Self-Aware Networks that Optimize Security, QoS and Energy

Erol Gelenbe, *Fellow IEEE*, Joanna Domanska, Piotr Fröhlich, Mateusz Nowak and Sławomir Nowak
Institute of Theoretical & Applied Computer Science
IITIS-PAN, 44-100 Gliwice, Poland

***Abstract*—The need to adaptively manage computer systems and networks so as to offer good Quality of Service (QoS) and Quality of Experience (QoE), with secure operation at relatively low levels of energy cnsumption is challenged by their sheer complexity and the wide variability of the workloads. A possible way forward is through self-awareness, whereby self-measurement and self-observation, together with on-line control mechansims which operate adaptively to attain the required performance and QoE. We survey the premises for these ideas coming from Cognitive Science and Active Networks, and review recent work on self-aware computer systems and networks, including those that propose the use of Software Defined Networks as a means to implement these concepts. Then we provide some examples from the literature on self-aware systems to illustrate the performance gains that they can provide. Finally we detail an example system and its working algorithms to allow the reader to understand how such a system may be implemented. Measurements showing how it can react rapidly to changing network conditions regarding Quality of Service (QoS) and security are also presented. Some conclusions and suggestions for further work are also presented.**

***Index Terms*—Self-Aware Networks, Artificial Intelligence, Random Neural Networks, Software Defined Networks, Quality of Experience, Quality of Service, Cognitive Packet Network, Reinforcement Learning**

I. Introduction

In the last few years, the telecommunications industry has woken to the potential use of artificial intelligence and machine learning to automate and simplify network design, management, and operations. Thus many organisations in the telecommunications industry have announced programs that aim at introducing machine learning into the next generation of packet and mobile networks. Recently, an industry publication [1] stated that "Advances in artificial intelligence are pushing .. an economically feasible Self-Driving Network^{TM} ... that's programmed to .. carry out your intentions ... eliminates the complex programming and management tasks ... self-configure, monitor, manage, correct, defend ...with very little human intervention ... predict performance issues ... eliminate burdensome operational tasks and free ... IT staff ... costs will drop. Security, reliability, and resiliency will improve ... speed of business will accelerate."

Indeed, in the same line, a recent industry blog post [2] asks the rhetorical question "So how does AI help? It starts at the top, with codifying .. the intent of the network operator ... in human language or through a more traditional interface ... translated into network and security policies ... It is often especially important to use machine reasoning ... domain-specific knowledge about networking to ... realize the desired intent in the given network context ... with a deep understanding of the network infrastructure ... automates the policies ... optimizing for performance, reliability, and security."

In [3] mobile radio access networks (RAN), especially for 5G, are discussed and it is suggested that AI can streamline RANs for Massive Multiple-Input-Mutiple-Output optimisation with Reinforcement Learning (RL) [4] so that each cell self-adapts to changing scenarios and traffic, increasing throughput by 20% and optimising speed for users that have low throughput.

Back in the 1990s, the research community had developed similar ideas [5]–[7] that have now met the technologies that allow their practical implementation. Thus this paper reviews the advances made since those early days, in support of the current vision to use data analytics and machine learning to automate network operations through Self-Aware Networks.

A. The Premise from Cognitive Science and Philosophy

Self-awareness has been studied by cognitive scientists [8] who have discussed "conscious attention" as having two aspects: one being directed "toward the self", while the other is directed "toward the environment". This early conceptualisation also emphasizes the importance of "discrepancies" between the internal model and the external reality. Discrepancies alert us on the differences between what we had thought about (from the internal model) and the reality we observe. Thus, discrepancies can be exploited via some form of RL because "the person will experience either negative or positive affect depending on whether attention is directed toward a negative or a positive discrepancy". This takes us back to René Descartes' [9] rationalism and his well-known affirmation that "I think therefore I am", since thought leads to self-representation to the ability to link the internal model to external cues and events and improve the internal model.

On the other hand, similar insight into the notion of "self-awareness" can be gained through an interesting article [10] from 1999, that defines the intentionality of an agent which may be "cued", i.e. triggered for instance by some sensory event or observation, or via a "detached" internal representation that models the world within which the agent exists. Links between the cued and detached representations are then based on experience. These links serve as a basis for action, and for the refinement of the internal or detached model. The paper argues about the importance of both the internal and

detached representation, that may include a self-model of the agent, which allows the agent to determine its capacity to have intentions and be able to act on these intentions. Both of these works [8], [10] provide insight into the way in which many "self-aware" computer networks and computer systems have been designed, with an internal model that is updated and corrected using observations based on measurement, and RL to make corrections in the internal model and take decisions.

