INTELLIGENT PACKETS FOR

DYNAMIC NETWORK ROUTING

by

Suihong Liang

Submitted in partial fulfillment of the requirements
For the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2002

DALHOUSIE UNIVERSITY
Faculty OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "Intelligent Packets for Dynamic Network Routing" by Suihong Liang in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated:____________________

Co-Supervisor:______________________
Dr. Nur A. Zincir-Heywood

Co-Supervisor:______________________
Dr. Malcolm I. Heywood

Reader: ______________________
Dr. Evangelos E. Milios

DALHOUSIE UNIVERSITY

DATE: April 2002

AUTHOR: Suihong Liang

TITLE: Intelligent Packets in Dynamic Routing Network

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: Master of Computer Science CONVOCATION: May YEAR: 2002

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

The author attests that permission has been obtained for the use of any copyrighted material appearing in the thesis (other than the brief excerpts requiring only proper acknowledgement in scholarly writing), and that all such use is clearly acknowledged.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Symbols Used

GA	Genetic Algorithms
TTL	Time to Live
RIP	Routing Information Protocol
OSI	Open System Interconnect Reference Model
OSPF	Open Shortest Path First
ISIS	Intermediate System to Intermediate System
EIGRP	Enhanced Interior Gateway Router Protocol
EGP	Exterior Gateway Protocol
BGP	Border Gateway Protocol
ABR	Border routers
AS	Autonomous system
ASBR	AS boundary router
SPF	Shortest Path First
SPS	Static Path Shortest
AP	Arrived Packet

List of Interchangeable Terms

Node & Router;
Path & Route;
Ant & Agent;
GA agent, Chromosome & Agent.

Abstract

A distributed GA (Genetic Algorithm) is designed for the packet switched network routing problem under minimal information, i.e., without information exchange, every node only knows the existence of its neighboring nodes. The requirements of such a problem mean that intelligent packets are required to possess more intelligence than was the norm. To this end a distributed GA approach is developed and benchmarked against the AntNet algorithm under similar information constraints. A profile of AntNet under local and global information is developed with the proposed distributed GA clearly improving on the AntNet algorithm under local information constraints.

Keywords: Network Routing, Swarm Intelligence, Genetic algorithm, evolutionary computing, decentralized.

Acknowledgements

I am very grateful to the people who helped me with this work. First of all, I would like to acknowledge Dr. Nur A. Zincir-Heywood & Dr. Malcolm I. Heywood, for their instructive guidance and their availability with their precious time. Many thanks are due to Min Shi, for her professional yet unique suggestions, and constant encouragement during all the phases of the work. For help with reviewing this thesis, I thank Phyllis Chubb MA & William Chubb.

1 Introduction

1.1 The problem – Routing

Network information systems and telecommunication in general rely on a combination of routing strategies and protocols to ensure that information sent by a user is actually received at the desired remote location. In addition, the distributed nature of the problem means that multiple users can make requests simultaneously. This results in delayed response times, information loss or other reductions to the quality of service objectives on which users judge network service.

For example, the Internet consists of a huge amount of local networks interconnected by gateways. Such gateways, generally called routers, usually have physical connections (e.g., Fiber, Satellite, coaxial cable) or network interface ports (e.g., Ethernet) onto many networks. The determination of the appropriate gateway and port for a particular data packet is called routing. By exchanging information among the other routers, a router usually maintains a list of Internet addresses and their corresponding location in the network. Such a list is called routing table. Routers near the center of a network generally have very large routing tables; those near the edges have small tables. Although the routing table may be configured by hand, it is usually configured to automatically use "Routing Protocols". The routing protocol allows routers to periodically exchange their knowledge of the network. After a period of time, the router becomes aware of all the possible ways to reach any end system in the network. It therefore updates its own routing table, building a picture of how to reach other parts of the network.

Protocols are used to implement handshaking activities such as error checking and receiver acknowledgements. In this work, we are interested in the routing problem on computer networks. In doing so, we do not consider protocol issues.

The problem of Routing usually refers to the process used to build the routing table on each router, and determine how a packet travels from its source to destination. From a single packet's point of view, the objective is to arrive at its destination in the shortest possible time, while from the whole network's point of view, the objective is to deliver maximum number of packets in minimum average trip time, use minimum network resources, such as memory, network link, router CPU, etc., and prevent traffic congestion from happening. What's more, we should not neglect some important facts of the problem, such as the local information constraint (section 3.3.1). Thus, routing is an optimization problem yet with many constraints.

1.2 Definitions and Characteristics of the Network Routing Problem

Before studying the routing problem of the data network, it is necessary to mention a bit the two basic connection strategies. The traditional telephone network uses circuit switching, whose key techniques are: must have initial path setup, path is maintained for the entire duration of communication; provides specific amount of dedicated bandwidth to each user, and messages are sent in order along the same path. The packet switching of the data network, on the contrary, needs no initial path setup, and allows multiple users to share the network facilities and bandwidth. Each message is fragmented into packets at the source hence each packet is independent. Packets may be routed along different paths. However, they are reassembled at the receiver end of the message. Therefore, traffic on the network is bursty and can be aggregated to maximize the use of the on demand bandwidth resources.

On computer networks, locations are called nodes. There is a link between two nodes if there is a path (of one or more connections) between the two locations. The nodes and the links form a network. Thus, the routing problem is that of determining routes for transferring information from source to destination nodes as efficiently as possible.

There are several queues within a router, such that for each network interface port, there are two queues assigned, one for incoming packets, one for the outgoing packets. The basic operation upon receiving a data packet *p* is: the router first put *p* into the incoming queue, when *p* gets the front the queue and the router processor finishes handling the packets with higher priority, the processor fetches *p* into the processor buffer, checks if *p* has destination to the local network, if true, then sends *p* to the its final destination; if false, the processor looks through the routing table, finds an appropriate next hop router, then put *p* into the outgoing queue of the corresponding port. The router sends packets one by one onto the outgoing links once the links get available. Subject to the limited capacity of memory space, queue lengths are limited. Once the limit is reached, any incoming packets are discarded. When doing simulation, treating all the incoming queues as one and assuming unlimited queue lengths will simplify the model, see Figure 11.

Several properties help make a route efficient. For example, routes passing through the same node more than once (a loop) are especially undesirable. Thus, a route (e.g., from *a* to *z*) with a loop can be represented as {*a, s, d, f, ..., w, d, ..., z*}. Removing the sub-route {*f, ..., w, d*} gives a shorter route both as measured in time and distance. Moreover, it is not unheard of for an infinite loop to occur. In this case, packets are continuously forwarded about a sub-path before their time-to-live (TTL) expires. Thus, a routing algorithm should prevent loops from happening.

The routing problem has several generic properties, which make it particularly challenging. Specifically, the problem is distributed in nature, every router independently stands alone, processing incoming packets, forwarding packets to outgoing links, and multiple users may make requests simultaneously, all these result in delayed response times, information loss or other reductions to the quality of service objectives. The problem is also dynamic; hence a solution that is sufficient for presently experienced network conditions may well be inefficient under other loads experienced by the network. Moreover, the traffic experienced by networks is subject to widely varying load conditions, making it impossible to design for a single 'typical' network condition. By means of the "Physical" and "Data Link" layers of OSI model, every router knows the existence of its neighboring nodes, and this is the only knowledge it has about the network before any information exchange performed in higher layers. In this thesis, this knowledge is called *local information*. One of the goals of routing is to help the nodes with the local information to acquire knowledge about the topology of the network. In this work, this knowledge is called *global information*. However, global information implies a priori knowledge, which is not realistic on computer networks. For example, nodes of the network are unaware of the network connectivity, number of nodes comprising the network, resources associated with each node or link. Hence a solution that assumes access to any form of global information is not desirable. Therefore, in this work, any further information is gathered using 'agents' that are responsible for collecting data as they travel across the network. In other words, an agent represents the intelligent packet in this context.

Since the problem is distributed and dynamic, centralized solutions (e.g., OSPF, EIGRP, BGP, etc.) are never good solutions, because such a mechanism would heavily rely on the central/designated routers to gather and exchange routing information (Minar *et. al.*, 1999). For example, the backbone routers, area border routers (ABRs), and anonymous system (AS) boundary routers (ASBRs) are the central routers in OSPF. Human configuration becomes inevitable for such kinds of solutions, as in a large network, it is nearly impossible to achieve the optimum configuration just by hand crafting. Each time network load changes sufficiently to impact quality of service, human intervention is necessary to reconfigure the routing protocols. This issue will be discussed in section 2.

1.3 Distributed Genetic Algorithms

Genetic Algorithms are a class of generic search algorithms that perform a parallel search over a fixed “population” of candidate solutions. To do so, survival of the fittest and observations from genetics are used to guide the general mode of operation. Specifically, a selection operator provides the pressure to improve the contents of the population, whereas search operators address the exploitation – exploration trade off associated with improving the performance of individuals. Such a scheme has proved robust both theoretically (Holland’s schema theory (Goldberg, 1989)) and in practice, with several international conferences held each year.

A genetic algorithm applies the principles of evolution found in nature to the problem of finding an optimal solution. To do so, the problem is encoded in a series of bit strings that are manipulated by the algorithm. A genetic algorithm for optimization is different from "classical" optimization methods in several ways:

- Stochastic Decision Making. Actions taken by the GA (selection of individuals for reproduction and application of search operators) are not deterministic. That is to say, decisions are stochastic, thus allowing individuals to survive that do not necessarily represent the current best-case solution alone. This avoids greedy search operators such as hill climbing, which often become ‘stuck’ at local minima.
- Population. Holland’s schema theory indicates that solutions are identified by different schema solving different components of the problem across multiple individuals. The GA therefore improves the overall fitness of the

population as a whole such that a single individual is able to solve the problem formulated by the fitness function.

- Mutation. Inspired by the mutation of DNA in natural evolution, evolutionary algorithms creates new solutions by periodically making random changes or mutations in one or more individuals of the current population, yielding a new candidate solution, which may be better or worse than existing population members. There are many possible ways to perform a mutation, but it typically takes the form of a single point operator applied to single genes, Figure 1. Mutation is therefore responsible for introducing new candidate solutions into the population (exploration operator).

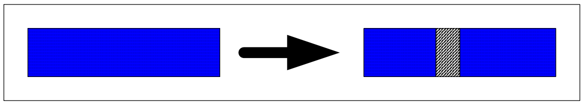

Figure 1: Mutation

- Crossover. Inspired by the crossover of DNA that occurs in reproduction of nature creatures. An evolutionary algorithm attempts to combine elements of existing solutions/individuals to create a new solution/individual, with some of the features from each "parent" (exploitation operator). As with mutation, there are many ways to choose the parents and perform crossover, figure 2 illustrates the action of a single point and figure 3 multipoint crossover operators. Crossover is therefore responsible for exploiting material (schema) currently available in the population to suggest new candidate solutions.

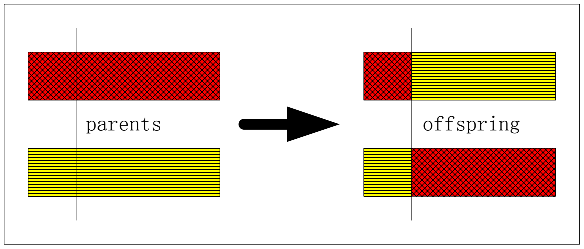

Figure 2: Single-point Crossover

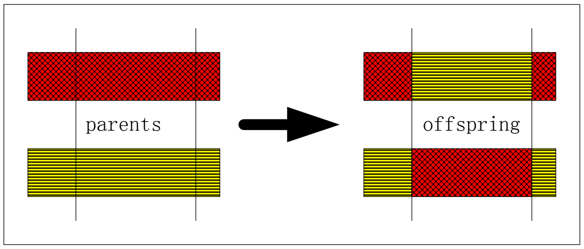

Figure 3: Multi-point Crossover

- Selection. Inspired by the natural selection in evolution -- an evolutionary algorithm performs a selection procedure in which the fittest members of the population are more likely to survive, and the least fit individuals are most likely to be eliminated. In a constrained optimization problem, the notion of fitness firstly depends on whether a solution is feasible (i.e. all constraints are satisfied), and secondly on the relative solution quality (as measured by the fitness function) value. The selection process is the step that guides the evolutionary algorithm towards ever-better solutions according to the fitness function.
- Representation. A key step in successfully applying GAs to solve problems is identifying a scheme for encoding solutions such that the search process itself is efficient. At the very least application of the search operators should not result in individuals that cannot be evaluated; whereas decoding an individual's into the equivalent genotype before fitness evaluation should not represent a significant computational overhead.

- Pragmatics. Most machine learning algorithms conduct a single point search. This is sufficient when the problem is well behaved to begin with, but will lead to either stalls at local minima in the search space (and require multipoint sampling processes to direct the restarting of the algorithm), or a prolonged search process when the problem non-deterministic. By non-deterministic we imply that the problem cannot be formulated in terms of an objective that is both smooth and continuous. Moreover, many-to-one mappings may exist between possible model solutions and their corresponding measure of error. Finally, the objective may be multimodel or deceptive, in the sense that hill climbing search algorithms will typically lead to local minima. If the problem does not conform to any of these types, then applying a GA will undoubtedly not be computationally efficient. Problems conforming to this context are typically only described in terms of data or an environment from which reinforcement is available. The objective is usually framed as minimizing or maximizing an *a priori* identified performance metric(s). As such there will be many, possibly an infinite, set of models capable of satisfying the objective, but only a much smaller set able to actually provide a 'generalized' model solution. It is for these reasons that GA is utilized in the case of the packet routing problem under local information constraints considered in this work.

Generally, a genetic algorithm can be outlined as follows (Mitchell, 1997):

```
GA (Fitness, Fitness threshold, p, r, m)
  Initialize: $P \leftarrow p$ random hypotheses;
  Evaluate: for each $h$ in $P$
              compute $Fitness(h)$;
  Evolve: While $[\max_h Fitness(h)] < Fitness\ threshold$
              Select: Probabilistically select $(1 - r) \times p$ members of $P$ to add to $P_S$;
```

```
                                                                  (Con't)
              where probability $\Pr(h_i)$ of selecting $h_i$ from $P$ is given by
                $\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_i)}$
            Crossover: Probabilistically select $r \times p/2$ pairs of hypotheses from $P$;
              For each pair $<h_1, h_2>$
                produce two offspring by applying the Crossover operator;
                add all offspring to $P_S$;
            Mutate: Invert a randomly selected bit in $m \times p$ random members of $P_S$;
            Update: $P \leftarrow P_S$;
            Evaluate: for each $h$ in $P$
                        compute $Fitness(h)$;
  Return the hypothesis from $P$ that has the highest fitness;
```

In order to enhance the parallel co-evolution, a single population is divided into a number of sub-populations or demes (Nowostawski, 1999). Demes exchange individuals at a certain rate, called the migration rate. By doing so it is possible to avoid premature domination of a single population by one schema that reaches a (relatively) high fitness too early in the evolutionary cycle.

1.4 The sections

In this study, we will first introduce the classical routing algorithms and protocols that are being widely used in practice. Other research approaches, with a discussion of their strengths and weaknesses, are summarized in section 2. In section 3, we will discuss AntNet, one of the most popular approaches to solving the routing problem in a distributed manner. This work belongs to the Social Insect Metaphor methodology and provides a major motivation for this study. Specifically, we introduce what happens when a strict local information constraint is enforced, the ensuing investigation effectively forming the case for incorporating "agents" into the routing process. Section 4 will introduce our own GA approach to the routing problem. Section 5 will talk about the experiments, and the experimental results will be presented. Section 6 is the conclusion and suggestions for future work.

1.5 Publications

The work of the thesis resulted in two published papers (with Dr. Nur Zincir-Heywood and Dr. Malcolm Heywood), they are:

- "The Effect of Routing under Local Information using a Social Insect Metaphor", The 2002 IEEE World Congress on Computational Intelligence (Hawaii, U.S.A, May 2002);

- "Intelligent Packets for Dynamic Network Routing Using Evolutionary Strategies", The 2002 Genetic and Evolutionary Computation Conference (New York city, U.S.A, July 2002). (Nominated for "Best paper" award)

In the first case, the drawback of the AntNet algorithm under local information is emphasized. In the second case, a methodology based on a Distributed GA is proposed for directly addressing the problem of routing with local information constraints alone.

