The Computer Journal Advance Access published June 5, 2009

doi:10.1093/comjnl/bxp053

The Cognitive Packet Network: A Survey

Georgia Sakellari[1,*]

[1]*Intelligent Systems and Networks Group, Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2BT UK*
**Corresponding author: g.sakellari@imperial.ac.uk*

Current and future multimedia networks require connections under specific quality of service (QoS) constraints which can no longer be provided by the best-effort Internet. Therefore, 'smarter' networks have been proposed in order to cover this need. The cognitive packet network (CPN) is a routing protocol that provides QoS-driven routing and performs self-improvement in a distributed manner, by learning from the experience of special packets, which gather on-line QoS measurements and discover new routes. The CPN was first introduced in 1999 and has been used in several applications since then. Here we provide a comprehensive survey of its variations, applications and experimental performance evaluations.

Keywords: cognitive packet networks; random neural network; quality of service; routing protocol; self-aware networks

Received 10 October 2008; revised 7 May 2009
Handling editor: Taskin Kocak

1. INTRODUCTION

The need to provide quality of service (QoS) to the users of current and future multimedia networks has led to the growth of 'smarter' networks that use QoS-driven architectures to provide a more stable and more reliable environment and guarantee packet delivery under specific and user-defined QoS constraints. Such an architecture is the cognitive packet network (CPN), a packet routing protocol designed to perform self-improvement and address QoS by using adaptive techniques based on on-line measurements, first introduced in [1] and inspired by an earlier approach on learning agents [2].

Contrary to conventional routing protocol mechanisms, in CPN it is the users rather than the nodes that are given control of the routing. Each user can specify the QoS criteria based on which data will be routed in the network, since the routing algorithm is directly related to the QoS desired by the end-user. Different users can have different QoS goals depending on the application and the service they want to use. This is done at the software level, and therefore CPN reduces the complexity of the routes, which in turn significantly reduces the cost of the network. The CPN is a distributed protocol, where the routing decisions are made at each node of the network and are based on adaptive learning techniques such as random neural networks (RNNs) with reinforcement learning (RL).

The remainder of this paper is organized as follows. Section 2 describes the functionality of the CPN in detail, while Section 3 describes the learning algorithms used to optimize user-specified QoS goals and to make routing decisions. Section 4 explains how the CPN collaborates with IP applications and other protocols and Section 5 reviews the research undertaken in its various aspects and describes some of its applications and enhancements. In Section 6 we review experimental evaluation results, and we conclude in Section 7 with a discussion on the role that CPN can play in the intelligent networks of the near future.

2. THE OPERATION OF CPN

CPN is an adaptive packet routing protocol with enhanced monitoring and self-improvement capabilities that address QoS by using adaptive techniques based on on-line measurements [3–7]. It is a distributed protocol with which users, or the network itself, declare their QoS requirements (QoS goals) such as minimum delay, maximum bandwidth, minimum cost etc. It is designed to perform self-improvement by learning from the experience of smart packets (SPs).

It makes use of three types of packets:

- SPs (also known as cognitive packets), for discovery;

- source routed dumb packets (DPs) to carry the payload; and
- acknowledgement (ACK) packets to bring back information that has been discovered by SPs, and is used in nodes to train neural networks.

SPs are generated by a user that requests to create a path to some CPN node or to discover parts of the network state, including location of certain fixed or mobile nodes, power levels at nodes, topology, paths and their QoS metrics. Their role is to explore the network and discover the 'best QoS routes' for each source–destination pair in the network according to goals assigned from the users. At each hop SPs are routed according to the experiences of previous packets with the same goals and the same destination. The decisions of the SPs are based on a learning algorithm. In order to explore all possible routes, at some hops, each SP makes a random routing decision, with a small probability (usually 5%). To avoid overburdening the system with unsuccessful requests or packets which are in effect lost, all packets have a life-time constraint based on the number of nodes they have visited. The term 'goal' is used instead of 'QoS specifications' to emphasize the fact that there are no QoS guarantees and that the CPN provides a best effort service [8]. The goal is a QoS criterion that the SPs try to achieve such as minimum delay, maximum bandwidth, minimum packet loss, minimum variance of the packet delay, maximum security level in a path, minimum power consumption in a wireless node or a weighted combination of these.

In the CPN, routers have restricted functionality. The tasks of the nodes include receiving packets from several ports, storing them into an input buffer and transmitting packets to other nodes via output buffers based on their priority discipline. In addition, routers store information they receive from SPs in special short-term memory stores, referred to as mailboxes (MBs) [3,4]. In a node, there may be more than one MB, since each one may be used by a certain class of SPs that have common characteristics, such as QoS goals or destination. SPs read their MBs and use the information contained in them to execute their code via the node, update their MBs in the node and decide their next hop movement. Routers are also responsible for discarding packets that have exceeded the allowed 'time-out' value.

Each SP or DP in a CPN contains all the usual fields one would find in usual TCP/IP packets, plus some fields that contain the code needed to interact with the nodes they visit [4,7]. Examples include reading the MB, computing the goal, running the adaptive learning algorithm, making the next hop decision. The learning algorithms used by the SPs are analysed in Section 3. An SP stores the route it follows and also keeps 'timestamp' information regarding the local time at which it visited a node.

When it arrives at its destination, an ACK packet is generated and the routing and measurement information collected by the SP are transferred to the ACK. The ACK packet will follow the reverse path back to the source, of the one followed by the corresponding SP (using a loop-removal algorithm to avoid circuits). The reverse path is not necessary the shortest one or the one resulting in the best goal satisfaction. As the ACK follows its route, it deposits QoS information in the MBs of the nodes it visits. At the source, the route carried by an ACK, along with its QoS data is cached in a table, the dumb packet route repository (DPRR).

The last path inserted in the DPRR is the one that the DPs will follow. DPs are the packets that carry the payload of a particular connection. Since the path discovery process is continuous, DPs select the most recently discovered best route from the source to the destination, which is a result of the SPs of the same QoS class previously sent in the network. DPs can also collect time information during their trip through the network, which can be brought back to a source node by a dumb acknowledgement (DACK) packet and can update the MBs of the nodes traversed.

3. LEARNING AND DECISION ALGORITHMS

Several algorithms have been used in the CPN as learning and decision techniques for SPs to find satisfactory routes from source to destination based on the desired goals. The simplest algorithm used is the bang-bang algorithm while other more sophisticated algorithms based on RNN have also been used, such as the learning feedforward algorithm and the RL algorithm. Finally, genetic algorithms (GAs) have recently been tested with CPN.

