Early Stage Assessment Report

Akinwande, Olumide J
Supervisor: Prof. Erol Gelenbe
Imperial College London
Department of Electrical and Electronic Engineering
Intelligent Systems and Networks Group
Email: olumide.akinwande13@imperial.ac.uk

1 Introduction

Packet switching is the more preferred strategy for communication in computer networks compared with the other popular circuit switching, due to its more efficient use of network resources. In circuit switching, a dedicated circuit in the network is first established between two nodes/hosts requiring a connection before communication begins. Because this circuit is maintained throughout the duration of the connection, network resources will be considered wasted during idle periods when messages are not being transmitted. Also, the dedicated circuit for the connection cannot be shared with other connections in the network. In a packet-switched network, messages are broken into smaller blocks called packets, which are routed independently through multiple intermediate nodes towards a destination node, where they are then re-assembled correctly into the original message. This approach allows for network resources to be shared by different connections, for example, in a first-come-first-serve manner.

While packet switching is more efficient in managing the limited network resources, it can require some sophisticated control measures in order to guarantee that packets are delivered safely and on time to their destinations. Some of these measures include the routing policy, flow control, error control, resource allocation and security. The focus of our research will mainly be on the problems of routing and security in communication networks. The routing policy defines the rules employed at the nodes in a network for the selection of paths for packets moving towards some destination. Therefore, the role of the routing algorithm is to compute paths between network nodes and ensure that the packets are correctly forwarded along these paths. In general, a routing algorithm should minimize the delay experienced by packets and also maximize the throughput of the network; other important design goals of routing algorithms can be found in [1]. Routing policies can be either static or dynamic. In static routing, the optimal paths are computed and known to nodes before the network starts operating and they remain unchanged. In order to compute these optimal paths, it is assumed that the behaviour of input traffic is known or can be estimated. In the present day networks, this assumption is no longer feasible and as a result, dynamic routing algorithms which compute paths based on the current network conditions are by far more preferable. A dynamic routing algorithm will generally incorporate a mechanism for regularly measuring information about the network state that is required for its route computation. The ARPANET project [2] which was one of the first successful packet switched networks and later evolved into the internet [3], also proposed important dynamic routing algorithms [2, 4].

Traditional dynamic routing algorithms like the distance vector and link-state algorithms try to maintain at the individual nodes of the network, information about the state of the network in order to keep the nodes connected, and to find the best paths. This means that changes in the network must be communicated to every node in the network when they occur. Due to the inherent delay in the network, it is not difficult

to see that this presents a huge problem which will become worse when the network size grows larger. Also, in trying to keep the nodes in the network connected at all times, there is the increased possibility of having unnecessary overhead in the network. These algorithms perform very well in small networks but their problems can become unmanageable or very costly to manage for larger networks. There is therefore the need to develop more intelligent cooperation among the network nodes for packet routing that will eliminate the overhead costs of synchronizing the network information among all the nodes in the network as obtainable in these traditional algorithms. This represents the goal of our research as we believe that any such level of cooperation can be better managed when the network size grows larger. The cognitive packet network (CPN) is a routing protocol that adopts such intelligent approach as it connects nodes on-demand and without prior information about the network, and it has been successfully implemented in large-sized networks.

Communication networks have become so critical to our daily activities which is evident from the increasing demand for mobile devices that provide ready access to the network. This has also been accompanied by constant threats from attackers that try to gain access to the network for criminal purposes. Common forms of attacks are the *Denial of Service* (DoS) attacks where the attacker unnecessarily burdens the network so as to deny genuine users access to network resources. Network security is therefore a big challenge for operators, in designing networks that can defend against new attacks while providing acceptable services to the users. This is also an area of interest for this research, concentrating on autonomic approaches.

In answering our research questions, the first stage of our research has focused on routing in wired packet networks and our first proposed algorithm exploits both the success of traditional routing algorithms in small-sized networks and the CPN's success in large networks. We also incorporate the sharing of connections among nodes whenever possible.

The remainder of this report is organised as follows. In the next section, we present a literature review of the different contributions to the routing problem for wired packet networks, from various fields of study. We also review the recent contributions relevant to autonomic communications, which is attracting a lot of interests. In the third section, we briefly describe the CPN protocol and the reason why it is suitable for our work. Our first proposed algorithm is detailed in the fourth section. We also present and discuss our simulation results. We draw conclusions in the fifth section and also provides some details of our future work.

2 Literature Review

Two well-known classes of routing algorithms for computer networks are the distance vector (DV) algorithms and the link-state (LS) algorithms. These two classes of algorithms employ already established solutions for computing shortest paths on graphs. The DV algorithms are based on the distributed version of the original Bellman-Ford algorithm [5] while the LS algorithms commonly use Dijkstra's algorithm [6] for path computation. It is easy to translate the routing problem to the problem of finding 'shortest paths' by viewing the computer network as a graph and defining a metric for characterizing the paths in the network. This similarity between packet routing and the shortest path problem, which is a much older problem, might explain why most of the initially proposed routing algorithms exploited graph theory.

In the DV algorithms, each node in the network maintains a vector containing the lengths of the shortest paths to every other node in the network, based on the defined path metric. This information together with the next-hop nodes along the shortest paths, will form the routing information at each node for the switching of data packets. Nodes will send this vector to their neighbours periodically and/or when a change is detected in an entry of the vector. Therefore, global information about the state of the network is propagated through local communication between nodes and their neighbours. Examples of the DV algorithms are the ARPANETs first routing algorithm [2, 7], Routing Information Protocol (RIP) [8], Interior Gateway Routing Protocol (IGRP) [9] and Enhanced Interior Gateway Routing Protocol (EIGRP) [10].

While the DV algorithm, in the basic form as described above, is simple and easy to implement, it can suffer

from some characteristic problems like the bouncing effect [11] and the counting-to-infinity [11, 3] problems which occur when link costs rise or there is total failure. These problems result in loops being formed in the paths chosen to route packets, which can seriously degrade network performance especially in large networks. The convergence properties of this algorithm are suitable only when changes in the network cause either a reduction or no change in the distances of any of the known shortest paths. Problems arise when a known shortest path increases in distance or becomes totally unusable due to link failures or a portion of the network becomes partitioned. When any of these events affects a node, it tries to find a new path (a new next-hop node) before propagating the information to its neighbours. This uncoordinated updating of the routing table increases the possibility for some nodes to believe old and incorrect information about the network over more recent updates and result in loops being formed. Also, since nodes only send updates to their neighbours, the possibility of old network information lingering in the system, before being purged out, is high.

Early proposed solutions for preventing routing table loops in DV algorithms were simple, easy to implement, and formed the basis for the more complex subsequently proffered solutions. *Holding down* is used in [7] where nodes will continue using a path to some destination for some given time (hold down time) after the path has become worse. This to allow for out-of-date information to be purged from surrounding nodes before nodes update their routing tables. In the *split horizon* method and its variations [12, 13, 8], nodes are configured not to divulge any routing information concerning a path to the neighbour from which they have learnt this path, that is, their next-hop node along the path. This is very efficient in eliminating two-node loops. Finally, there is the *triggered updates*, where nodes immediately send updates to their neighbours after noticing a change in metric in any path on their routing table. By quickly sending these updates, old information can be purged faster which can speed up convergence. These solutions were proposed on their own for the small networks available at the time and will easily become insufficient for very large networks. Cisco's IGRP [9] combines all these solutions in order to make the DV algorithm more efficient for a large network.

Ref. [13] also proposed that the full information of the shortest paths, in the form of the set of nodes along the paths, should be included in the routing information exchanged by nodes in order to completely avoid loops. This approach can drastically increase the size of the packets carrying the routing information, but it is straightforward in guaranteeing loop freedom and is applied in the Border Gateway Protocol (BGP) [14] used in the internet. One of the most important solutions for DV algorithms that guarantee loop freedom at all times is the diffusing update algorithm (DUAL) [15]. DUAL belongs to a class of solutions which use the diffusion computations technique [16]. This technique had been earlier proposed in [16] as a signalling scheme which guarantees that a system in a network that starts a diffusing computation [16] would be able to detect when the computation terminates. Basically, this class of algorithms [15, 17, 18] initiate some internodal coordination among the nodes affected by an increase in metric of a shortest path. This idea is similar to holding down, but rather than using a timer, the coordination between the nodes, through the exchange of special messages, ensures that all the affected nodes are informed of the current network state before they update their routing tables, thereby eliminating loops. The DUAL is implemented in Cisco's EIGRP [10] and is the most successful of this class because the nodes can select new paths under some sufficient conditions (called the *feasibility conditions*) that guarantee loop freedom and therefore not participate in the internodal coordination. Also, it provides a better algorithm, compared to the other solutions in the class, for handling multiple changes in the routing table entries. Although these solutions can effectively eliminate loops, they can invoke a considerably large amount of message transfers that can use up a lot of network resources. This will be worse when a sensitive path metric like delay is being used as the internodal coordination will become easily triggered. Ref. [19, 20] use similar feasibility conditions to the ones proposed in [15] and attempt to minimise the overall overhead costs of the DUAL.

While the feasibility conditions introduced in the DUAL guarantee loop free paths when nodes update their routing tables, it is easy to show scenarios where they are too restrictive such that alternative loop-free paths exist but fail the feasibility conditions. This implies that it is possible for a node to lose connection to some destination and be prevented from immediately selecting an existing loop-free alternative, a problem referred to as 'starvation' in [21]. Ref. [21] also proposed the BABEL routing protocol that solves this problem by

combining DUALs feasibility condition and the use of sequence numbers, earlier introduced in [22]. The sequence number idea is implemented such that they designate the age of the routing updates so that nodes can distinguish recent updates from older ones. This essentially avoids loops, and reduces starvation. BABEL requires periodic updates to perform well which can burden the network and also increase the chances of occurrence of sequence number wrap-around. The basic DV algorithm might be a simple algorithm but there have been no easy solutions to its characteristic problems.

The structure of the LS algorithm suggests that it was designed to avoid some of the problems of the DV algorithm. This approach is more centralized as each node maintains a copy of the full network topology, consisting of the cost of all links, from which it computes the shortest path tree to every other node. To make it adaptive, each node must periodically initiate a broadcast of the costs of its outgoing links to all the other nodes and also update its view using the incoming network information. The LS algorithm can still suffer from temporary routing loops caused by inconsistent information at the nodes but it is free from the counting-to-infinity phenomena. Some examples of the LS algorithm are the SPF (ARPANET's second routing algorithm)[4, 23], OSPF protocol [24] and IS-IS protocol [25].

Since nodes make routing decisions solely based on their copies of the network topology, it is essential that all these copies are identical and up-to-date. Therefore, nodes must constantly measure the costs of their outgoing links and with a reliable broadcast algorithm, communicate this information to all other nodes. This is not a trivial problem especially in a large network that experiences regular changes in its topology. Also, since nodes can only truly vouch for the information regarding the links they are connected to, it is essential that a mechanism for identifying the most recent updates received from any other node is implemented. This problem of maintaining correct and identical global information at every node is known as the *routing information problem* [26] or the *distributed database problem* [27]. This means that LS algorithms can be decoupled into two separate algorithms; an algorithm to solve the routing information problem, and the shortest path computation algorithm. The routing information algorithm is the most critical to the overall convergence of the entire routing algorithm and most LS algorithms differ in their approach to solving this problem. The ability of nodes to recognise the most recent updates in the presence of regular topological changes, and the inherent delay in the network presents a more interesting problem in maintaining synchronized topology information among network nodes. Some of the proposed solutions for the routing information problem can be found in [27, 28, 29].

LS algorithms possess better convergence properties compared to the DV algorithms but they incur more overhead in computation and communication. A popular mechanism for reducing the communication overhead of flooding in LS algorithms is the spanning tree approach [5]. Here, each node maintains a spanning tree connecting all the nodes in the network with itself as the root, and LSUs are forwarded using these trees. This approach incurs extra computational cost for maintaining these spanning trees [5]. LS algorithms [30, 31] have also been proposed that use partial information about the network topology in path computation in order to minimise overhead.