2 Literature Survey

2.1 Classical Routing Algorithms

As we know, a network can be denoted as a graph, which consists of a set of nodes/vertices and a set of links/edges, which connect the nodes in the manner that each link joins two nodes. The following graph (Figure 4) represents the network of the Japanese backbone (NTTNet). NTTNet is the NTT (Nippon Telephone and Telegraph company) fiber-optic corporate backbone. NTTNet is a 55-node, 162-bidirectional link network. Link bandwidth is 6Mbit/sec, while propagation delays range around 1 to 5 msec. It is a narrow long configuration in which the degree of connectivity is low (from 1 to 5), when compared to the US backbone. Hence the Japanese network provides a more demanding configuration for testing routing algorithms, as higher degrees of connectivity lower the possibility of packet loss due to loops, timeouts, i.e., in a narrow long shaped network, once a packet is forwarded in a wrong direction, it might never have the chance to be routed to the desired destination.

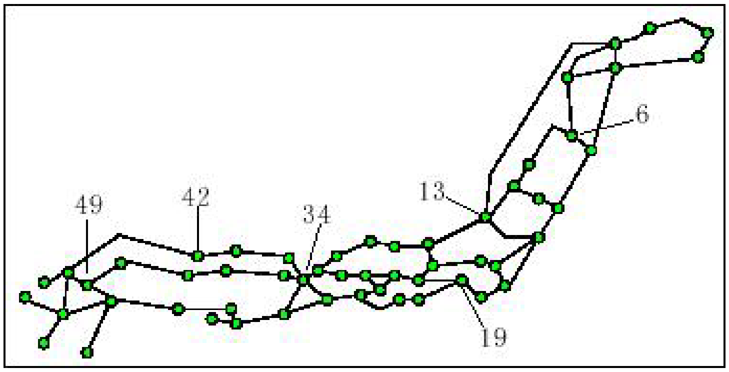

Figure 4: NTTNet topology

The nodes and links have capacities, such as buffer size and processing time for nodes, bandwidth for links. A non-directed graph $G = \{N, A\}$ with a node set N and an arc set A provides a formal framework for describing network connectivity. Finding the shortest paths among nodes can be solved in polynomial time (using Dijkstra's algorithm, Bellman-Frod's algorithm), while flow optimization, i.e., maximizing packets delivery (throughput) when links have transmission limitations is known to be a *NP*-complete problem (Ahuja *et. al*., 1993). Note, however, this classical definition of the problem assumes a static (worst case) load and complete information. In practice neither are necessarily known and the problem becomes more difficult.

The routing protocols are responsible for exchanging routing information between routers, and helping each router build a routing table for each possible destination sub-network. Packet destinations are therefore expressed in terms of sub-networks (Norris, Pretty, 2000). Figure 4 represents the node connectivity above the sub-network level. It is only at this level that we are interested in routing.

The routing protocols being widely used on the Internet are usually based on one of the following general principles: Static Routing, Distance Vector Routing, Link State Routing, or Path Vector Routing. In small networks, for example, a small network of a small business with leased line connection to the Internet, Static Routing is commonly used to configure the default route. When the topology of a network changes frequently, static routing is no longer suitable for such a dynamic environment; distance vector routing and link state routing have advantage over static routing. Distance vector routing relies on the regular updates of routing information to keep the routing tables on every router up to date. The objective of link state routing is to let every router maintain a map of the network topology.

Routing protocol RIP2 (Routing Information Protocol version 2, RFC2453, STD0056) is widely used in small networks. As the original Interior Gateway Routing Protocol (IGRP), RIP is a kind of Distance Vector Routing algorithm, more specifically, based on the distributed Bellman-Ford algorithm for the Graph Shortest Path problem.

RIP works well in small networks, but becomes increasingly less efficient as network size increases. It also suffers from the count-to-infinity and slow convergence problems. Count-to-infinity is an issue with hop counts, it happens in some subtle network failure situation resulting from mutual deception routing information updates. All distance vector protocols are susceptible to this well-known "count to infinity" problem (Perlman, 1992). Look at the following example (Figure 5):

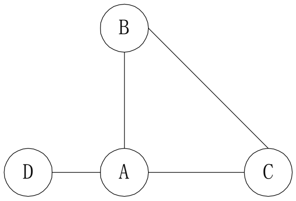

Figure 5: A simple example of the count to infinity problem

After convergence, *A*, *B*, and *C* has an entry for route to *D*, Table 1.

Routing table of *A*			Routing table of *B*			Routing table of *C*		
Destination	# of Hops	Next hop	Destination	# of Hops	Next hop	Destination	# of Hops	Next hop
……			……			……		
D	1	*D*	*D*	2	*A*	*D*	2	*A*
……			……			……		

Table 1: Routing tables after convergence

Consider the situation that link between *A* and *D* is down. Suppose link *AB* is much slower than link *AC*. Then the Update events are:

(1) *A* sets path to *D* with cost ∞, and sends update to *B* and *C*;

(2) *C* gets update information from *A*, sets path to *D* with cost ∞;

(3) *B* tells *C* path to *D* with cost 2; *C* updates;

(4) *B* gets update information from *A*, sets path to *D* with cost ∞;

(5) *C* tells *A* path to *D* with cost 3; *A* updates;

(6) *A* tells *B* path to *D* with cost 4; *B* updates;

(7) *B* tells *C* path to *D* with cost 5; *C* updates;

……

This update cycle continues and the cost to *D* goes to infinity. Slow convergence problem is caused by the "count to infinity" problem, in such a way that, routers *A*, *B*, and *C* waste time in updates before they realize the route to *D* is unavailable. There are many other network topologies suffer these problems. Split Horizon and Split Horizon with poison were then proposed, but they can only decrease the possibility of count to infinity problem.

OSPF (Open Shortest Path First, RFC2328, STD0054) is a more modern protocol from the IGRP family, which is based on the Dijkstra's algorithm. OSPF is much more successful than RIP and is used in many networks, although it requires human configuration. That is, a series of assumptions, based on global information, is required to configure the protocol.

RIP and OSPF belong to IGRP. IGRP protocols are routing protocols for autonomous systems (ASs). These include: RIP, EIGRP, ISIS, OSPF, and SPF. An AS is a group of routers that are within one administrative domain and that run the same routing protocol. The public Internet nowadays is composed of ASs, and EGP (Exterior Gateway Protocol), which are designed for routing among the ASs. BGP (Border Gateway Protocol) is a kind of EGP. BGP uses path vector routing, where a path is an ordered list of AS numbers. Every entry in the routing table contains the destination network, next router, and path to reach the destination.

As shown above, a range of different routing protocols exist, each with their own strengths and weaknesses. Static routing is simple, but has poor scalability and robustness properties (which is a key advantage of dynamic routing). RIP suffers count-to-infinity and slow convergence problems, and takes up a lot of bandwidth. All these make RIP (or other distance vector protocols) only good for small networks, and not competent for larger networks; OSPF (or other link state protocols) are designed with scalability, but their complexity makes it hard to design and configure the network efficiently. The path vector routing attribute of BGP leads to some attractive features, such as policy routing, loop prevention, and so forth. OSPF and BGP have a common weakness in that the design relies on several core routers. As discussed in section 1.2, such a centralized design has many drawbacks for the case of highly distributed networks.

2.2 Recent Research Approaches

Several approaches have been proposed for addressing these objectives including: active networking (Tennenhouse *et. al.*, 1997), social insect metaphors (Di Caro, Dorigo, 1998), (Dorigo *et. al.*, 1996), cognitive packet networks (Gelenbe *et. al.*, 1999), evolutionary approaches (Sinclair, 1993), (Munetomo *et. al.*, 1997), and what might be loosely called other ‘adaptive’ techniques (Corne *et. al.*, 2000). The evolutionary approaches usually represent a route/path by an ordered list of nodes, and then try to achieve the routing problem by evolving the “paths” or “routing tables”. Moreover, Evolutionary and ‘adaptive’ techniques typically involve using evolutionary or neural techniques to produce a ‘routing controller’ as opposed to a ‘routing table’ at each node, where the controller typically requires knowledge of the global connectivity to ensure a valid route. Both the social insect metaphor and the cognitive packet approach provide a methodology for routing, without constraints; by using the packets themselves to investigate and report network topology and performance. Similarly, mobile agents discover edges by traversing them, and update the routing table on the landed hosts (Minar *et. al.*, 1999).

All methods as currently implemented suffer from one drawback or another. For example, cognitive packet networks and active networking algorithms attempt to provide routing programs at the packet level, hence achieving scalable run time efficiency becomes an issue. The Social Insect Metaphor approach is discussed in the following section, and limitations investigated under a strict local information constraint. This will form the basis for combining a multi-agent approach with Genetic Algorithms for avoiding any reference to global information.

3 Social Insect Metaphor Approach

3.1 Introduction

As indicated above active networking (Tennenhouse *et. al.*, 1997) and cognitive packet (Gelenbe *et. al.*, 1999) based approaches emphasize a per packet mechanism for routing. The aforementioned Evolutionary and 'adaptive' techniques (Corne *et. al.*, 2000) tend to emphasize adding 'intelligence' to the routers leaving the packets unchanged. A social insect metaphor provides a middle ground in which the concepts of a routing table and data packet still exist, but in addition, intelligent packets – *ants* – are introduced that interact to keep the contents of the routing tables up to date. To do so, the operation of ant packets is modeled on observations made regarding the manner in which worker ants use chemical trails as a method of indirect *stigmergic* communication. Specifically, ants are only capable of simple stochastic decisions influenced by the availability of previously laid stigmergic trails. The chemical denoting a stigmergic trail is subject to decay over time, and reinforcement proportional to the number of ants taking the same path. Trail building is naturally a bi-directional process, ants need to reach the food (destination) and make a successful return path, in order to reinforce a stigmergic trail (Forward only routing has also been demonstrated (Heusse *et. al.*, 1998). Moreover, the faster the route, then the earlier the trail is reinforced. An ant on encountering multiple stigmergic trails will *probabilistically* choose the route with greatest stigmergic reinforcement. Naturally, this will correspond to the 'fastest' route to the food (destination). The probabilistic nature of the decision, however, means that ants are still able to investigate routes with a lower stigmergic signature (probability).

This approach has proved to be a flexible framework for solving a range of problems including the traveling sales man problem (Heusse *et al.*, 1998) and the quadratic assignment problem (Maniezzo *et al.*, 1999). The work reported here follows the ‘AntNet’ algorithm of Di Caro and Dorigo (Di Caro, Dorigo, 1998), and is informally summarized as follows:

1) Each node in the network retains a table of packet destination frequency as seen on data packets passing through that node. This is used to periodically, but asynchronously, launch ‘forward’ ants with destinations stochastically sampled from the collected set of destinations;
2) Once launched a forward ant uses the routing table information to make probabilistic decisions regarding the next hop to take at each node. While moving, each forward ant collects time stamp and node identifier information and this is later used to update the routing tables along the path followed;
3) If a forward ant re-encounters a node previously visited before reaching the destination, it is killed (case of a loop);
4) On successfully reaching the destination node, total trip time is estimated and the forward ant converted into a backward ant;
5) The backward ant returns to the source using exactly the same route as recorded by the forward ant. Instead of using the data packet queues, however, the backward ant uses a priority queue;
6) At each node visited by the backward ant the corresponding routing table entries are updated to reflect the relative performance of the path;
7) When the backward ant reaches the source it dies.

Although providing for a robust ant routing algorithm under simulation conditions, an assumption is made, which inadvertently implies the use of global information (Di Caro, Dorigo, 1998). Table 2 is the routing table in AntNet for the case of a node *k* with *L* neighbors. The routing table is organized as in vector-distance algorithms, but represents probabilities. The definition of a routing table is such that it assumes every node (destination) has a unique location in the routing table. In practice this is never the case. To do so would assume that it is first feasible, and secondly, should the network configuration ever change, then all nodes should be updated with the new configuration information. And, the probabilities of every column add up to 1.0, which means a certain event that there must be some path from any node to any node – a connected graph; but, that fact is, the network could be broken into two or more separate components. Moreover, as forward ants propagate across the network, the amount of information they need to 'carry' also increases (node identifier and time stamp). Finally, the availability of globally synchronized time is also assumed. Section 3.3 discusses the implementation of the AntNet algorithm without recourse to global information.

	All Network Nodes (Possible Destinations)						
The *L* Neighbors	$P_{N1,1}$	$P_{N1,2}$	---	$P_{N1,k\text{-}1}$	$P_{N1,k+1}$	---	$P_{N1,55}$
	$P_{N2,1}$	$P_{N2,2}$	---	$P_{N2,k\text{-}1}$	$P_{N2,k+1}$	---	$P_{N2,55}$
	---	---	---	---	---	---	---
	$P_{NL,1}$	$P_{NL,2}$	---	$P_{NL,k\text{-}1}$	$P_{NL,k+1}$	---	$P_{NL,55}$

Table 2: Original AntNet Routing Table T_k at network node k ($1 \leq k \leq 55$) on the NTTNet

3.2 AntNet Algorithm

It is assumed that routing tables, T_k, exist at each node, k, in which a routing decision is made. Tables consist of '*L*' rows, one row for each neighboring node/link. As far as a normal data packet is concerned, if the destination d from the current node k, is a neighbor then the routing is still a stochastic decision. In all other cases, a router is selected based on the neighbor node probabilities.

a) New forward ants, F_{sd}, are created periodically, but independently of the other nodes, from source, s, to destination node, d, in proportion to the destination frequency of passing data packets. Forward ants travel the network using the same priority structures as data packets, hence are subject to the same delay profiles;

b) Next link in the forward ant route is selected stochastically, $p'(j)$, in proportion to the routing table probabilities and length of the corresponding output queue.

$$p'(j) = \frac{p(j) + \alpha l_j}{1 + \alpha\left(|N_k| - 1\right)}$$

where $p(j)$ is the probability of selecting node j as the next hop; α weights the significance given to local queue length versus global routing information, $p(j)$; l_j is the queue length of destination 'j' normalized to the unit interval; and N_k is the number of links from node k;

c) On visiting a node different from the destination, a forward ant checks for a buffer with the same identifier as itself. If such a buffer exists the ant must be entering a cycle and dies. If this is not the case, then the ant saves the previously visited node identifier and time stamp at which the ant was serviced by the current node in a buffer with the forward ant's identifier. The total number of buffers at a node is managed by attaching "an age" to buffer space and allowing backward ants to free the corresponding buffer space;

d) When the current node is the destination, $k = d$, then the forward ant is converted into a backward ant, B_{ds}. The information recorded at the forward ant buffer is then used to retrace the route followed by the forward ant;

e) At each node visited by the backward ant, routing table probabilities are updated using the following rule,

```
IF (node was in the path of the ant)
THEN $p(i) = p(i) + r\ \{1 - p(i)\}$
ELSE $p(i) = p(i) - r\ \mathrm{P}(i)$
```

where r $\in$ (0, 1] is the reinforcement factor central to expressing path quality (length), congestion and underlying network dynamics.

As indicated above, the reinforcement factor should be a factor of trip time and local statistical model of the node neighborhood. To this end (Di Caro, Dorigo, 1998) recommended the following relationship:

$$r = c_1\left(\frac{W_{best}}{t_{ant}}\right) + c_2\left\{\frac{I_{\sup} - I_{\inf}}{(I_{\sup} - I_{\inf}) + (t_{ant} - I_{\inf})}\right\}$$

where t_{ant} is the actual trip time taken by the ant; c_1 and c_2 are constants that weigh the importance of each term; W_{best} is the best-case trip time to destination d over a suitable temporal horizon, W;

$$I_{inf} = W_{best};$$
$$I_{sup} = \mu_d + \{\sigma_d / [W(1 - \gamma)]^{0.5}\}.$$

In the above equation, γ is a constant, determining the confidence interval. An array $M_k(\mu_d, \sigma_d; W_d)$, of data structures defines a statistical model for the traffic distribution over the network as seen by the local node k. For each possible destination d in the network, the estimates for mean, μ_d, and variant, σ_d, of the trip time reflect the expected trip time to d, and the stability. These adaptive values are also made iteratively, using the trip time information. Thus,

$$\mu_d = \mu_d + \eta(o_{kd} - \mu_d)$$
$$(\sigma_d)^2 = (\sigma_d)^2 + \eta\{(o_{kd} - \mu_d)^2 - (\sigma_d)^2\}$$

where o_{kd} is the newly observed trip time from current node k to destination d. From the above algorithm, it is, therefore, apparent that ants are required to make decisions under more uncertainty than was previously the case. Moreover, the trip time information is updated incrementally based on the recorded trip duration between current node, *k*, and ultimate destination, *d*. This means that it is no longer necessary to carry all node and duration information as a ‘stack’ to the target duration as in the original model (Di Caro, Dorigo, 1998). Only the previous step information is therefore necessary.

