window size, number of acknowledgements, and others.

Genetic Operators

GATO takes an initial population generated with purely random individuals. At each run, two members of the population are randomly chosen from a mating pool, which is comprised of individuals that in average have a higher fitness value than the average fitness value of the population. This mating pool is generated using a selection scheme called tournament selection. On tournament selection, *s* individuals (competitors) hold a tournament, with *s* being the tournament size. The winner of the tournament is the individual with the highest fitness among the *s* competitors; that individual is then inserted into the mating pool. Tournaments are performed until the mating pool has reached a certain size. When compared to other selection schemes, the advantages of tournament selection include (a) its simplicity and (b) the ability to increase or decrease the selection pressure (intensity with which a GA tends to eliminate individuals) by increasing or decreasing the tournament size, which allows GAs to adapt to different problem spaces (Goldberg & Deb, 1991; Miller & Goldberg, 1995).

The algorithm generates a single child by combining the genome of both parents using single-point crossover. The genetic information of the resulting individual may be modified based on a mutation probability that is proportional to the degree of similarity

between parents, meaning that the probability of the mutation operator is dynamically adjusted according to the diversity of the population. Adapting the probability of genetic operators has been shown to be effective at improving the convergence rate of GAs and preventing GAs from converging to local optimum (Srinivas & Patnaik, 1994).

Based on the degree of similarity between parents, the more similar the parents are, the higher the mutation probability is. When the algorithm starts to converge, the mating pool would consist of individuals who are similar to each other. In this case, the mutation probability increases, allowing the algorithm to randomly search other regions of the problem space. On the other hand, if the diversity of the population is too high, the mutation probability decreases to allow the algorithm to stabilize and refine the best individuals in the population.

GATO also follows a steady-state replacement policy, where the new individual replaces the worst member of the population in the current generation. Steady-state replacement generally allows finding a solution much faster than the generational approach because the algorithm can immediately exploit new individuals (Montana & Redi, 2005). In particular, the use of such replacement policy is important in the case of the proposed algorithm because of the elevated cost of the fitness function, which requires the protocol to operate for a certain amount of time in order to estimate its performance given the current configuration.

Furthermore, GATO also maintains a fixed group of the best individuals across generations. This strategy, commonly known as *elitism*, aims to avoid the loss of the best individuals during selection, crossover, and mutation by keeping on the next generation an individual whose performance is better than the performance of other population members in the current generation. Elitism has been shown to provide better solutions and higher convergence speeds for different types of problems (Ahn & Ramakrishna, 2003; Montana & Redi, 2005; Vasconcelos et al., 2001).

Fitness Function

To evaluate an individual, GATO uses the parameter values encoded in the genome of the individual to configure the protocol and measure how well it performs in the current environment. More precisely, GATO creates and configures a connection to measure a variety of potential performance metrics given the operating conditions of the network. A performance metric can be based in the number of retransmissions, the transmission delay, the throughput, or a combination of them. For some of the metrics, such as throughput or transmission delay, GATO requires a mechanism to monitor the performance of the protocol at the opposite end-point of the connection. In this case, the receiver node collects information about the connection and sends it back to the transmitter so that the algorithm can compute the fitness value of the individual.

Stopping Criteria

The main goal of GATO is to find an individual who leads to an improvement of one or more performance metrics. Hence, the algorithm stops when it cannot find a better individual after a given number of consecutive generations have passed. In other words, the group of elite members in the population remains the same after a certain number of iterations. GATO also considers the total number of generations to prevent the system from diverging. In this case, the algorithm returns the best individual from the population when the maximum number of generations has passed.

Case Study

The validation of the algorithm is made by using the algorithm to automatically tune the parameters of the Reliable UDP (RUDP) transport protocol. The objective of the optimization is to maximize the throughput for end-to-end communication under different conditions of mobility and interference.

The proposed algorithm performs the parameter optimization of RUDP over a multi-hop wireless ad hoc network, which is sufficient to demonstrate (a) the ability of the algorithm to optimize the protocol parameters in the presence of interference caused by other nodes in the network given that the algorithm does not need to determine the source of interference and (b) the effectiveness of the algorithm given that the objective of the proposed optimization is to improve the protocol performance and not the overall performance of the network.

RUDP is a simple but flexible transport protocol designed to support applications that require reliable in-order transport of packets (Bova, Krivoruchka, & Cisco Systems, 1999). In terms of implementation, this protocol is less sophisticated than TCP but provides a minimal set of QoS services for network transport. Similar to other protocols for ad hoc networking, RUDP has a set of modifiable parameters that can be used to configure the protocol in response to a variety of conditions, allowing applications to satisfy different QoS requirements.

More precisely, RUDP is a simple connection-oriented and packet-based transport protocol originally designed to support transport of telecommunication signaling protocols across IP networks. This UDP-based protocol provides a set of QoS enhancements that allow a variety of applications to maintain a good quality stream without the overhead of more sophisticated transport protocols, even in the presence of severe network congestion or packet losses caused by link failures. Similar to TCP, RUDP provides reliable in-order packet delivery, as well as basic flow control, error detection, and keep-alive mechanisms. Additionally, RUDP allows the characteristics of each connection to be individually configured so that applications with different transport requirements can adapt themselves to the operating conditions of the network.

RUDP uses sequence numbers to guarantee in-order delivery and a three-way handshake to synchronize sequence numbers between two peers. When a connection is

first opened, each peer randomly chooses a sequence number and increments it before sending a data packet. Before the connection is fully established, both peers negotiate a set of parameters that includes (a) the maximum segment size, (b) the maximum number of unacknowledged segments, and (c) the maximum number of segments that can be received out of sequence. The last two parameters are used as a mean of flow control, whereas the maximum segment size indicates the maximum size of a single RUDP segment or data packet.

Moreover, the protocol specifies two types of segments to acknowledge received data packets. The first type is the ACK segment, which is used to acknowledge segments that were received in order. The second type, called Extended ACK (EACK), is used to acknowledge segments that were received out of sequence. Additionally, data packets may also be acknowledged by piggybacking its sequence number in the header of a segment that the receiver sends to the transmitter.

RUDP provides active and passive keep-alive mechanisms. The active keep-alive mechanism consists of periodic transmissions of null (empty) segments if no data segments are being transmitted, whereas the passive keep-alive mechanism makes use of a counter to keep track of how many times a packet has been retransmitted. The connection is considered broken if the receiver's null segment timer expires or the retransmission count for a data packet exceeds its maximum.

Bova et al. (1999) define the following protocol parameters:

1. *Maximum segment size*. It specifies the maximum size for an RUDP segment or data packet, including the length of the RUDP header. The valid range for the value of this parameter is 0 to 65,535.
2. *Maximum number of outstanding segments*. It specifies how many packets can be transmitted without getting an acknowledgment. The valid range is 1 to 255.