3.1. Bang-bang algorithm

This algorithm makes use of the most recently available data stored in the MBs of the present node and makes the decision that requires the lowest cost or highest gain. To illustrate how bang-bang algorithm works we assume that our goal is a weighted combination of packet loss (L) and delay (W). This means that the QoS Goal will be calculated as: $G = aW + bL$. Each time a decision has to be made the SPs read the MB of the node and estimate a running average of the delay to destination D of the form: $W_\mathrm{d} = \alpha W_\mathrm{d} + (1 - \alpha)V_\mathrm{d}$, where V_d is the most recent value of the delay for the particular destination. Similarly, information is also collected regarding the loss and a running average for it, l_d, is computed as well [1]. In order for the SP to take a decision, it makes an evaluation of the average delay and loss it would require to the destination D_d and L_d respectively. Note that D_d and L_d are *a priori* information since their evaluation requires the knowledge of the grid structure by the SPs [2]. If $L_\mathrm{d} < l_\mathrm{d}$ or $D_\mathrm{d} < W_\mathrm{d}$, then the SPs select at random an output link since they assessed that better paths to the destination exist while if the aforementioned conditions are not valid, then the SP takes the direction followed by an SP with exactly the same characteristics (e.g. same destination D and QoS class). A significant disadvantage of the bang-bang algorithm is the fact that it uses *a priori* information. Later algorithms addressed this issue.

3.2. RNN-based algorithms

The RNN is a biologically inspired neural network model, which is characterized by the existence of positive (excitation) and negative (inhibition) signals in the form of spikes of unit amplitude that circulate among nodes and alter the potential of the neurons. Each neuron can be connected to another neuron and each connection is characterized by an excitatory or inhibitory weight [2]. It was originally introduced in [9] and extended in [10–15]. Although the RNN model was initially inspired by biophysical neural networks, it has been successfully applied in many areas such as associative memory [16–18], image processing [19–21], texture generation [22,23], video QoS and compression [19,24–29], as well as task assignment [30] and resource allocation [31], and has inspired the use of negative customers in queuing networks, which has led to G-networks [32–36].

Apart from the CPN, RNN-based routing is employed in [37], where RNN is used to learn the characteristics of the minefield by fusing data from multiple sensors and navigating autonomous vehicles, robots or agents. The predecessor of CPN was the routing of agents described in [2]. Here agents are routed according to predefined goals, such as traversing a dangerous metropolitan grid safely and rapidly, using the RNN to learn from their own observations and from the experience of other agents with whom they exchange information. The RNN is also used to control the movement of agents within a realistic live scene of a simulator [38].

In a RNN, the state q_i of the ith neuron, which represents the probability that the ith neuron is excited, satisfies the following system of nonlinear equations:

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)}, \tag{1}$$

where

$$\lambda^+(i) = \sum_j q_j w^+_{ji} + \Lambda_i, \tag{2}$$

$$\lambda_-(i) = \sum_j q_j w^+_{ji} + \lambda_i, \tag{3}$$

$$r(i) = \sum_j \left[w^+_{ij} + w^-_{ij} \right], \tag{4}$$

in which w^+_{ji} is the rate at which neuron j sends 'excitation spikes' to neuron i when j is excited, w^-_{ji} is the rate at which neuron j sends 'inhibition spikes' to neuron i when j is excited and $r(i)$ is the total firing rate from the neuron i. Λ_i and λ_i are the constant rates of the external positive and negative signal arrivals, respectively, which follow stationary Poisson distributions. For an N neuron network, the network parameters are these N by N 'weight matrices' $W^+ = \{w^+(i, j)\}$ and $W^- = \{w^-(i, j)\}$, which need to be 'learned' from input data. The weights w^+_{ij} and w^-_{ji} as well as Λ_i and λ_i are assumed to be known for every i, j.

Several learning techniques have been proposed for the RNN. Hebbian learning was tested at the early stages of the CPN development and was shown to be inefficient and slow [8]. Other algorithms include feedforward learning RNN with the use of a gradient descent quadratic error function, as well as RL. These two algorithms are briefly described next.

3.2.1. *Learning feedforward random neural networks*

These networks update their weights W^+ and W^- based on a gradient descent quadratic error function, which is used to minimize the error observed when a new input-output pair of training data is introduced. In CPN, recent samples of the goals (e.g. packet loss and transit delay) are used to train the learning feedforward random neural networks. When an update of the weights has taken place, the CPN provides its own estimate of the cost or reward incurred in each output direction and a decision is made based on the direction associated with the highest reward [2]. This algorithm is computationally less efficient than the bang-bang and the RL algorithm, because it requires computation at every step and also because mathematical analysis of the model leads to a 'back-propagation type algorithm' that requires the solution of a linear and a nonlinear system of N equations each time [10].

3.2.2. *Reinforcement learning random neural networks*

This is the algorithm that eventually prevailed in the implementations of the CPN. The RL algorithm was initially based on an RL algorithm with internal expectation for the RNN that was developed for use in maze navigation [39]. The arrival of an SP triggers the execution of the RL algorithm. More specifically, each router stores a specific RNN for each QoS class and for each active source–destination pair. Each RNN node, which represents the decision to choose a given output link for a SP, has as many neurons as the possible outgoing links [5]. Decisions are taken by selecting the output link j for which the corresponding neuron is the most excited, i.e. $q_i \leq q_j$ for all $i = 1, \ldots, N$, where N is the number of neurons (possible outgoing links). The state q_i of the ith neuron in the network represents the probability that the ith neuron is excited and thus the probability that the ith outgoing link will be selected for the SP's routing.

In CPN, the RL process changes the weights to reward or punish a neuron according to the level of goal satisfaction measured on the corresponding output, and thus according to whether the SP succeeded or failed to achieve its QoS goal. Each QoS class, for each source–destination pair, has a QoS Goal G, which expresses a function to be minimized. The level of goal satisfaction is expressed by a reward. Given some goal G that a packet has to minimize, the reward R is formulated simply as $R = 1/G$. The RNN weights are updated based on the following threshold T:

$$T_k = \alpha T_{k-1} + (1-\alpha)R_k, \quad (5)$$

where R_k, $k = 1, 2, \ldots$, are successive measured values of reward R and α is some constant ($0 < \alpha < 1$) that is used to tune the responsiveness of the algorithm: for instance, $\alpha = 0.8$ means that on the average five past values of R are being taken into account. Neurons are rewarded or punished based on the difference between the current reward R_k and the last threshold T_{k-1}. Hence, if the most recent value of the reward R_k, which corresponds to neuron j, is larger than the previous value of the threshold T_{k-1}, then the excitatory weights going into that neuron are significantly increased (in order to reward it for its new success), and the inhibitory weights leading to other neurons are slightly increased. If the new reward is not greater than the previous threshold, then all excitatory weights leading to all neurons are moderately increased, except for the previous winner, and the inhibitory weights leading to the previous winning neuron are significantly increased, in order to punish it for not being very successful this time. Thus, if r_i is the firing rate before the update takes place for every neuron i it is given by

$$r_i = \sum_{m=1}^{n}[w^+(i,m) + w^-(i,m)]. \quad (6)$$

First T_{k-1} is computed and then the network weights are updated for all neurons $i \neq j$ as follows:

- If $T_{k-1} \leq R_k$
  - $w^+(i,j) \leftarrow w^+(i,j) + R_k$,
  - $w^-(i,l) \leftarrow w^-(i,l) + \dfrac{R_k}{n-2}$, if $l \neq j$.
- Else
  - $w^+(i,l) \leftarrow w^+(i,l) + \dfrac{R_k}{n-2}$, if $l \neq j$,
  - $w^-(i,j) \leftarrow w^-(i,j) + R_k$.