A class of routing algorithms that share similarities with both DV and LS algorithms are the path-finding algorithms [32, 33, 34, 35, 36]. In these algorithms, nodes include in their routing tables the second-to-last hop node, known as the predecessor node, on all their known shortest paths. This information is always included in the update messages sent by a node to its neighbours. By keeping the predecessor information, an idea first suggested in [32], a node can basically reconstruct all its shortest paths from its routing table. In other words, each node can always build a shortest-path tree (which includes other nodes in the network for which the shortest paths are known), with itself as the root node,. As a result, path-finding algorithms do not suffer from the counting-to-infinity problem of the DV and have improved convergence properties. While these algorithms can still suffer from path loops when multiple topological changes occur, these loops are always short-lived. Ref [34] eliminates all loops by incorporating feasibility conditions for the routing table updates and internodal coordination, identical to the DUAL. Path-finding algorithms are comparable to the DV algorithms in the way messages are exchanged in the network, and since each node essentially constructs a shortest path tree, they also resemble the LS algorithms.

Contributions to the problem of routing in computer networks have also come from other fields of study. Ref. [37] provides an extensive survey of routing algorithms from different communities of researchers. The trade-off between exploration and exploitation in some given environment towards eventually obtaining or learning some optimal or near-optimal policy, is very fundamental in the field of reinforcement learning (RL) [37]. It is not difficult to see that routing in packet networks involves exploration and exploitation processes, since to guarantee that the best path between two nodes in a network is known, all possible paths should have been explored. In large computer networks, where the network state can change very quickly and possibly result in an initially good path becoming inadequate, exploration must be an endless process. The trade-off between exploration and exploitation in packets networks becomes very significant when packets carrying the actual payload are also used in the exploration of the network. This represents the general view from which researchers in RL have approached the routing problem.

Q-routing [38], which was derived from the Q-learning algorithm, is the first routing algorithm from this field of study. In Q-routing, there is an initial exploration phase where nodes make random decisions when forwarding packets to some destination. With every decision a node makes, its learning algorithm updates a Q value corresponding to that decision. Eventually, the algorithm should converge to a routing policy which is expected to give the best path between two nodes. The routing policy at a node for some destination is the decision, that is, the next-hop node, with the optimal Q value. Once this policy is learnt, nodes will keep making the same decisions and only the corresponding Q values will be updated by the learning algorithm. This means that there is no way the algorithm can detect improvements on unknown paths. As a result, nodes will have to learn new policies only after the known path deteriorates such that the Q values are no longer optimal. This affects the adaptability of Q routing, and there are also no guarantees for the convergence to the best paths. Predictive Q-routing [39] was introduced to address the limitations of Q-routing. It improves the exploration and routing policy of Q-routing by incorporating an estimation of the recovery rate of paths in its decision algorithm. Ref. [40, 41] combine Q-routing with the ϵ-greedy strategy, where the routing policy is such that nodes either forward packets using the optimal Q values with a probability (1- ϵ) or make a random decision with probability ϵ (ϵ is usually chosen to be small). This is to improve the exploration process and the adaptability of the routing algorithm. Backward exploration is integrated with Q routing in [42] towards improving its convergence. In the algorithm, called the dual reinforcement Q-routing, a node, before forwarding any packet, appends its best Q value corresponding to the source of the packet. The implication is that nodes learn about paths to both the source and the destination of a packet after receiving and forwarding it. While the looping properties of these routing algorithms using RL are barely investigated, the randomness of the initial exploration phase and the fact that learning occurs after a packet has already been forwarded, make the occurrence of temporary path loops inevitable.

Another important group of routing algorithms that have enjoyed considerable attention from researchers are the agent-based routing algorithms. In these algorithms, a mobile agent traverses the network in order to learn the network topology towards establishing and improving connections between the nodes in the network. The agents either experience the network conditions themselves like an actual data packet in order to learn the topology, or they learn from local estimates made by the nodes they traverse. These algorithms are also commonly referred to as *ant-based* because the mobile agents imitate some properties of biological ants when they perform complex tasks like finding the best path to a food source [43]. Here, the exploration and exploitation dilemma, as tackled in the RL based routing algorithms, only exists in the routing of the agents, as data packets are completely exempted from the exploration process. Therefore, reinforcement learning techniques are commonly applied in agent-based routing algorithms. Separating the data packets from exploration should improve their overall performance compared to RL based routing algorithms, but at the cost of increased control overhead introduced by the agents. Agent-based routing algorithms generally use either of two strategies [44]: Round trip agent routing [45, 46] or Forward agent routing [47, 44, 48]. In round trip agent routing, forward agents are regularly launched by nodes to some destinations to explore the network and learn the best paths to these destinations. These forward agents have local memories where they store routing (link costs) and the full path information of the paths they use towards their destinations. When a forward agent reaches its destination, a backward agent is generated that carries the information

in the forward agent's memory back to the source, travelling along the reverse route used by the forward agent. The Backward agent uses the path information to route itself and the routing information to update the routing tables at the visited nodes. In the *AntNet* algorithm and its variations [45, 46], nodes maintain a parametric statistical model that represents a local view of the entire network traffic; a pheromone table for routing forward agents; and a data routing table for forwarding data packets. An entry in the tables is a probability value describing the 'goodness' of choosing a neighbour node as the next-hop node for a packet or an agent moving towards some destination. Loops are prevented from the forward agents by deleting information about the nodes forming the loop once a cycle is detected. The information brought back by the backward agents is used to update the statistical models at the nodes. Also, on arrival of a backward agent at a node, the information it brings back, which corresponds to the routing decision made by its forward agent, together with the updated statistical model, are used to generate a reinforcement signal for updating both the data routing table and the pheromone table. The size of the reinforcement signal is proportional to how good the most recent information brought by the backward agent is compared to the previously obtained information that is summarised in the statistical model.

The forward agent strategy for agent-based routing algorithms was proposed to improve the convergence and reduce the overhead costs of the round trip agent routing strategy. This approach was inspired by an earlier and similar successful implementation in [49] for routing and achieving load balancing in circuit-switched telecommunication networks. In this strategy, forward agents are launched regularly by the network nodes and they update the routing tables of nodes as they traverse the network, thereby eliminating the need for backward agents. Forward agents explore paths in the reverse, that is, they find paths beginning from the nodes they traverse in the network and leading to their originating nodes. In the *Regular Ant* algorithm in [47], the forward agent measures the costs of the links it uses has it moves in the network. It updates at each node, the routing table entry corresponding to the goodness of choosing the link the agent arrived on for forwarding data packets destined for the agent's originating node. This algorithm assumes that the network links are symmetrical in terms of link cost, which means that the path an agent traverses from its originating node to a destination node carries the same cost as its inverse path. This is a comfortable assumption as it also allows both data packets and the agents to be forwarded using the same routing table. Using the same routing table for forwarding both data packets and forward agents can be detrimental especially after the algorithm has converged, as this will result in the agents only exploring the learned optimal paths. Lack of exploration of other paths makes it difficult for the algorithm to be fully adaptive to subsequent changes in the network. The ϵ-greedy approach is employed in [47] to ensure continuous exploration and prevent this situation. Ref [47] also proposes the *Uniform Ant* routing algorithm for the more general case where the symmetry assumption is removed. Here, forward agents are not assigned any destination on creation but a time-to-live and they are terminated after this time expires. Forward agents are always switched randomly at each node with an equal probability of choosing any of the outgoing links. This means that agents only explore the network and exploitation is restricted to the data packets alone. This suggests that a considerable size of agents will be required to ensure good performance of the algorithm. The agents learn the estimated link costs of the links in the opposite direction to the ones they use and update the routing tables as appropriate. The authors do not make it clear how the agents obtain these costs but they can be easily be learnt from the nodes if every node monitors and estimates the link costs of their outgoing edges. In the *Cooperative Asymmetric Forward* (CAF) algorithm [44], data packets on arrival at a node, estimate and leave at the node, the cost of the link they arrived on. This is then carried by forward agents moving in the opposite direction and used to update the routing table. A separate routing table called the reverse routing table is maintained at each node for forwarding the agents. The reverse routing table ensures that CAF agents originating from a source node to some destination node are routed along the possible paths used by information packets moving in the opposite direction.

Another interesting routing algorithm with the forward agent strategy is the *Beehive* routing algorithm [48], which is based on the organizational principles used by honey bee colonies especially in foraging. In this algorithm, there are two types of forward agents; the short distance agents, which are restricted to a small number of hops after they are launched; and the long distance agents, which are expected to traverse the

entire network. Also, the network is subdivided into; foraging regions, which are non-overlapping partitions of the network, with each region having a representative node; and foraging zones, maintained by each node as the set of nodes reachable by their small distance agents. Representative nodes regularly launch long distance agents only, while non-representative nodes only launch the short distance agents. Agents are intelligently flooded in the network, with short distance agents eliminated once they have reached their maximum hop, and they explore paths beginning from the nodes they traverse and leading to their originating nodes. This implies that every node has path information for reaching the nodes in their foraging zone and for reaching all the representative nodes in the network. For destinations not in a node's zone, the node forwards packets towards the representative node of the region the destination node belongs to. A special table, called the *foraging region membership*, is maintained at each node and it maps known destinations to their foraging regions.

These agent-based routing algorithms generally use a stochastic policy for forwarding packet, in contrast with the deterministic policies of the DV and LS algorithms. This essentially makes them multi-path algorithms and they combine load balancing in the network with other good performances. A disadvantage of the stochastic policy is that there is the possibility, though with small probability, of loops occurring in the paths assigned to the data packets.

A simple routing algorithm that shares similarities with the agent-based routing algorithm using the forward agent strategy, but avoids the use of RL techniques is proposed in [50]. Special packets called *scouts*, which carry their source node address, a sequence number, and the path cost for reaching the source node, are regularly generated by each node and intelligently flooded in the network. Nodes will update the path cost before forwarding a received scout packet. A scout broadcast is used to set up connections originating from the other nodes in the network to the source of the scout packet. The network improves its knowledge of the best paths to a node with every scout broadcast originating from the node.

Genetic algorithms which code evolutionary operations of selection, crossover and mutation based on the work of Charles Darwin in biology, in order to solve optimisation problems [51], have also found application in network routing. In a routing algorithm, a path between two nodes can represent a chromosome or an individual and its fitness is the cost of directing a packet along it. Any routing algorithm that can compute full path information, in terms of the sequence of nodes along the path, can easily apply genetic algorithms. By using the operators, the best (fittest) discovered paths can be exploited, subject to the network topology, in order to generate more and possibly better paths. By subject to the topology, we mean for example, in order to use the crossover operator on two paths, they must have the same source and destination nodes and at least one other common node appearing in their sequences.

In the *Genetic Adaptive Routing Algorithm* (GARA) [52, 53], each node maintains full topological information and initial paths are generated using the Dijkstra's algorithm as in the LS methods. Genetic operators are applied at specific intervals to find more paths and special packets called *delay query packets* are sent along the discovered paths to evaluate their costs. Nodes also send their discovered routes to neighbour nodes at regular intervals. The routing algorithm in [54] applies genetic algorithms to paths discovered using an agent-based round trip routing strategy similar to AntNet. A more original approach is proposed in [55], where the genetic operators can be applied independent of the network topology. Agents (Chromosomes) launched from each node to explore the network are each assigned a string of integers, with each integer value of the string chosen from a closed interval beginning with 0. Each node initialises by generating a population of agents with random string lengths and values, and then sending out half of the population in search of paths. The number of agents generated is directly proportional to the number of outgoing links present at a node. An agent determines its next hop node by making a clockwise count on the neighbouring nodes, starting from the node it arrived from. The value of the count is selected from the position in the agent's integer string corresponding to the hop count the agent is about to make since its launch. A random next hop node is chosen if the agent's count falls on the node it arrived from, and the agent resets its integer string to reflect this decision. An agent stores information about the nodes it traverses and the trip times to reach these nodes from its originating node. After making a number of hops corresponding to the length of

its integer string, or reaching a node with only one outgoing link, an agent becomes a backward agent and returns to the source node with all its information. Agents that find the shortest paths to the frequently used destinations are considered the fittest and the genetic operators are applied to their integer strings to generate more agents. In contrast to the two aforementioned genetic routing algorithms which use source routing, DGA uses next-hop routing.