3. *Maximum number of retransmissions*. It specifies the maximum number of consecutive retransmissions of a packet that will be attempted before a connection is considered broken. The valid range is 0 to 255. A value of 0 indicates that the sender will continue to retransmit the packet forever.
4. *Maximum number of cumulative acknowledgements*. It specifies the maximum number of packet acknowledgements that will be accumulated before sending an ACK or EACK. The valid range is 0 to 255. A value of 0 indicates that an acknowledgment must be sent immediately after a data packet is received.
5. *Maximum number of out of sequence packets*. It specifies the maximum number of out of sequence packets that will be accumulated before sending an EACK segment. The valid range for this value is 0 to 255. A value of 0 indicates an EACK segment will be sent immediately if an out of sequence packet is received.
6. *Null segment timeout.* It specifies the number of milliseconds a sender must wait before sending a null segment if another segment is not sent. The valid range is 0 to 65,535.
7. *Retransmission timeout*. It specifies the number of milliseconds a sender must wait before retransmitting a packet that has not been acknowledged. The valid range is 100 to 65,535.
8. *Cumulative acknowledgement timeout.* It specifies the number of milliseconds a receiver must wait before sending an acknowledgment segment if another segment is not sent. The valid range is 100 to 65,535, but it should be smaller than the value of the retransmission timeout.

An object-oriented implementation of the protocol was developed in Java as there was not a public reference implementation available at that time. This open-source implementation (Granados, 2009) is available for download at

https://sourceforge.net/projects/rudp. The original design of the protocol has been changed to simplify its implementation and provide capabilities that make possible its integration with the Java Networking API (*Java Networking Overview*, n.d.) and the proposed optimization algorithm. The modifications and additions to the protocol include the following:

1. Support for error detection. In its original design, the protocol ensures data integrity by calculating a checksum on the header and body of the packet; however, the data integrity check performed at the UDP layer should be sufficient to guarantee a normal operation of the protocol. Therefore, error detection was not implemented.
2. Support for redundant connections. When a connection fails, applications using RUDP can initiate the transfer of the state of the connection to another connection and resume transmission ensuring that packets are not duplicated or lost. Although having support for redundant connections is an interesting feature, it was not considered as it makes the implementation unnecessarily complex.
3. Support for auto reset of active connections. Either side of an RUDP connection can initiate an auto reset when the number of retransmissions exceeds a maximum value. In the event of an auto reset, both peers reset their states, renegotiate the connection parameters, and then resume normal transmission of packets. Auto reset was partially implemented and can only be initiated by the application to renegotiate parameters on active connections, preventing the application from sending new packets until all previously accepted packets are delivered and the connection is reopened.
4. Support for connection monitoring. The original protocol was extended to provide monitoring capabilities. Essentially, the protocol can be configured to

monitor the rate at which the sender is transmitting data and the rate at which the receiver is receiving data. This information is shared between peers and is also exposed to higher level applications so that decisions can be made regarding the adjustment of the protocol parameters to satisfy a variety of QoS requirements.

5. Support for Java API integration. The original specification of RUDP makes a clear distinction between client and server connections, each of them having a different behavior. This implementation of the protocol provides a mechanism by which each side of the connection behaves in the same manner as the other peer. In this case, a peer assumes the role of a client by actively opening the connection, whereas the peer assuming the role of a server performs the passive opening. This modification to the protocol makes possible the integration with the Java Networking API so that RUDP sockets can be used transparently by existing Java applications.

CHAPTER IV

EXPERIMENTS AND ANALYSIS OF RESULTS

Experiments were run to measure the performance of RUDP in multi-hop wireless networks. Also, experiments were run to evaluate the effectiveness of the proposed algorithm for optimizing the parameters of RUDP in order to maximize the throughput for end-to-end transport over multiple hops. The description of the experiments and the analysis of the results are presented in the next two sections.

Experimental Setup

The validation of the proposed algorithm was made over an emulation testbed developed at the Institute for Human and Machine Cognition. This testbed uses theoretical models to simulate the radio and interference signals that can be found in wireless networks. Nodes were configured and used to emulate different multi-hop wireless communication paths (Figure 8).

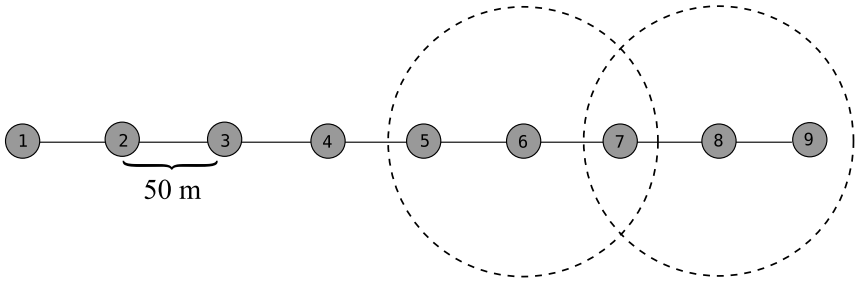


Figure 8. Emulated wireless network. There is a maximum of 8 hops (from node 1 to node 9). Circles indicate the transmission range of a node.

For each multi-hop path, nodes were positioned to form a line topology with a separation of 50 meters between nodes. The transmission power of each node was set to 6.5 milliwatts to provide a transmission range of approximately 75 meters using the Free Space radio propagation model. With this configuration, nodes could only communicate directly with its immediate neighbors and indirectly (through packet forwarding) with nodes that were more than one hop away.

The testbed was configured to use a basic interference model that computes the interference at node *N* based on the sum of all the simultaneous transmissions that reach, but are not targeted to node *N*. Additionally, the capacity of all links was limited to 11 megabits per second, the rate of a IEEE 802.11b wireless network. For each evaluation, the same propagation and interference models were used to ensure that differences in the performance of RUDP were due solely to different parameter settings. Additionally, in some scenarios, multiple experiments were run in parallel by segmenting the line topology into shorter multi-hop paths, ensuring that the nodes on each network segment could not interfere with nodes from other segments.

The RUDP parameters considered for optimization consist of (a) maximum number of outstanding segments, (b) maximum number of cumulative acknowledgements, (c) maximum number of packets received out of sequence, and (d) retransmission timeout. The valid range for the value of the first three parameters was $[1, 2, \ldots, 8]$ whereas the valid range for the retransmission timeout was $[200, 400, \ldots, 1,000]$, generating a search space that consisted of $8 \cdot 8 \cdot 8 \cdot 5 = 2,560$ parameter sets.

At each node, a Java server process was listening for RUDP connections on the (emulated) wireless interface. In order to establish a connection with the server process, a client RUDP socket had to actively initiate the connection and negotiate its initial parameters. After a connection was established, the server's connection handler would start reading bytes continuously until signaled to gracefully close the connection.

An initial experiment consisted of measuring the maximum throughput that can be achieved by RUDP when using the recommended parameter settings. RUDP was originally designed as a reliable transport mechanism for telecommunication signaling, which does not require high throughput. Therefore, the recommended parameter settings for RUDP were proposed to provide the minimal QoS capabilities for signaling protocols requiring transport. Thus, a second experiment consisted of running an exhaustive search over the 2,560 parameter sets to find the best configuration for the given scenario and network operating conditions from 1 to 8 hops. The main objectives of this experiment were to determine if the parameters of the RUDP protocol could be optimized for throughput and to use the results as a baseline for comparing the performance of GATO over the same search space. The rest of the experiments were designed to analyze the performance and effectiveness of GATO for optimizing throughput over 2, 4, and 8 hops in static and dynamic environments.