To avoid large weights, which would lead to numerical difficulties when running the algorithms, and since the relative size of the weights of the RNN, rather than the actual values, determine the state of the neural network, the weights can be re-normalized by carrying out the following operations. First, for each i r_i^* is computed by

$$r_i^* = \sum_{m=1}^{n}[w^+(i,m) + w^-(i,m)], \quad (7)$$

and then the weights are re-normalized with:

$$w^+(i,j) \leftarrow w^+(i,j) * \frac{r_i}{r_i^*}, \quad (8)$$

$$w^-(i,j) \leftarrow w^-(i,j) * \frac{r_i}{r_i^*}. \quad (9)$$

Having computed the new values of the weights the nonlinear system of equations (1–4) can be solved in order to obtain the q_i and choose the most excited, which will provide the link that the SP will follow. This procedure is repeated for each SP, QoS class and source–destination pair.

3.3. Genetic algorithms

Recently, GAs have been applied in the CPN for adaptive route discovery. The authors of [6,40] use GAs to modify, filter and combine the paths already found by the SPs in order to generate new undiscovered but valid source–destination paths and select the most advantageous ones. In the CPN, the populations of individuals (or genotypes), which in GAs are the possible solution to the problem being investigated, are the routes from a source to a destination, represented as a list of the nodes' names in a particular route. Accordingly, the fitness function, which estimates the usefulness of each individual to the solution of the problem, is the measured or evaluated QoS or goal value of a given path. Therefore, the CPN protocol is interested in paths that have the best fitness and can use the crossover operator of the GA to combine paths already discovered and create new valid undiscovered paths.

When SPs discover valid routes, these are returned back to the source via ACK packets to become individuals. The individuals' population is stored at the source in a repository of finite length called stack, where individuals are placed in order of ascending goal value. The fittest individual is deposited first in the stack. New valid paths are generated from the GA using the crossover operator. Note that paths can only be combined if they contain the same source and destination as well as same intermediate nodes. Since we are only interested in the fittest paths, only paths with small goal values are combined while their resulting goal value of a new path is evaluated. Each time a DP needs to be forwarded to the destination the fittest path is taken from the top of the stack and is used as the source route of the particular DP.

4. INTEGRATION WITH IP APPLICATIONS AND OTHER PROTOCOLS

In CPN, both SPs and DPs contain all the usual fields one would find in usual TCP/IP packets plus some fields containing the code needed to interact with the nodes they visit [7]. When this additional information is discarded, the smart and DPs can be viewed as ordinary TCP/IP packets. However, ACKs are specific to the CPN framework, even though they too are derived from TCP/IP packets. Since the routing decisions are based on the code carried by the CPN packets, the nodes in a CPN network do not need the routing tables or routing algorithms, which are typically stored in TCP/IP routers, and therefore can be implemented with much simpler and less costly hardware and software than conventional TCP/IP routers. The CPN is a replacement for the IP layer, since it introduces a different manner to handle routing.

Integrating with IP applications and end-hosts is an important characteristic of any networking protocol. Since most network environments use IP as standard, it is important that CPN is compatible with it [41]. On operating systems such as Linux, which is used in most routers, a great majority of applications have their source code available. When this is the case, adding CPN support can amount to a few very simple changes to the code that creates and connects a socket [41]. Since the size of the address in the CPN is kept at 32 bits, the CPN and IP addresses are completely compatible, making it very simple to convert an IP application to a CPN one. Even for applications for which the source code is not available, the CPN can still be used. By intercepting the various socket functions and redirecting them to their CPN-specific counterparts, an application which has been written to use IP can use the CPN without any requirements from the application's side.

Another approach to bridging IP with CPN, presented also in [41], is tunnelling. This involves modifying the IP routing table such that before an IP packet is sent out on the wire, it is first encapsulated into a CPN packet. This process requires the designation of a tunnel source and a tunnel edge by an administrator. This way conventional IP packets may tunnel through the CPN to seamlessly operate mixed IP and CPN networks.

By using an overlay network, the benefits of the CPN routing can be introduced into existing networks with low overhead and without modifying their underlying routing mechanisms [41]. Thus, the CPN protocol is fully compatible at its edges with the IP protocol, while internally it offers dynamic routing based on on-line sensing and monitoring. Furthermore, the CPN technology could be used to build private user networks, which are implemented on top of existing TCP/IP, Asynchronous Transfer Mode or other forms of network communication.

5. ENHANCEMENTS AND APPLICATIONS

The CPN has been shown to be effective for a variety of uses, including traffic balancing, power-based routing in mobile *ad hoc* networks, denial of service (DoS) protection and admission control (AC). This section describes some of the applications that make use of the CPN protocol.

5.1. CPN in wireless networks (mobile *AD HOC* CPN)

The CPN architecture has also been implemented in mobile networks [42]. However, in mobile *ad hoc* environments, node neighbours may change unexpectedly because of the link breaks caused by changes in the position of the nodes or the environment. To deal with such situations, the SPs of the *ad hoc* CPN (AHCPN) allow the use of broadcasts in addition to unicast decisions to continue exploring the network whenever they reach a node with no information, or stale information about its neighbourhood [42].

The use of broadcasts allows SPs to reach all neighbouring nodes without requiring the explicit selection of one. It also avoids the need of an explicit neighbour discovery process (e.g. by a periodic broadcast of hello packets) that may unnecessarily use network resources. AHCPN assumes that there is an insufficient knowledge about the neighbourhood if the number of known neighbours is less than two when the node is the source of the traffic, or three otherwise [43].

Neighbours receiving the packet repeat this decision process, and thus the SP may use a broadcast again or select a given neighbour to continue network exploration. It is interesting to note that, in general, unicast decisions are preferred by SPs [44]. Unicasts have the advantage of being more reliable and are able to restrict the number of nodes involved in the route discovery. Broadcasts, on the other hand, are prone to collide, although a small random delay is introduced before transmitting a broadcast to reduce the probability of collision. AHCPN defines specific goals for wireless environments, which take into account link reliability and energy consumption.

5.2. Traffic engineering with CPN

The use of CPN to achieve traffic engineering (TE) goals is described in [45]. TE allows the internet service providers (ISPs) to optimize resource utilization and network performance. With the CPN technology an ISP is capable of providing value-added IP services, such as QoS, to its customers and perform TE on the network. For the end-user CPN provides QoS at the application layer, based on the delay constraint, and at the service level it redistributes the traffic in a flow-basis over the edge nodes (similar to IntServ and RSVP). Thus, the network minimizes congestion at the link level and minimizes the delay experienced by the packets. The authors of [45] also provide experimental results of how the ISPs can fulfil TE requirements through the use of CPN.

5.3. Hardware implementation of CPN

The use of CPN with the routers currently used in Internet-based network architectures may impose limitations due to higher and higher traffic rates introduced to the networks, while at the same time the processing demand of the packets could significantly increase. Therefore, a hardware implementation of the CPN protocol (a 'CPN router') would be low cost and provide a more efficient processing of the packets. The main problem in the hardware implementation of the CPN is the complexity and the memory requirements of the RNN with RL.