Another desirable property for routing algorithms is the ability to find and use multiple paths to forward packets. Using multiple paths helps to improve load balancing in the network and the efficient use of the limited network resources. [56] established necessary and sufficient conditions for achieving overall minimum delay performance in the network. In this work, optimising the overall system performance was considered rather than just for some connection. While their approach is not very workable in practice, because of the assumptions of known stationary traffic inputs and an unchanging or slowly changing network, it suggests that for near optimal performance in terms of network delay, the possibility must exist at each node for sharing flows meant for some destination node, among the outgoing links, such that there are no path loops [57]. This review has focused primarily on single path algorithms since multipath routing algorithms are generally extensions of some single path algorithm. The *MP Scout* algorithm [50] is the multipath version of the Scout algorithm; [57, 58] and [59] extend the LS algorithm and DV algorithm respectively to provide multiple paths; while [60] is a multipath version of the path-finding algorithm. These algorithms find multiple loop free paths for packets. [57, 59, 60] achieve loop freedom by applying loop free invariants when updating the routing tables in the network. These loop free invariants are similar to the feasibility conditions of the DUAL but adapted for multipath routing. The extensions of these algorithms to find multiple paths come at a price of increased communication overheard in finding these extra paths and keeping them loop free.

More recently, research has focused on developing autonomic networks that are self-managing, self-configuring, and self-regulating [61]. Sensor networks are a good example of where autonomic communication might be required, especially when the sensors are located in places that can be dangerous for humans. For such environments, it is essential that the sensor network is efficient, for example, in its energy consumption, as the network would most likely be replaced as a whole in the event of failure. It can also be desirable that the sensor network provides a better level of service to packets that originate as a result of an higher priority event [62]. This means that the sensor network should be able to differentiate between events and have quality of service (QoS) [63] capabilities. With autonomic networks, the possibility increases of having more intelligent, user-centric networks, for example, which can connect users to desired services based on their specified QoS needs, as presented in [64, 65]. It is easy to see that traditional routing algorithms will be inadequate for such intelligent networks as the overhead of maintaining network information in order to provide different levels of services can become unmanageable. It is therefore necessary to consider methods for searching for paths in a network on-demand and with little or no prior knowledge of the topology [66, 67]. The *Cognitive Packet Network* (CPN) [68, 69] is a routing protocol that adopts such strategy. In CPN, nodes obtain network information only when they need it, thereby eliminating the need for any form of network-wide synchronization process. The CPN routing protocol uses a similar strategy to the round trip strategy of agent-based algorithms in finding paths. Exploration packets, called *smart packets* (SPs) are sent out by a node requiring a connection in order to gather relevant information necessary for path discovery and maintenance. In most implementations, the SPs are routed using the *random neural network* (RNN)[70] sitting at each node. For continuous exploration, an SP will be allowed to make a random decision rather than the RNN's, based on some probability value known as the *drift parameter*. [71] studied the convergence properties of the CPN based on this parameter. CPN is also quality of service driven and has been successfully implemented to provide different level of services to packets, based on user defined path metrics, in both wired [72, 73, 74] and wireless networks [73]. More recently, CPN was shown to be capable of providing the quality requirements necessary for real-time traffic [75, 76]. [77] presented a machine learning approach, called *SMART*, which uses CPN-inspired overlay routing strategy to improve the forwarding delays and loss rates of intercontinental IP routes. Ideas from CPN have also been applied in QoS-driven task allocations in cloud computing using online measurements [78] that track the state of the cloud system [79, 80]. CPN, like most adaptive routing algorithms, experiences routing oscillations where different paths are chosen for

packets belonging to the same connection. [81, 82] showed that routing oscillations do not severely degrade performance in CPN, and they also propose intelligent measures to control these oscillations in network. These attributes make CPN an important contribution to autonomic communications.

Efficient use of energy has become important for communication networks, and schemes for improving energy consumption have been studied analytically for both wireless and wired networks [67, 83, 84, 85] using probability models [86, 87, 88]. In order to balance the QoS requirements and the energy consumption in a network, [89, 90, 91, 92] combined the energy consumption per job, and QoS per job using a composite cost function and studied, analytically, for the distribution of load in the network that optimised both energy consumption and QoS. An energy aware routing protocol (EARP) is presented in [93, 94] that uses the functionalities of the CPN in gathering required network information in order to choose paths that minimise the total power consumption in the network while respecting the QoS of individual flows. For autonomic networks, like sensor networks, where energy is very critical to the network performance, energy harvesting [95, 96] must be combined with this optimal load distribution. The *Energy Packet Network* (EPN) introduced in [97, 98, 99], can also be adapted to such systems to intelligently manage the energy demands in the network. In EPN, energy is dispatched using *energy packets* [97] according to consumer and storage requests, under the control of *Smart Energy Dispatching Centres* (SEDCs).

Also critical to autonomic networks, is the ability to maintain network stability and reliability in the face of network threats, like denial of service attacks, worms, viruses. A probability model for the interaction between cells (computer software) and viruses, in the presence of an antiviral agent, is developed in [100]. They allow the antiviral agent to be made up of different proportions of agents that target different strains of the virus. Their analysis showed that for proper control of the infection, the appropriate mix for antiviral agent must take into account the resistance of each strain of virus to the antiviral agent. CPN's resilience to network worms, compared with the OSPF routing protocol, is successfully demonstrated in [101, 102]. Their results show that CPN can still provide acceptable QoS by quickly adapting to the changes in the network caused by the worms. An attack is detected when the repetition count reached some set threshold and the offending terminal will have all its communication blocked for some fixed time duration. The authors in [103] use CPN to detect and defend against distributed denial of service attacks by detecting unusual bandwidth usage by suspect connections and dropping all associated packets. Opportunistic communication (oppcomms), proposed in [104, 105] to assist in emergency evacuation, allows mobile nodes, carried by the evacuees, to maintain communication among themselves as they come in close proximity thereby creating an autonomic network. Their goal is for these mobile nodes to learn and improve their view of the environment, through participation in oppcomms, in order to properly direct the evacuees to safe locations. The absence of a central structure and the possibility of any node participating, make oppcomms easily susceptible to network attacks. [106, 107] propose techniques for identifying and rejecting packets originating from malicious nodes. The popularity of smart mobile devices has made them attractive targets for cyber-criminals. In addition to having access to personal information by compromising these devices, cyber-criminals can also gain entry to the mobile networks. A recent project [108] has focused on developing novel strategies to improve the mobile network security against such cyber-attacks.

3 Cognitive Packet Network

The performance of algorithms like the DV and LS algorithms is highly dependent on the accuracy of the network information available at each individual node. The methods that have been proposed for maintaining accurate routing information at the nodes, apart from introducing fresh problems, generally generate more overhead in the network which increases with the size of the network. It is clear that the problems associated with routing algorithms that require that nodes regularly update their knowledge of the state of the network can become unmanageable or very costly to manage as the size of the network grows. Hierarchical routing is used in large networks to reduce the routing information maintained at individual nodes. This is achieved by grouping the network nodes into different levels of clusters so that nodes maintain full information about the

nodes close to them and lesser information concerning those further off [109]. Because of the routing protocol being employed, reducing network information as a result of hierarchical routing immediately translates into degradation in the network performance. It is therefore necessary to consider routing algorithms that do not require any form of synchronization involving all the nodes in the network, but where nodes only seek network or routing information when it is needed. This is one of the motivations for the development of the CPN routing protocol [69, 72]. In CPN, nodes initiate connections on-demand, using special packets which gather the relevant network information required for establishing and maintaining the connections.

As earlier mentioned, CPN is an agent-based routing algorithm that consists of three types of packets: *smart packets* (SPs), *acknowledgement packets* (ACKs) and *dumb packets* (DPs). When a node requires a connection, it sends out some amount of SPs towards the desired destination. A SP, on arrival at a node, will update itself with the relevant routing information, including the identity of the node, and then decide its next-hop node based on the learning algorithm at the current node. The learning algorithm commonly used in the CPN is the random neural network (RNN) with reinforcement learning, which directs SPs to the outgoing link corresponding to its most excited neuron. If a SP arrives at a node with no prior information concerning paths to the SP's destination, the SP constructs a RNN with as many neurons as the number of outgoing links at the node and then makes a random decision. Therefore, in the simplest implementation, a node will have, when it is required, an RNN for a destination node. An ACK packet is generated when a SP arrives at its destination. The ACK carries back to the source node, the information obtained by the SP, along the inverse path used by the SP after all loops have been removed. ACKs update the RNNs at the nodes they traverse as appropriate and this update triggers the RNNs to update their weights. The weights are updated to either reward or punish the decisions made by the SPs [110]. The ACK will provide the source node with the full path information for the requested connection. The DPs which carry the payload will use the highest ranked path information at their source node to route themselves. ACKs are also generated for DPs when they arrive at their destinations, so DPs can also be used to obtain routing information. After setting up the connection, subsequent SPs are sent at a reduced rate, to maintain and possibly improve the connections. A comprehensive review of the variations, applications, and performance evaluations of the CPN can be found in [111]. More recently, CPN has been successfully applied in emergency navigation algorithms for confined spaces, to find and distribute evacuees along safe egress paths [112, 113, 114].

4 Ongoing Research

The focus of our research is to develop ideas to improve routing in packet networks. In the first phase, we focus on wired networks were the topological changes are restricted to changes in link costs or total link failures. Note that the failure of a node can be easily modelled as the failure of its incident links. Our goal is to develop algorithms that allow for more intelligent cooperation among the nodes in a network such that nodes can share and benefit from already established connections. The approach of maintaining the current topological information at all nodes for effective routing, as adopted in the traditional routing algorithms, becomes less efficient and less manageable in large networks. There is therefore the need for more intelligent cooperation among network nodes and this is where our work seeks to make some contributions.

The CPN provides a solid framework to implement our ideas because of its ability to find paths in a network only on-demand and without prior information about the network topology. We believe that this approach will, overall, be more efficient in routing in large flat networks and this is evident in the success of CPN in its application in emergency navigation algorithms [112, 113, 114], where a building with more than 200 nodes in its graphical representation was considered. Our first proposed algorithm, which we present in the next section, combines the idea of hierarchical routing where nodes maintain full information for nodes close to them, and the CPN's ability to establish connections on-demand without the need for periodical updates, to establish connections between nodes further apart.

4.1 Proposed Algorithm

In our first algorithm, we propose a coordination among nodes that allows nodes in some defined vicinity to intelligently share their connections. The network will be divided into non-overlapping partitions (groups) and nodes within a partition will engage in this coordination when they desire communication with nodes outside their partition. Every node maintains a *group routing table* (GRT) for paths leading to the nodes in its group. This can be implemented using any simple next-hop routing algorithm like the distance vector algorithm. Since the number of nodes in a group will be relatively small compared to the size of the network, it would be easier to manage the problems associated with any next-hop routing algorithm implemented locally in the group. Our idea is for nodes within a group to have ready access to each other but initiate a connection setup process when they have to communicate with nodes outside the group, and also to share these connections whenever possible.

The algorithm is such that a node requiring a connection will first check if any of the nodes in its group has initiated or is actively processing a connection to the same destination node. If such connection already exists, this node will send its packets to use the connection instead of setting up its own connection. We also employ special packets, called *carrier packets* (CP), to carry packets sharing the same connection when possible. By this we mean that a CP will only contain packets originating from different sources but belonging to the same group, and sharing the same connection. The CP should make the network traffic appear lighter in terms of the number of packets in the network. Our aim is to improve network performance by considerably reducing communication overhead in the network through intelligent cooperation among nodes. The CPN routing protocol will be used to set up connections between nodes belonging to different groups. We now give a more detailed description of the proposed routing algorithm, including also the conditions when a carrier packet is initiated.

I Assumptions: For our first set of simulations, we divide a network into different non-overlapping but connected groups of nodes and assume that each node knows the identities of the nodes in its group.

II Initialization and local routing tables: The only initialization performed in the network is by the local routing algorithm employed within a group in order to find paths that connect all the members. Nodes store and maintain these paths in their GRT. Also, the local routing algorithm will use periodic or triggered updates to maintain the routing table.