Self-awareness is also often discussed through the development of young animals and children [11], [12] who go through of self-awareness that change with age. Another area that is often studied relates to anomalies in self-awareness. Examples include the commonly known "self-denial" which is viewed as a psychological protection mechanism whereby individuals refuse to acknowledge a weakness or an illness, and anosognosia [13] where a neurological disorder (i.e. malfunctioning of brain neural networks) can cause an individual's lack of awareness or denial of a well identified physical incapacity or illness. Other work has considered the relation between internal representation of self-awareness and the capacity to take action [14].

B. Self-Aware Networks and Quality of Experience

In the context of packet networks self-awareness should include the ability of a network to pursue objectives or goal functions. It should be able to observe itself – via data from measurements that the network collects – and "criticize" its own behaviour, so as to meet the objectives that have been set for it, and improve its behaviour based on these observations. The network's primary goal should be to convey traffic flows from source to destination nodes, and to retrieve and convey content to its users. However, this cannot be done without consideration for the Quality of Experience (QoE) [15] of its "users", which are not just the end users that convey traffic or download content through the network, but also the network's human operators, and the people in charge of troubleshooting the network.

At the speed with which the network operates forwarding GBytes of data with hundreds of thousands of simultaneous users, the analysis of QoE does not mean that we can poll users about their satisfaction with the network [16] in real-time, though some of that may happen at a slower pace by sampling some users. However, early work has successfully linked network bandwidth and perceived video quality, with the traffic characteristics and network QoS [17].

At the high-end, one may monitor the end-users' emotions as they use the network [18] to evaluate QoE, though the same emotions may be related to the content they receive rather than the network's own performance. However substantial work has been conducted on linking the quality of the sound that the network users may be hearing through their connections, to the QoS that is directly measured [19], [20]. A related strain of work has linked the perceived quality of video that is downloaded by the users to the usual traffic QoS measurements, such as packet loss and delay [21].

Another major element that enters into QoE, from the perspective of the network operator as a "user", is the network's energy consumption, because energy may represent close to 70% of a network's operating costs [22], [23]. Thus the minimization of energy consumption [24], [25] needs to be included in the "goal" or objective of a self-aware network. Its measurement and evaluation is a key issue [26], [27] and it should also include the Cloud support that is indispensable for modern communication networks [28].

Resilience [29] and security [30] are also key issues for the network operator, who is the "user" that needs to assure the seamless non-stop operation of the system. Poor resilience and security are also directly felt by the end user. Thus substantial research has been conducted in this area [31], while network software must be protected with appropriate recovery techniques [32]. While the actions of the environment on a network arise mainly from the legitimate "good" users, "malicious" users can launch attacks or infiltrations which can have dramatic effects on the QoS perceived by end users. Thus cybersecurity remains one of the core aspects of self-aware network management, with the help of appropriate attack detection and mitigation techniques [33], [34].

Thus in the sequel, when we express a Self-Aware Network's Goal or Objective function, we will include both QoS, Security and Energy, to handle the different issues related to QoE in a holistic manner.

C. Content of this Paper

In the next section we will consider the design of self-aware networks, and discuss the first systematic attempt to incorporate intelligence in networks, known as *Active Networks*, which have provided many of the ideas that are still being pursued today.

Next, we will examine how self-aware networks can be built thanks to the rising technology of software defined networks (SDN), and how machine learning techniques such as RL can help implant self-awareness into such systems. This will lead to some detailed examples that will also quantitatively illustrate how they may be used in network overlay routing to reduce end-to-end packet delays in the global Internet, or to reduce the average execution time of jobs that are allocated to edge or fog clusters, and in the Cloud.

Then we will go further into how SDN can incorporate self-awareness using the Cognitive Packet protocol based on Smart Packets that gather measurements and information in the network and using a RL based decision engine to modify the paths of flows dynamically to achieve better QoS.