In all runs, GATO was configured to use a tournament size of 30% the number of individuals in the population. On the other hand, the size of the elite group at each generation was 5% the size of the population. Additionally, the maximum number of generations was 100. Finally, GATO was also configured to return the best individual of the current population if no better solution could be found after three consecutive generations.

Preliminary Analysis

The RUDP draft specifies the recommended values for each of the protocol settings. In particular, the default values for the maximum number of outstanding segments, maximum number of cumulative acknowledgements, maximum number of packets received out of sequence, and retransmission timeout are 3, 3, 3, and 600 ms, respectively. Hence, the first consideration was to identify if, in comparison with the

default configuration, optimizing the protocol parameters can significantly increase the overall throughput of the protocol. In this preliminary analysis, an experiment was conducted to determine the maximum throughput that can be achieved by using the recommended RUDP settings and to identify the level of improvement that can be achieved by modifying the protocol parameters (Figure 9).

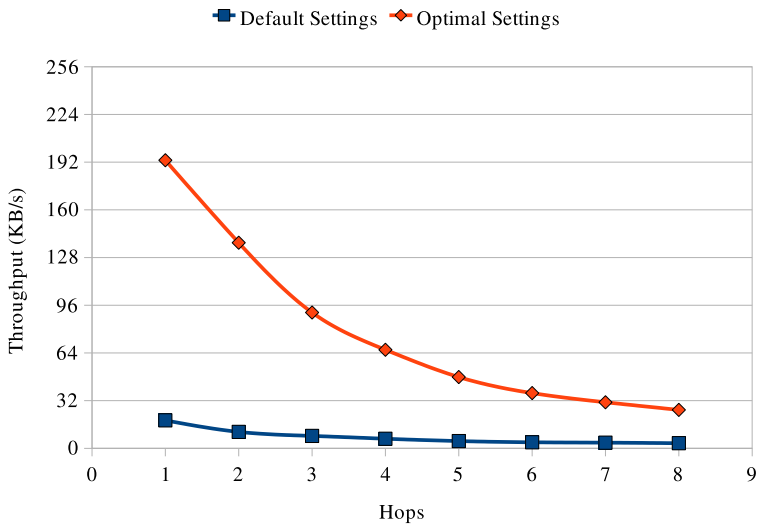


Figure 9. Maximum throughput achieved by the Reliable User Datagram Protocol from 1 to 8 hops.

The exhaustive search results show that there was at least one set of parameter values that could significantly increase the throughput of RUDP over multiple hops. By modifying the parameter settings, the average throughput had a maximum improvement between 650% (8 hops) and 930% (1 hop) when compared to the average throughput that was achieved using the default configuration.

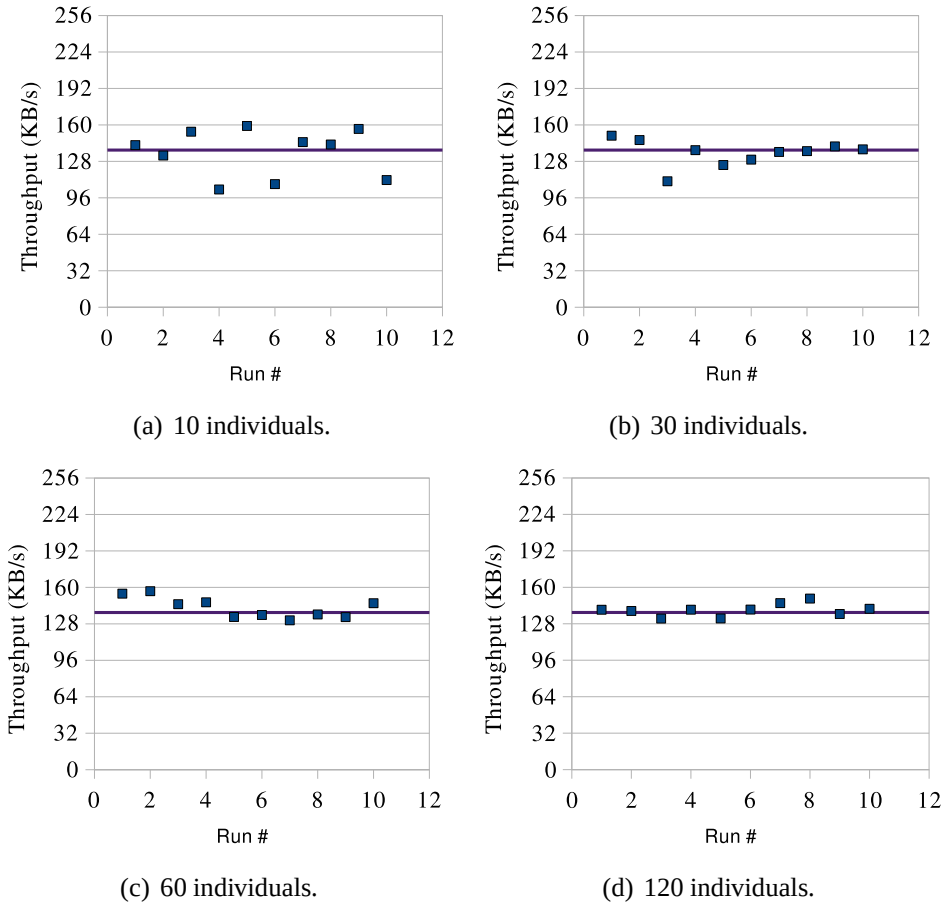
For example, in a 2-hop scenario, the protocol achieved an average throughput of 11.03 kilobytes per second (KB/s; SD = 3.68) using the default parameter values, whereas the average throughput increased to 137.94 KB/s (SD = 10.10) when using the best

configuration that was found during the exhaustive search. For an 8-hop scenario, the protocol achieved an average throughput of 3.44 KB/s (SD = 0.89) using the default parameter values, whereas the average throughput increased to 25.81 KB/s (SD = 13.08) using the optimum configuration.

It can be observed also that the overall throughput decreased as packets were transmitted over longer paths. This reduction in performance can be explained by the increase in packet drops resulting from the interference caused by packet retransmissions at each intermediate node. However, with an alpha level of .05 for all statistical tests, the difference between average throughputs for a 2-hop scenario was statistically significant, $t(58) = 64.66, p < .0001$. In the case of the 8-hop scenario, the difference between average throughputs was also statistically significant, $t(58) = 9.34, p < 0.0001$.

These results verify the hypothesis that the protocol performance can be improved by tuning the protocol parameter values. Thus, these preliminary results are used to evaluate the effectiveness and performance of GATO over the same reduced problem space. In the validation of GATO, a 2-hop static path and population sizes of 10, 30, 60, and 120 individuals were considered.

Optimizing the parameter values of RUDP using GATO show that the proposed algorithm found sets of parameter values that allowed the protocol to achieve, in all cases, an average throughput that was statistically equivalent to the average throughput obtained with an optimal protocol configuration (Figure 10).



(a) 10 individuals. (b) 30 individuals.

(c) 60 individuals. (d) 120 individuals.