III Establishing connections between groups

- When a node requires some connection to a node outside its group, it first sends a *request packet* as a broadcast message to its group nodes for the possibility of sharing an already established connection. A *request packet* will only be allowed some given number of hops in order to restrict connection sharing to between nodes in close proximity and to ensure the termination of the flooding process.
- A node on receiving a request packet, will reply if it is processing the desired connection. The node will continue to flood the request packet if it has not reached its limit hop.
- A waiting time is also set after which, if no response is received for a *request packet*, a node will initiate its own connection using CPN.
- If a positive response is received, the node begins the process of using the already established connection. The response will also carry the current known full path information for the connection, if this information is available. If more than one positive response is received, the first response is always chosen. The only exception is when the first response does not include path information. In this case, the first response with path information is selected.
- After selecting the connection to use, the node informs the initiator of the connection and switches its packets towards this node. Any node initiating a connection will only permit some given number

of nodes in its group to share the connection. The initiating node also updates the nodes it is sharing its connection with every time the best path information for the connection changes.

- A node that receives full path information after making a request to share a connection, will first check if its address or the address of any of its neighbour nodes lies along the received path. If this is the case, in order to avoid unnecessary path loops, the node extracts the appropriate route information and source routes its packets instead of switching them towards the initiator of the connection.
- The initiating node will continue maintaining its connection as long as any of the nodes sharing the connection has packets to send to that destination. Therefore, a node sharing a connection must inform the initiator when it no longer desires the connection.

IV Packet switching and the Carrier packet

- Source routed packets are forwarded normally according to their path information when they enter nodes not belonging in the same group as their sources
- The format for the packets that carry the payload will include an additional field for a second destination address. This will help the nodes in differentiating between packets actually destined for nodes within the same group as their sources, and packets moving towards group member nodes in order to share some connection. A packet that is being switched to share a connection will have the address of the initiator of the connection in its second destination field.
- In order to prevent loops, once a node begins to share its connection, its packets will leave their path information at the group member nodes they traverse as they move towards their destination. This ensures that packets moving towards a node in order to share its connection can be source routed once they enter a node along the connection path and loops are avoided. This path information will be stored temporarily at these nodes.
- As earlier mentioned, CPs are used to carry packets sharing the same connection but from different sources. A CP will contain the payloads of the different packets and their source addresses. CPs will be source routed. We assume that destination nodes can properly process the CPs. CPs can be initiated at any node but only for packets originating from group member nodes. While processing a packet, a node checks its queue for a packet using the same connection as the packet being processed. If such a packet is found, it creates a CP that carries the two packets. Note that a CP can only be created if at least one of the packets carries the path information for the connection or the node creating the CP has this information.
- A node will discard any packet without full path information and whose source belongs to a different group.

  The flowchart in Fig. 1 describes the packet forwarding logic at a node.

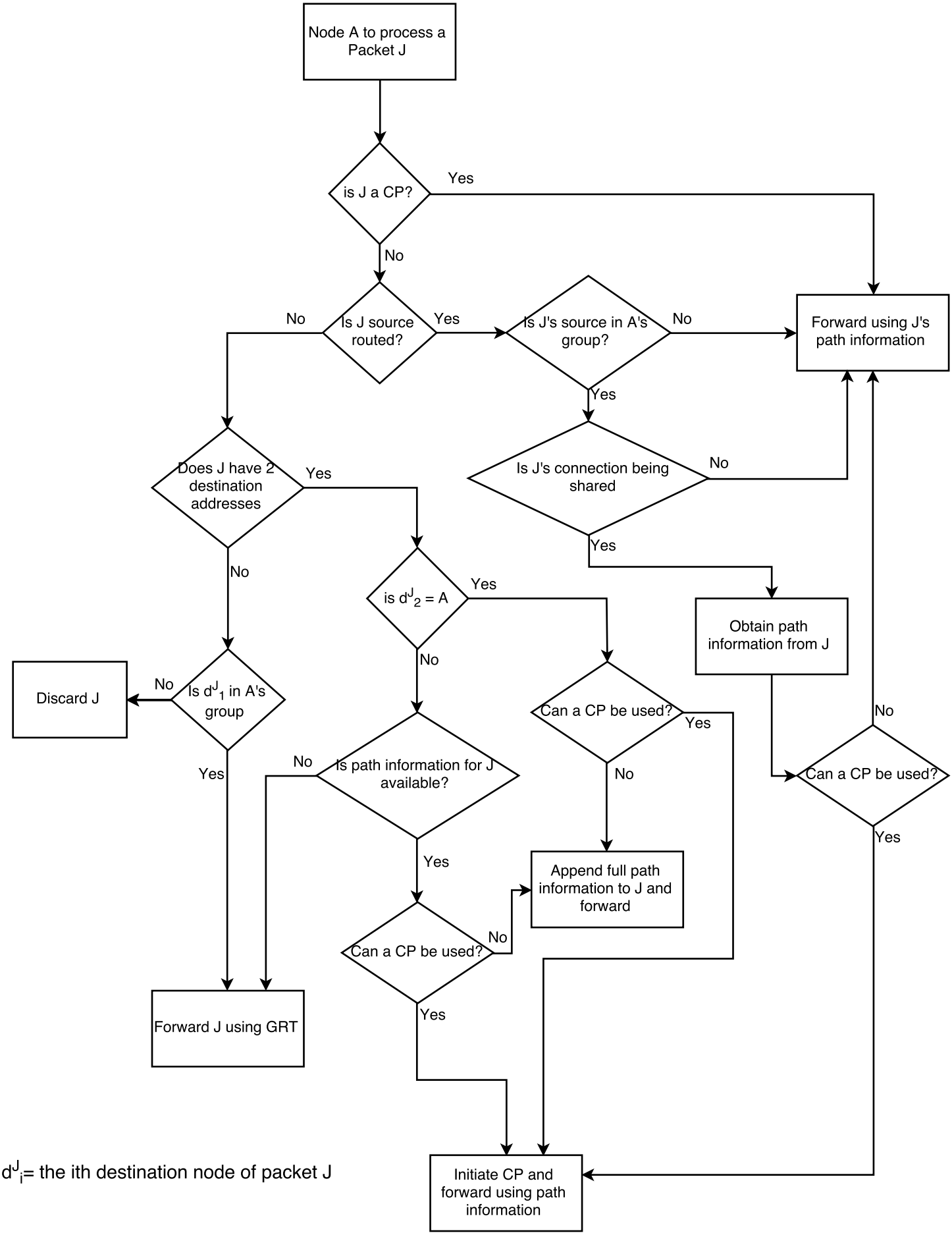

Figure 1. Packet processing logic at a node A, where A is not the source of the packet

4.2 Simulation

Our simulations were carried out using the OMNET++ discrete event simulator [115]. OMNET++ provides a good platform to compare different routing algorithms under similar load conditions. In our first set of simulations, we compare the performance of our proposed algorithm with the setting where no partitions exist and nodes find all paths using only the CPN protocol.

4.2.1 Modelling a node

In the simulations, a node is modelled as consisting of three layers: the Application layer, the Routing layer and an array of queues.

The application layer creates packets, in sessions, which are destined for a randomly selected node and then forwards them to the routing layer. The number of packets in any session is also a random value between 15 to 20 packets. The average time for creating a complete session is between 19 *seconds* to 25 *seconds* depending on the number of packets in the session. There is a rest time between sessions with an average time of about 1.5 *seconds*. This layer also processes packets destined for the node.

The routing layer implements the rules for switching packets, including the CPN algorithm. Packets requiring connection using CPN are first placed in an internal memory of the routing layer until a path is found for the connection. After deciding the neighbour node to forward a packet, the routing layer sends the packet to the queue connected to the appropriate interface. Packets arrive and leave a node through the queues. Therefore, the number of queues at a node is the same as the number of incident links at the node. It is assumed that the routing layer processes packets at zero time, so no queues are formed for packets arriving at a node.

We do not set limits for the size of the queues so packets are never dropped as a result of full queues. Due to the high data rates we have used for the links in the simulations, this assumption is reasonable. Also, the maximum queue length recorded at any point in the simulations never exceeded 40 packets even at high traffic levels. The different traffic levels are explained later on. In all our simulations, we do not allow for nodal or link failures to occur

4.2.2 Network topology

The network topology we have used for the simulations, shown in Fig. 2, is a 98-node network extracted from the ITC Deltacom network topology obtained from [116]. The extracted network was also modified by including extra links to increase the connectivity. The data rates for all the links are set to $100 Mbps$. We also set a propagation delay for each link to a random value between $0.1 ms$ and $1 ms$.

4.2.3 Dividing the network into areas

The nodes in the network have been labelled as shown in Fig. 2. In implementing our proposed algorithm, we divided the network into 5 areas. In Table 1, the range of nodes in each of the areas is given. All the areas are connected and every node knows the identities of the nodes in its area. In addition, a node also knows the identities of all its neighbours including those outside its area.

Area	Node range
1	0 - 19
2	20 - 38
3	39 - 54

4	55 - 72
5	73 - 97

Table 1: Range of nodes in each area

4.2.4 Traffic Levels

The routing algorithms are evaluated under three different traffic levels: Low traffic, Medium traffic and High traffic levels. Under low traffic levels, only 10 nodes are allowed to initiate packet sessions throughout a simulation. For medium traffic levels, 25 nodes initiate packet sessions, while 45 nodes create packet sessions for high traffic levels. The set of nodes that create sessions are randomly selected before carrying out the simulation.

For a scenario, we carried out 10 simulations and different combinations of nodes are used in each of the simulations. The same combination of nodes is used for all the different algorithms.

4.2.5 Performance metrics

The two main metrics we have used in comparing the different algorithms are *delay* and *routing overhead*.

Delay, measured in *seconds*, is computed by subtracting the packet creation time in the application layer of the packet's source from the time the packet is received at the application layer of the packet's destination. We only record the delay values for the data packets and for each simulation we evaluate the average delay. Also, because of the size of the network, and CPN's process of finding paths, we also record and compare the maximum packet delay.

The routing overhead for a simulation, measured in *packets/second*, is computed by summing the total counts of non-data packets processed by each of the nodes and dividing the result by the model simulation time. The model simulation time for all our simulations is one hour.

4.2.6 CPN implementation

We send 25 smart packets (SPs) for the path discovery process of the CPN routing protocol and set the maximum hop limit for an SP to 60 hops. We also set a reconnection time after which if no path has been brought back by an ACK, the path discovery process is restarted. From our test results, we observed that reducing this reconnection time significantly reduced the maximum packet delay seen. However, continued reduction of the reconnection time beyond some point had no effect on the maximum delay, so for all our CPN implementations, we set the reconnection time to $500ms$. We also set the size of a data packet with no source routing information to $1016MB$.

4.3 Results

Since we run 10 simulations for each scenario, the results presented are averages over 10 simulations. For our first set of results, presented in Fig. 3, Fig. 4 and Fig. 5 , we compare three routing algorithms.

In the first algorithm, all paths are found using the CPN routing protocol.