This discussion will lead directly to a detailed presentation of the working example of a networked system whose aim is to provide quality of service and energy aware network connectivity with active protection from network attacks. A test-bed that embodies these concepts will be detailed, together with several measurements that illustrate its ability to react rapidly to quality of sevice degradation and to security alarms.

The paper ends with some conclusions and suggestions for further research.

II. Related Work

The first mention of a "self-aware network" appears in 2003 [35], [36], while reference to an overlay network's "lower level

network awareness" appears in [37], and context-awareness in wireless ad-hoc networks is proposed in [38]. Other work has discussed "bodily self-awareness" with regard to humans and machines [14]. Self-aware services that can detect anomalies in Internet-based services are presented in [39], and web-aware tools are proposed in [40].

Yet the earlier concepts of Cognitive Radio (CR) [41] and Cognitive Packet Networks (CPN) from 1999 [7], also describe networks which are self-aware. Indeed, in CR [42], [43] the network's packet forwarders (in this case radio transmitters) directly sense the communication channel to enhance their awareness of the conditions under which they are communicating, before they transmit or forward packets. Their purpose is to optimize both the utilization of the channels, and the performance or QoS of the users' communications by reducing possible interference between distinct communications.

The work in [44] describes CPN [45] where "intelligent capabilities for routing and flow control are concentrated in the ... Cognitive Packets (which) route themselves. They are assigned goals ... and pursue these goals adaptively ... learn from their own observations about the network and from the experience of other packets with whom they exchange information via mailboxes". The analogy between a network path and a sequence of "letters" in the genetic code was also exploited in [46] to choose the best paths in CPN with a genetic algorithm that uses QoS as the fitness function.

Even earlier, the ALOHA network [47], was a pioneering initiative in self-aware "multiple access" networks to connect terminal consoles to computer servers via space satellite based communications. Ideas from ALOHA were rapidly incorporated into the widely used local area network Ethernet [48], and also used in the first fiber-optics random access network Xantos [49], as well as the well known space-satellite based network Inmarsat for boats and ships [50].

ALOHA could be "slotted" so that time was synchronised throughout the network and all participants transmitted at the beginning of a slot, so that they could not be aware of each other prior to the transmission, but they could optimize the network's collisions by tracking the length of the silent periods to estimate the network's traffic rate [51], [52]. ALOHA could also be "unslotted" and operated in continuous time. In the latter case the "carrier sense multiple access" (CSMA-CD) protocol [53] required the different users to sense the channel before transmitting, and refrain from transmitting if they "heard" that there were other ongoing transmissions on the channel, so as to minimize the probability of interference and collisions with other users [54].

A. Active Intelligent Networks

In the mid 1990's, the traditional role of IP networks that had emerged as the main follow-up to wired telecommunications, were being seriously challenged. Indeed, Internet Protocol (IP) networks were limited to transporting data passively and opaquely between systems, without taking advantage of the possibilities offered by the knowledge acquired about users and their requirements, and about the data content of packets. The capabilties offered by technological imrpovements that had increased the computational power of routers were not beeing exploited, and the routers' role was limited to managing packet headers and to signalling functions needed to manage connections.

This tradition was broken by the novel concept of Active Networking (AN) [6], [55] to perform useful tasks in a network to improve the end users' quality of service (QoS) and the network's performance, using software that is adaptively activated in the network. Examples include compressing high throughput packet flows (such as video) with knowledge of the data content, grouping the routing of multiple flows that carry identical media content into multicast trees to save network bandwidth and reducing end-to-end delay, using knowledge about a group of users in video-conferencing to optimize network topologies and minimize delay, or tactivating enhanced security and attack detection at nodes when they carry sensitive traffic.

AN places active "capsules" into IP packets that are either programs that can be run by routers, or data that activates or instantiates a program that is already resident at a node. It was suggested [56] that AN could enhance security by activating packet filters dynamically to authentify traffic flows. Capsules could also be used dynamically activate data caches in the network nodes. One major application of AN that was proposed was to deploy and update IP networks based on policies and specific network conditions [57], [58]. The worldwide interest it generated resulted in a global network Planetlab network of servers that survives to today [59].