A hardware implementation of the RNN-based routing engine of a CPN network processor chip, called the smart packet processor (SPP), is presented in [46]. The SPP is a dual port device that stores, modifies and interprets the defining characteristics of multiple RNN models. The authors of [46] suggest hardware design improvements over the software implementation, such as the dual access memory, output calculation step and reduced output calculation module, and introduce a modification over the reinforcement learning random neural networks such that the number of weight terms are reduced from $2n^2$ to $2n$, in order to not only save memory, but also simplify the calculations for the steady-state probabilities. Simulations have been conducted in [46] for the functionality of the isolated SPPs design as well as for the multiple SPPs in a networked environment.

Alternative algorithms to the RNN were also proposed in [47,48] in order to keep the benefits of the RNN while being less complex and less resource demanding, and therefore, more suitable for implementation in hardware. The approximated RNN (aRNN) [48] is not a neural network model, but attempts to approximate properties of the RNN while using only simple arithmetic operators such as addition and subtraction, and bitwise shifts. A sensible routing (SR) policy (also called 1-SR), introduced in [49], bases its routing decisions on the expected QoS of a possible outcome and selects a path with a probability that is inversely proportional to the expected QoS metric of that path. In an m-SR policy the inverted expectation of the QoS metric is raised to the power of m in order to increase the probability of selecting the better performing routes. Therefore m-SR shares the RNN's problem of costly divisions, both from the inversions and the probability calculations. However, if $m \to \infty$ the probability of selecting the best route equals 1, which makes the implementation simpler. Thus, by avoiding divisions, the ∞-SR and the approximated RNN offer smaller and faster implementations than the RNN. They also require less memory, and scale better with respect to the number of neighbours. This is beneficial as it makes the CPN suitable for implementation in dedicated hardware for high-speed routers, and also enables it to be used in software-driven devices that lack powerful processors and large memories, such as PDAs or wireless sensor network devices. The authors in [47,48] also propose an architecture for a field programmable gate array-based hardware CPN router. This design demonstrates that the traditional architectural approaches used in high-speed IP routers are applicable to CPN routers, and thus demonstrates the possibility for hybrid IP/CPN routers in hardware.

5.4. CPN in sensor networks

Wireless sensor networks consist of many devices of limited energy, memory and processing capabilities. The author of [50] proposed a simplified CPN-based routing protocol for sensor motes. This scheme uses smart routing in order to minimize the transmission power use and also enables on-line aggregation of data based on the locality of motes. The tinyCPN protocol tracks the success of messages in reaching the sink, the length of the route they discover and the power required.

Similar to the wired version of CPN, in tinyCPN smart messages attempt to find a reliable low-power route to the destination. They are forwarded to the neighbour that has the lowest recorded energy requirements to the sink. When the smart message reaches the sink, a smart acknowledgement message is sent along the reverse route to the source. It updates the routing tables and route power requirements for the sink at each node in the route.

In contrast to the CPN, apart from the end-to-end acknowledgements that are sent when smart messages reach their destinations to update the routing and QoS tables, hop-by-hop acknowledgements are also used every time a node has received a message. Due to the fact that the probability of a message being lost or corrupted in transit is higher in sensor networks, when a node receives a message, it will reply to the previous hop with a hop-by-hop acknowledgement. If no acknowledgement is received, the message can be re-sent. The number of re-sent messages is recorded and is used as a multiplier when determining the hop-by-hop power requirements of the route for smart messages. Whenever a data carrying dumb message is sent, there is a small probability of a smart message also being sent. This ensures that power and route data remain current.

5.5. Defence mechanisms against DoS attacks in CPN

Recognizing the fact that the networks of the near future will feature self-awareness and on-line interaction with users, the authors of [51] investigated the application of defence techniques on the resilience of the CPN against DoS attacks. They introduced a generic framework of DoS protection based on the dropping of probable illegitimate traffic, and presented a mathematical model with which one can measure the impact that both attack and defence have on the performance of a network. More analytically, the CPN-based distributed DoS defence technique exploits the ability of the CPN to trace traffic going both downstream and upstream, owing to SPs and ACK packets. When a node detects an attack, it uses the ACKs to ask all intermediate nodes upstream to drop the packets of the attack flow. Each node is allowed to select the maximum bandwidth that it will accept from any flow that terminates at the node and the maximum bandwidth that it allocates to a flow that traverses the node. These parameters may vary dynamically as a result of other conditions, and they can also be selected based on the identity and the QoS needs of the flows. When a node receives an SP or DP from a flow that it has not previously encountered (e.g. with a new source–destination pair, or a new QoS class), it sends a Flow-ACK packet back to the source along the reverse path and informs the source of its bandwidth allocation. The node monitors the flows that traverse it and drops packets of any flow that exceeds the allocation; it may also inform upstream nodes that packets of this flow should be dropped. Other possible actions include diverting the flow into a ‘honeypot’ or to a special network. The mathematical results of [51] were validated with simulation results and experimental measurements in a CPN environment. This generic defence was further improved by using prioritization and rate-limiting instead of simple dropping [52,53]. To further improve the resilience against DoS attacks, the same authors have also introduced a DoS detection mechanism that makes use of on-line statistics collected by the CPN protocol's monitoring system and fused them with a RNN [54]. More analytically, the scheme uses input features to capture both the instantaneous behaviour and the longer-term statistical properties of the traffic. In an off-line information gathering step, it obtains the probability density function estimates, and evaluates the likelihood ratios for the input features. During the real-time decision step it measures and calculates the features of the incoming traffic, finds the likelihood ratios corresponding to those values and aggregates these likelihood values using an RNN. The overall architecture outputs a numerical value that is a measure of having an on-going attack in the network, which is consequently used in the prioritization and rate-limiting mechanisms previously mentioned [55,56].

5.6. CPN with sorting

In the original CPN implementation, the route brought back by an SP's ACK is considered as the best route and would be immediately used by the data traffic. However, this could lead to poor data packet performance since it does not take into consideration the random component of SP exploration, where each router will, with a small probability, sometimes choose a random next hop for an SP rather than the one decided by the RNN. Therefore, the authors of [57] propose to compare the quality of routes brought back by an ACK packet against the current route and switch routes only when the reward of the new route is greater than that of the current route. They named this version 'CPN with Sorting' and the experiments show that it performs better than the original CPN algorithm.

5.7. Recursive CPN

A QoS-based recursive routing algorithm, which can break a large-scale routing problem into some smaller routing problems, has been proposed in [58]. Partial routes are cached in the intermediate nodes, which can be used to provide a fast estimate of the "best" QoS-based route that is needed by an arriving packet. The experiments conducted on a 46-node network test-bed showed that when recursive routing is applied, the average time for a SP to discover a valid route is reduced dramatically, the QoS that the users experience is also slightly improved and more routes are discovered by the SPs. They also show that selecting the paths based on QoS rather than on the 'freshness' of data does not reduce the adaptivity of the CPN, and that it improves the QoS experienced by the users by recommending only the best routes.