3.3 AntNet Algorithm With Local Information

3.3.1 Assumptions

Research has demonstrated that the AntNet algorithm outperforms OSPF (Di Caro, Dorigo, 1998), which belongs to link state routing algorithm. However, as indicated above it makes some assumptions in terms of the information available in the routing table that make it unrealistic in practice. First, every node in the network has a column in the AntNet algorithm. This implies that every router must know of the existence of the rest of the world. That is to say, all routers are aware of the number of routers comprising a network and their addresses. This violates the shortsighted property (section 1.2) of the routers. Indeed, gaining such knowledge of the network is one of targets of the routing problem!

Secondly, its routing table implicitly implies the network connectivity assumption. The probabilities of each column in the routing table (Table 2) always sum up to 1.0, which means a certain event, that is, from any source node to any destination node, there is at least one path connecting them. The network will never be broken into separate components. But this is possible to happen.

In the following, we introduce a local information constraint into the AntNet algorithm with the objective of providing a platform for investigating these limitations further, section 4.

3.3.2 Removing the Global Information Assumption

One of the goals of our work is to investigate routing algorithms without global information. We remove the first assumption, i.e., the global information assumption (i.e., the number of routers in the whole network, the IDs of all the routers), to identify the sensitivity of the AntNet algorithm to this property.

In the AntNet algorithm without the first assumption (i.e., global information), the routing table is revised in such a way that it has 2 columns respectively, one column for the router's neighbors, and the other for the rest of the network nodes. The elements in the routing table are still probabilities. For example, $P_{n,d}$ is the probability to reach destination *d* via neighbor *n*. Table 3 is the routing table at node *k* for the AntNet algorithm with local information, i.e., every node only "sees" its neighbors.

	Neighbors as destinations	Destinations other than neighbors
The *L* Neighbors	$P_{N1, N1}$	$P_{N1,d}$ $d \neq N1$
	$P_{N2, N2}$	$P_{N2,d}$ $d \neq N2$
	------	------
	$P_{NL, NL}$	$P_{NL,d}$ $d \neq NL$

Table 3: Proposed Routing Table at any network node k on the NTTNet

This modification brings changes to the routing table in original AntNet algorithm (Liang *et. al.*, May 2002).

- Routing tables only detail the neighboring nodes. Such a limitation therefore places greater emphasis on the learning capacity of the ant. This is particularly significant during step *b)* of the ant forward pass (section 3.2). Table 2 and 3 illustrate the difference in available information for a node;

- Each node has a buffer in which forward ants deposit time stamp and identifier for the previous node. It is only the inter-node information, which is important;

- Time synchronization is treated as a protocol issue. That is to say, during low load conditions each node is responsible for letting neighboring nodes know what their current time clock is. Moreover, whenever interruptions to services are sustained, then the first step once a node returns to operation will be to reinitiate local time references. Usually the time synchronization is achieved by applying the Network Time Protocol (RFC 1305) to the routers.

4 GA Approach With Local Information to the Routing Problem

4.1 Discussion of the Problem and Previous Research Work

As indicated in the introduction, the routing problem is an *NP*-complete problem, and, it has several properties, such as its distributed and dynamic nature, which imply that one single static solution does not exist.

Static Routing, limited in its ability as its name indicates and is nowadays only used in simple situations; Distance Vector routing algorithms and link state routing algorithms are far away from handling the dynamic network conditions efficiently, especially when network equipment have many failures or network topology changes very often. The traffic and overhead caused by the routing protocols will become burdens of the network. Moreover, all the currently used routing protocols rely on some critical routers to collect, exchange and distribute the routing information to achieve the goal. This centralized nature makes the routing protocols incapable for distributed nature of the routing problem, because if some of the central routers or just some links of them have problem, the network will be affected seriously. Thus, experienced network engineers are required to configure the routing protocols each time network conditions change.

We have discussed the drawbacks of the AntNet algorithm in a previous section. Yet, there are other problems with even the modified AntNet algorithm. For example, the routing table is fixed, while the network topology is highly dynamic! What if a new node, or a new link is added? In order to let other nodes adapt to the change, and be able to find a route to it, according to the AntNet algorithm, modification of the node structures (T_k & M_k) on every node of the whole network must be carried out: every node must add a new entry in the traffic distribution statistics array M_k, every node must add a new column in the routing table T_k for the new node, the new node's neighbors even have to add a new row in T_k. This would be a protocol overhead not explicitly addressed in the AntNet algorithm. More significantly, simulation results using the local AntNet algorithm indicate that the algorithm no longer is able to correctly route data packets without significant loss. In effect, the positive feedback mechanism central to the AntNet algorithm is disrupted.

4.2 Objective and the GA Approach

Our goal in this work is to remove the significance of global information in the AntNet routing algorithms with the help of Genetic Algorithm co-evolutionary mechanism (Liang *et. al.*, July 2002).

The objective of this work is to investigate a scenario in which the entries themselves are identified dynamically. This will be a first step towards a co-evolutionary model capable of evolving solutions to the packet switched routing problem. The ants, in this case, take the form of individuals from a distributed Genetic Algorithm (GA), hereafter referred to as GA-agents. Individual chromosomes travel the network using a variable length string of next hop offsets (detailed in Figure 8: Processing GA-agents), e.g., {1, 5, 0, 4, 2, 3, 5} over the interval [0, *L*], where '*L*' is selected to enable indexing of node connectivity. In all the experiments of section 5.3.4, '*L*' is set to 6. On entering a node, a gene (next hop offset) is used to identify the next link using a clockwise count from the link, the GA-agent entered the node i.e. the next link is selected as the modulus of (gene % # of links). Such a representation is then independent of the specific network connectivity, unlike say the GA approach in (Munetomo *et. al.*, 1997). For each node encountered, a record of the trip time and node ID is made. The process naturally continues until the GA-agent executes its last gene, at which point it becomes a backward agent, returning to its original source node. In the special case of a GA-agent attempting to return down the same link as it entered a node, the router randomly selects the next hop from the available links, and changes the gene to the new value (deterministic mutation). If no next hop is available, then the chromosome is truncated, and the GA-agent becomes a backward agent (see the algorithm "processing agents"). Note, unlike the AntNet algorithm, modification of routing tables only takes place once the GA-agents have returned to their original source, and modifications only affect the source node routing table. The above representation supports single point crossover, resulting in variable length individuals. Mutation randomly selects a gene and adds/ subtracts an integer such that the new gene is still in the interval [0, L].

Agent ID	Agent Fitness	Trip Time (ms) and node ID
95	0.32	(3, *J*), (9, *C*), (21, *W*)
234	0.39	(1, *B*), (7, *A*), …, (432, *Y*)
…	…	…
31	0.71	(5, *C*), (9, *K*), …, (871, *X*)

Table 4: GA-agent Routing Table

At initialization, a router sends out half of the population of GA-agents to explore the network. Whenever the number of returned GA-agents reaches four, the fitness of the four agents is evaluated.

The fitness function is defined as the normalized node popularity:

$$\frac{\sum_{for\_each\_\exp lored\_node\_i} NP_k(i) * trip\_time_i}{\sum_{for\_each\_\exp lored\_node\_i} trip\_time_i}$$

where the node popularity $NP_k(i)$ is defined as:

$$NP_k(i) = Dest(i) \,/\, TD_k$$

where TD_k is the total number of data packets passing through node '*k*'; and *Dest*(*i*) is the number of data packets with destination '*i*'.

The chromosome fitness measures the popularity of nodes visited as well as the time taken to reach nodes encountered by GA-agents, both of these properties are measured with respect to the original source node.

Node popularity $NP_k(i)$ is a dynamic property, measured at the original source node k by recording the frequency of different data packet destinations as seen by the source node k over a fixed time window (the time window is set 50 seconds in this work). It reflects the most recent trend of the desired destinations. Trip time to each of the explored node is a dynamic property too; the returned agents bring back the most recent knowledge of the network condition.

Chromosomes, which find shortest paths to frequently used destinations, are therefore favored. The best two agents are then chosen – as in a steady state tournament, as parents for routing table update and population evolution (See Figure 9).

The routing table (Table 4) in the GA approach consists of a list of returned agents, every entry corresponds to an evaluated returned agent. On routing a data packet (see Figure 10), the router checks the routing table for the agent that had experienced shortest trip time to the desired destination (third column of Table 4); if such an entry is not found, the entry with the highest fitness (second column of Table 4) will be selected as the default route for this data packet. The neighboring node, which corresponds to the first gene of the selected route/agent, is the next hop for this data packet.

The above constitutes our basic GA-agent approach. In addition, three further concepts are introduced. The first is that of demes (Nowostawski, 1999). This provides a mechanism for passing useful chromosomes between neighboring nodes, thus the nodes share their knowledge of the network. To do so, every node will propagate best-case chromosomes to neighboring nodes every 500 or 700ms (tuneable parameter, see “propagate freq” in section 5.3.2). Secondly, in order to avoid stagnation in the routing tables an aging mechanism is introduced. This takes the form of an incremental penalty applied to each entry of the routing table, Figure 9. The motivation for such an aging mechanism is to ensure that routing tables remain sensitive to the dynamic nature of the environment (e.g., changes to network topology, network node/link failure, network congestion). Such a mechanism is introduced during updates to routing tables: making each routing table subject to decaying fitness and an increasing trip time, figure 6.

```
for each agent in routing table
do      fitness = original_fitness × $c_2$;
        for each node in the entry
        do      trip_time = original_trip_time / $c_2$;
        end for
end for
where $c_2$ is a constant $\in (0.0, 1.0)$
```

Figure 6: GA agent aging mechanism

Finally, when initializing the populations of chromosomes at each node, a node with a higher connectivity degree naturally represents a larger search problem. Thus, the number of chromosomes of a node is initialized in proportion to the square of the number of neighbors (more discussion in section 6.3).

The algorithm is outlined as follows from Figure 7 to Figure 10: (c_1, c_2, and c_3 are constants.)

```
Init
    initialize first generation of agents;
    #agents = #links$^2$ × $c_1$;
    string of offsets of an agent is a string of non-negative integers, e.g., {3, 1, 5, 2, …}
    clear routing table;
    clear flow pattern statistics;
    send out half population of individuals/chromosomes;
```

Figure 7: GA Initialization

```
Processing agents
    if it's a backward agent
    then    if it arrives the source
            then    if it timeouts
                    then    discard it;
                    else    put it into "back" list;
                    end if
            else    if the next hop is down
                    then    discard it;
                    else    forward it to the link;
                    end if
            end if
      else  agent records the trip time info;
            take out an offset from the appropriate position;
            if the corresponding link is available and no loop (section 1.2) caused
            then    send the agent to the link;
            else    randomly (each available link has equal probability) select an
            available link and causing no loop;
```

```
                end if                                                    (Con't)
                if no such link found
                then    convert the agent into a backward agent;
                else    set the offset to the new value;
                        send agent to the corresponding outgoing buffer;
                end if
        end if
```

Figure 8: Processing GA-agents

Updating routing table & population (once 4 agents are back**)**

```
    update the performance table by aging mechanism:
    for each agent in routing table
    do          fitness = original_fitness × $c_2$;
                for each node in the entry
                do      trip_time = original_trip_time / $c_2$;
                end for
    end for
    use the fitness function to evaluate the fitnesses of back agents;
    select the best two agents as parents;
    put/update the fitness's of the parent agents in the routing table;
    delete the entries of the worst two agents in the routing table;
    use standard crossover and mutation on the parents to generate two children;
    put the children into the population;
    delete the worst two agents from the population;
    if current time > last clear time + $c_3$
    then        clear flow statistics;
    end if
    randomly launch 4 agents from the population to explore the network;          (Con't)
```

Figure 9: Updating routing table and population

```
Routing data packets
    if routing table is empty
    then    randomly choose a link to forward;
    else    search routing table for the shortest trip time to desired destination;
            if no entry found ever explored the desired destination
            then    choose agent with best performance;
            end if
    end if
    if no route is found
    then    discard the packet;
    else    forward packet to the neighbor that corresponds to the first gene of selected agent;
    end if
```

Figure 10: Routing data packets

4.3 Data Structures

Every agent consists of a string of next hop offsets, and time stamp records. Every router (Figure 11) consists of an incoming buffer, an outgoing buffer for each neighboring router, a processing buffer (stores a packet at a time), and memory space for routing table. For the GA approach, every router has a population of chromosomes, a routing table, a flow pattern statistics table, and a fitness table. The number of chromosomes per population is in direct proportion to the square of number of neighbors. The routing table, which is updated whenever four chromosomes return, consists of current fittest individuals. The flow pattern estimates the popularity of data packets passing through the node. The fitness table stores the fitness of every chromosome, currently a member of the routing table.

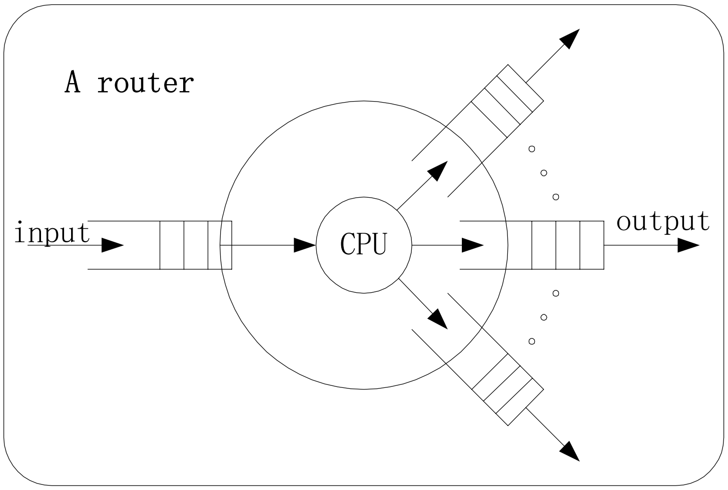

Figure 11: Schematic representation of a network node (router)

5 Experiments

5.1 Settings

The discrete event driven simulation models the network as routers (nodes) and links. The simulation in this work can be considered on the “network” layer of OSI model, where only the routing issues are addressed. The memory space is assumed always large enough for the buffers and routing table of the router (Figure 11). A priority queue is used to store the events. Both AntNet (local model and global model) and GA-agent algorithms are simulated under the same environmental conditions. That is, an event generator is used to generate the events, such as a new packet time of generation, or router’s availability. The following are the parameters used in the simulations:

- Network topology takes the form of the Japanese backbone, figure 4;
- Forward ants are launched every 300ms;
- The AntNet and GA algorithms are given 5 seconds at the beginning of the simulation to converge the initial routing tables, during this period, routing packets (ants and GA-Agents) are the only packets traversing the network;
- Data packets are generated by Poisson distribution (mean of 35ms);
- The seven parameters for the GA based scheme are given in Table 5;
- Any packets, including data packets, are killed should a loop be detected. Given the probabilistic nature of the routing tables this represents a rather harsh constraint, but it is utilized to emphasize the properties of different routing strategies.

5.2 Baseline – Static Shortest Path algorithm

To establish a baseline for AntNet and GA-agent algorithms, we implemented the Static Shortest Path (SPS) algorithm for this routing problem. The SPS pre-calculates the all-pairs shortest paths, can be regarded as the converged routing tables on each router using RIP. Indeed, SPS is different from RIP critically in that the latter takes the number of hops to judge the routes, the former takes the transmission time, which is more preferable, because some links are much slower than others, which should not be equally considered as fast links. In order to simplify the problem, we use the Floyd-Warshall algorithm (Cormen *et. al*., 1989) to construct all the shortest path pairs, Figure 12.

```
Floyd-Warshall (network $W$)
  n = rows($W$);
  distance array $D = W$; (n × n, filled with element ∞)
  next hop array $Next$; (n × n, filled with element ∞)
  for i = 1 to n
    for j = 1 to n
      if i and j are adjacent
        then      $d_{ij}$ = cost(i, j);
                  $next_{ij} = j$;
      end if
    end for
  end for
  for k = 1 to n
    for i = 1 to n
      for j = 1 to n
        if ($d_{ij} > d_{ik} + d_{kj}$)
        then      $d_{ij} = d_{ik} + d_{kj}$;
                  $next_{ij} = next_{ik}$;
        end if
  return $D$ & $Next$;
```

Figure 12: Floyd-Warshall algorithm

Each simulation is run for 1250s. As a result, 1985536 data packets are generated within 1250s. The queue length is the total number of waiting packets, which includes the data packets and the routing packets. In this paper, the routing packets refer to the ants in the AntNet algorithm, and to the GA-agents in the GA approach.