While AN initially encountered much enthusiasm, the idea did not fully survive and certain aspects were progressively incorporated into other technologies and research areas. AN had presented several difficulties, including the need to revisit the computing power and storage space available at network nodes, especially in those years when computing power was significantly lower than today.. The unlimited capabilities of AN implied that the corresponding routers could find themselves overloaded with tasks, turning themselves into network servers with a role going way beyond routing, so that research had to be conducted into the design of more powerful network routers and corresponding test-beds [60], [61]. In additon, AN could potentially create processing overload in the network nodes, that came in addition to the load caused by the traffic itself, despite the fact that active routers could also interoperate with legacy routers, which transparently forward datagrams in the traditional manner [62].

Another major issue concerned security [63]. Indeed, the fact that AN proposed to use packets to inject programs into the network, with these programs being possibly specific to various users, could create tremendous security risks. Thus, each entering "caplet" would have to be monitored, either with regard to its provenance, with some form of admission control, or would have to be verified with some form of deep packet inspection. Both of these functions are not available in the Internet Protocol (IP), so that AN required a comprehensive review of IP. Thus AN was raising even more questions regarding the means to mitgate the threats that it itself created, even though it could offer certain security services [64].

However many good ideas that AN developed have been

incorporated into current research. For instance, the Cognitive or Smart Packets in CPN [65] do not carry code but do carry data to instantiate programs that are already resident in the network nodes. The idea about caching data or creating services in routers or nodes has also been incorporated in many systems, and Fog and Edge computing carry some of the functions that were initially proposed for AN routers [66].

1) Self-Aware Networks that use SDN: The rise of Sofware Defined Networks (SDN) [67], [68] has provided a practical opportunity to experiment with networks that have some of the features that were suggested by AN. SDN can operate across various technologies incuding fiber-optic networks [69], and a combination of WiFi and wired connections [70].

In [71] an SDN controller incorporates machine learning based decisions for routing so as to optimize Quality of Service (QoS), and tested to dynamically route flows between two fixed end servers in a small network with regard to its reactions to sudden changes of delay between nodes. A similar approach whose objective is to minimize energy was reported in [25]. However, the measurements show that a SDN controller operating with the CPN algorithm [65] switches paths rapidly and allows the test-bed to reach its new performance level, within a few percent of the optimum end-to-end delay, in a few seconds. Such transients can be long in terms of the characteristic time constants of networks which operate at the millisecond level [72], and they should be studied further to determine their impact on the overall system optimization and the system's reaction delays.

There are several differences between the results in [71], [73] which implement a RL based SDN controller and measure its performance, and the interesting work in [74] which presents numerical results concerning a RL algorithm's internal numerical values but not the system's resulting performance. Also [74] discusses a hierarchical structure, but then shows internal numerical values for the algorithm within a single controller so that the behaviour of the system as a whole is not evaluated. The "single tree" in [71] can be used for one instance of a sub-tree in [74]. From an algorithmic perspective, in [74] only QoS is considered (and not security or energy), and it uses Markov decision processes with a soft-max decision rule, while in [73] a Goal Function is used and direct system measurements are carried out on a test-bed.

The research in [75] suggests the use of RL for load balancing in an IP network that is used to control the Smart Grid. It proposes to use an SDN controller for the network with two classes of data packets: those that carry monitoring traffic for the Smart Grid, and those that carry control commands which have a higher priority. A policy iteration approach based on Markov decision processes is proposed for the RL algorithm. The authors present discrete event simulations with Poisson traffic for a three node network to show the improvements that can be obtained regarding the network's QoS. Thus while the work in the present paper presents an actual network implementation on a multiple node test-bed with real traffic and experimental results both for QoS and security, the presentation in [75] covers QoS only, and it is illustrated with a discrete event simulation of three network nodes with idealised Poisson traffic.

In [76], [77], an approach that combines SDN using CPN [65] is proposed, and outlines a hierarchical vision of SDN similar to [74], without the algorithmic aspects and results of the present paper. In [78] a steady-state analysis of the radio transport layer for 5G is presented, based on prior mathematical work on signal to noise plus interference ratio calculations. A SDN configuration is then suggested to support base stations in order to maximize steady-tate throughput at the radio level. This work differs from the problem we deal in this paper which mainly fouses on an experimental investigation of wired networks and their dynamic real-time operation.

III. Examples of Self-Aware Systems and Networks

The global Internet and the resulting possibility to create large scale services with the help of the web, encouraged early work [79] on the design of systems that could dynamically monitor and improve QoS experienced by services for a large number of users, and examples were developed in web services for managing map data [40] and in detecting QoS and other anomalies in Internet services [39]. The latter example illustrates the inconsistency discussed earlier between what is expected based on an internal model, and the observations obtained from attention that is directed toward reality [8]. Early work [37] also focused on building network middleware without changes in the existing Internet substrate.