Figure 10. Maximum throughput achieved by the Reliable User Datagram Protocol over 2 hops when using the parameter values found by the Genetic Algorithm for Transport Optimization. The horizontal line indicates the average maximum throughput when using the optimal configuration.

However, it can be observed that the variance of the average throughput increased for the cases where the GA was initialized with a random population of 10 and 30 individuals (Figure 11). These results show that the algorithm tends to find more local optima when the size of the population is small because the genetic operators are unable to prevent the algorithm from converging too rapidly. If the size of the population is larger, the algorithm may find better solutions but at the cost of a slower convergence.

Consequently, if carefully chosen, a given population size may provide good solutions in a reasonable amount of time.

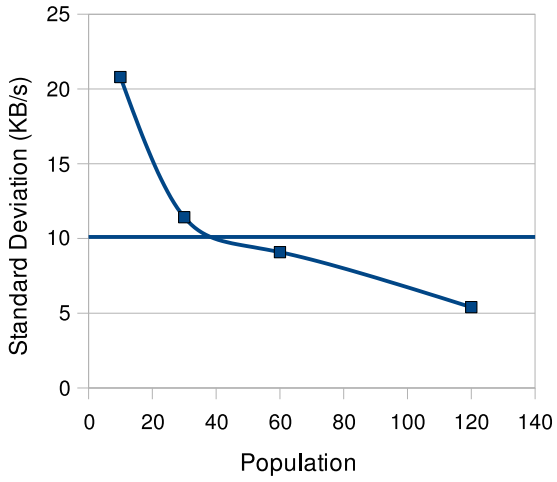


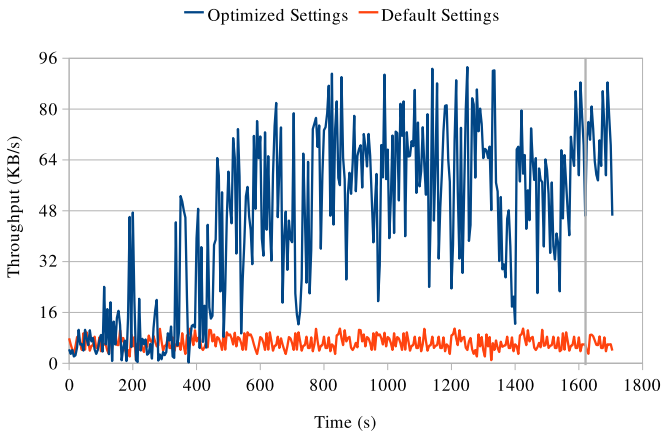
Figure 11. Standard deviation of the throughput achieved by the Reliable User Datagram Protocol over 2 hops when using the parameter values found by the Genetic Algorithm for Transport Optimization. The horizontal line indicates the standard deviation of the maximum throughput when using the optimal configuration.

Based on these assumptions, a population size of 50 individuals should be sufficient to optimize the parameter values of RUDP in comparison with the results obtained from the exhaustive search.

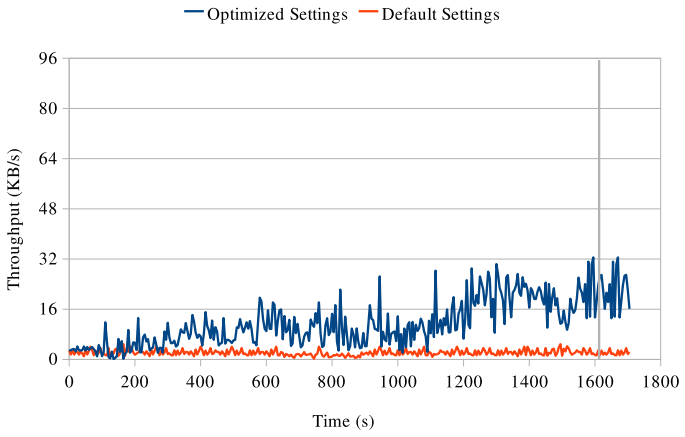
Simulation Results and Discussion

The proposed GA is designed to optimize end-to-end transport, which means that GATO adjusts the protocol's parameters and monitors its performance at the sender and receiver end-points, respectively. Simulation results show that GATO is capable of optimizing the parameter values of RUDP over longer communication paths, without increasing the cost of the optimization task.

Results show, for example, that the algorithm's convergence time for both 4-hop and 8-hop static paths, with a population of 50 individuals (Figure 12), was approximately 27 min (1620 s). However, in the process of searching for the optimal configuration, the algorithm always tended to improve the throughput of the protocol.



(a) 4 hops.



(b) 8 hops.

Figure 12. Online optimization of the Reliable User Datagram Protocol using the Genetic Algorithm for Transport Optimization over 4 and 8 hops. The vertical line indicates the time at which the algorithm converged.

This tendency to gradually improve the performance of the protocol means that the algorithm does not require, for example, 27 min of continuous traffic. Moreover, if the state of the current population is maintained across multiple flows, the algorithm could also gradually improve the overall performance of the protocol because, as long as packets are sent, the evolution of individuals continues until an optimum solution is found.

Highlighted by these experimental results, the main shortcoming of GATO is the time that the algorithm takes to converge. The fitness function of the algorithm adjusts the protocol's parameters based on the parameter values encoded by the individual. Then, it monitors the performance of the protocol by sampling the throughput at the receiver end-point. However, this sampling must be performed over a period of time that is sufficient enough to ensure the accuracy of the throughput estimation. As a result, the cost of the fitness function significantly increases the overall running time of the algorithm.

Improving the Efficiency of the Optimization Algorithm

There are several approaches that could lead to an improvement of the algorithm's performance. A first approach could be to assign to new offspring the same fitness value of a similar or identical individual found in the population, which prevents GATO from performing a new fitness evaluation. However, because of the unpredictability in wireless communications and the characteristics of the proposed fitness function, throughput estimations at a given time can vary significantly even when the protocol's parameters are not modified. Hence, when a good individual is evaluated in the presence of higher packet drop rates, subsequent similar or identical individuals will be assigned the same bad fitness value even under normal operating conditions. Assigning the same values could cause the algorithm to converge to local optima and never find the global optimal solution.