The second algorithm implements our proposed algorithm as presented earlier on. The network is divided into 5 areas according to Table 1. On initialization, the GRT of each node is updated using the Dijkstra's

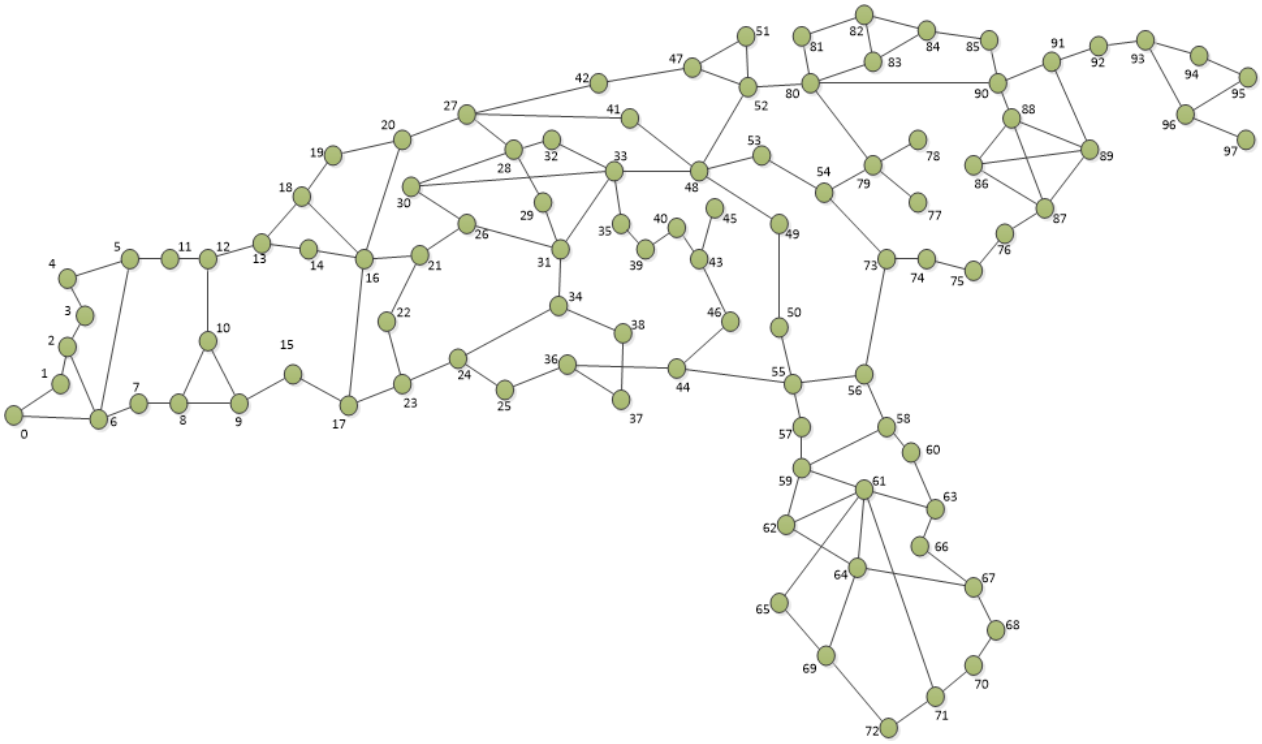

Figure 2: Network Topology.

algorithm to ensure that the nodes in an area can reach each other. To achieve this, we let each node intelligently flood packets, similar to link-state packets in OSPF, carrying the costs of all its incident links that connect to area member nodes. The flooding process is such that link information is never sent outside an area. On receiving all the link information for an area, a node, using the Dijkstra's algorithm, will determine the next-hop nodes along the best paths to all the other nodes in the area. For the connection sharing process, we set the waiting time, after which a node requesting for a connection to share will initiate its own connection if no response is received, to $0.5ms$. This value was arrived at by considering the size and the maximum hop of both the request and response packets, the speed of the links, and possible queueing delays. The maximum hop allowed for a path request packet was set to 5 hops. Also, the temporary path information recorded by nodes from packets of shared connections have an expiration time set at $5ms$. Later on, we discuss the importance of this parameter, and completely eliminating temporary path information recording have on performance. It is important to mention at this point that a node that has neighbours that are outside its area, known as an *area border router* (ABR) in OSPF, will always respond to a path request packet seeking connection to any of these neighbours. We restrict an initiator of a connection to not allow more than 2 nodes to share the connection but this does not hold when an ABR is sharing a connection destined for any of its neighbours.

For the third algorithm, we do not allow for connection sharing so that when a node desires an inter-area connection, it immediately begins the path discovery process using CPN.

Fig. 3 shows that combining area routing, implemented by a LS routing algorithm, and CPN can improve the average delay from almost $9ms$, when only CPN is used, to below $7ms$. These results also show that CPN can maintain good performance with increasing traffic. By having more connections being initiated, the implication is that more SPs will be sent out to gather network information. This will provide more information to the RNNs located at the nodes and generally improve learning and SP routing.

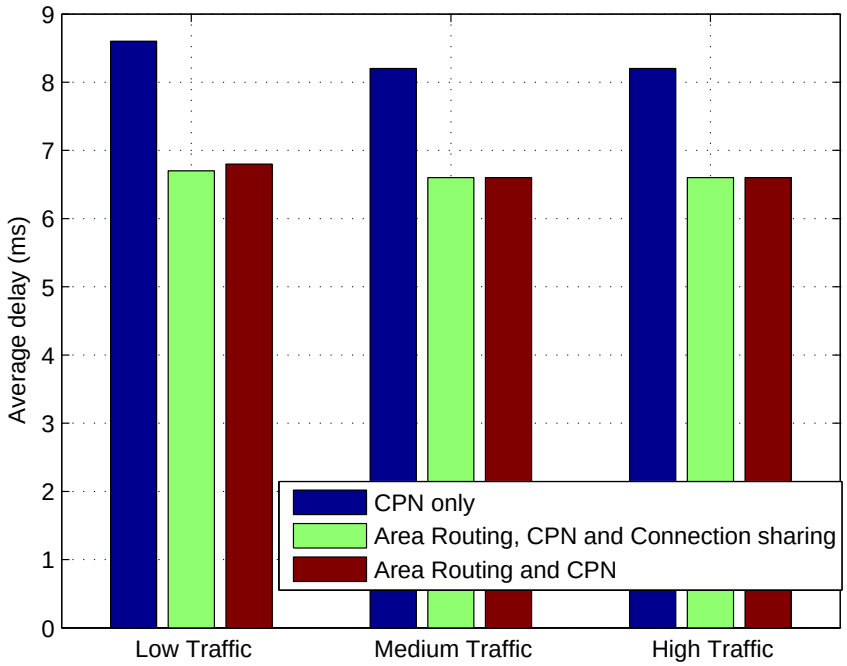

Figure 3: Average delay.

Fig.5 compares the average value of the total CPN connections required by the algorithms under similar load conditions. Reconnections are also included in this sum. It is clear that having area routing can considerably reduce the number of times CPN is used. As expected, the results also show that most of the connections required are inter-area connections which confirms the importance of the CPN algorithm to the overall performance.

The routing overhead incurred by the algorithms for the different traffic levels is shown in Fig. 4. In computing this overhead, the overhead caused by the initialisation process that updates the GRTs of the nodes have not been included. Therefore, the overhead reported is only due to the CPN packets and the local packets used in implementing the connection sharing mechanism. Since more connections are initiated when only CPN is used, it is not surprising that it generates more overhead than the other algorithms. While our proposed algorithm can reduce the connections created compared to the algorithm where no connections are being shared, the overhead incurred by both algorithms still appear similar. This can be explained by the overhead caused by flooding request packets every time a connection is required.

In summary, our first results confirm that combining area routing and CPN for our large test network will, overall, produce a better network performance when compared to using only CPN to find paths. The results also suggest that employing the connection sharing mechanism as we have described earlier is not very efficient.

In trying to improve the efficiency of our algorithm, we observed from the results that connection sharing can be considered as a low probability event. Therefore, rather than increasing the waiting time at the beginning of the process in order to increase the possibilities of connection sharing, which is also less optimal, the algorithm should be modified so that nodes only initiate the process when there is a very high chance that connection sharing is possible. In our modified algorithm, rather than having nodes flood request packets when they require a connection, nodes will advertise their newly created connections using a similar flooding

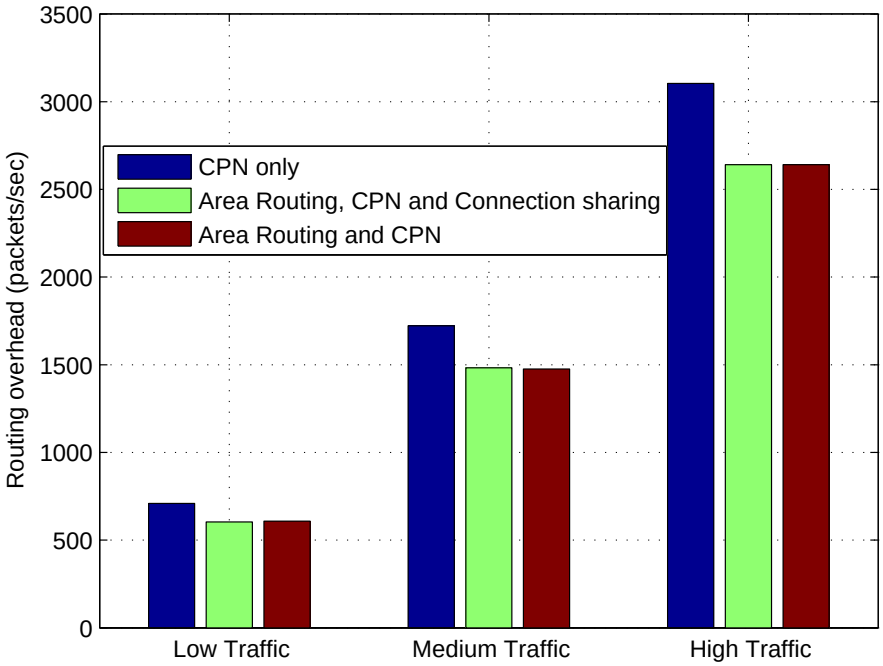

Figure 4: Routing Overhead.

algorithm. By similar we mean that, the advertisement packet will also be restricted to 5 hops in order to maintain connection sharing between nodes in close proximity. This means that each node will have a location in its memory for recording advertised connections. When a node requires a connection, it first checks its memory to see if the connection has been advertised by another node. If the connection has been advertised, the node then sends a request packet to the advertising node in order to begin the connection sharing process. This implies that the request packet in the modified algorithm is a unicast packet. Also, when a node ends a connection, another flooding process is initiated to inform the concerned nodes which then delete the connection. Another important modification is in the behaviour of the ABRs. As part of the initialisation process, ABRs will advertise, once, all their neighbour addresses that are outside their area. These ABR adverts remain in the memories of all the nodes that receive them for the entire duration of the simulation as they are never ended.

For our second set of results, we compare the modified algorithm with both the initial algorithm and the algorithm where the connection sharing mechanism is not used.

From Fig. 6, the modified algorithm offers slight improvements in average delay at both medium and high traffic levels. The efficiency of the modified algorithm in increasing connection sharing in the network compared to the initial algorithm is also evident from the results in Fig. 7. By ensuring that connections are shared whenever possible, our modified algorithm, despite involving two flooding processes per connection created, incurs lower overhead than the other algorithms as shown in Fig. 8.

In the following results, we show how performance can be further improved by taking advantage of the local area routing algorithm in the forwarding of the SPs. In the previous implementations of our algorithm, SPs are always routed by the RNNs located at the nodes. Note that the RNN can also decide to randomly forward an SP in order to ensure continuous exploration of the network. When an SP arrives at a node without RNN, the node creates one, and then randomly forwards the SP. This implies that nodes will create RNNs for

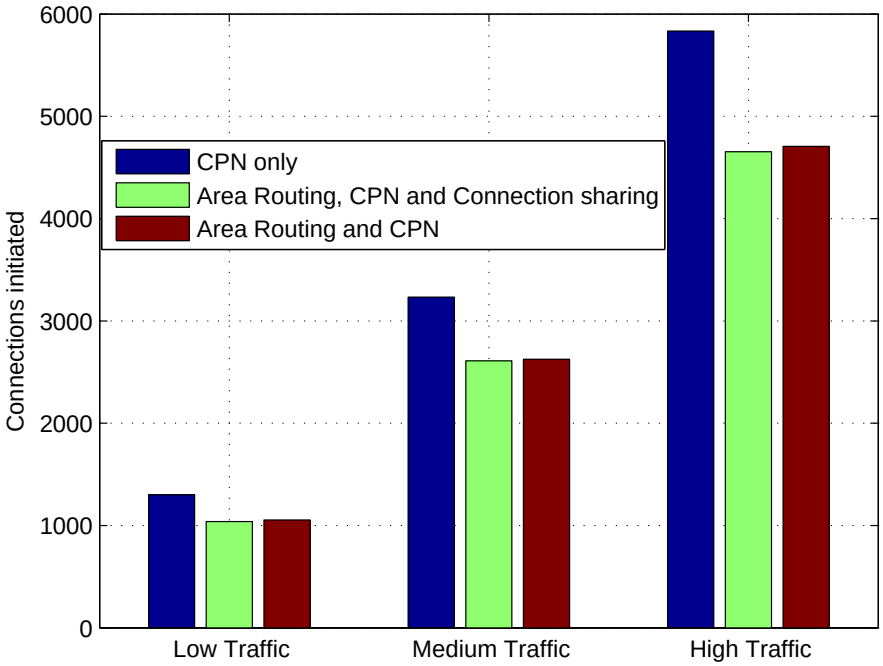

Figure 5: Total connections initiated.