5.8. AC with CPN

CPN offers QoS-based best-effort routing and therefore, high demand and network congestion, can prevent multimedia applications and users from obtaining the network service they require for a successful operation. A way to control traffic congestion and satisfy all users' QoS requirements, without overprovisioning, is AC. The authors in [59,60] propose a centralized, measurement-based, multiple criteria, AC algorithm that is based on measurements of the QoS metrics on each link of the network. This does not require any special mechanism since CPN already collects QoS information on all links and paths that the SPs have explored and on all paths that any user is using in the network. A novelty of the proposed AC algorithm is that the users can be the ones that specify the QoS constraints they need in order to obtain the network service they require for a successful connection, so that each user can have different QoS criteria and QoS values.

The scheme decides whether a new call should be allowed to enter the network based on measurements of the QoS metrics on each link of the network before and after the transmission of probe packets. Contrary to existing methods, the scheme estimates the new flow's impact by probing at a small rate, so that probe packets will not contribute noticeably to the network's congestion. Finally, the decision on whether to accept a new flow or not is based on an algebra of QoS metrics, inspired by Warshall's algorithm [61], which searches whether there is a feasible path to accommodate the new flow without affecting the existing users. Experimental results showing the efficiency of the algorithm, under highly congested circumstances, in a 46-node real test-bed were presented in [62].

5.9. Autonomic auctions and CPN

The role which CPN can play in the future network-based markets, where autonomic auctions will be part of the web-based economy, is described in [63,64]. The auctions considered, involve network auctions in which any bidder, except for the one who has made the most recent bid and is waiting for a response from the seller, is allowed to move at any time from one auction to another one. A mathematical analysis of such automated networked auctions is given in [65,66], where the author studies different auction types, both in terms of the price a good will fetch and the income per unit time provided by the auction, and provides mathematical models of automated bidders and sellers who interact through a network.

A number of CPN nodes, geographically distributed, could constitute a network market. This network can provide the users with information such as where and when to buy or sell an item. The network could consider application-level requirements, for instance price-related or history-related factors, along with network-level ones like routing or communication factors [64,67,68]. Decisions could be based on 'sensible decision' algorithms, studied in [49], where the estimations are based on probabilities that mix auction-specific factors, such as the current price of the item under auction, the average selling price in the past or the average wait-time for acceptance of the bid, and network or routing factors, such as the current average delay in the path towards the destination and the average loss probability in the path. Therefore, with the CPN protocol, each user (seller or buyer) of a networked auction would self-adapt their own behaviour according to heterogeneous factors either auction or network based.

5.10. CPN and next-generation battlespace information services

The CPN protocol is used for decision making in the communication layer of an agent-based architecture implemented as part of the HYPERION project, funded by the UK Ministry of Defence, which aims to provide an automated and adaptive information management capability embedded in defence networks [69]. The overall system architecture is designed to improve the situational awareness of field commanders by providing the ability to fuse and compose information services in real time. The key technologies adopted to enable this include: autonomous software agents, self-organizing middleware, a smart data filtering system and a 3D battlespace simulation environment. The role of CPN in this project is to provide an adaptive network architecture with enhanced functionality and resilience for networks for battlespace communication and information services. Since network security is of primary concern for such communication networks, the DoS defence mechanism described in a previous section is a vital aspect of this scheme.

6. EXPERIMENTAL PERFORMANCE EVALUATION

Apart from the individual experiments conducted for each enhancement and application presented in the previous sections, the CPN has been thoroughly investigated for a variety of performance metrics. All performance evaluation work has been carried out in a real networking test-bed, running the CPN as a module of the Linux kernel.

6.1. Adaptability

The ability of CPN to adapt to changing network conditions, such as changes in traffic load, link failures or buffer overflows has been experimentally evaluated in [70]. The experiments showed that the CPN managed to find new routes in order to avoid obstructing traffic introduced in some of the links used by the data traffic and also avoided links that were under failures. Another issue studied experimentally in [70] was the impact of the ratio of SPs to total packets, on the overall performance of CPN. The experiments concluded that in order to achieve the best performance for the DPs the percentage of SPs that should be sent for discovery is 10% to 20% of the data packets' rate. Going beyond these values does not significantly improve the QoS values for DPs. This was further investigated in [71], where experimental data also show that a relatively small fraction of SPs and ACKs, compared to total user traffic, is needed to serve the users' QoS goals. One must bear in mind that in CPN, SPs and ACKs are not full sized Ethernet packets, but are actually 10% of the DPs' size. If 20% of SP traffic is added, this will result in 14% traffic overhead, when ACKs are generated by both DPs and SPs, and only 4% of traffic overhead when ACKs are only generated in response to SPs. Additionally it was shown that a small number of SPs would suffice to initially establish a connection. In addition to the experiments on the test-bed in [71], simulations were conducted for a 1000-node network with results similar to the real experiments.

6.2. QoS goals

The choice of a 'goal' and 'reward' function for packetized voice applications is discussed in [5] and experiments conducted for 'voice over CPN' are presented. The performance of CPN is detailed via several measurements, and the resulting QoS is compared with that of the IP routing protocol under identical conditions showing the gain resulting from the use of CPN.

Measurements indicating how the CPN protocol can respond to different QoS goals are also presented in [6,8]. Composite goal functions for taking into account both delay and packet loss are proposed. In [8], the measurements suggest that CPN networks effectively adapt routing behaviour to the QoS goal that is specified; [6] also provides experimental results when a GA is used to create new routes from the information discovered by the SPs. The results showed that the GA daemon significantly improves QoS under light network traffic but not under high traffic conditions. An explanation given by the authors is that the GA tends to delay decision making, since it stores more information and makes recommendations based on longer-term trends.

The experiments in [72] show that the CPN can implement distributed adaptive shortest-path routing and approximately find the shortest paths. Extensive experiments compare the shortest-path CPN, where the QoS goal is the minimum hop count, with a CPN routing using minimum delay and a version where routing is based on a combination of hop count and forward delay. The experiments where conducted under low, medium and high background traffic and show that the use of criteria more complex than the shortest number of hops can provide better overall QoS.

The use of delay implies a collection of timestamps along packets' paths, which add overhead to the packets that may become significant in long routes. Therefore the authors of [73] implemented a composite QoS goal metric that consists of path length and buffer occupancy of nodes to achieve traffic balancing and to identify low-delay paths in a network. Experimental results in a wired test-bed and wireless *ad hoc* simulations show that a routing goal that combines path length and buffer occupancy in nodes offers the advantage of producing approximately the same performance as that of using delay but with a smaller packet overhead.

6.3. Realistic environments

A set of experiments, which demonstrate how the CPN performs in a realistic environment of a 46-node test-bed, have been presented in [57]. The performance of CPN was compared to that of an industry standard routing protocol, the open shortest path first (OSPF) routing protocol, the current industry standard and widely used in IP networks. A 46-node test-bed was used, the topology of which represents a real-world topology, the Swiss Education and Research Network (SWITCHlan), which is used by universities and some education sites in Switzerland. The administrators of this network provided the authors of [57] with details on their 46-router backbone, complete with bandwidth, OSPF costs and link-level delays. Because the cost of each link is proportional to its delay, OSPF routing converges to the minimal delay path, giving a baseline for comparison. The experiments show that the routes computed by CPN are as good as those computed *a priori* using administrator-defined costs. Furthermore, the paper gives experimental results showing that an RNN with RL can autonomously learn the best route in the network simply through exploration in a very short time-frame and demonstrates that the CPN protocol is able to adapt to changes in the network environment quickly, by switching to a new optimal route in the network. The experiments also show that the original algorithm of the CPN can be improved when data traffic is re-routed only when a better route is found (CPN with sorting).