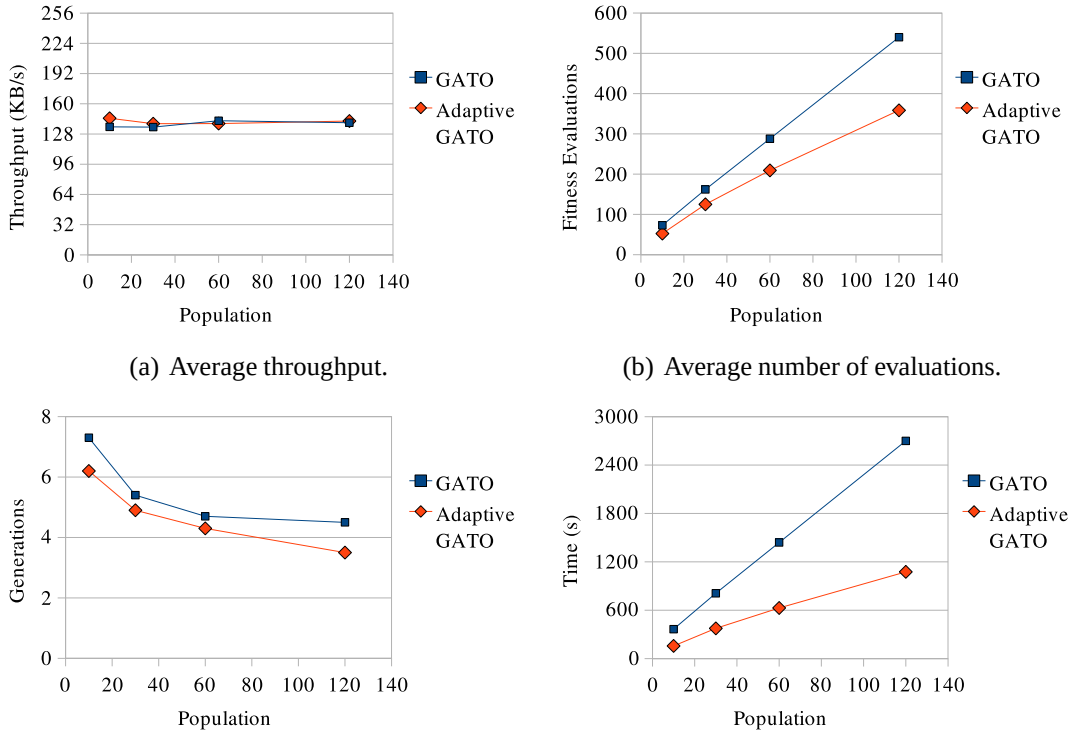
A second approach could be to dynamically adjust the size of the population to reduce the number of evaluations. More precisely, the algorithm identifies similar or

identical individuals and generates populations that consists of just unique members. In this case, only the best member of a group of similar individuals survives and passes to the next generation. The advantage of this approach is that it still allows for repeated evaluations of individuals that are alike. These repeated evaluations helps GATO to recognize individuals that may have had bad fitness values in previous evaluations, but that in the long term are actually good individuals.

Finally, a third approach could be to reduce the time that it takes to evaluate an individual by sampling the throughput of the protocol at higher rates, in a shorter period of time. Reducing the sampling time could make the fitness evaluation more sensitive to bursts in traffic, increasing the variance of the estimated average throughput. Throughput estimations at shorter intervals can be performed accurately if a constant and sufficient rate of bytes is transmitted during the sampling time.

A variation of GATO that reduces the time for fitness assessment and dynamically adjusts the size of the population shows a significant reduction in the overall execution time. This variation of the algorithm, called Adaptive-GATO, also converges to statistically equivalent solutions as the original algorithm. Adaptive-GATO computes a degree of similarity between individuals to identify unique members in the population. Given two individuals, the algorithm measures the distance between their two parameter vectors by calculating, for each pair of corresponding parameter values, the number of incremental steps that are required to transform a value into the other one. Then, the algorithm calculates a normalized distance between the two vectors by dividing the sum of all the parameter distances by the maximum possible distance between any two individuals. A value in the range $[0-1]$ determines how similar the individuals are. Two individuals are identical if the degree of similarity is equal to 0, and they are completely different if the degree of similarity is equal to 1.

A performance comparison between GATO and Adaptive-GATO (Figure 13), shows that, similarly to GATO, Adaptive-GATO converged to a solution statistically equivalent to the optimal solution found by the exhaustive search. However, Adaptive-GATO required fewer fitness evaluations, which reduced the time needed for the optimization task. For example, in the scenario where Adaptive-GATO was initialized with a random population of 60 individuals, the algorithm required, in average, 28% fewer evaluations than GATO, which reduced the average execution time by 58.33%, from 24 min (1440 s) to 10 min (600 s).



(a) Average throughput.

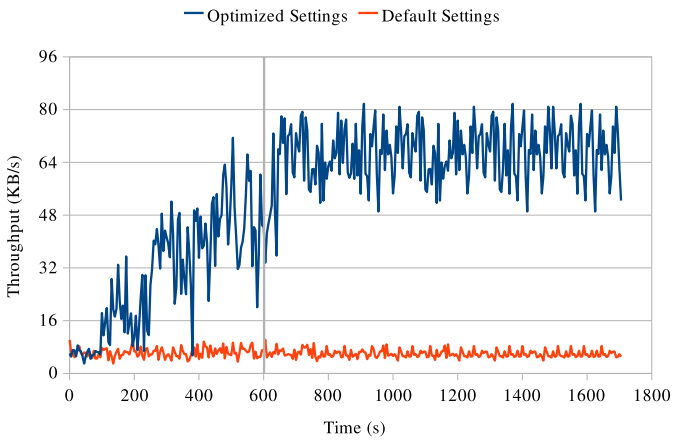
(b) Average number of evaluations.

(c) Average number of generations.

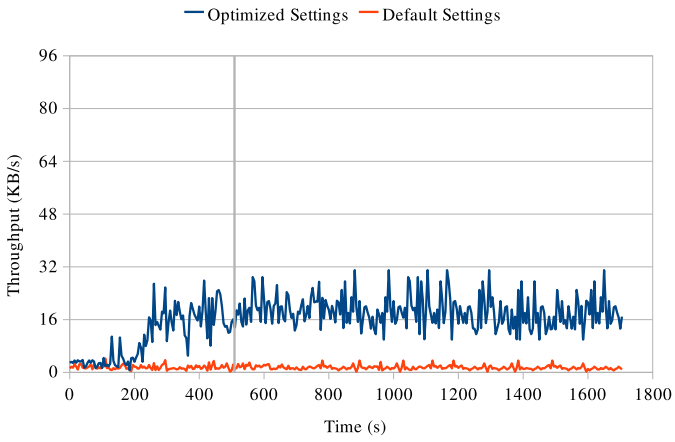
(d) Average convergence time.

Figure 13. Performance comparison of the Genetic Algorithm for Transport Optimization (GATO) and its adaptive version (Adaptive-GATO) over a 2-hop path, using population sizes of 10, 30, 60, and 120 individuals.

Adaptive-GATO gave the same results in the 4-hop and 8-hop scenarios that were used to evaluate the performance of GATO (Figure 14). However, the execution times of Adaptive-GATO were shorter than the execution times of GATO because of the dynamic resizing of the population, which reduces the number of fitness evaluations that are performed at each iteration.



(a) 4 hops.



(b) 8 hops.

Figure 14. Online optimization of a flow using the Adaptive Genetic Algorithm for Transport Optimization over 4 and 8 hops. The vertical line indicates the time at which the algorithm converged.

Because Adaptive-GATO adjusts the size of the population based on the diversity of the individuals and how they evolve, the number of fitness evaluations can vary between executions, independently of the number of hops. For example, in the case of the 4-hop scenario, the algorithm took approximately 10 min (600 s) to converge, whereas in the 8-hop scenario, the algorithm needed fewer evaluations and converged in 8.5 min (510 s).