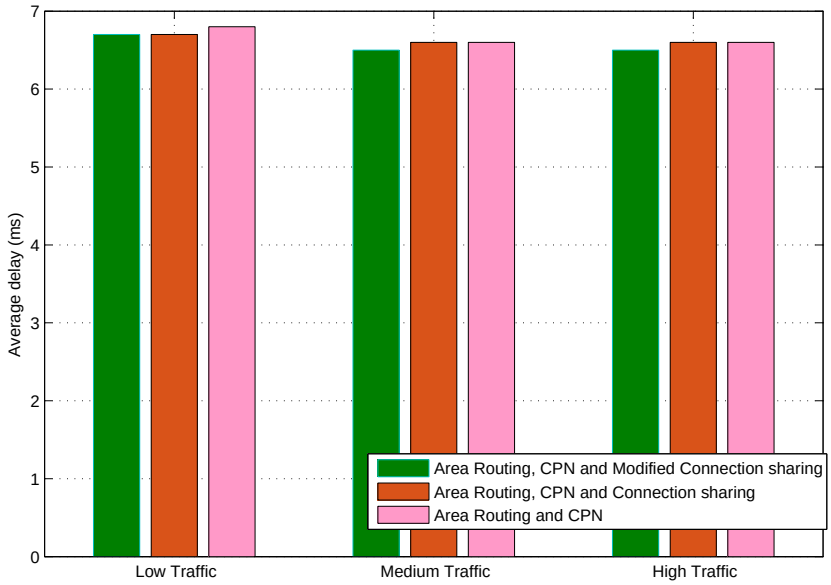

Figure 6: Modified Algorithm: Average Delay.

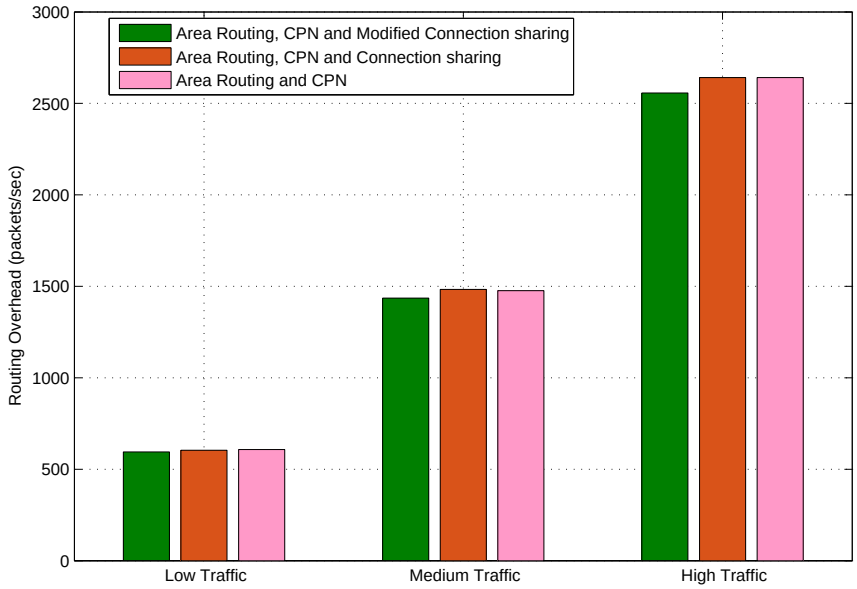

Figure 7: Modified Algorithm: Routing Overhead.

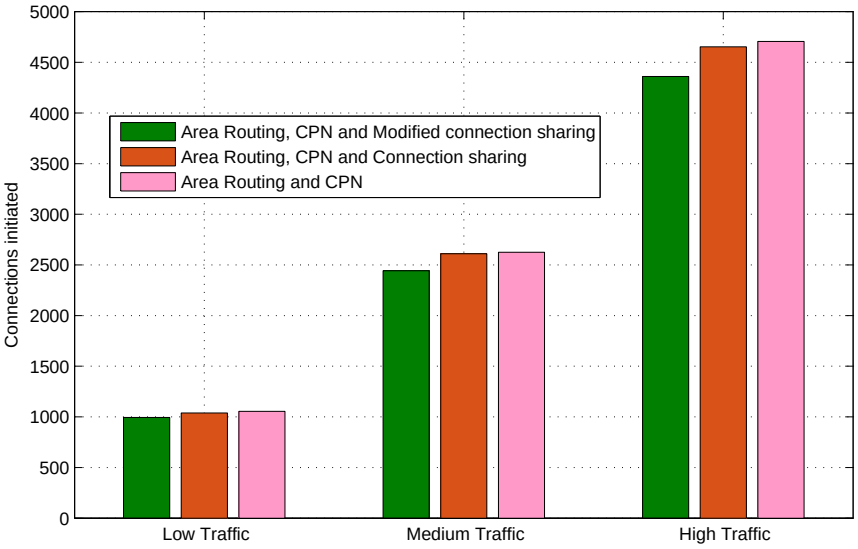

Figure 8: Modified Algorithm: Connections initiated.

connections destined for area member nodes despite having group routing table information. We now modify the routing of SPs such that when they reach the area where their destination resides, they are forwarded using the GRT rather than creating RNNs. Even though we have not considered processing overhead, having less RNNs will reduce the processing at the nodes.

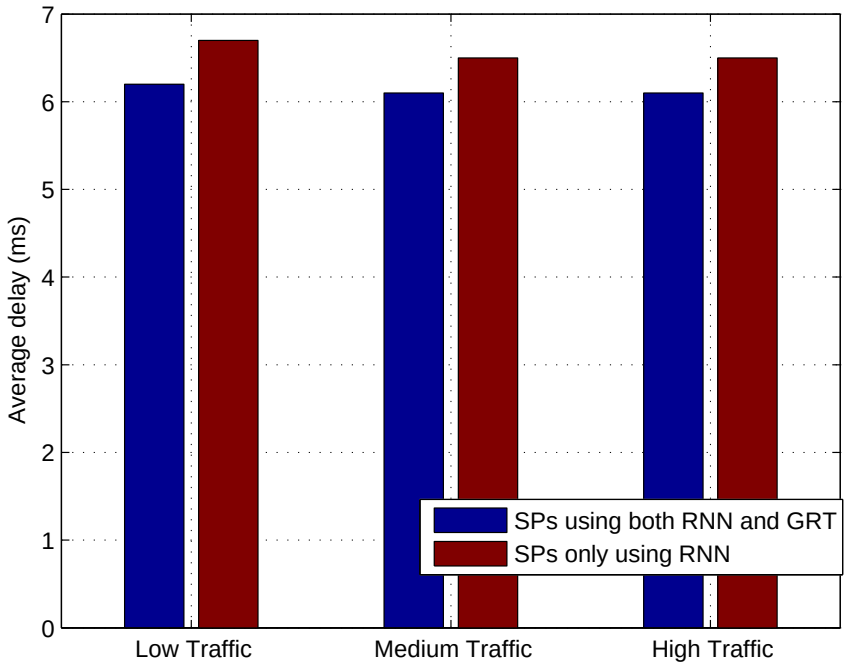

Figure 9: Routing SPs with the Group routing table: Average Delay.

From Fig. 9, routing SPs with the GRT once they arrive at their destination node's area further improves the average delay in the network for all traffic levels. This makes sense since routing SPs in this way implies that they are basically searching for anyone of a set of nodes rather than a single node which improves their search conditions.

Another important observation is in the average maximum delay seen for any packet in the simulations. We have not considered this in the past comparisons because of the noticeable variation in the average value of the maximum delay for different runs of the same scenario. The average maximum delay for all our previous algorithms lied in the range $0.6 secs$ - $1.5 secs$, which is always greater than the reconnection time of the CPN setup process. The result in Fig. 11 therefore shows that improving the search conditions of the SPs by allowing them to use the GRT ensures that CPN will generally not require a reconnection process in finding any path despite the size of the network. This also reflects in the reduced overhead as can be seen in Fig. 10.

Finally we discuss the importance of having temporary storage of paths of connections that are being shared. In the implementation of our proposed routing algorithm, a node will record the path information carried by a packet if the connection the packet belongs to is being shared, and the path information was assigned to the packet by the initiator of the connection. Also, path information will only be recorded if the connection originates from the area. Since CPN improves on the quality of paths its finds, it is important that this recorded information be expired after some time in order to ensure, as much as possible, that packets going to share a connection use the most recent path information as known to the initiator. This becomes more crucial when node/link failures are considered in the network. Clearly, removing this path recording, which implies that all packets will be assigned paths by the initiating node, gives the most robust implementation of our proposed algorithm even though it increases the possibilities of having path loops for the packets going to share a connection. These loops are reduced by restricting connection sharing between nodes in close proximity and having the initiator send path updates to the nodes sharing its connection. We conclude by

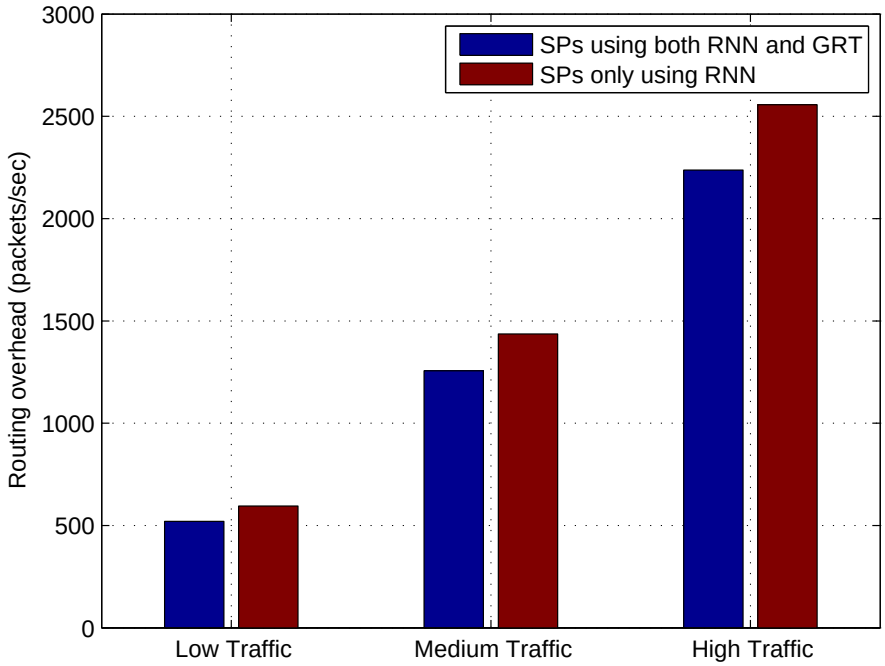

Figure 10: Routing SPs with the Group routing table: Routing Overhead.

stating that it is better not to have the path recording than to set its expiry time to a large value and for our simulations, 1 *second*, for example, can be considered a large value.

5 Conclusion and Future work

The first part of our research is focused on making contributions in routing in communication networks. Autonomic communication is receiving lots of interest because it can better manage the demands of the 'modern' network. We have provided a review of relevant literature where we highlighted the developments made in the traditional routing algorithms. We also reviewed some of the current research areas in autonomic communication especially in QoS, efficient energy consumption and harvesting, and in security.