In this work, we will compare three algorithms (see Figure 13): Static Shortest Path algorithm (SPS), AntNet (with global information – GlobalAnt, with local information – LocalAnt), and GA (with 100% crossover & 100% mutation – certainGA, with classical probabilistic 90% crossover & 10% mutation – probGA).

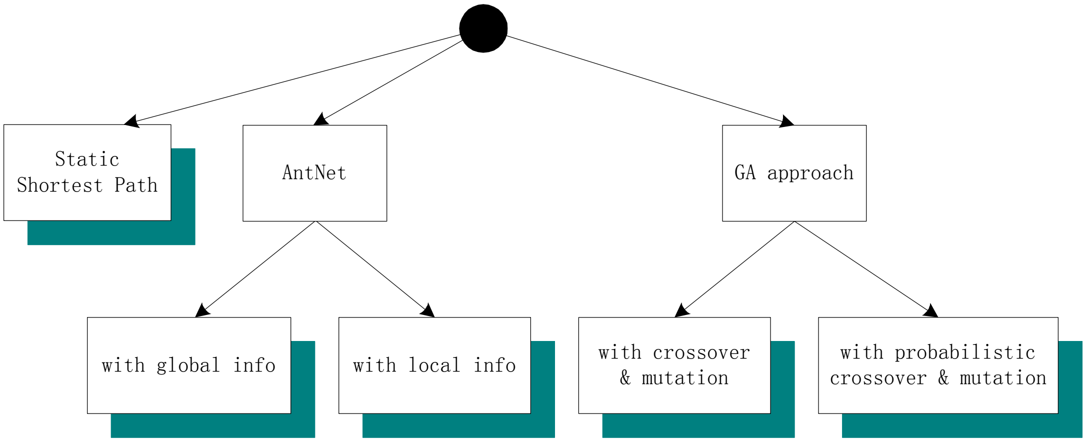

Figure 13: Algorithms for comparison

5.3 Experimental Results

5.3.1 AntNet Parameters

The following is the list of parameters of AntNet algorithms being used:

$\alpha = 0.3;$

$c_1 = 0.7;$

$c_2 = 0.3;$

$\eta = 0.005;$

$\gamma = 0.654;$

where these values follow the recommendation of (Di Caro, Dorigo, 1998).

5.3.2 GA Approach Parameters

In the case of routing using GA-agents, there are five basic parameters,

- Rates of crossover and mutation;
- # Agents / link^2 – a constant c_1, which determines the population of chromosomes on every node;
- Aging – a constant $c_2 \in (0.0, 1.0)$, rate by which fitness of individuals currently populating the routing tables decay;
- Propagate ratio – the number of chromosomes exchanged between populations, expressed as a % node population size;
- Propagate freq – constant rate/frequency of exchange of chromosomes between populations;
- Flow clear freq – a constant c_3, time interval over which data packet destination statistics are collected.

Eight different combinations of the above parameters are considered, these are initially selected to enable qualification of the sensitivity to population size, rate of aging etc. and remain the same across all experiments. Table 5 summarizes these parameters:

Combinations *Parameters*	No.1	No.2	No.3	No.4	No.5	No.6	No.7	No.8
Probability$_{crossover}$	1.0				0.9			
Probability$_{mutation}$	1.0				0.1			
# agents / link^2	32	32	40	48	32	32	40	48
Aging rate	0.8	0.9	0.9	0.9	0.8	0.9	0.9	0.9
Propagate ratio (%)	5	3	3	2	5	3	3	2

Propagate freq (ms)	500	500	700	700	500	500	700	700
Flow clear freq (sec.)	50							

Table 5: combinations of GA parameters

Simulations of the eight combinations are conducted (see Table 5) on five network conditions (Table 6 - 10). Combinations involving minimal agent count, propagation frequency and ratio with highest aging rate (combination No.2 and No.6, table 5) appeared to provide the most reliable performance independent of scenario (Appendix B). These two cases are therefore detailed in the following results.

5.3.3 Performance Measurements

A total of 5 simulation scenarios are considered for the SPS, AntNet and GA approaches, all of which utilize the Japanese backbone network topology, figure 4. In the first case, all routers remain available, Table 6. The remaining experiments investigate plasticity of the network by removing different router combinations, Tables 7 - 10. First, router R34 is removed at a time step of 500s, Table 7. From figure 4, it is apparent that router R34 represents a significant node in the topology, although alternative paths certainly exist. In Table 8, two routers (R49, R13) are removed, whereas in Table 9 the same two routers (R49, R13) are first put *down* asynchronously, but put *up* later. Finally, in Table 10, three routers (R42, R19, R6) experience failure at time 500s.

On measuring the performance of routing algorithms, we focus on the following metrics:

- Network throughput, which is defined as number of data packet bytes successfully received at their destination in a two second window;
- Network Queue size, which is defined as the number of data packets and routing packets in the incoming buffers and outgoing buffers on

all the routers. So in the case of SPS, there is no routing packets; in the case of AntNet algorithms, routing packets refer to the ants; and in the case of GA approaches, routing packets refer to the GA agents;

- Total time to deliver all the data packets (finish time);
- Number of arrived data packets (short as AP);
- Average trip time of arrived data packets;
- Number of routing packets generated during simulation;
- Average chromosome length on each node (Appendix A);
- Chromosome fitness on each node (Appendix A);
- Number of arrived data packets on each node (Appendix A).

In this work, packet loss happens in these conditions: immediately loss due to network failures, e.g., packets which are existing in a failed node, or existing in the outgoing buffer to a failed node, or being transmitted via links to a failed node; discarded by a node if the packet enters a loop.

5.3.4 Results

5.3.4.1 No Network Failure

No network failure					
Algorithm	SPS	GlobalAnt	LocalAnt	CertainGA	ProbGA
Finish time (ms)	1318526	1252740	1267002	1252815	1252000
# routing packets		198346	218539	1027884	960601
Arrived packets (AP)	1985536	1979268	902884	1585185	1692813
Lost packets	0	6268	1082652	400351	292723
Average trip time of AP (ms)	1387	566	398	905	1171

Table 6: All network devices remain available

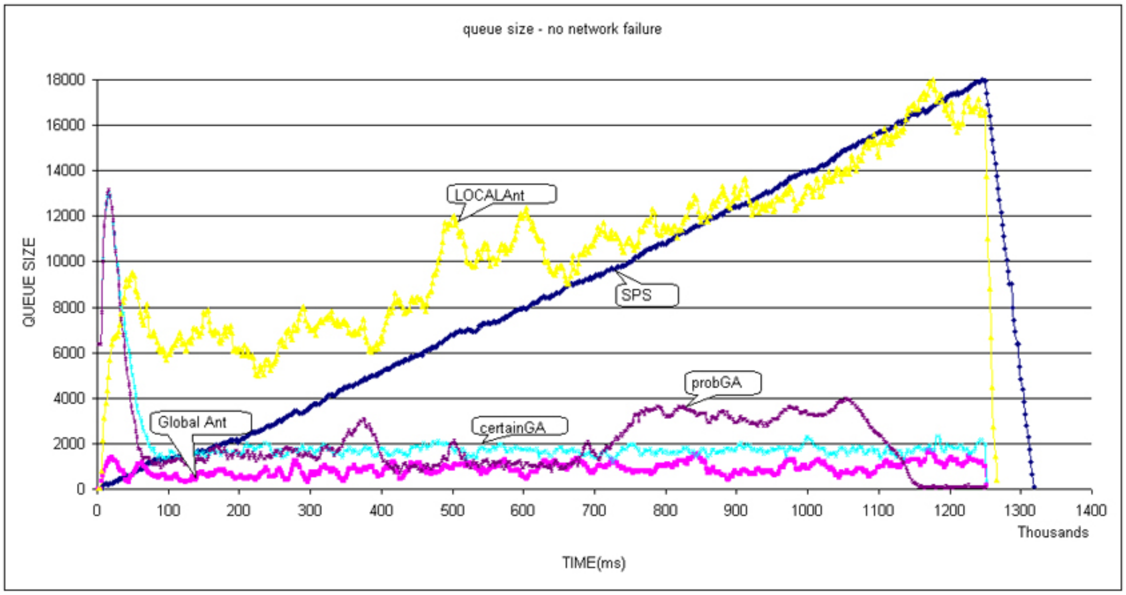

Figure 14: queue size - all network devices remain available

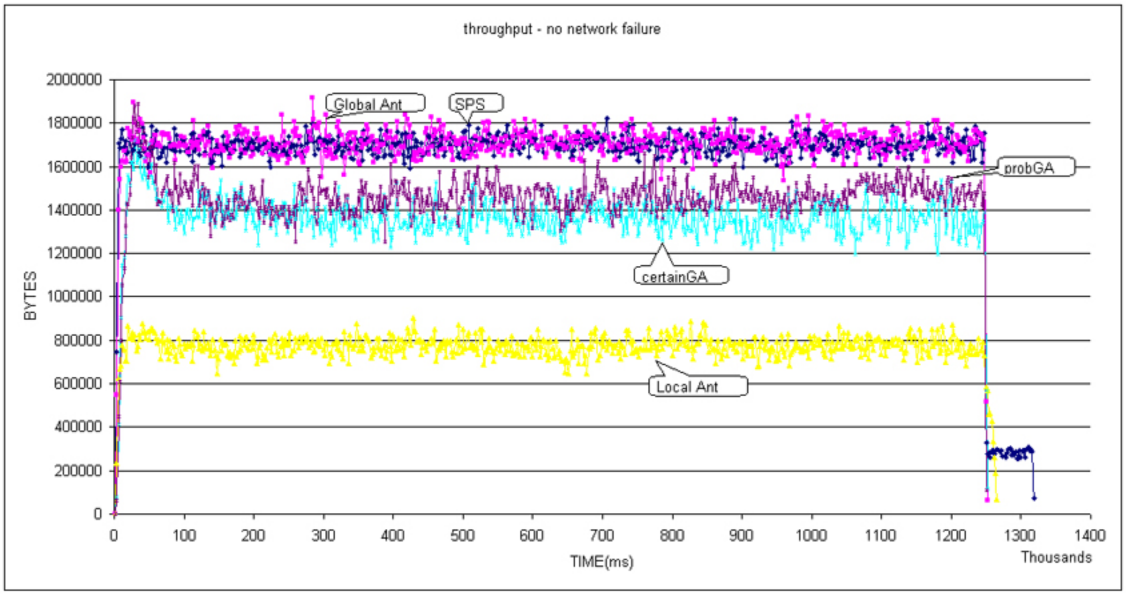

Figure 15: throughput performance - all network devices remain available

From Figure 14, Queue Length, the following observations are made. SPS observes a linear increase in queue length, indicating that the algorithm is unable to control this parameter, and suggesting a reliance on routing buffers of sufficient size to absorb this property. From Table 6, it is evident that no packets are lost (no loops) – unlike the other algorithms. Throughput, figure 15, is high relative to that achieved using the other algorithms.

The LocalAnt algorithm is not able to improve on the queue length performance of SPS, figure 14. In addition looses more packets than it successfully routes, table 6 (packets with loops now exist on account of poor routing and probabilistic nature of the routing tables) and returns the lowest throughput rates.

After an initial configuration period (typically 100 seconds for the GA schemes) the remaining algorithms control queue length effectively, figure 14. The global Ant algorithm "looses" the least packets (0.3% as opposed to 20% and 15% respectively for certainGA and probGA respectively), table 6, and maintains the highest levels of throughput. Figure 15. However, the classical selection of crossover and mutation rates significantly benefits probGA.

In short, the sequence of comprehensive performance is: GlobalAnt – SPS – probGA – certainGA – LocalAnt.

5.3.4.2 R34 Down at 500s

R34 down at 500s					
Algorithm	SPS	GlobalAnt	LocalAnt	CertainGA	ProbGA
Finish time (ms)	1250245	1668469	1369366	1306682	1506900
# routing packets		199075	218823	1086689	1170219
Arrived packets (AP)	1464815	1833184	813913	1298426	1400861
Lost packets	520721	152352	1171623	687110	584675
Average trip time of AP (ms)	1956	998	2899	2613	356

Table 7: Router R34 fails to work at time 500s

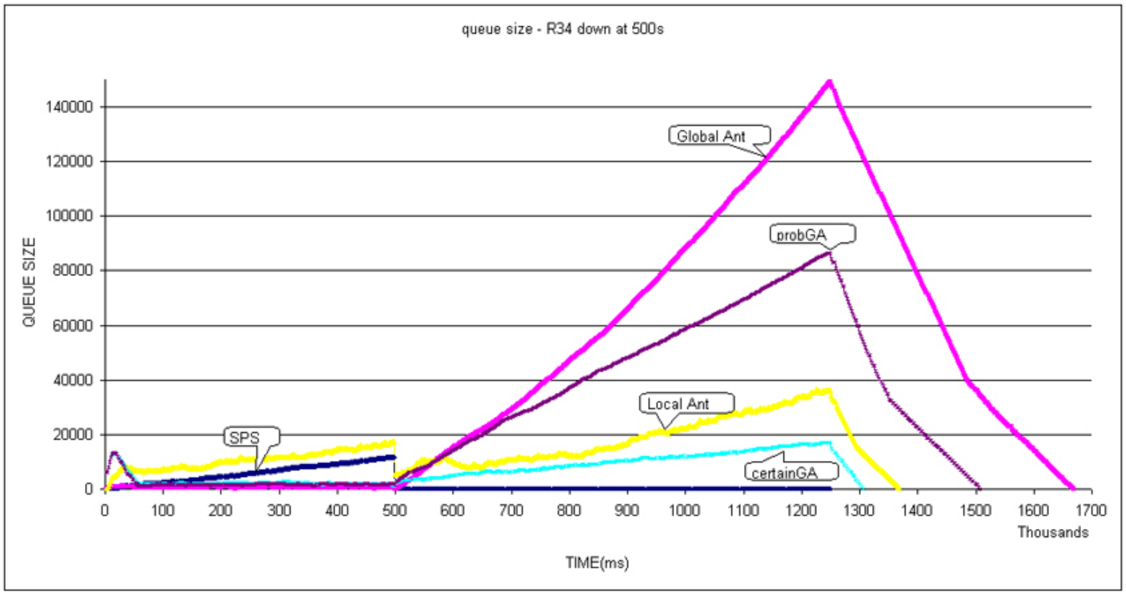

Figure 16: queue size - Router R34 fails to work at time 500s

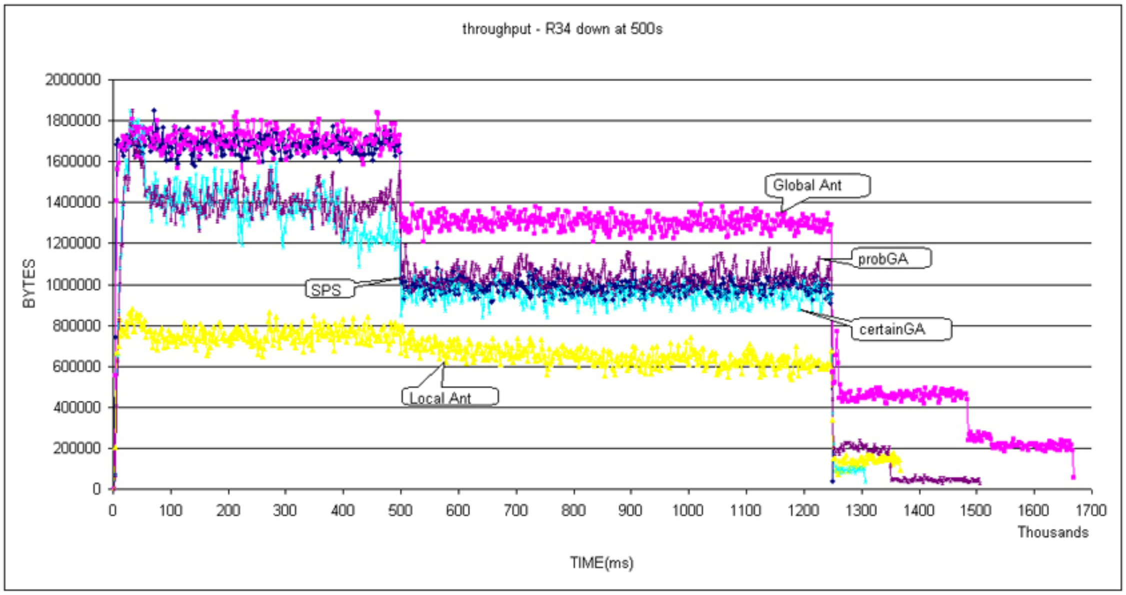

Figure 17: throughput performance - Router R34 fails to work at time 500s

In this scenario node 34 is removed at time step 500, where node 34 represents a critical node for connectivity, but bypass routes do exist, figure 4. The SPS algorithm is naturally not able to adapt to the change in configuration. As such SPS throughput changes from joint best to third, figure 17; SPS queue length gets very small (much smaller than 20000) as soon as node R34 is removed, reflects that in the static SPS routing tables of the nodes direct most data packets through R34, once again shows the importance of R34 in the NTTNet.