When compared to GATO, Adaptive-GATO requires significantly less time to converge and reduces the variance of the achieved throughout because the algorithm gradually reduces the size of the population by eliminating similar individuals (Figure 15). This reduction increases the probability of individuals with higher fitness values to be chosen as parents, shrinking the size of the sample space. However, if the size of the population is greatly reduced, Adaptive-GATO may converge to local optimum since the time that the algorithm takes to find the sub-optimal solution is not enough to diversify the population and explore other regions of the search space where better solutions may be found.

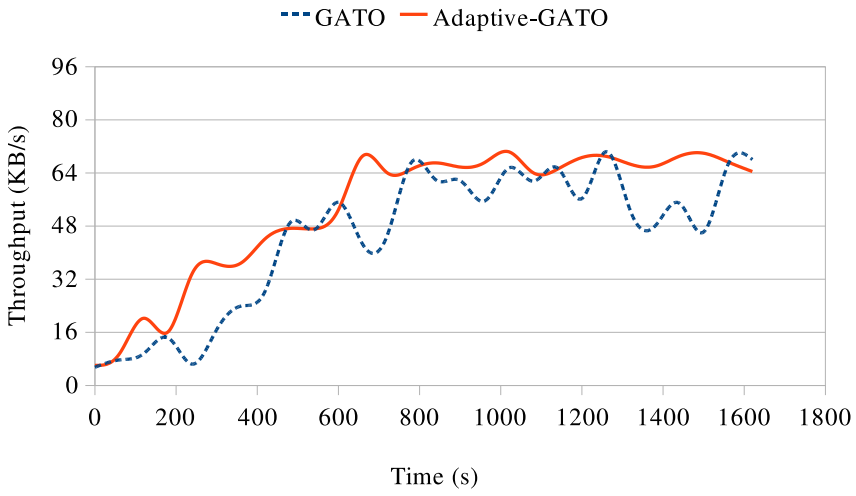
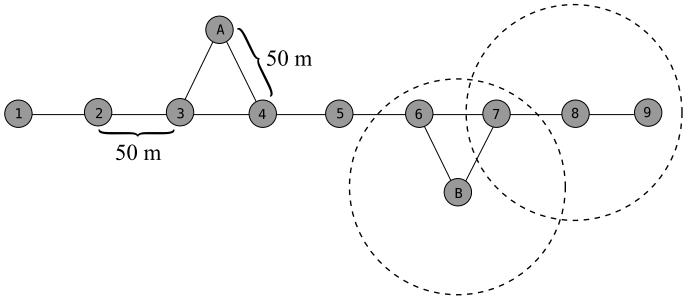


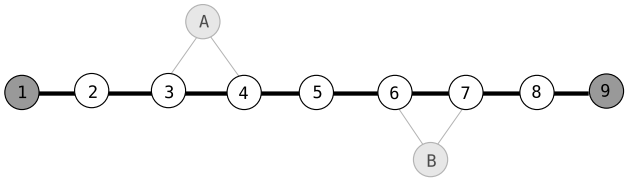
Figure 15. Online optimization comparison of the Genetic Algorithm for Transport Optimization (GATO) and its adaptive version (Adaptive-GATO) over a 4-hop path (using averaged data points to facilitate readability).

Online Adaptation to Network Conditions

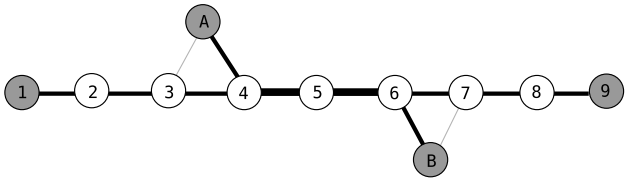
When the conditions of the network change, the algorithm needs to be restarted to optimize the parameter values to adapt the protocol to these new conditions. Two tests were conducted to verify that the algorithm can dynamically adjusts the parameter values of the protocol to provide maximum throughput when network conditions change. In these experiments, two additional nodes were added to the emulated wireless network to create secondary flows that interfered with the flow targeted for optimization (Figure 16) .



(a) Emulated wireless network topology.



(b) A flow going from node 1 to node 9.



(c) A secondary flow going from node A to node B.

Figure 16. Another emulated wireless network with two additional nodes. A secondary connection interferes with the existing flow going through the 8-hop path.

For each test, an RUDP connection was opened between nodes 1 and 9. Then, after the flow going through this first connection was optimized, another connection was opened between nodes A and B to interfere with the existing flow, triggering a second run of the optimization algorithm.

In the first test, the secondary flow increased the induced interference over the flow targeted for optimization, reducing the throughput of the optimized connection by 58%. The change of operating conditions and the reduction of the throughput triggered a new optimization task. Results show that Adaptive-GATO improved the performance of the protocol in the presence of higher packet drops caused by the induced interference (Figure 17). The average throughput increased by 40%, from 8.65 to 14.36 KB/s.

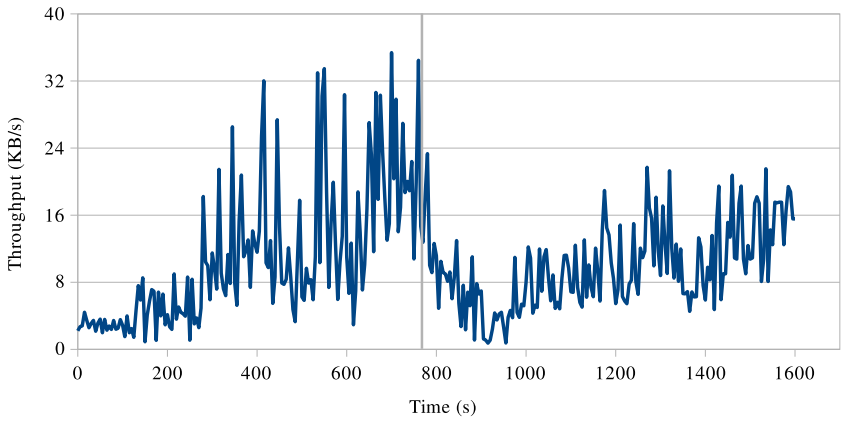


Figure 17. Dynamic online optimization of a flow over an 8-hop path. At time 780 (vertical line), a secondary flow is started from node A to node B, interfering with the optimized flow and triggering a new optimization task.

Each time the optimization algorithm is restarted, a new population of purely random individuals is generated. However, it is possible that the time to converge could be reduced by inserting into the initial population a group of individuals that are known to

improve the performance of the protocol. This group of individuals could be, for example, the elite group of the last generation from a previous run of the algorithm.

In the second test, the secondary flow that also increased the interference after the algorithm had already performed the optimization task. When the protocol detected a degradation in the performance of the initial connection, the algorithm was restarted using a population that contained 40 random individuals and the 10 best individuals of the previous run (Figure 18). In this case, results show that Adaptive-GATO improved the performance of the protocol by 37%, and the time to converge was reduced by 14%.