Another strain of work that has given impetus to the use of self-awareness in networks for various purposes, is the field of "adhoc networks" [80], [81] based on wireless nodes that are geographically distributed, yet relatively close to each other, and which dynamically create temporary links and network graph topologies with other nodes [38], [82]. Here the nodes are often battery powered and mobile [83]. Since their energy consumption is of concern, if one wishes to maximize the failsafe operation of the network links [84], and neural network based methods were implemented and tested to create self-aware methods to dynamically manage routes and maximize the energy life-time of the network [44].

A. Self-Aware SMART Overlay Routing

In this section we review results that were obtained when self-aware QoS oriented routing using RL was used with network overlays [85], motivated by the fact that many measurements have shown that the Internet Protocol (IP) typically yields sub-optimal network paths for QoS metrics [86].

The idea of overlay networks that use software installed in servers, or in machines that are connected to routers, so as to take routing decisions that supersede the ones taken by the routers themselves, has been around for some time [31]. One simple approach is to capture the packets coming from a given router, and then forward them to an IP address via the IP protocol that is not the "natural" next hop of the given packet, had it stayed within the given router. This decision can be taken based on prior knowledge that the modified next hop can provide better QoS for the packet towards the final destination. Various proposals and experiments are available in the literature regarding this idea [87], [88]. A simple approach

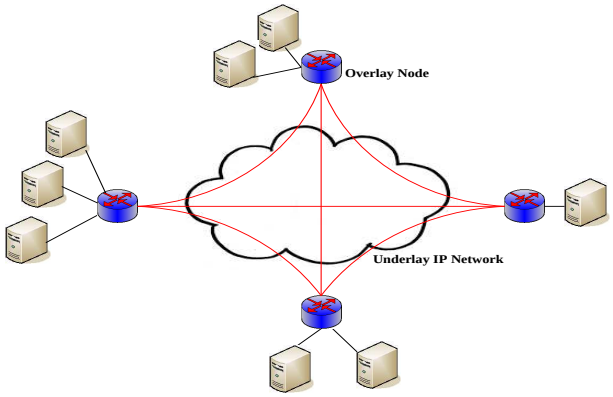


Fig. 1. Schematic Representatin of an Overlay Network.

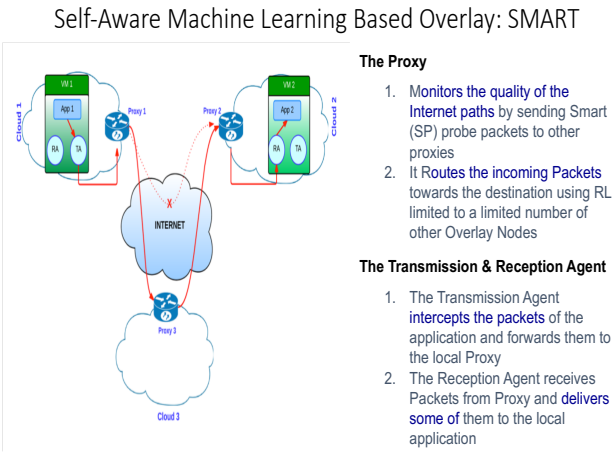


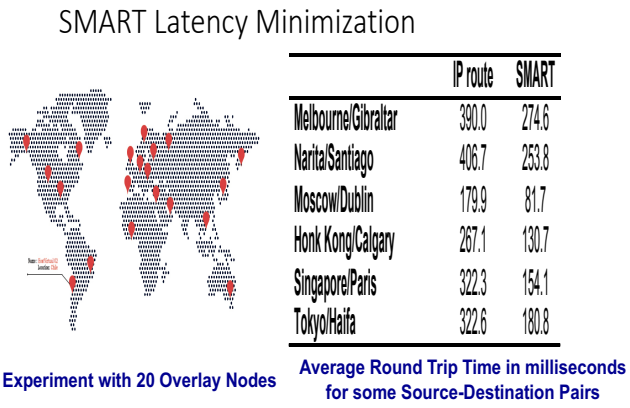
Fig. 2. The software architecture of the SMART overlay node.

would be to exploit "next hops" that are known to offer some QoS guarantee based on admission control schemes that were popular in earlier generation networks [89]. However the approach taken here is based on self-awareness and on-line measurement.