The LocalAnt algorithm continues to lose more packets than it delivers (implying that more packets enter a loop than find a direct path) and in addition returns the longest trip time for those packets that are delivered, table 7. Possibly on account of the reduction in the number of packets delivered, the queue length profile is now better than GlobalAnt, figure 16, whereas throughput is still the worst, figure 17. The linear increase in global Ant queue length appears to indicate that only small changes to the routing strategy have been made to accommodate the new network condition. That is to say, the algorithm – under the current parameterization – is making use of unconstrained queue lengths to soak up the reduced connectivity.

In the case of GA-agents, a clear preference for probGA as opposed to certainGA exists, hence comments are made for probGA alone. ProbGA returns the fastest average trip time for delivered packets and almost betters SPS in terms of ‘lost’ packets, table 7. Queue length, figure 16, observes a continued linear increase once the fault is introduced, indicating that a suitable alternative routing strategy has not been identified. Both local Ant and certainGA have minimal queue length profiles on account of the high packet losses, table 7.

In summary, the distinction between GlobalAnt and probGA is less clear. All the remaining algorithms however perform significantly worse.

5.3.4.3 R13 and R49 Down at 500s

R49 & R13 down at 500s					
Algorithm	SPS	GlobalAnt	LocalAnt	CertainGA	ProbGA
Finish time (ms)	1250239	1465837	1300468	1444549	1252000
# routing packets		198030	218713	972622	1025141
Arrived packets (AP)	1370605	1871469	827125	1369264	1417205
Lost packets	614931	114067	1158411	616272	568331
Average trip time of AP (ms)	2161	1325	1617	1301	861

Table 8: Routers R49 and R13 fail at time 500s

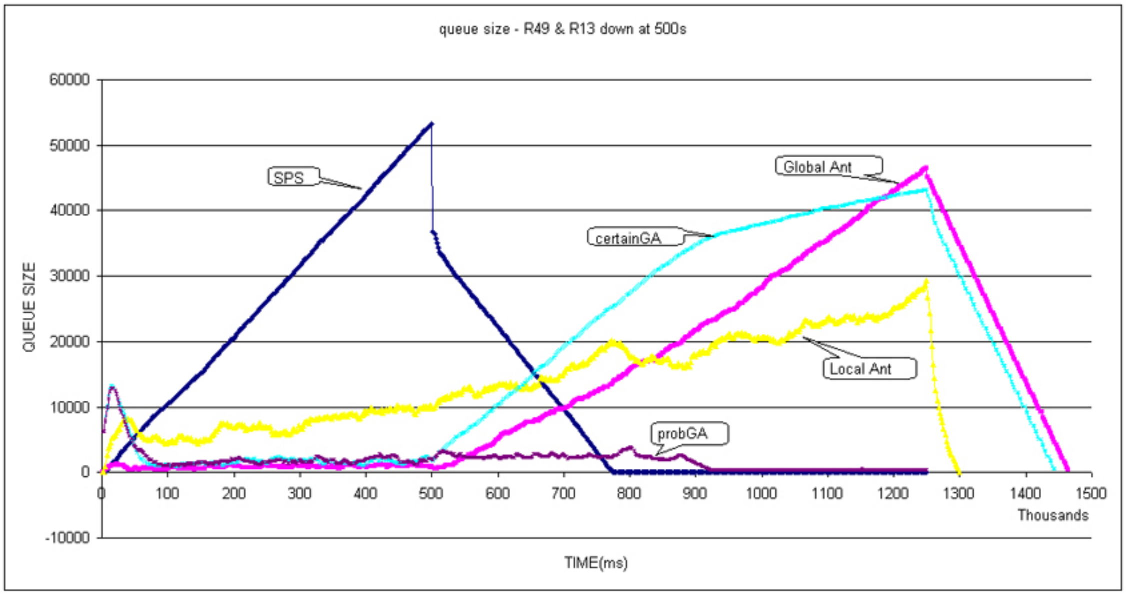

Figure 18: queue size - Routers R49 and R13 fail at time 500s

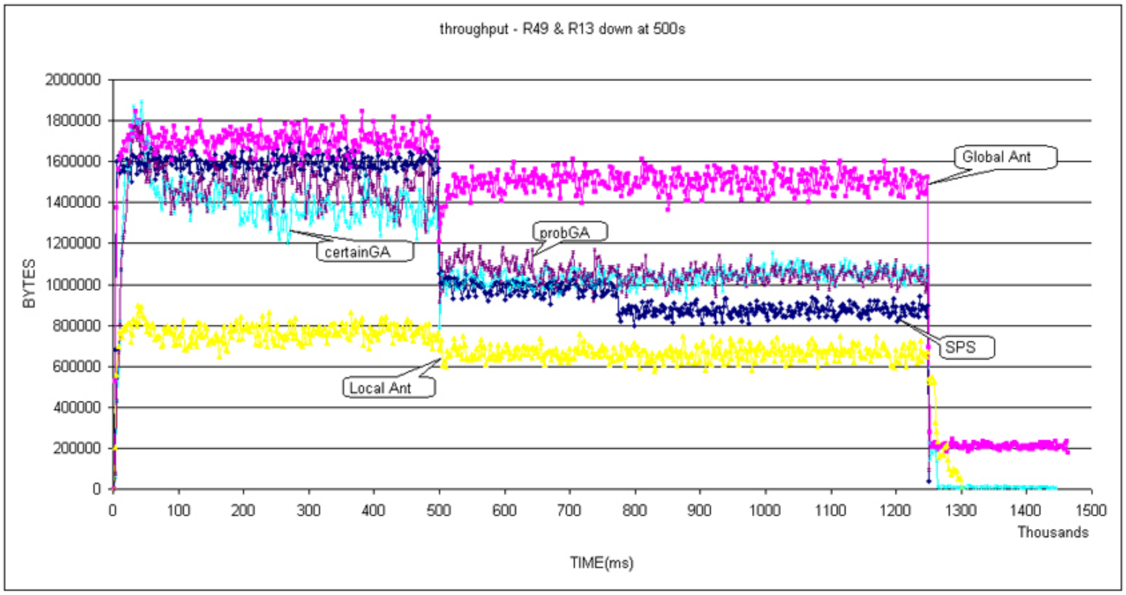

Figure 19: performance - Routers R49 and R13 fail at time 500s

In this experiment Routers R13 and R49 were put down at 500s. Routers R13 and R49 are not as important as R34 (R13 has connectivity degree 4, R49 has connectivity degree 3, whereas R34 has connectivity degree 5.). However, we wanted to know how the algorithms would behave under multiple network failure conditions.

SPS is again naturally unable to reconfigure following the introduction of faults. Moreover, the number and position of faults results in the highest number of lost packets, table 8 (in the case of SPS due to the retention of routes which lead to nodes which no longer function). Furthermore, average trip time of the delivered packets is now the worst of the five algorithms. LocalAnt is still losing far more packets than it is delivering, table 8, which naturally results in low throughput and queue length profiles, figures 18 and 19. The GlobalAnt algorithm still looses the least number of packets, table 8. The probGA algorithm provides the best queue profile, figure 18, whilst returning the second best lost packet count and best case average trip time, table 8. In effect queue length minimization appears to be prioritized more in the GA scheme.

Thus the sequence of comprehensive performance is: GlobalAnt – probGA – certainGA – SPS – LocalAnt.

5.3.4.4 R13 Down at 300s, R49 Down at 500s, Both Up at 800s

R13 down at 300s, R49 down at 500s, both up at 800s					
Algorithm	SPS	GlobalAnt	LocalAnt	CertainGA	ProbGA
Finish time (ms)	1284363	1252262	1288776	1261363	1252000
# routing packets		198742	217659	1043279	1082799
Arrived packets (AP)	1740080	1915162	865033	1334514	1554835
Lost packets	245456	70374	1120503	651022	430701
Average trip time of AP (ms)	1503	677	3259	1202	1012

Table 9: Router R13 fails at 300s, R49 fails at 500s, both up at 800s.

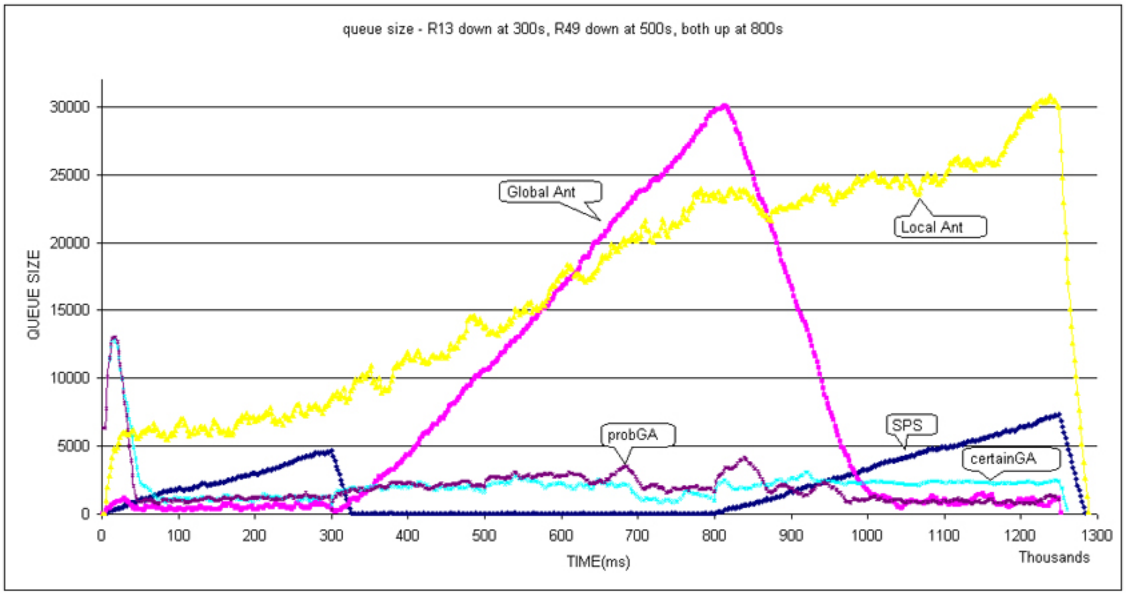

Figure 20: queue size - Router R13 fails at 300s, R49 fails at 500s, both up at 800s.

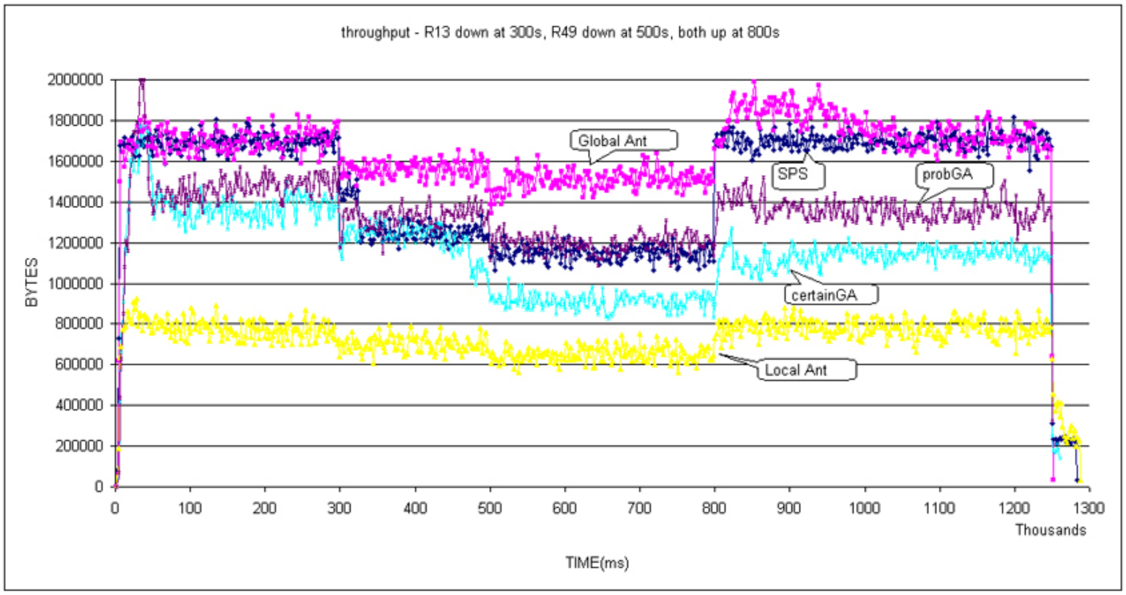

Figure 21: throughput performance - Router R13 fails at 300s, R49 fails at 500s, both up at 800s.

This scenario is to test the ability of the algorithms to adapt themselves to the more drastically dynamic network situations. In this case, routers experience failures, and are recovered.

SPS again establishes the baseline performance level. In contrasting Ant verses GA-agent algorithms significant differences are evident. Both the Ant type algorithms make use of queues, figure 20, possibly implying the utilization of a small number of preferred routes. GA-agent based strategies appear to minimize queue lengths at the expense of higher lost (looping) packet counts, table 9. Throughput profiles follow the same general pattern as previously encountered – GlobalAnt consistently has the highest throughput, with SPS dropping to the same level as probGA under fault conditions and the remaining algorithms returning significantly lower throughputs – figure 21.

From Table 9, we can see that GlobalAnt is the best among the five algorithms, LocalAnt is the worst, and the two GA approaches are better than LocalAnt.

5.3.4.5 R42, R19 and R6 Down at 500s

R42, R19 & R6 down at 500s					
Algorithm	SPS	GlobalAnt	LocalAnt	CertainGA	ProbGA
Finish time (ms)	1297102	1250849	1283775	1253750	1252504
# routing packets		198360	219116	1122024	954765
Arrived packets (AP)	1645627	1845743	816190	1410263	1413077
Lost packets	339909	139793	1169346	575273	572459
Average trip time of AP (ms)	1931	381	375	1065	2085

Table 10: Routers R42, R19 & R6 fail at 500s

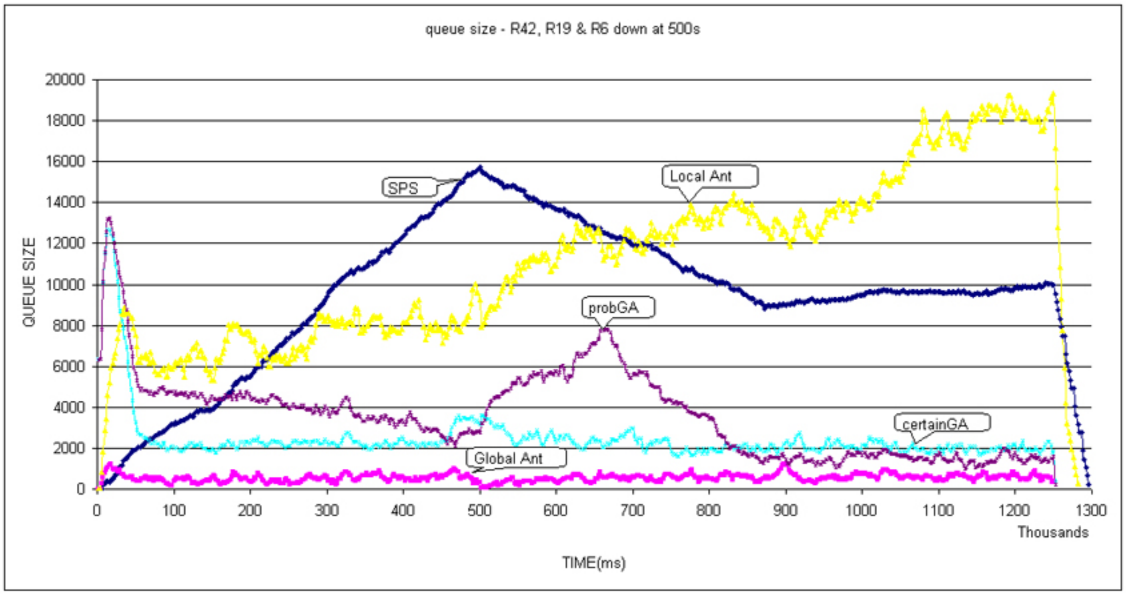

Figure 22: queue size - Routers R42, R19 & R6 fail at 500s.