6.4. Intermittent node failures

In [74], the authors compare the performance of CPN in the existence of intermittent node failures, caused by the spread of a network worm, with that of the OSPF routing protocol. The experimental results showed that the CPN performs much better than the OSPF routing, by adapting more quickly to the network changes and avoiding both the failed nodes and the congestion created by the failures, while keeping the average delay that the users experience at smaller levels. The authors of [74] also describe a failure detection element used to enhance the resilience of the CPN during node failures.

6.5. Routing oscillations

Although oscillations are generally considered a weakness of a network, performance evaluations indicate that routing oscillations do not severely degrade performance as would be expected, and high performance can still be obtained even in the presence of oscillations [75,76]. The way oscillations can be controlled was studied, and two different parameters that largely impact the rate at which oscillations are observed were tested: the use of probabilistic path switching, which can be used both to make path switching more asynchronous and to vary the rate at which switching decisions are made, and the introduction of a decision threshold, which will only allow path switching if the gain expected from switching exceeds a certain minimal value. Both of these control schemes are easy to implement and provide an effective way to limit oscillations and their negative consequences, although [75] brings into question whether there are actually such negative consequences in routing protocols that are based on self-monitoring and adaptation such as the CPN protocol.

7. CPN AND FUTURE INTELLIGENT NETWORKS

Scalability issues barricade the broad use of QoS mechanisms in the Internet, since it is impossible for each Internet router to deal with the QoS needs of each individual connection of the network. The scalability issue can be tackled with the use of intelligent network routers (INRs) that would be responsible only for the local users and services [77]. Source routing removes the burden of routing decisions from all but the local INR, which reduces overhead and removes the need of 'per flow' information handling apart from the INRs where the flows are initiated. Thus, the CPN algorithm, which runs at the packet transport level, can be abstracted to a higher level where it searches for services. Users and services can specify their requests in terms of the services that they seek and QoS criteria that they need, and also the price that they are willing to pay. In turn the services and the network would dynamically try to find the best-suited service for the users, satisfy them as best as they could and inform them of the level at which their requests are being satisfied and at what cost. Thus an intelligent network of the future, according to [77], can be viewed as an overlay network composed of INRs (CPN routers) that offer a flexible and self-organizing communication environment for users and services and can be used for finding services and users, for routing through the network and for self-observation and network monitoring in order to obtain the best QoS and performance.

8. CONCLUSIONS

We have described the CPN routing protocol, the underlying mathematical principles and the learning and decision algorithms that it uses.

In CPN, the routing decisions are made at each node of the network. Thanks to this distributed nature and the fact that it uses real-time QoS measurements, it is more resilient to link or node failures and more adaptive to network changes, such as network congestion.

Also, since it is the users rather than the nodes that control the routing, the traffic of different users can be routed under different QoS criteria, depending on the application and the required service. This is a unique contribution that CPN brings in computer network research.

Finally, its numerous applications and extensive experimental implementations in wired, wireless communication and sensor networks, indicate that it is a highly adaptable protocol that is easy to integrate with existing networking systems.

FUNDING

The Hyperion cluster project funded by the UK MoD's Data and Information Fusion Defence Technology Centre (DIF DTC) research program and British Telecommunications plc (BT). The CASCADAS project (EU FP6-027807). The ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project which is jointly funded by a BAE (British Aerospace) Systems and EPSRC (Engineering and Physical Sciences Research Council) strategic partnership (EP/C548051/1).

REFERENCES

[1] Gelenbe, E., Xu, Z. and Seref, E. (1999) Cognitive Packet Networks. *Proc. 11th Int. Conf. Tools with Artificial Intelligence* (*ICTAI'99*), Chicago, IL, USA, 8–10 November, pp. 47–54. IEEE Computer Society Press, Washington, DC, USA.

[2] Gelenbe, E., Seref, E. and Xu, Z. (2001) Simulation with learning agents. *Proc. IEEE*, **89**, 148–157.

[3] Gelenbe, E., Lent, R., Montuori, A. and Xu, Z. (2000) Towards Networks with Cognitive Packets. *Proc. 8th Int. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (*IEEE MASCOTS*), San Francisco, CA, USA, August 29–September 1, pp. 3–12. IEEE Computer Society Press, Washington, DC, USA. Opening Invited Paper.

[4] Gelenbe, E., Lent, R. and Xu, Z. (2001) Design and performance of cognitive packet networks. *Perform. Eval.*, **46**, 155–176.

[5] Gelenbe, E., Lent, R., Montuori, A. and Xu, Z. (2002) Cognitive Packet Networks: QoS and Performance. *Proc. 10th IEEE Int. Symp. Modeling, Analysis, and Simulation of Computer and Telecommunications Systems* (*MASCOTS'02*), Fort Worth, Texas, USA, October 11–16, pp. 3–9. IEEE Computer Society, Los Alamitos, CA, USA. Opening Keynote Paper.

[6] Gelenbe, E., Gellman, M., Lent, R., Liu, P. and Su, P. (2004) Autonomous Smart Routing for Network QoS. *Proc. First Int. Conf. Autonomic Computing* (*ICAC*), New York, NY, USA, May 17–18, pp. 232–239. IEEE Computer Society Press, Washington, DC, USA.

[7] Gelenbe, E. (2004) Cognitive Packet Network. US Patent 6804201 B1.

[8] Su, P. and Gellman, M. (2004) Using adaptive routing to achieve quality of service. *Perform. Eval.*, **57**, 105–119.

[9] Gelenbe, E. (1989) Random neural networks with negative and positive signals and product form solution. *Neural Comput.*, **1**, 502–510.

[10] Gelenbe, E. (1993) Learning in the recurrent random neural network. *Neural Comput.*, **5**, 154–164.

[11] Gelenbe, E. (1990) Stability of the random neural network model. *Neural Comput.*, **2**, 239–247.

[12] Gelenbe, E. and Stafylopatis, A. (1991) Global behavior of homogeneous random neural systems. *Appl. Math. Model.*, **15**, 534–541.

[13] Gelenbe, E. and Fourneau, J. M. (1999) Random neural networks with multiple classes of signals. *Neural Comput.*, **11**, 953–963.

[14] Gelenbe, E. and Hussain, K. (2002) Learning in the multiple class random neural network. *IEEE Trans. Neural Netw.*, **13**, 1257–1267.

[15] Gelenbe, E. and Timotheou, S. (2008) Random neural networks with synchronised interactions. *Neural Comput.*, **20**, 2308–2324.