Routing overlays have a long history, and have also been suggested to improve the reliability and resilience of the network in case of path outages or network attacks. However they have the main advantage of overriding those routes that are selected by IP and route traffic based on the QoS considerations of the applications that are generating and utilising the network connections. Potentially, many different applications may use overlays differently with different QoS considerations. The Resilient Overlay Network (RON) [31] was the first such overlay for wide-area networks that demonstrated this advantage. However RON had the disadvantage of having to monitor $O(M^2)$ connections for M overlay nodes, and is thus was limited by the amount of overhead it imposed on each overlay node.

The SMART (Self-MAnaging Routing) overlay overcomes this difficulty for an M node overlay network by limiting to 4 or another relatively small number of overlay neighbour nodes that a single overlay node can use as a next hop. Because of the use of self-awareness gained through the use of SPs and the RL algorithm described previously, SMART has demonstrated substantial improvements over IP in terms of both delay and throughput, while being very scalable.

The software overlay used by SMART is shown in Figure 2, where we show various neighbouring overlay nodes that are communicating with each other. The Transmission Agent intercepts packets entering the node and forwards them to the Proxy to be forwarded according to the SMART algorithm. The Reception Agent receives packets from the Proxy, and if they are to be received locally, they are forwarded to the local application. Otherwise, the Proxy forwards them one more hop according to the SMART algorithm. The Proxy generates SPs that are used to monitor the network paths, and forwards them, as well as other passing SPs towards the relevant neighbouring node Proxy. The Proxy also operates the RNN [90] based RL algorithm, with the data that is brought by the SPs.



	IP route	SMART
Melbourne/Gibraltar	390.0	274.6
Narita/Santiago	406.7	253.8
Moscow/Dublin	179.9	81.7
Honk Kong/Calgary	267.1	130.7
Singapore/Paris	322.3	154.1
Tokyo/Haifa	322.6	180.8

Fig. 3. Average Round-Trip Delay measured during a one week experiment that compared the use of the IP protocol with SMART for several intercontinentally distributed overlay nodes. The data shows a distinct reduction of delay when SMART is used.

SMART's data driven intercontinental packet routing scheme regularly over time, say every two minutes, collects round-trip delay data between each overlay node and the other nodes to which it forwards packets. Each overlay node updates the RL agorithm's state, and then updates the state of corresponding RNN [91] whose number of neurons is limited to the number of allowable neighbouring overlay nodes. When it needs to send a packet to a given destination, the PROXY computes the activation probabilities of the corresponing RNN, and selects the next neighbouring overlay node to be used by the packet based on the neuron with highest probability, as described in Section III-C.

Experiments that were run by installing 20 overlay nodes in a large network offered by the NLNOG Ring test-bed [92] are reported in [85]. Measurement results obtained with a fairly long one week experiment are shown in Figure 3, and they exhibit a substantial reduction of round-trip packet delay as compared to the IP protocol, showing the advantage of a measurement based self-aware scheme over a standard commonly used approach. More results related to these experiments can be found in [85] where results are also reported

using overlays on the Amazon Cloud [93]. They all confirm the superiority of the self-aware approach for improving both delay and throughput in the network.

B. Self-Aware Scheduling of Tasks in the Cloud

Static algorithms for task allocation [94], [95] have long been preferred due to their low overhead and simplicity. However, they are only suitable for stable environments, and cannot easily adapt to dynamic changes in the Cloud [96]. Dynamic algorithms [97], which can sometimes be quite complex, use the applications' characteristics prior to, and at, run-time, but often result in overheads that also cause performance degradations. Thus, typically dynamic adaptive schemes are evaluated through simulations [98] rather than in practical experiments. An example of such a fairly complex, nature inspired task assignment and load balancing algorithm can be found in [99]. Complex scheduling schemes that also use multiple hosts or servers, must often be avoided because of the synchronisation issues that can result in significant slowdowns [100].

Thus in this section, we turn to the exploration of how self-awareness can be applied to the design of adaptive schemes that exploit on-line measurement and take decisions with low computational overhead to assign tasks to Cloud services [101], [102].