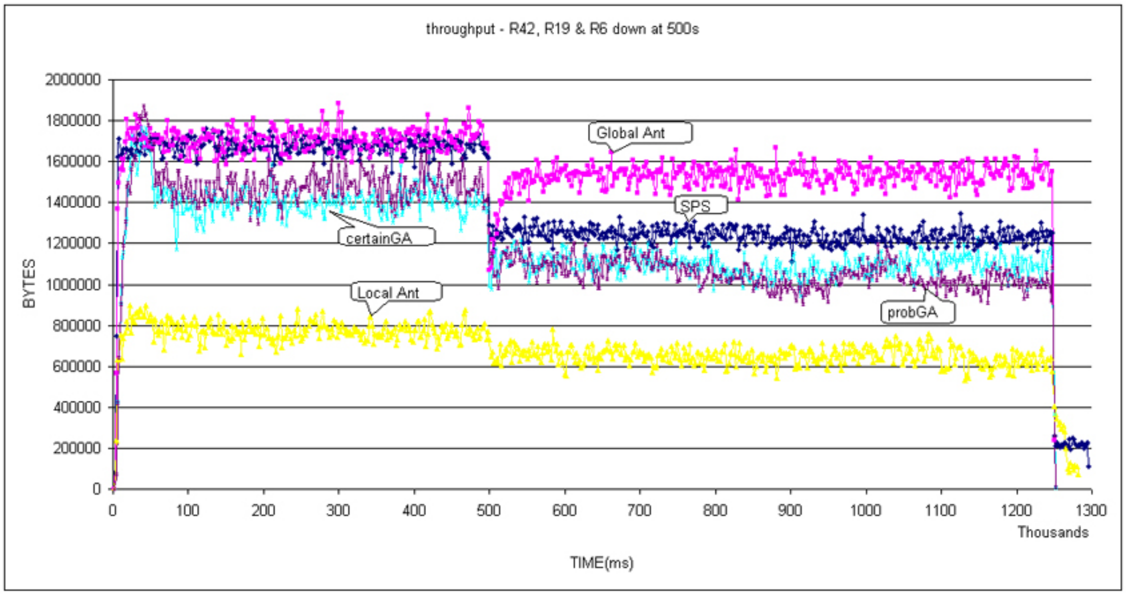

Figure 23: throughput performance - Routers R42, R19 & R6 fail at 500s.

In this scenario, the failures of three routers (R42, R19, & R6) happen at the same time (500s) in three different areas of NTTNet. Moreover, the failures occur at nodes associated with the boarder of the topology.

As in previous fault conditions, the GlobalAnt algorithm returns the lowest 'lost' packet count, but queue length, figure 22, no longer observes the linearly increasing characteristic associated with each of the previous fault scenarios. Moreover, the queue length and throughput profiles are very similar to those in the no failure scenario. In effect the GlobalAnt algorithm is able to maximize its global information benefits to solve the routing problem with little modification to the pre-fault strategy.

On the other hand, probGA queue lengths increase following the disturbance; until a new strategy is identified around the 675th second and the original queue profiles are achieved around the 840th second, figure 22. It is interesting to note that the GA-agent scheme with 100%

mutation provides a better solution than probGA in this specific scenario, table 10.

By way of an overall ranking, it is clear that the utilization of global information in the ant algorithm plays a central role in its performance. Without this – LocalAnt – more packets are lost (loops are identified) than delivered, irrespective of whether there are missing links or not. GA-agents clearly perform far better than LocalAnt, typically delivering twice as many packets, irrespective of the scenario. It is also apparent, however, that the different cost functions controlling AntNet and GA-agents may also be playing a role in determining the characteristics of the approaches. The AntNet algorithm explicitly incorporates temporal and queue lengths, with a pre-selected parameter, α, determining the relative weighting given to each. Di Caro and Dorigo recommend a value in the interval 0.2 to 0.5, where a value of 0.3 was used in this work. The fitness function of the distributed GA is much more straightforward, with no direct representation of queue lengths. Future work will investigate the utility of more complex cost functions.

6 Conclusions and Future Work

6.1 Criteria

On comparing the routing algorithms, we should not focus on a separate measurement index, but consider them comprehensively. An ideal algorithm would be able to deliver more data packets (number of arrived packets) irrespective of the network scenario, send the packets to their destinations using shorter trip times (average trip time, finish time, and throughput), while the queue size is minimized. In order to achieve these goals, the routing algorithm must be capable of finding the appropriate routes, recognizing dynamic changes to network traffic and topology, adapt the routers to the new conditions, route the data packets efficiently while distributing the work load among the network. Therefore, we believe that network resources must work as a cooperative team. In addition, it is important to include system/network overheads, such as buffer occupations, CPU usage, or network resources needed to support the algorithm.

6.2 Conclusions

The performance (arrived packets, network queue size, network throughput, etc.) reflects the routing ability and adaptability of the algorithms. AntNet was proven to outperforms OSPF (Di Caro *et al.*, 1998). We first looked at the comparison of the following pairs.

6.2.1 Static vs. Adaptive

As we discussed in section 2, RIP is a form of static routing algorithm, thus the routing tables of the routers will not change after the convergence, as long as the network topology is not changed. The difference between RIP and SPS is that RIP makes use of hop counts in measuring distance/cost between two routers, while SPS makes use of the trip time, which is pre-calculated based on the assumption of global information.

The first experimental scenario (Section 5.3.4.1, Table 6, Figure 14 and Figure 15) represents a static network topology without network failure. Even in this static situation, SPS does not give perfect queue size performance, the queue size keeps growing linearly, which is effectively indicating that the algorithm is not able to control traffic load. This indicates the necessity of adaptive routing algorithms, which can make use of additional paths as network load varies.

However, the other four adaptive routing algorithms, except Local AntNet, did demonstrate (section 5.3.4) abilities to adaptively route data packets with load balancing. Their principal weakness is the loss of some data packets. For the AntNet algorithm, its probabilistic character causes the loops (section 1.2). For the GA approaches, the possibility of loop exists in the situation that no GA-agent ever explored the route to the desired destination, then, by following the default gateway may lead the data packet into a loop or, wrong direction. We could expect the elimination of such possibility for GA in the future by improving the routers' sense of direction.

6.2.2 Global AntNet vs. Local AntNet

The most significant difference between these two algorithms is in their routing tables (Table 2, Table 3), i.e., global information assumption in the Global AntNet is removed in the Local AntNet.

The experimental program (section 5) demonstrated the importance of the global information assumption:

- The LocalAnt loses many more data packets than the GlobalAnt;
- The GlobalAnt has the highest throughput, and the LocalAnt has the lowest throughput under the five network scenarios;
- The queue size graphs (section 5.3.4) imply that the LocalAnt can not route the data packets correctly, because in the normal situation and

light or medium network failure situations, its queue size keeps increasing linearly in trend and arrived data packets are less then loss packets, and in the serious network failure conditions, low queue size and small amount of arrived data packets imply that a lot of packets are identified as looping very soon after they enter the network.

- The global AntNet shows more capable to recognize the network failure – least amount of lost data packets in all experiments. But only in the light failure situation (section 5.3.4.5), global AntNet shows ability in adapting nodes to new efficient routing tables. The large queue size in the other three failure (medium to severe level) experiments (section 5.3.4.2 – 5.3.4.4) implies that it's not highly adaptive.

In real world situations, however, it is impossible for the routers to acquire the global information because each router knows only the existence of its neighbor routers, by means of the physical and data link layers in the OSI model. However, the removal of global information was shown to degrade the performance of AntNet algorithm.

6.2.3 Local AntNet vs. GAs

The Local AntNet algorithm follows the Global AntNet algorithm in every way other than the global information assumption. The distributed Genetic Algorithm (D-GA) approach proposed in this work makes use of only the local information. Their performance is significantly different, both D-GAs are much better than Local AntNet in that:

- They successfully deliver much more data packets in shorter average trip time (section 5.3.4);
- Their network throughput is much higher than Local AntNet (section 5.3.4);

- Their network queue sizes performance is also better than Local AntNet in the experiments of the five network situation scenarios (section 5.3.4).

The price for higher performance in D-GA is that the greater autonomy represents an overhead to network and router resources. There are about four times more routing packets in D-GAs than in Local AntNet. The statistics in the previous section indicate this does not detract from the quality of service guaranteed, because the queue size (defined as the number of data packets and routing packets) performance of D-GAs is more desirable than that of Local AntNet even when comparing D-GA to the Global AntNet algorithm.

6.2.4 Summary of Contribution

This work presents a D-GA approach to solve the challenging network routing problem subject to the local information constraint. Access to Global information is never the case in the real world; however, this is the first work we are aware of to reach these objectives using a multi-agent framework.

The original (Global) AntNet algorithm (Di Caro, Dorigo, 1998) was proved to outperform some currently used (in practice) routing protocols (such as OSPF). However, if its global information assumption is removed, leading to the so-called Local AntNet algorithm studied as part of this work, we show that performance degrades by about half (section 5.3.4). On the other hand, the proposed D-GA approach yields superior performance in many aspects, such as number of arrived data packets, network throughput, average trip time of arrived data packets, and, it appears to minimize network queue size at the expense of higher packet loss (section 5.3.4). Moreover, the second assumption of AntNet algorithm (network connectivity assumption, section 3.3.1) is not used in the D-GA algorithm.

The advantages of the D-GA approach over the existing routing protocols are that GA-agents are completely distributed, thus routing is not dependent on the designated central routers; and are purely automatic, i.e., no human configuration is necessary, which avoids the mis-configuration or non-efficient configuration problems caused by insufficient human knowledge of the network.

Indeed, security is a problem in mobile agent methods as a whole (Minar *et. al.*, 1999). In short, the task is to protect hosts from agents, protect agents from hosts, and protect agents from each other. Since the agents in the D-GA approaches are not executable binary codes, the major problem is almost avoided. Moreover, an agent carries a string of genes, together with the time stamps of every visited router; it is not likely to be harmed by other agents. The principle problem of the method, however, lies in routers permitting agents to update routing table information. This problem is true for both AntNet and the D-GA solution detailed here.

6.3 Future Work

The GA approach has given us some encouraging results, yet there are still many unanswered questions. For example, fitness function plays a very important role in GA, the current factors taken into account by the fitness function are trip times and destination statistics. A more comprehensive fitness function has the potential to incorporate a wider range of quality of service objectives.

Another example would be the population of agents on each router. At the very beginning of the D-GA approach, every router has same number of agents, that is, no matter how important a router is in the network, it still has the same number of agents as other routers. The routing ability turned out pretty low, just at the comparable level of Local AntNet. We found that the important routers did not sufficiently evolve. A direct proportional relation was then used (section 5.3.2), performance improved, but critical routers still did not evolve in line with their importance to the network. The quadratic relation, which is currently being used in D-GA, finally lets the routers make corresponding contribution. The question is how to adjust the population of agents to a router? What other factors should be included?

The size of the routing table is always an important factor, because it has direct influence on the efficiency of a router. Nowadays a core router (e.g., a ASBR) in the Internet may have one hundred twenty thousand or even more routes in its routing table (Huston, 2001). And according to http://bgp.potaroo.net/, the most recent size is over one hundred thirty thousand. This is a huge routing table. We now look at the size of the routing table and the next hop look up time of the algorithms, for a router having *l* neighbors in a network having *n* routers.

- A SPS router will have $(n - 1)$ records, each record has two fields: destination and the next-hop router, so the size of the routing table is $\Theta(n)$; Sequential search of the routing table will take $\Theta(n)$ time.
- A GlobalAnt router has *l* records, each has $(n - 1)$ fields, each for a node in the network, thus, the size of the routing table is $l \times (n - 1)$, i.e., $\Theta(l \times n)$, usually $l << n$, so, the size of the routing table is $\Theta(n)$; Since the routing table is a two-dimensional array, the next hop look up time is only $\Theta(1)$.
- A LocalAnt router has *l* records (number of neighboring links), each has only two fields, one for the neighbor itself, one for the rest of network. Thus, the size of the routing table is $l \times 2$, i.e., $\Theta(l)$; the next hop look up time is also only $\Theta(1)$.
- A D-GA router has a population of $c_1 \times l^2$ chromosomes, thus the routing table has $O(l^2)$ records, and each represents an explored route. According

to the statistics of the experiments, the routes have 2 to 12 genes, approximately, this fits relation Θ(l). Thus the size of routing table is O(l^3). Sequential search of the routing table will take O(l^3) time.

But in our experiments with NTTNet, the routing table of D-GA is quite a lot larger than that of GlobalAnt, because n and c_1 are very close (n = 55, c_1 = 32), which means D-GAs takes more memory space and requires more next hop look up time. Thus, the question rises: how to decrease the size of the routing table and the next hop look up time in D-GA?

One emphasis should be placed on the co-evolution mechanism of the GA approach, which means the algorithm would make the whole network work like a team by keeping sharing their knowledge (topology, congestion, etc.) about the network. The outcome would be that the routers need not know the exact routes to any destination across the whole network, but only to the next local node that is more likely to know the correct route. The current GA approach provided have such a mechanism, or be it indirectly – demes (section 4). We believe the whole performance of the routing ability would be greatly improved if stronger forms of co-evolution were introduced.

Finally, we note that the discrete event simulation utilized here deletes *any* packet that enters a loop, where such a property was introduced to emphasize differences in routing strategy. A more practical approach would be to apply such a constraint to Ant or GA-agent as opposed to data packets. This would have the effect of increasing the network load for the adaptive schemes and therefore increase the average time taken for packet delivery. Moreover, the only lost packets would then be those, which are routed down links subject to failure, or whose TTL expires.

References

Ahuja R.K., Magnanti T.L., Orlin J.B., “Network Flows: Theory, Algorithms and Applications”, Prentice-Hall, 1993.

Cormen T.H., Leiserson C.E., Rivest R.L., “Introduction to Algorithms”, McGraw-Hill, 1989, ISBN 0-07-013143-0.

Corne D.W., Oates M.J., Smith G.D., “Telecommunications Optimization: Heuristic and Adaptive Techniques”, John Wiley & Sons, 2000, ISBN 0-471-98855-3.

Di Caro G., Dorigo M., “AntNet: Distributed Stigmergetic Control for Communications Networks” Journal of Artificial Intelligence Research, 9, pp. 317-365, 1998.

Dorigo M., Maniezzo V., Colorni A., “Ant System: Optimization by a Colony of Cooperating Agents”, IEEE Transactions on Systems, Man and Cybernetics – B, 26(1) pp. 29-41, Feb 1996.

Gelenbe E., Xu Z., Seref E., “Cognitive Packet Networks”, Proceeding of 11th IEEE International Conference on Tools with Artificial Intelligence, pp 47-54, 1999.

Goldberg D.E., “Genetic Algorithm in Search, Optimization and Machine Learning”, Addison-Wesley, 1989.

Heusse M., Snyers D., Guerin S., Kuntz P., “Adaptive Agent-driven Routing and Load Balancing in Communication Networks”, Advances in Complex Systems, 1, pp 237-254, 1998.

Huston G., “Analyzing the Internet's BGP routing table”, Internet Protocol Journal, 4(1), 2001.

Liang S., Zincir-Heywood A.N., Heywood M.I., “The Effect of Routing under Local Information using a Social Insect Metaphor”, IEEE International Congress on Evolutionary Computation, May 2002, (in press).