Our first algorithm successfully combines the LS routing algorithm and CPN in managing a large autonomous network. The network is divided into areas that run, locally, the LS algorithm and the nodes initiate the CPN routing protocol when inter-area communication is desired. Area routing is already a solution in OSPF for managing communication in a large autonomous system. In OSPF, another area, called the *Backbone area* or *Area 0* consisting of all the ABRs must be formed. The role of the backbone area is to connect the other areas of the network. For inter-area communication, packets are first sent into the backbone area which then directs them to their destination area. Since it is important for the backbone area to be connected, a separate LS algorithm must be implemented in the area. Virtual links are also sometimes required to connect the ABRs that do not share a common link. With our approach, using autonomic communication, we have eliminated the need for the backbone area and the overhead of keeping it connected through link-state updates and virtual links. Also, all nodes are at the same level since ABRs do not have any special roles apart from advertising non-area neighbours. We have also shown that a proper implementation of the

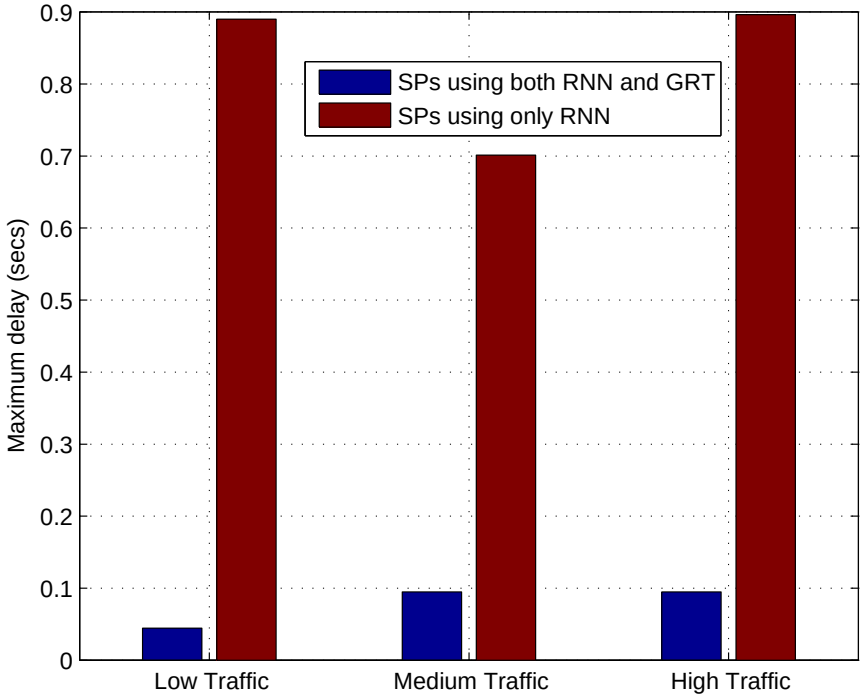

Figure 11: Routing SPs with the Group routing table: Maximum Delay.

proposed connection sharing mechanism can give a much more efficient performance.

The continuous improvement being made in the physical layer as regards the speed of the links means that the performances of different routing algorithms will become closer. Therefore, the efficiency of the algorithms should become more important. Also, these improvements has allowed for the possibility of the connection sharing mechanism that we have proposed in this work.

In the above implementations, we have not considered failures in the network so this our next focus. We will not seek to investigate CPNs ability to find new paths in the event that the best path fails as this capability has been shown in past works. More crucial to our algorithm is the failure of the initiating node or if it becomes unreachable by the nodes sharing its connection. An easy solution is to have special acknowledgements for packets moving towards their second destination in order to share connection. This acknowledgement will be sent by the initiating node once it receives such packets or by any other node in the area if temporary path recording is used. When a node does not receive such acknowledgements, it assumes the initiating node has become unreachable and initiates its own connection. This approach will be ineffective when the most recent path update sent to the node sharing the connection is such that the node can source route packets without sending them to the initiator. In such situations, to continuously check for the availability of the initiating node, 'probe' packets must be sent towards the initiating node.

For the fault-tolerant demonstration of our routing algorithm we will perform experiments on an actual CPN test-bed.

References

[1] C. S. Inc., *Internetworking Technologies Handbook, Third Edition*, 4th ed. Cisco Press, 2003.

[2] J. M. McQuillan and D. C. Walden, "The ARPA network design decisions," *Computer Networks (1976)*, vol. 1, no. 5, pp. 243–289, 1977.

[3] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[4] J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET," *Communications, IEEE Transactions on*, vol. 28, no. 5, pp. 711–719, 1980.

[5] D. Bertsekas and R. Gallager, *Data Networks (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[6] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[7] J. M. McQuillan, G. Falk, and I. Richer, "A review of the development and performance of the ARPANET routing algorithm," *Communications, IEEE Transactions on*, vol. 26, no. 12, pp. 1802–1811, 1978.

[8] C. L. Hedrick, "Routing information protocol," 1988. [Online]. Available: http://dx.doi.org/10.17487/RFC1058

[9] C. Hedrick and L. Bosack, "An introduction to IGRP," *Rutgers-The State University of New Jersey Technical Publication, Laboratory for Computer Science*, 1991.

[10] R. Albrightson, J. Garcia-Luna-Aceves, and J. Boyle, "@miscrekhter2006rfc, title=RFC 4271: Border gateway protocol 4, author=Rekhter, Y and Li, T and Hares, S, year=2006, publisher=Technical report, IBM, Cisco Systems, 2006. Last checked on May 19th ," *Interop 94*, 1994.

[11] D. Medhi, *Network routing: algorithms, protocols, and architectures*. Morgan Kaufmann, 2010.

[12] T. Cegrell, "A routing procedure for the TIDAS message-switching network," *Communications, IEEE Transactions on*, vol. 23, no. 6, pp. 575–585, 1975.

[13] K. G. Shin and M.-S. Chen, "Performance analysis of distributed routing strategies free of ping-pong-type looping," *Computers, IEEE Transactions on*, vol. 100, no. 2, pp. 129–137, 1987.

[14] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)," Tech. Rep., 2005.

[15] J. J. Garcia-Lunes-Aceves, "Loop-free routing using diffusing computations," *IEEE/ACM Transactions on Networking (TON)*, vol. 1, no. 1, pp. 130–141, 1993.

[16] E. W. Dijkstra and C. S. Scholten, "Termination detection for diffusing computations," *Information Processing Letters*, vol. 11, no. 1, pp. 1–4, 1980.

[17] P. M. Merlin and A. Segall, "A failsafe distributed routing protocol," *Communications, IEEE Transactions on*, vol. 27, no. 9, pp. 1280–1287, 1979.

[18] J. M. Jaffe and F. H. Moss, "A responsive distributed routing algorithm for computer networks," *Communications, IEEE Transactions on*, vol. 30, no. 7, pp. 1758–1762, 1982.

[19] J. G. Garcia-Luna-Aceves *et al.*, "Distributed routing with labeled distances," in *INFOCOM'92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*. IEEE, 1992, pp. 633–643.

[20] G. DAngelo, M. DEmidio, and D. Frigioni, "A loop-free shortest-path routing algorithm for dynamic networks," *Theoretical Computer Science*, vol. 516, pp. 1–19, 2014.

[21] J. Chroboczek, "The BABEL routing protocol, rfc 6126," 2011.

[22] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM computer communication review*, vol. 24, no. 4. ACM, 1994, pp. 234–244.

[23] A. Khanna and J. Zinky, "The revised ARPANET routing metric," *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 4, pp. 45–56, 1989.

[24] J. Moy, "RFC 2178: OSPF version 2," *IETF, Jul*, 1997.

[25] D. Oran, "RFC 1142: OSI IS-IS intra-domain routing protocol," 1990.

[26] J. M. Spinelli, "Broadcasting topology and routing information in computer networks," DTIC Document, Tech. Rep., 1985.

[27] E. C. Rosen, "The updating protocol of ARPANET's new routing algorithm," *Computer Networks (1976)*, vol. 4, no. 1, pp. 11–19, 1980.

[28] R. Perlman, "Fault-tolerant broadcast of routing information," *Computer Networks (1976)*, vol. 7, no. 6, pp. 395–405, 1983.

[29] J. M. Spinelli and R. G. Gallager, "Event driven topology broadcast without sequence numbers," *Communications, IEEE Transactions on*, vol. 37, no. 5, pp. 468–474, 1989.

[30] J. Garcia-Luna-Aceves and J. Behrens, "Distributed, scalable routing based on vectors of link states," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 8, pp. 1383–1395, 1995.

[31] J. Garcia-Luna-Aceves and M. Spohn, "Scalable link-state internet routing," in *Network Protocols, 1998. Proceedings. Sixth International Conference on.* IEEE, 1998, pp. 52–61.

[32] J. Hagouel, "Issues in routing for large and dynamic networks," Ph.D. dissertation, New York, NY, USA, 1983, aAI8327222.

[33] P. A. Humblet *et al.*, "Another adaptive distributed shortest path algorithm," *IEEE transactions on communications*, vol. 39, no. 6, pp. 995–1003, 1991.

[34] J. J. Garcia-Luna-Aceves and S. Murthy, "A path-finding algorithm for loop-free routing," *IEEE/ACM Trans. Netw.*, vol. 5, no. 1, pp. 148–160, Feb. 1997. [Online]. Available: http://dx.doi.org/10.1109/90.554729

[35] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves, "A loop-free extended bellman-ford routing protocol without bouncing effect," in *Symposium Proceedings on Communications Architectures &Amp; Protocols*, ser. SIGCOMM '89. New York, NY, USA: ACM, 1989, pp. 224–236. [Online]. Available: http://doi.acm.org/10.1145/75246.75269

[36] S. Murthy and J. Garcia-Luna-Aceves, "A more efficient path-finding algorithm," in *Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on*, vol. 1, Oct 1994, pp. 229–233 vol.1.

[37] H. F. Wedde and M. Farooq, "A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks," *Journal of Systems Architecture*, vol. 52, no. 89, pp. 461 – 484, 2006, nature-Inspired Applications and Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762106000178

[38] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," *Advances in neural information processing systems*, pp. 671–671, 1994.

[39] S. Choi and D.-Y. Yeung, "Predictive q-routing: A memory-based reinforcement learning approach to adaptive tra c control," *Advances in Neural Information Processing Systems*, vol. 8, pp. 945–951, 1996.

[40] S. Khodayari and M. J. Yazdanpanah, "Network routing based on reinforcement learning in dynamically changing networks," in *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on.* IEEE, 2005, pp. 5–pp.

[41] D. Ouzecki and D. Jevtic, "Reinforcement learning as adaptive network routing of mobile agents," in *MIPRO, 2010 Proceedings of the 33rd International Convention.* IEEE, 2010, pp. 479–484.

[42] S. Kumar and R. Miikkulainen, "Dual reinforcement q-routing: An on-line adaptive routing algorithm," in *Artificial neural networks in engineering*, 1997.

[43] A. Costa, "Analytic modelling of agent-based network routing algorithms," Ph.D. dissertation, The University of Adelaide, 2002. [Online]. Available: http://hdl.handle.net/2440/37738

[44] M. Heusse, D. Snyers, S. Guerin, and P. Kuntz, "Adaptive agent-driven routing and load balancing in communication networks," *Advances in Complex Systems*, vol. 1, no. 02n03, pp. 237–254, 1998.

[45] G. Di Caro and M. Dorigo, "Antnet: Distributed stigmergetic control for communications networks," *Journal of Artificial Intelligence Research*, pp. 317–365, 1998.

[46] G. Di Caro *et al.*, "Ant colony optimization and its application to adaptive routing in telecommunication networks," Ph.D. dissertation, PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[47] D. Subramanian, P. Druschel, and J. Chen, "Ants and reinforcement learning: A case study in routing in dynamic networks," in *IJCAI (2).* Citeseer, 1997, pp. 832–839.

[48] H. Wedde, M. Farooq, and Y. Zhang, "Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior," in *Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science, M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Sttzle, Eds. Springer Berlin Heidelberg, 2004, vol. 3172, pp. 83–94.

[49] R. Schoonderwoerd, O. Holland, and J. Bruten, "Ant-like agents for load balancing in telecommunications networks," in *Proceedings of the First International Conference on Autonomous Agents*, ser. AGENTS '97. New York, NY, USA: ACM, 1997, pp. 209–216. [Online]. Available: http://doi.acm.org/10.1145/267658.267718

[50] J. Chen, P. Druschel, and D. Subramanian, "A simple, practical distributed multi-path routing algorithm," *TR98-320, Department of Computer Science, Rice University*, 1998.