[16] Gelenbe, E., Stafylopatis, A. and Likas, A. (1991) An extended Random Network Model with Associative Memory Capabilities. *Proc. Int. Conf. Artificial Neural Networks* (*ICANN'91*), Helsinki, Finland, June 24–28, pp. 307–312. Elsevier, Amsterdam, The Netherlands.

[17] Likas, A. and Stafylopatis, A. (1996) High capacity associative memory based on the random neural network model. *Int. J. Pattern Recognit. Artif. Intell.*, **10**, 919–937.

[18] Stafylopatis, A. and Likas, A. (1992) A pictorial information retrieval using the random neural network. *IEEE Transactions on Software Engineering*, **18**, 590–600.

[19] Cramer, C., Gelenbe, E. and Gelenbe, P. (1998) Image and video compression. *IEEE Potentials*, **17**, 29–33.

[20] Bakircioglu, H., Gelenbe, E. and Koçak, T. (1998) Image enhancement and fusion with the Random Neural Network model. *Elektrik*, **5**, 65–77.

[21] Gelenbe, E., Bakircioglu, H. and Koçak, T. (1997) Image Processing with the Random Neural Network (RNN). *Proc. 13th Int. Conf. Digital Signal Processing*, Santorini, Greece, July 2–4, pp. 243–248. SPIE, San Jose, CA, USA.

[22] Atalay, V. and Gelenbe, E. (1992) Parallel algorithm for colour texture generation using the random neural network model. *Int. J. Pattern Recognit. Artif. Intell.*, **6**, 437–446.

[23] Atalay, V., Gelenbe, E. and Yalabik, N. (1992) The random neural network model for texture generation. *Int. J. Pattern Recognit. Artif. Intell.*, **6**, 131–141.

[24] Gelenbe, E., Cramer, C., Sungur, M. and Gelenbe, P. (1996) Traffic and video quality in adaptive neural compression. *Multimedia Syst.*, **4**, 357–369.

[25] Cramer, C., Gelenbe, E. and Bakircloglu, H. (1996) Low bit-rate video compression with neural networks and temporal subsampling. *Proc. IEEE*, **84**, 1529–1543.

[26] Feng, Y. and Gelenbe, E. (1998) Adaptive object tracking and video compression. *Netw. Inform. Syst.*, **1**, 371–400.

[27] Cramer, C. and Gelenbe, E. (2000) Video quality and traffic QoS in learning-based subsampled and receiver-interpolated video sequences. *IEEE J. Sel. Areas Commun.*, **18**, 150–167.

[28] Bakircioglu, H., Gelenbe, E. and Koçak, T. (1997) Image enhancement and fusion with the random neural network. *Elektrik*, **5**, 65–77.

[29] Mohamed, S. and Rubino, G. (2002) A study of real-time packet video quality using random neural networks. *IEEE Trans. Circuits Syst. Video Technol.*, **12**, 1071–1083.

[30] Aguilar, J. and Gelenbe, E. (1997) Task assignment and transaction clustering heuristics for distributed systems. *Inf. Sci.*, **97**, 199–219.

[31] Gelenbe, E. and Shachnai, H. (2000) On G-networks and resource allocation in multimedia systems. *Eur. J. Oper. Res.*, **126**, 308–318.

[32] Gelenbe, E. (1994) G-networks: a unifying model for queuing networks and neural networks. *Ann. Oper. Res.*, **48**, 433–461.

[33] Fourneau, J., Gelenbe, E. and Suros, R. (1996) G-networks with multiple classes of positive and negative customers. *Theoret. Comput. Sci.*, **155**, 141–156.

[34] Gelenbe, E. and Labed, A. (1998) G-networks with multiple classes of signals and positive customers. *Eur. J. Oper. Res.*, **108**, 293–305.

[35] Gelenbe, E. (2000) The first decade of G-networks. *Eur. J. Oper. Res.*, **126**, 231–232.

[36] Gelenbe, E. and Fourneau, J. M. (2002) G-networks with resets. *Perform. Eval.*, **49**, 179–191.

[37] Gelenbe, E. and Cao, Y. (1998) Autonomous search for mines. *Eur. J. Oper. Res.*, **108**, 319–333.

[38] Gelenbe, E., Hussain, K. and Kaptan, V. (2005) Simulating autonomous agents in augmented reality. *J. Syst. Softw.*, **74**, 255–268.

[39] Halici, U. (2000) Reinforcement learning with internal expectation for the random neural network. *Eur. J. Oper. Res.*, **126**, 288–307.

[40] Gelenbe, E., Liu, P. and Laine, J. (2006) Genetic Algorithms for Autonomic Route Discovery. *Proc. IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, Prague, Czech Republic, June 15–16, pp. 371–376. IEEE Computer Society, Los Alamitos, CA, USA.

[41] Gellman, M. (2007) Quality of service routing for real-time traffic. PhD Thesis, Imperial College London, London, UK.

[42] Gelenbe, E. and Lent, R. (2004) Power-aware ad hoc cognitive packet networks. *Ad Hoc Netw. J.*, **2**, 205–216.

[43] Lent, R. (2007) Linear QoS goals of additive and concave metrics in ad hoc cognitive packet routing. *IEEE Trans. Syst. Man Cybern.*, **36**, 255–260.

[44] Lent, R. (2005) Smart Packet-Based Selection of Reliable Paths in Ad Hoc Networks. *Proc. 5th Int. Workshop on the Design of Reliable Communication Networks* (*DRCN 2005*), Island of Ischia, Naples, Italy, October 16–19, pp. 497–501. IEEE, Piscataway, NJ, USA.

[45] Gelenbe, E. and Nunez, A. (2003) Traffic engineering with cognitive packet networks. *Simul. Ser.*, **35**, 514–518.

[46] Koçak, T., Seeber, J. and Terzioglu, H. (2003) Design and implementation of a random neural network routing engine. *IEEE Trans. Neural Netw.*, **14**, 1128–1143.

[47] Hey, L., Cheung, P. and Gellman, M. (2005) FPGA Based Router for Cognitive Packet Networks. *Proc. IEEE Int. Conf. Field-Programmable Technology*, Kent Ridge, Singapore, December 11–14, pp. 331–332. IEEE, Piscataway, NJ, USA.

[48] Hey, L. (2008) Reduced complexity algorithms for cognitive packet network routers. *Comput. Commun.*, **31**, 3822–3830.

[49] Gelenbe, E. (2003) Sensible decisions based on QoS. *Comput. Manage. Sci.*, **1**, 1–14.

[50] Hey, L. (2008) Power Aware Smart Routing in Wireless Sensor Networks. *Proc. Conf. Next Generation Internet Networks* (*NGI 2008*), Krakow, Poland, April 28–30, pp. 195–202. IEEE, Piscataway, NJ, USA.

[51] Gelenbe, E., Gellman, M. and Loukas, G. (2005) An Autonomic Approach to Denial of Service Defence. *Proc. Sixth IEEE Int. Symp. World of Wireless Mobile and Multimedia Networks* (*WoWMoM 2005*), Taormina, Italy, June 13–16, pp. 537–541. IEEE Computer Society, Washington, DC, USA.