Liang S., Zincir-Heywood A.N., Heywood M.I., “Intelligent Packets for Dynamic Network Routing Using Evolutionary Strategies”, The Genetic and Evolutionary Computation Conference, July 2002, (in press).

Maniezzo V., Colorni A., “The Ant System Applied to the Quadratic Assignment Problem”, IEEE Transactions on Knowledge and Data Engineering, 11(5), pp. 769-778, Sept/ Oct 1999.

Minar N., Kramer K. H., Maes P., “Cooperating Mobile Agents for Dynamic Network Routing”, Chapter 12, Software Agents for Future Communications Systems, Springer-Verlag, 1999, ISBN 3-540-65578-6.

Mitchell T., “Machine Learning”, Chapter 9, McGraw Hill, 1997, ISBN 0070428077.

Norris M., Pretty S., “Designing the Total Area Network: Intranets, VPN'S and Enterprise Networks Explained”, Chapter 6, John Wiley & Sons, 2000, ISBN: 0-471-85195-7.

Nowostawski M., Poli R., “Dynamic Demes Parallel Genetic Algorithm”, 1999. Proceedings of the Third International Conference on Knowledge Based Intelligent Information Engineering Systems (KES'99), IEEE, pp. 93 - 98.

Perlman. R., "Interconnections: Bridges and Routers", Addison-Wesley, 1992, pp. 210-211.

(RFC 1305) Mills D.L., “Network Time Protocol (Version 3) Specification, Implementation”, 1992

(RFC 1771) Rekhter Y., Li T., “A Border Gateway Protocol 4 (BGP-4)”, 1995.

(RFC 2328, STD 0054) Moy J., “OSPF Version 2”, 1998.

(RFC 2453, STD 0056) Malkin G., “RIP Version 2”, 1998.

Tennenhouse D., Smith J., Sincoskie W., Wetherall D., Minden G., “A Survey of Active Network Research”, IEEE Communications Magazine, 35(1), pp. 80-86, Jan 1997.

Appendices

A. Auxiliary Measure Metrics

In addition to the measure metrics discussed in section 5, some other auxiliary features of the D-GA were recorded for analysis, they are:

- Agent fitness on each node;
- Average agent length on each node;
- Number of arrived data packets on each node;

For example, the following graphs are collected under the no network failure scenario of probGA Figure 24 to Figure 26.

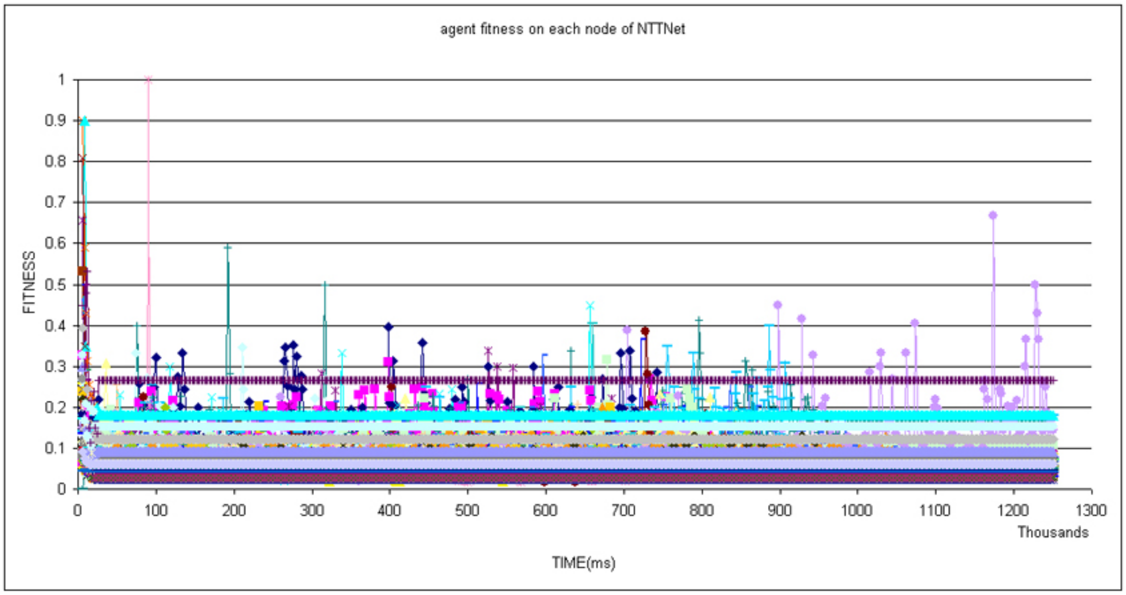

Figure 24: agent fitness - all network devices remain available

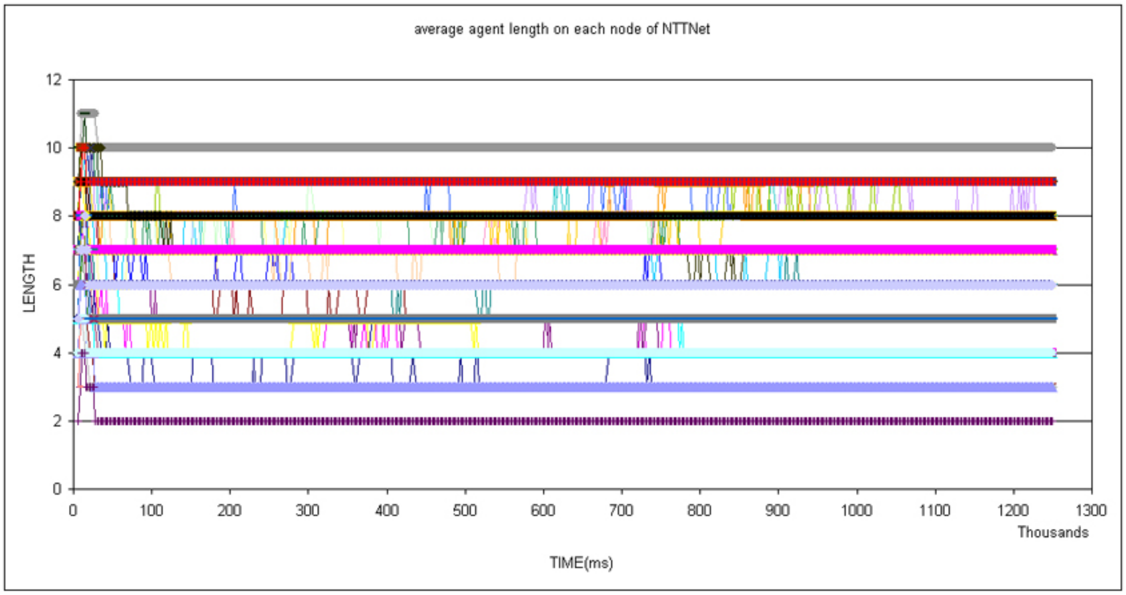

Figure 25: average agent length - all network devices remain available

The distribution of the average agent length on each node is between 2 and 10, Figure 25, and their average is 6.33. It's interesting that although the allowed maximum length of each chromosome is 30 genes, the average length is far shorter. And the other interesting fact is the average agent length of the five edge nodes (whose connectivity degree is 1) is 4.8. These reflect the adaptivity of the D-GA from the agent length aspect.

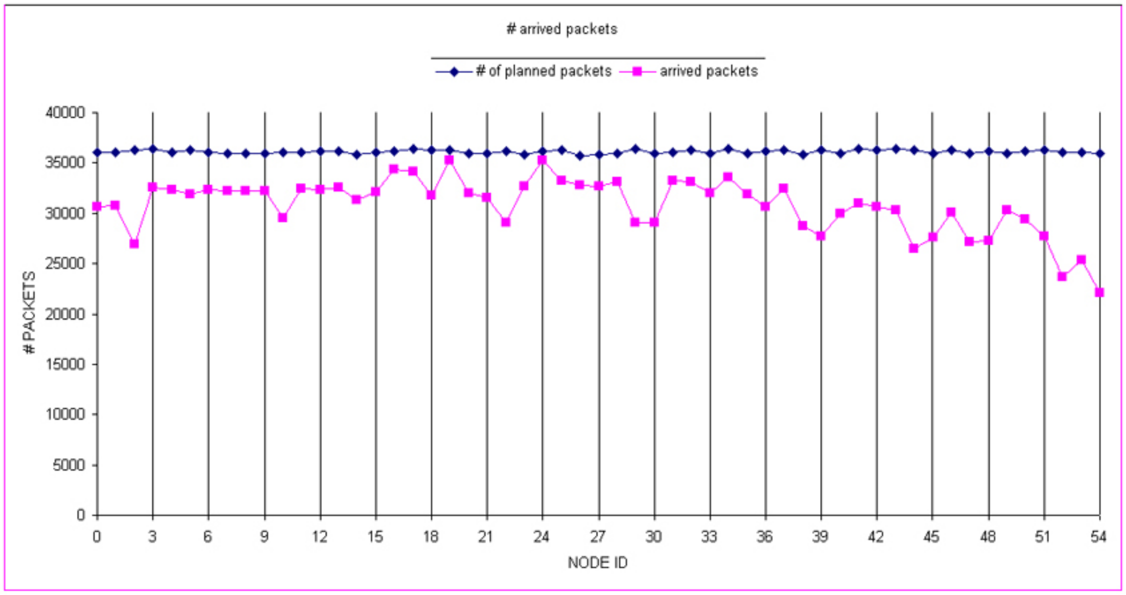

Figure 26: number of arrived packets - all network devices remain available

The upper curve tells the numbers of packets heading to each node, and the lower one shows the number of actual arrived packets. Those edge nodes (connectivity degree 1) have fewer arrived packets, for example, the nodes R52, R53, and R54 have degree 1 in Figure 4, they have least arrived packets, as shown in Figure 26.

B. Performance Graphs of the 8 Combination of GAs

Figure 27 to Figure 36 are the queue size and throughput performance graphs of the eight combination (Table 5) of the GA algorithm.

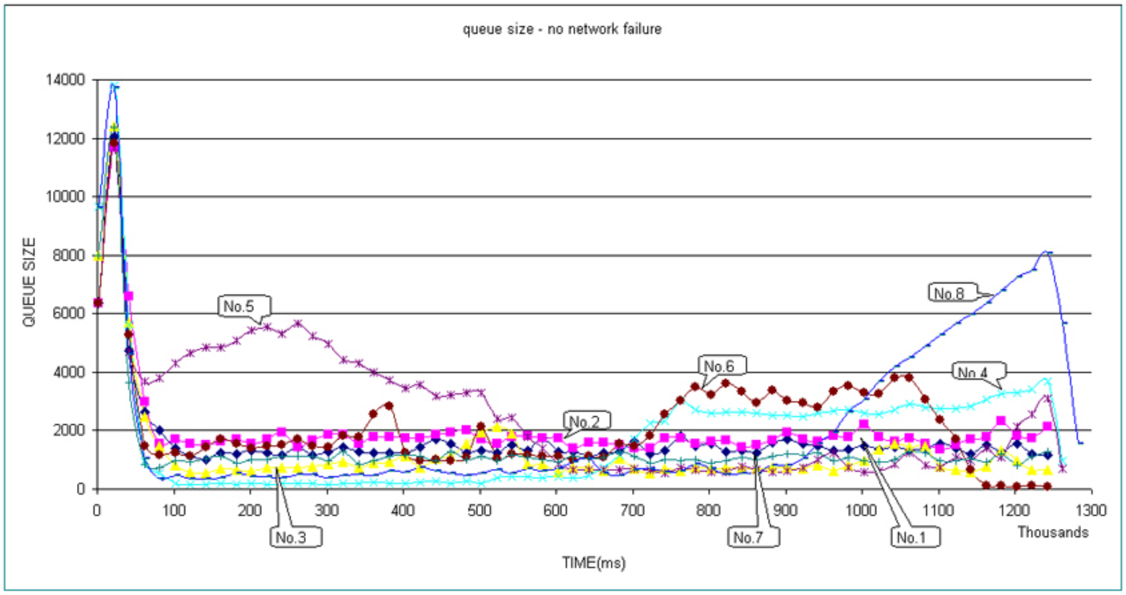

Figure 27: queue size of the eight combinations of GA - all network devices remain available

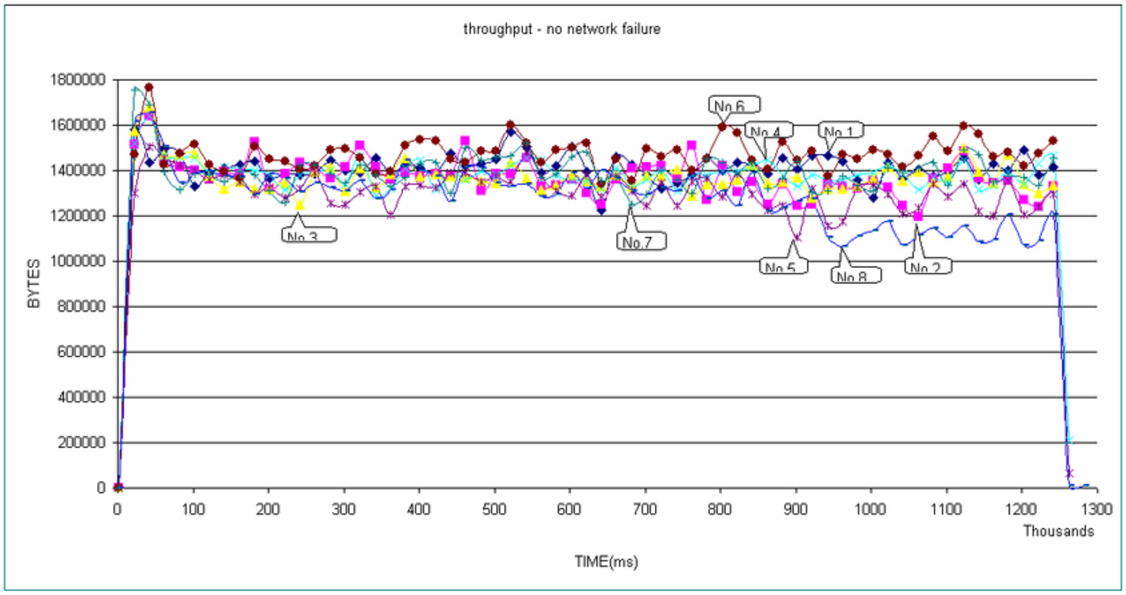

Figure 28: throughput of the eight combinations of GA - all network devices remain available