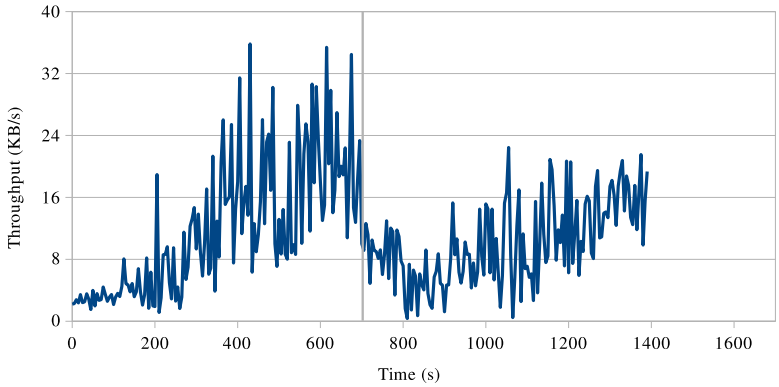


Figure 18. Dynamic online optimization of a flow over an 8-hop path using a predefined initial population. At time 700 (vertical line), a secondary flow is started from node A to node B, interfering with the optimized flow and triggering a new optimization task. The elite group of individuals from the previous run was inserted into the initial population of the next run of the optimization algorithm.

When comparing to the case where the initial population of the algorithm was comprised of purely random individuals, using the previous elite group of individuals did not have a significant effect in the convergence time of the algorithm (Figure 19). Although, assuming that the optimal configurations of small variations in the operating

conditions are located close to each other in the search space, this heuristic could provide a rapid recovery when there are small changes in the conditions of the network.

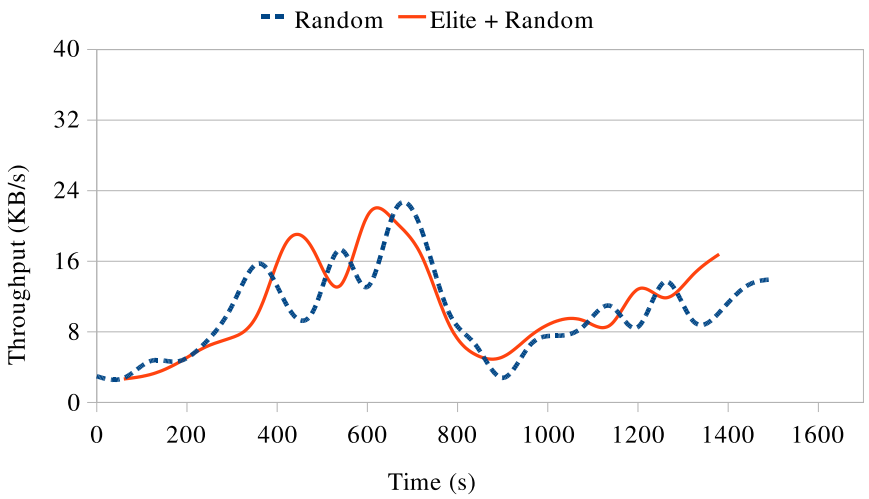


Figure 19. Comparison of a dynamic online optimization of a flow over an 8-hop path (using averaged data points to facilitate readability). At each restart of the algorithm, the initial population was comprised of either purely random individuals (Random) or a combination of random individuals and the elite members of the previous run of the algorithm (Elite + Random).

On the other hand, if the configuration that was obtained in the previous run of the algorithm is located far away in the search space from the optimal solution given the new conditions, then using the best individuals from the previous run may cause a premature convergence to local optima or increase the convergence time since more generations would be needed to diversify the initial population of individuals.

CHAPTER V

CONCLUSIONS

In mobile ad hoc networks, the variation of link characteristics and frequent and unexpected changes in topology greatly decreases the performance of commonly used transport protocols. Transport protocols usually have a set of modifiable parameters that can be adjusted to provide better performance for specific scenarios, although they are usually fixed and rarely changed after the protocol has been deployed. Because the connectivity and operating conditions in wireless networks cannot be predicted during the network design stage, the problem of tuning the parameters of the protocol becomes an important and difficult one.

The experimental results demonstrate the feasibility of automatic parameter tuning for transport protocols based on GAs. They show that the proposed algorithm was able to optimize the parameters of a transport protocol to maximize throughput for end-to-end communication over multiple hops. The algorithm requires no knowledge of the characteristics of the network and can optimize for different scenarios as long as the following two conditions are met.

First, the rate of bytes transmitted must be sufficient enough to induce detectable variations in performance when parameters are modified, otherwise the algorithm is unable to distinguish between good sets and bad sets of parameters. And second, the rate of changes in network conditions must be slower than the time the algorithm requires to converge. Even if the network conditions change faster, the algorithm can still optimize as it always tends to gradually improve the performance of the protocol.

In the real world, an optimization time in the order of minutes is not practical for MANETs, especially when there are frequent changes in topology. The use of heuristics can help reduce the convergence time of a GA to make it more suitable for dynamic environments. However, the effectiveness of such heuristics is tightly related to the characteristics of the problem and the design of the GA.

For example, the results show that for the considered scenarios, allowing only unique individuals to survive helped the adaptive algorithm to converge much faster than the implementation of GATO working with populations of constant size. To dynamically adjust the size of the population, Adaptive-GATO used a degree of similarity between individuals based on a distance metric between corresponding parameter values, assuming that resembling individuals would have similar fitness. However, depending on the type and range of possible values, short distances between parameters do not always indicate a similarity between individuals, and the heuristic would result ineffective in these cases.

Even though heuristics can speed up the execution time of a GA, the cost of the fitness function still constitutes a major challenge for real-time optimization of transport protocols. As an alternative, GAs can quickly provide suboptimal solutions that could allow protocols to satisfy more flexible QoS requirements. For instance, a stopping criteria could be specified so that the GA would return the first set of parameters that provides a minimum desired throughput given the current operating conditions.

Another challenge for real-time optimization of transport protocols using GAs is that the fitness evaluation of the same set of parameters may return different values over multiple runs. This inconsistency is a result of the variation of link characteristics at the time the evaluation is performed. If the operating conditions change quite often, the algorithm may never converge to an optimal solution because it cannot refine the characteristics of the best individuals in the population.

However, elitism is shown to be useful on retaining the best individuals so that they can be tested again and thus, over time, gain increasingly reliable fitness estimates when the fitness function is noisy. Moreover, in the presence of small amounts of noise, the accumulation of fitness statistics over multiple runs of the algorithm may help the GA to recover quickly from changes in the conditions of the network, although the inclusion of good individuals in the initial population can be ineffective if the optimal configuration of the protocol is located far away in the search space from previously found solutions.

Despite the difficulties associated with performing online optimization using GAs, the proposed algorithm was effective at improving the performance of the transport protocol for multi-hop scenarios. Also, even though the execution time of GATO prevents it from quickly adapting itself to changes in the operating conditions of the network, the use of problem specific heuristics can help GATO to further reduce the time required for a single optimization task in order to provide better performance over dynamic MANETs.

REFERENCES

Aggelou, G. (2005). *Mobile ad hoc networks: From wireless LANs to 4G networks*. New York: McGraw-Hill.

Ahn, C. W., & Ramakrishna, R. S. (2003). Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation*, *7*(4), 367–385.

Akan, O. B., & Akyildiz, I. F. (2004). ATL: An adaptive transport layer suite for next-generation wireless Internet. *IEEE Journal on Selected Areas in Communications*, *22*(5), 802–817.

Al Hanbali, A., Altman, E., & Nain, P. (2005). A survey of TCP over ad hoc networks. *IEEE Communications Surveys & Tutorials*, *7*(3), 22–36.