[51] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[52] M. Munetomo, "An adaptive network routing algorithm employing path genetic operators," in *Proc. of the Seventh Inter. Conf. on Genetic Algorithms, 1997*, 1997.

[53] M. Munetomo, Y. Takai, and Y. Sato, "A migration scheme for the genetic adaptive routing algorithm," in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, vol. 3. IEEE, 1998, pp. 2774–2779.

[54] X. Cheng and Y.-B. Hou, "A study of genetic ant routing algorithm," in *Machine Learning and Cybernetics, 2003 International Conference on*, vol. 4. IEEE, 2003, pp. 2041–2045.

[55] S. Liang, A. N. Zincir-Heywood, and M. I. Heywood, "Intelligent packets for dynamic network routing using distributed genetic algorithm." in *GECCO*, 2002, pp. 88–96.

[56] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," *Communications, IEEE Transactions on*, vol. 25, no. 1, pp. 73–85, 1977.

[57] S. Vutukury and J. Garcia-Luna-Aceves, *A simple approximation to minimum-delay routing.* ACM, 1999, vol. 29, no. 4.

[58] H. Geng, X. Yin, X. Shi, and Z. Wang, "Mlsa: A link-state multipath routing algorithm," in *Computers and Communications (ISCC), 2013 IEEE Symposium on.* IEEE, 2013, pp. 000 330–000 335.

[59] S. Vutukury and J. J. Garcia-Luna-Aceves, "Mdva: A distance-vector multipath routing protocol," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 2001, pp. 557–564.

[60] S. Vutukury and J. Garcia-Luna-Aceves, "An algorithm for multipath computation using distance-vectors with predecessor information," in *Computer Communications and Networks, 1999. Proceedings. Eight International Conference on.* IEEE, 1999, pp. 534–539.

[61] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 2, pp. 223–259, 2006.

[62] E. Gelenbe, E. Ngai, and P. Yadav, "Routing of high-priority packets in wireless sensor networks," *IEEE Second International Conference on Computer and Network Technology, IEEE*, 2010.

[63] E. Gelenbe, "Sensible decisions based on qos," *Computational management science*, vol. 1, no. 1, pp. 1–14, 2003.

[64] ——, "Users and services in intelligent networks," in *Technologies for Advanced Heterogeneous Networks.* Springer Berlin Heidelberg, 2005, pp. 30–45.

[65] ——, "Towards autonomic networks," in *Applications and the Internet, 2006. SAINT 2006. International Symposium on.* IEEE, 2006, pp. 6–pp.

[66] ——, "A diffusion model for packet travel time in a random multihop medium," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 2, p. 10, 2007.

[67] ——, "Search in unknown random environments," *Physical Review E*, vol. 82, no. 6, p. 061112, 2010.

[68] S. E. Gelenbe, "Cognitive packet network," October 2004, US Patent 6,804,201.

[69] E. Gelenbe, "Steps toward self-aware networks," *Communications of the ACM*, vol. 52, no. 7, pp. 66–75, 2009.

[70] ——, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[71] A. Desmet and E. Gelenbe, *Information Sciences and Systems 2014: Proceedings of the 29th International Symposium on Computer and Information Sciences.* Cham: Springer International Publishing, 2014, ch. A Parametric Study of CPN's Convergence Process, pp. 13–20.

[72] E. Gelenbe, R. Lent, and A. Nunez, "Self-aware networks and qos," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1478–1489, 2004.

[73] E. Gelenbe, R. Lent, M. Gellman, P. Liu, and P. Su, "CPN and qos driven smart routing in wired and wireless networks," in *Performance Tools and Applications to Networked Systems, Revised Tutorial Lectures [from MASCOTS 2003]*, ser. Lecture Notes in Computer Science, M. Calzarossa and E. Gelenbe, Eds., vol. 2965. Springer, 2003, pp. 68–87.

[74] E. Gelenbe and Z. Kazhmaganbetova, "Cognitive packet network for bilateral asymmetric connections," *IEEE Trans. Industrial Informatics*, vol. 10, no. 3, pp. 1717–1725, 2014. [Online]. Available: http://dx.doi.org/10.1109/TII.2014.2321740

[75] L. Wang and E. Gelenbe, "Demonstrating voice over an autonomic network," in *2015 IEEE International Conference on Autonomic Computing, Grenoble, France, July 7-10, 2015*. IEEE, 2015, pp. 139–140. [Online]. Available: http://dx.doi.org/10.1109/ICAC.2015.14

[76] ——, "Real-time traffic over the cognitive packet network," in *Proceedings of 23rd International Science Conference on Computer Networks CN2016*, Poland, June 2016.

[77] O. Brun, L. Wang, and E. Gelenbe, "Data driven self-managing routing in intercontinental overlay networks," *Accepted for publication in the IEEE J. on Selected Areas in Communications, 2016.*

[78] G. Gorbil, D. Garcia Perez, and E. Huedo Cuesta, "Principles of pervasive cloud monitoring," in *Information Sciences and Systems 2014*, T. Czachorski, E. Gelenbe, and R. Lent, Eds. Springer, sep 2014, pp. 117–124.

[79] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *Cloud Computing, IEEE Transactions on*, 2015.

[80] ——, "Experiments with smart workload allocation to cloud servers," in *Network Cloud Computing and Applications (NCCA), 2015 IEEE Fourth Symposium on*, June 2015, pp. 31–35.

[81] E. Gelenbe and M. Gellman, "Can routing oscillations be good? The benefits of route-switching in self-aware networks," in *15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2007), October 24-26, 2007, Istanbul, Turkey*. IEEE Computer Society, 2007, pp. 343–352. [Online]. Available: http://dx.doi.org/10.1109/MASCOTS.2007.13

[82] ——, "Oscillations in a bio-inspired routing algorithm," in *IEEE 4th International Conference on Mobile Adhoc and Sensor Systems, MASS 2007, 8-11 October 2007, Pisa, Italy*. IEEE Computer Society, 2007, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/MOBHOC.2007.4428681

[83] E. Gelenbe and C. Morfopoulou, "A framework for energy-aware routing in packet networks," *The Computer Journal*, vol. 54, no. 6, pp. 850–859, 2011.

[84] E. Gelenbe and S. Silvestri, "Optimisation of power consumption in wired packet networks," in *Quality of Service in Heterogeneous Networks*. Springer Berlin Heidelberg, 2009, pp. 717–729.

[85] O. H. Abdelrahman and E. Gelenbe, "Time and energy in team-based search," *Physical Review E*, vol. 87, no. 3, p. 032125, 2013.

[86] E. Gelenbe and A. Labed, "G-networks with multiple classes of signals and positive customers," *European Journal of Operational Research*, vol. 108, no. 2, pp. 293–305, 1998.

[87] E. Gelenbe and R. R. Muntz, "Probabilistic models of computer systemspart i (exact results)," *Acta Informatica*, vol. 7, no. 1, pp. 35–60, 1976.

[88] E. Gelenbe, "Probabilistic models of computer systems," *Acta Informatica*, vol. 12, no. 4, pp. 285–303, 1979. [Online]. Available: http://dx.doi.org/10.1007/BF00268317

[89] E. Gelenbe and C. Morfopoulou, "Power savings in packet networks via optimised routing," *Mobile Networks and Applications*, vol. 17, no. 1, pp. 152–159, 2012.

[90] E. Gelenbe and R. Lent, "Trade-offs between energy and quality of service," in *Sustainable Internet and ICT for Sustainability (SustainIT), 2012*. IEEE, 2012, pp. 1–5.

[91] E. Gelenbe, R. Lent, and M. Douratsos, "Choosing a local or remote cloud," in *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*. IEEE, 2012, pp. 25–30.

[92] E. Gelenbe and R. Lent, "Energy–QoS trade-offs in mobile service selection," *Future Internet*, vol. 5, no. 2, pp. 128–139, 2013.

[93] E. Gelenbe and T. Mahmoodi, "Distributed Energy-Aware Routing of Traffic," in *Proceedings of the 26th International Symposium on Computer and Information Sciences (ISCIS'11)*, London, UK, 26-27 September 2011.

[94] ——, "Energy-aware routing in the cognitive packet network," *ENERGY*, pp. 7–12, 2011.

[95] E. Gelenbe, "A sensor node with energy harvesting," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 2, pp. 37–39, 2014.

[96] E. Gelenbe, D. Gesbert, D. Gunduz, H. Kulah, and E. Uysal-Biyikoglu, "Energy harvesting communication networks: Optimization and demonstration (the E-CROPS project)," in *Digital Communications-Green ICT (TIWDC), 2013 24th Tyrrhenian International Workshop on*. IEEE, 2013, pp. 1–6.

[97] E. Gelenbe, "Energy packet networks: smart electricity storage to meet surges in demand," in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012, pp. 1–7.

[98] ——, "Energy packet networks: ICT based energy allocation and storage," in *Green Communications and Networking*. Springer Berlin Heidelberg, 2012, pp. 186–195.

[99] ——, "Energy packet networks: adaptive energy management for the cloud," in *Proceedings of the 2nd International Workshop on Cloud Computing Platforms (CloudCP'12)*. Bern, Switzerland: ACM, New York, NY, USA, 10 April 2012, pp. 1–5.

[100] ——, "Keeping viruses under control," in *Computer and Information Sciences-ISCIS 2005*. Springer Berlin Heidelberg, 2005, pp. 304–311.

[101] G. Sakellari and E. Gelenbe, "Demonstrating cognitive packet network resilience to worm attacks," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 636–638.

[102] ——, "Adaptive Resilience of the Cognitive Packet Network in the presence of Network Worms," in *Proceedings of the NATO Symposium on C3I for Crisis, Emergency and Consequence Management*. Bucharest, Romania: NATO Research & Technology Organisation, 11-12 May 2009, pp. 16:1–16:14.

[103] E. Gelenbe, M. Gellman, and G. Loukas, "An autonomic approach to denial of service defence," in *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*. IEEE, 2005, pp. 537–541.

[104] G. Gorbil and E. Gelenbe, "Opportunistic communications for emergency support systems," *Procedia Computer Science*, vol. 5, pp. 39–47, 2011.

[105] ——, "Disruption tolerant communications for large scale emergency evacuation," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, 2013, pp. 540–546.

[106] ——, "Resilience and security of opportunistic communications for emergency evacuation," in *Proceedings of the 7th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. ACM, 2012, pp. 115–124.

[107] ——, "Resilient emergency evacuation using opportunistic communications," in *Computer and Information Sciences III*. Springer, 2013, pp. 249–257.

[108] E. Gelenbe, G. Görbil, D. Tzovaras, S. Liebergeld, D. Garcia, M. Baltatu, and G. Lyberopoulos, "NEMESYS: Enhanced network security for seamless service provisioning in the smart mobile ecosystem," in *Information Sciences and Systems 2013*. Springer International Publishing, 2013, pp. 369–378.

[109] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks performance evaluation and optimization," *Computer Networks (1976)*, vol. 1, no. 3, pp. 155–174, 1977.

[110] U. Halici, "Reinforcement learning with internal expectation for the random neural network," *European Journal of Operational Research*, vol. 126, no. 2, pp. 288–307, 2000.

[111] G. Sakellari, "The cognitive packet network: a survey," *The Computer Journal*, p. bxp053, 2009.

[112] H. Bi, A. Desmet, and E. Gelenbe, "Routing emergency evacuees with cognitive packet networks," in *Information Sciences and Systems 2013*. Springer, 2013, pp. 295–303.

[113] A. Desmet and E. Gelenbe, "Capacity based evacuation with dynamic exit signs," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*. IEEE, 2014, pp. 332–337.

[114] O. J. Akinwande, H. Bi, and E. Gelenbe, "Managing crowds in hazards with dynamic grouping," *Access, IEEE*, vol. 3, pp. 1060–1070, 2015.

[115] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European simulation multiconference (ESM2001)*, vol. 9, no. S 185. sn, 2001, p. 65.

[116] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, october 2011.