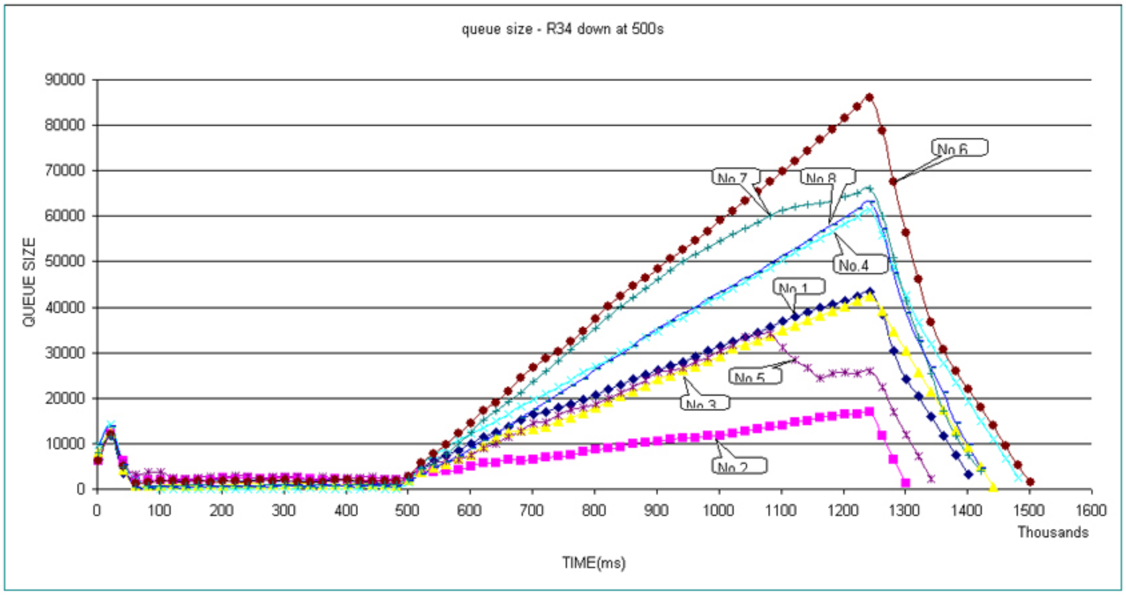

Figure 29: queue size of the eight combinations of GA – R34 down at time 500s

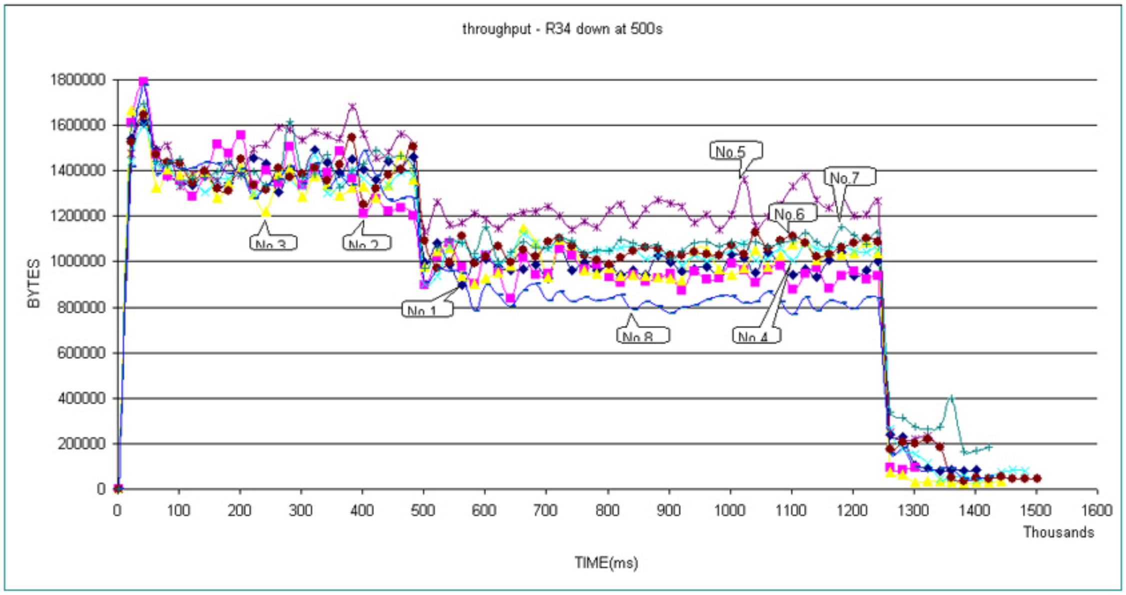

Figure 30: throughput of the eight combinations of GA – R34 down at time 500s

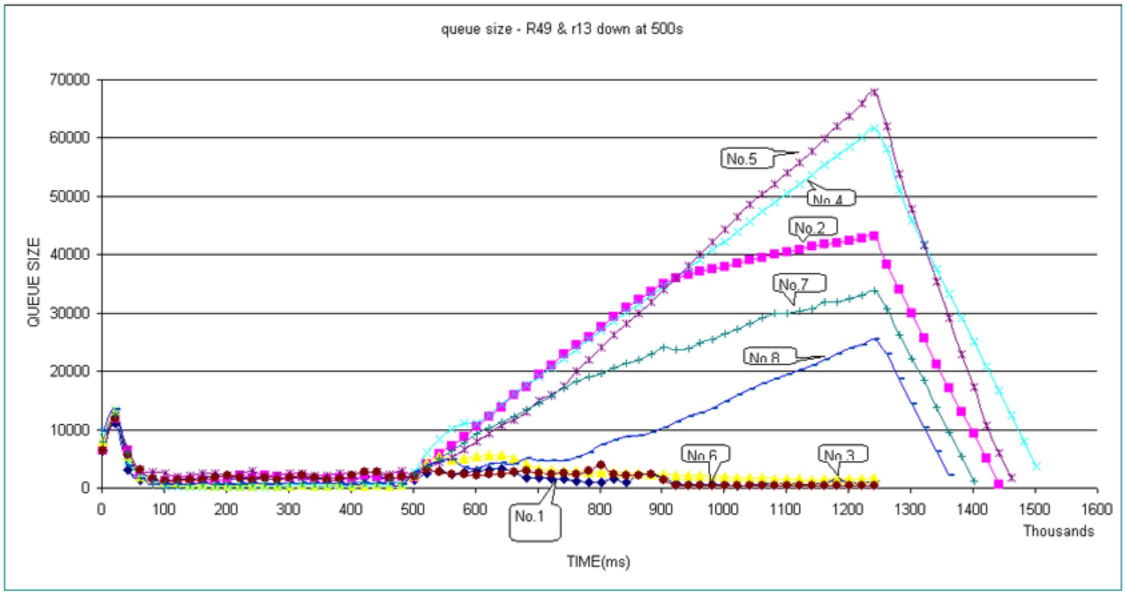

Figure 31: queue size of the eight combinations of GA – R49 & R13 down at time 500s

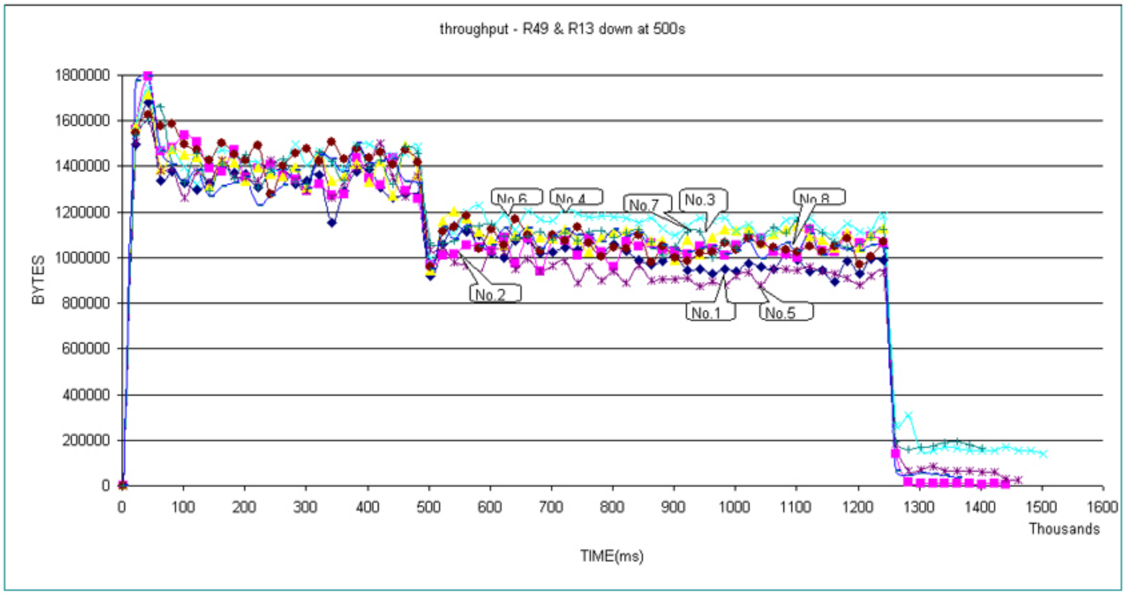

Figure 32: throughput of the eight combinations of GA – R49 & R13 down at time 500s

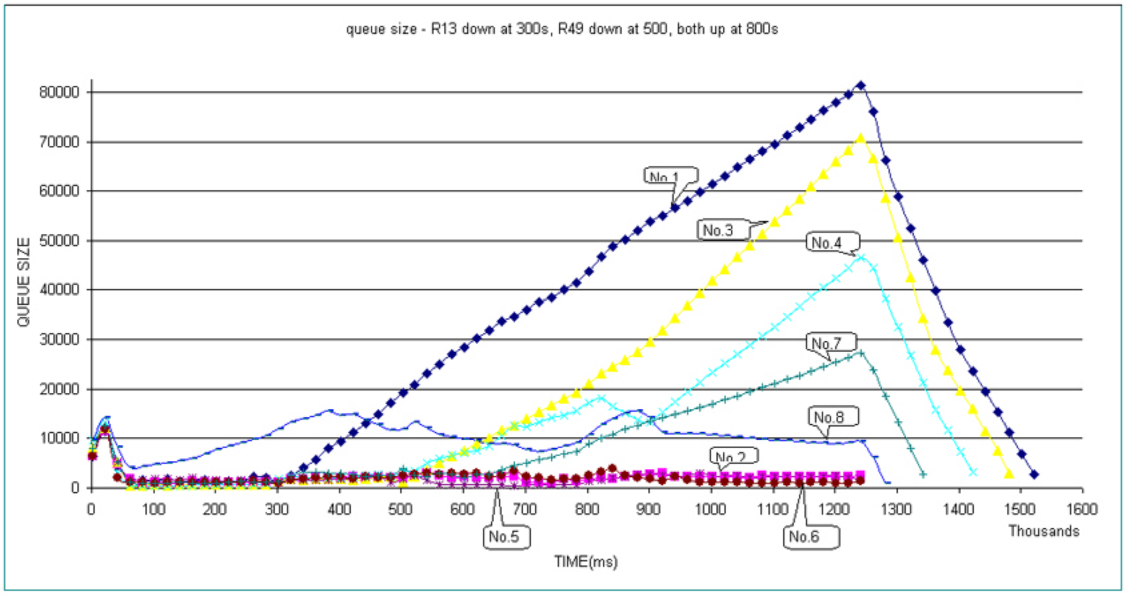

Figure 33: queue size of the eight combinations of GA – R13 down at time 300s, R49 down at 500s, both up at 800s

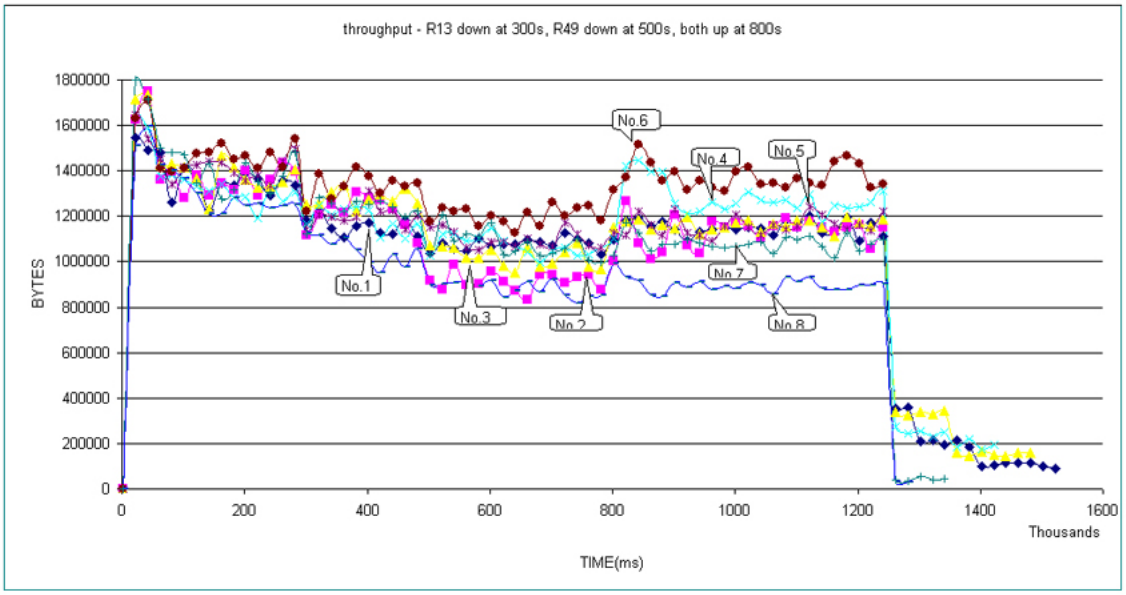

Figure 34: throughput of the eight combinations of GA – R13 down at time 300s, R49 down at 500s, both up at 800s

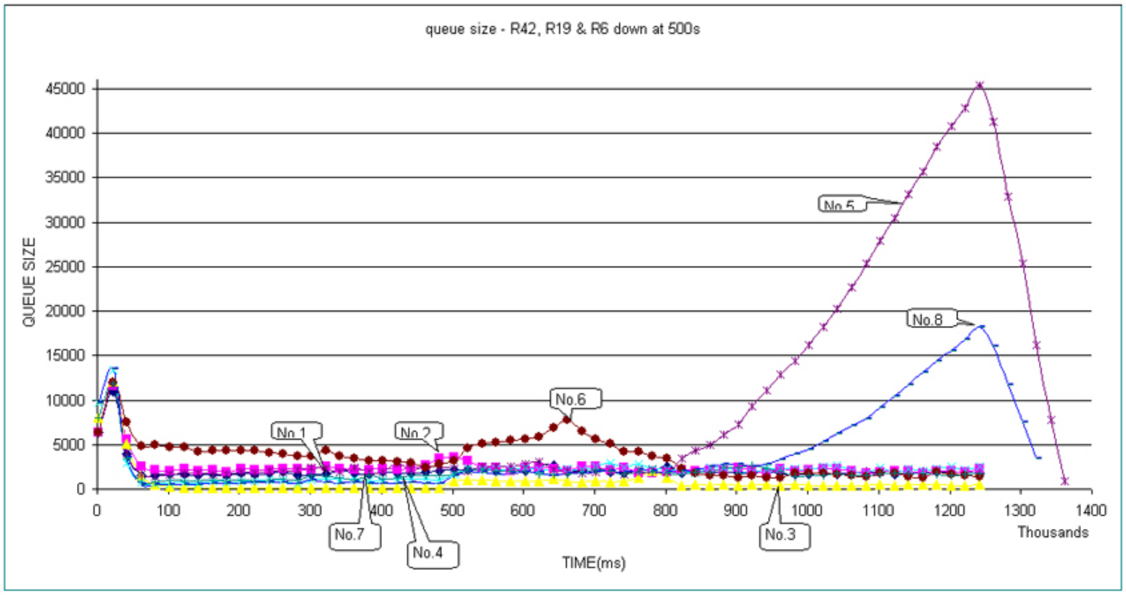

Figure 35: queue size of the eight combinations of GA – R42, R19 & R6 down at time 500s

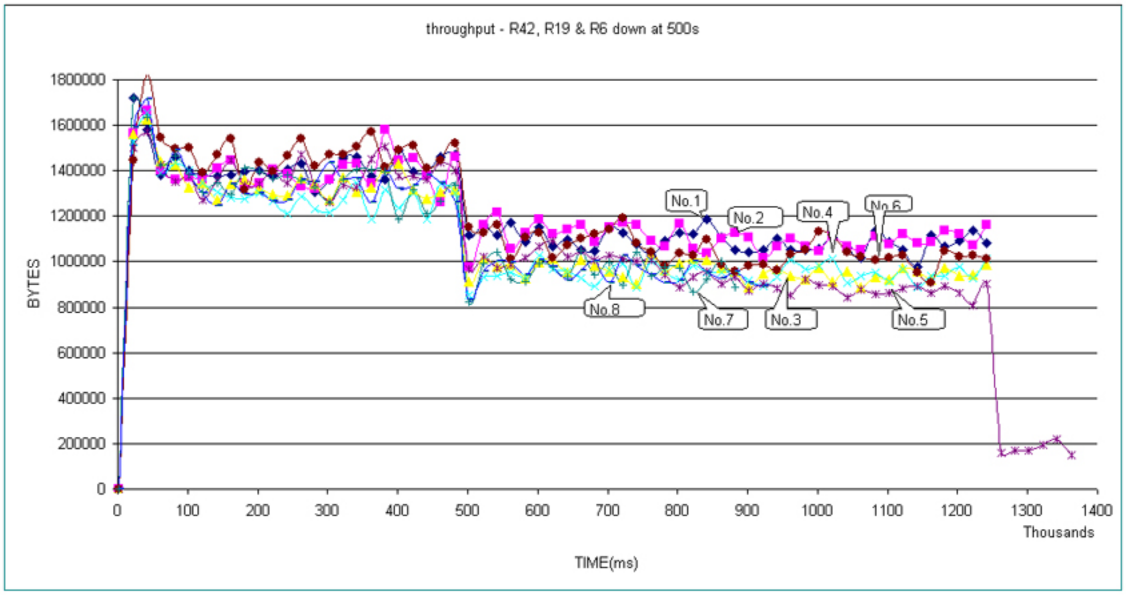

Figure 36: throughput of the eight combinations of GA – R42, R19 & R6 down at time 500s