Barolli, L., Koyama, A., & Shiratori, N. (2003). A QoS routing method for ad-hoc networks based on genetic algorithm. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications* (pp. 175–179). Washington, DC: IEEE Computer Society.

Bova, T., Krivoruchka, T., & Cisco Systems. (1999). *Reliable UDP Protocol.* Retrieved January 11, 2009, from the Internet Engineering Task Force Web site: http://www.ietf.org/proceedings/99mar/I-D/draft-ietf-sigtran-reliable-udp-00.txt

Calagaz, J., Chatam, W., Eoff, B., & Hamilton, J. (2004). On the current state of transport layer protocols in mobile ad hoc networks. In *Proceedings of the 42nd Annual Southeast Regional Conference* (pp. 76–81). New York: ACM.

Cordeiro, C., & Agrawal, D. (2006). *Ad hoc & sensor networks: Theory and applications*. Hackensack, NJ: World Scientific.

De Jong, K. A. (2006). *Evolutionary computation: A unified approach.* Cambridge, MA: The MIT Press.

Elaarag, H. (2002). Improving TCP performance over mobile networks. *ACM Computing Surveys (CSUR)*, *34*(3), 357–374.

El Khayat, I., Geurts, P., & Leduc, G. (2005). Improving TCP in wireless networks with an adaptive machine-learnt classifier of packet loss causes. *Lecture Notes in Computer Science*, *3462*, 549–560.

ElRakabawy, S., Klemm, A., & Lindemann, C. (2005). TCP with adaptive pacing for multihop wireless networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing* (pp. 288–299). New York: ACM.

Fu, Z., Luo, H., Zerfos, P., Lu, S., Zhang, L., & Gerla, M. (2005). The impact of multihop wireless channel on TCP performance. *IEEE Transactions on Mobile Computing*, *4*(2), 209–221.

Galily, M., Roudsari, F. H., & Riazi, A. (2005). Applying fuzzy sliding mode control based on genetic algorithms to congestion avoidance in computer network. *International Journal of Information Technology*, *11*(10).

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, *1*, 69–93.

Granados, A. (2009). Simple Reliable UDP (Version 1.2.1) [Computer software]. Retrieved May 3, 2009, from http://sourceforge.net/projects/rudp/

Gurtov, A., & Floyd, S. (2004). Modeling wireless links for transport protocols. *ACM Computer Communication Review*, *34*(2), 85–96.

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

Huang, Y. C., Bhatti, S., & Parker, D. (2006). Tuning OLSR. In *Proceedings of the 17th International IEEE Symposium on Personal, Indoor and Mobile Radio Communications* (pp. 1–5). Washington, DC: IEEE Computer Society.

Information Sciences Institute. (1981). *Transmission Control Protocol.* Retrieved April 25, 2009, from the Internet Engineering Task Force Web site: http://www.ietf.org/rfc/rfc0793.txt

Jain, K., Padhye, J., Padmanabhan, V. N., & Qiu, L. (2003). Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking* (p. 66-80). New York: ACM.

Java networking overview. (n.d.). Retrieved June 13, 2009, from http://java.sun.com/j2se/1.5.0/docs/guide/net/overview/overview.html

Li, Y., Qiu, L., Zhang, Y., Mahajan, R., & Rozner, E. (2008). Predictable performance optimization for wireless networks. In *Proceedings of the ACM Conference on Data Communication* (pp. 413–426). New York: ACM.

Lochert, C., Scheuermann, B., & Mauve, M. (2007). A survey on congestion control for mobile ad hoc networks. *Wireless Communications and Mobile Computing*, *7*(5), 655–676.

Michalewicz, Z., Eiben, A. E., & Hinterding, R. (1999). Parameter control on evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *3*(2), 124–141.

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, *9*(3), 193–212.

Mitchell, M. (1996). *An introduction to genetic algorithms*. Cambridge, MA: The MIT Press.

Montana, D., & Redi, J. (2005). Optimizing parameters of a mobile ad hoc network protocol with a genetic algorithm. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation* (pp. 1993–1998). New York: ACM.

Mulabegovic, E., Schonfeld, D., & Ansari, R. (2002). Lightweight Streaming Protocol (LSP). In *Proceedings of the 10th ACM International Conference on Multimedia* (pp. 227–230). New York: ACM.

Navaratnam, P., Cruickshank, H., & Tafazolli, R. (2007). A link adaptive transport protocol for multimedia streaming applications in multi hop wireless networks. In *Proceedings of the 3rd International Conference on Mobile Multimedia Communications* (pp. 1–6). Brussels, Belgium: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.

Rao, N. S., & Iyengar, S. (2004). On throughput stabilization of network transport. *IEEE Communication Letters, 8*(1), 66–68.

Roy, A., & Das, S. K. (2004). QM2RP: A QoS-based mobile multicast routing protocol using multi-objective genetic algorithm. *Wireless Networks, 10*(3), 271–286.

Russell, S., & Norvig, P. (2002). *Artificial intelligence: A modern approach* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

Sipper, M. (1996). *A brief introduction to genetic algorithms.* Retrieved January 11, 2009, from http://www.cs.bgu.ac.il/~sipper/ga.html

Srinivas, M., & Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics, 24*(4), 656–667.

Suydam, A. J. (2004). Combined numerical optimization of TCP and routing protocol parameters in wireless ad hoc networks. In *Proceedings of the 20th Annual Rensselaer at Hartford Computer Science Conference.* Hartford, CT: Rensselaer Polytechnic Institute.

Tanenbaum, A. S. (2002). *Computer networks* (4th ed.). Upper Saddle River, NJ: Prentice Hall.

Turgut, D., Das, S. K., Elmasri, R., & Turgut, B. (2002). Optimizing clustering algorithm in mobile ad hoc networks using genetic algorithmic approach. In *Proceedings of the IEEE Global Telecommunications Conference* (pp. 62–66). Washington, DC: IEEE Computer Society.

Vasconcelos, J. A., Ramirez, J. A., Takahashi, R. H., & Saldanha, R. R. (2001). Improvements in genetic algorithms. *IEEE Transactions in Magnetics*, *37*(5), 3414–3417.

Wright, A. H. (1991). Genetic algorithms for real parameter optimization. *Foundations of Genetic Algorithms*, *1*, 205–218.

Wu, Q., & Rao, N. S. (2005). A class of reliable UDP-based transport protocols based on stochastic approximation. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communication Societies* (Vol. 2, pp. 1013–1024). Washington, DC: IEEE Computer Society.

Xylomenos, G., Polyzos, G. C., Mahonen, P., & Saaranen, M. (2001, April). TCP performance issues over wireless links. *IEEE Communications Magazine*, *39*(4), 52–58.

Ye, T., & Kalyanaraman, S. (2001). *An adaptive random search algorithm for optimizing network protocol parameters.* Unpublished manuscript, Rensselaer Polytechnic Institute, Troy, NY.

Ye, T., & Kalyanaraman, S. (2004). A recursive random search algorithm for network parameter optimization. *ACM SIGMETRICS Performance Evaluation Review*, *32*(3), 44–53.