[52] Gelenbe, E. and Loukas, G. (2007) A Self-aware approach to denial of service defence. *Comput. Netw.*, **51**, 1299–1314.

[53] Oke, G. and Loukas, G. (2008) Distributed Defence Against Denial of Service Attacks: A Practical View. *Proc. 1st BCS Int. Academic Conf., Visions of Computer Science*, London, UK, September 22–24, pp. 153–162. The British Computer Society, Swindon, UK.

[54] Oke, G. and Loukas, G. (2007) A denial of service detector based on maximum likelihood detection and the random neural network. *Comput. J.*, **50**, 717–727.

[55] Oke, G., Loukas, G. and Gelenbe, E. (2007) Detecting Denial of Service Attacks with Bayesian Classifiers and the Random Neural Network. *Proc. IEEE Int. Fuzzy Systems Conf.* (*FUZZ-IEEE 2007*), London, UK, July 23–26, pp. 1–6. IEEE, Los Alamitos, CA, USA.

[56] Loukas, G. and Oke, G. (2007) Likelihood Ratios and Recurrent Random Neural Networks in Detection of Denial of Service Attacks. *Proc. Int. Symp. Performance Evaluation of Computer and Telecommunication Systems* (*SPECTS 2007*), San Diego, CA, USA, July 16–18, pp. 16–18. Simulation Councils Inc., San Diego, CA, USA.

[57] Gellman, M. and Liu, P. (2006) Random Neural Networks for the Adaptive Control of Packet Networks. *Proc. 16th Int. Conf. Artificial Neural Networks* (*ICANN 2006*), Athens, Greece, September 10–14, pp. 313–320. Springer, Berlin, Heidelberg, Germany.

[58] Liu, P. and Gelenbe, E. (2007) Recursive Routing in the Cognitive Packet Network. *Proc. 3rd Int. Conf. Testbeds and Research Infrastructures for the Development of Networks and Communities* (*TridentCom*), Orlando, FL, USA, May 21–23, pp. 1–6. IEEE, Piscataway, NJ, USA.

[59] Sakellari, G., D' Arienzo, M. and Gelenbe, E. (2006) Admission Control in Self Aware Networks. *Proc. 49th Annual IEEE Global Telecommunications Conf.* (*GLOBECOM 2006*), San Francisco, CA, USA, November 27–December 1, pp. 1–5. IEEE, Piscataway, NJ, USA.

[60] Gelenbe, E., Sakellari, G. and D' Arienzo, M. (2008) Admission of QoS aware users in a smart network. *ACM Trans. Auton. Adapt. Syst.*, **3**, 4:1–4:28.

[61] Warshall, S. (1962) A theorem on Boolean matrices. *J. ACM*, **9**, 11–12.

[62] Sakellari, G. and Gelenbe, E. (2008) A Multiple Criteria, Measurement-Based Admission Control for Self-Aware Networks. *Proc. Third Int. Conf. Communications and Networking in China* (*CHINACOM'08*), Hangzhou, China, August 25–27, pp. 1060–1064. IEEE, Piscataway, NJ, USA.

[63] Gelenbe, E., Lent, R., Di Ferdinando, A. and Manzalini, A. (2006) Distributed Auctions in Autonomic Communication Services. *Proc. Second IEEE ComSoc/CreateNet Int. Workshop on Guaranteed Optical Service Provisioning* (*GOSP 2006*), San Jose, CA, USA, October 1–5, pp. 1–6. IEEE, Piscataway, NJ, USA.

[64] Di Ferdinando, A., Lent, R. and Gelenbe, E. (2007) A Framework for Autonomic Networked Auctions. *Proc. 1st Int. Conf. Autonomic Computing and Communication Systems, Workshop on Innovative Service Technologies (INSERTech '07)*, Rome, Italy, 28–30 October, pp. 1–10. ICST, Brussels, Belgium.

[65] Gelenbe, E. (2009) Analysis of single and networked auctions. *ACM Trans. Internet Technol.*, **9**, 1–24.

[66] Gelenbe, E. (2006) Analysis of Automated Auctions. *Proc. 21th Int. Symp. Computer and Information Sciences* (*ISCIS' 06*), *Lecture Notes in Computer Science*, Istanbul, Turkey, November 1–3, pp. 1–12. Springer, Berlin, Heidelberg, Germany.

[67] Di Ferdinando, A., Rosi, A., Zambonelli, F., Lent, R. and Gelenbe, E. (2008) A Platform For Pervasive Combinatorial Trading With Opportunistic Self-Aggregation. *Proc. Second IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications* (*AOC 2008*), Newport Beach, CA, USA, June 23–26, pp. 1–6. IEEE, Piscataway, NJ, USA.

[68] Gelenbe, E., Lent, R., Di Ferdinando, A. and Manzalini, A. (2006) An Autonomic Networked Auction System. *Proc. Int. Conf. Self-Organization and Autonomous Systems in Computing and Communications* (*SOAS 2006*), Erfusr, Germany, 18–21 September, pp. 1–14.

[69] Ghanea-Hercock, R., Gelenbe, E., Jennings, N.R., Smith, O., Allsopp, D.N., Healing, A., Duman, H., Sparks, S., Karunatillake, N.C. and Vytelingum, P. (2007) Hyperion — next-generation battlespace information services. *Comput. J.*, **50**, 632–645.

[70] Gelenbe, E., Lent, R. and Xu, Z. (2001) Measurement and performance of a cognitive packet network. *Comput. Netw.*, **37**, 691–701.

[71] Gelenbe, E., Lent, R. and Nunez, A. (2004) Self-aware networks and QoS. *Proc. IEEE*, **92**, 1478–1489.

[72] Gelenbe, E. and Liu, P. (2005) QoS and Routing in the Cognitive Packet Network. *Proc. First Int. IEEE WoWMoM Workshop on Autonomic Communications and Computing (ACC'05)*, Taormina, Italy, June 13–16, pp. 517–521. IEEE Computer Society, Washington, DC, USA.

[73] Lent, R. and Liu, P. (2005) Searching for Low Latency Routes in CPN with Reduced Packet Overhead. *Proc. ISCIS 2005, Advances in Computer Science and Engineering Series*, Istanbul, Turkey, October 26–28, pp. 63–72. Imperial College Press, London, UK.

[74] Sakellari, G. and Gelenbe, E. (2009) Adaptive Resilience of the Cognitive Packet Network in the presence of Network Worms. *Proc. NATO Symp. C3I for Crisis, Emergency and Consequence Management*, Bucharest, Romania, May 11–12.

[75] Gellman, M. (2008) Oscillations in self-aware networks. *Proc. R. Soc.*, **464**, 2169–2185.

[76] Gelenbe, E. and Gellman, M. (2007) Oscillations in a Bio-Inspired Routing Algorithm. *Proc. IEEE Int. Conf. Mobile Ad hoc and Sensor Systems* (*MASS 2007*), *BIONETWORKS Workshop*, Pisa, Italy, October 8–11, pp. 1–7. IEEE Computer Society, Washington, DC, USA.

[77] Gelenbe, E. (2006) Users and services in intelligent networks. *IET Proc. Intell. Transp. Syst.*, **153**, 213–220.