The experimental results that are reviewed in this are based on the work in [103], [104] that develops a self-aware Task Allocation Platform (TAP), implemened asa portable Linux based system that dynamically allocates user tasks to available servers, exploiting online performance measurements of task and system performance. TAP attempts to meet the workload's Service Level Agreements (SLA), and it can support both static and dynamic allocation and load balancing schemes [105]. It collects measurements to provide performance reports, and exploits these measurement results to take adaptive decisions.

TAP is used to allocate tasks to a collection of host servers, some of which may be distant from the others and remotely accesible through the Internet, while others may be collectively accessible as a Cloud server, as shown in Figure 4. It runs on a host server, and embeds measurement agents into each host server that it uses, either in a Cloud, or at individual servers, to observe the system's state. The great variety of short, medium and long tasks, and their different measuerd resource requirements are shown on the left-hand-side of Figure 5, while on the right-hand-side of the same figure we show how different applications may have distinct service-level or QoS requirements that need to be respected, as discussed in [104].

These observations are then collected by smart packets (SPs), as described in Section III-C. The SPs are forwarded by TAP at regular intervals to identify the sub-systems which provide better performance, and each SP generates an ACK packet that comes back to TAP and carries the required measurement data. This provides TAP with a constant flow of information that TAP can use in its decision making.

The same TAP platform has been used to compare different task scheduling schemes, in order to see whether a self-aware approach would have distinct performance advantages. Although the work in [104] considers a larger set of task scheduling algorithms, including some that are based on a queueing analysis of the expected response times, here we will summarize results obtained with three schemes:

- (a) A round-robin allocation of incoming tasks to distinct hosts, so that irrespective of their size or of the host servers' workload, the tasks are shared equally among the servers. Round-robin provides a simple and practical baseline for comparison.
- (b) A "sensible" scheme [106] that allocates tasks to server i among a set of S servers, with a probability p_i that is inversely proportional to the measured average response time R_i, which includes the queueing and service delays at the server i:

$$p_i = \frac{\frac{1}{R_i}}{\sum_{j=1}^{S} \frac{1}{R_j}} \ , \tag{1}$$

  so that the system tends to load more those servers that provide better service. The assignments are randomized using a random number generator with the probability (1) for server i. Note that this algorithm will yield an average response time of:

$$R = \sum_{j=1}^{S} p_i.R_i = [\sum_{j=1}^{S} \frac{1}{R_j}]^{-1} \ . \tag{2}$$

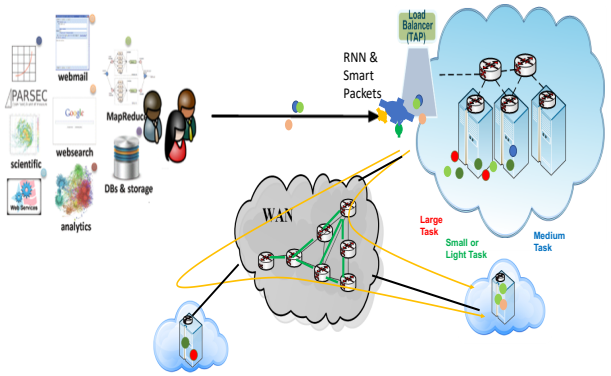
  This algorithm also exploits the self-aware capabilities of TAP because it assigns load to a particular server as a function of *all* of the servers' ongoing measured performance.
- (c) Finally, a self-aware scheme that runs the RL algorithm in Section III-C, based on the observed task *total execution times, including any wait time* measured at the different servers.

This last approach differs from the "sensible" scheme in several ways. While the sensible scheme tends to spread the load over all of the servers but varies the fraction of tasks being assigned as a function of measured QoS, even if some of the servers provide poor performance, the last approach focuses at any time on a small set of the "best ones", and can stick to one "best" host server for some time, until that host server may become itself overloaded and provide poor performance. Furthermore, the use of equation (5) to update the RL scheme, offers an in-built technique to encourage changes in decision making by "forgetfulness". This makes the self-aware scheme (c) more responsive to new data, and more reactive with its ability to focus one the *current* small number of host servers that are able to provide the best performance.

Experimental data from [104] summarized in Figure 6, shows that the RL based task assignment scheme (c) results in significantly lower average response time (y-axis) for tasks at different levels of load, represented by the task arrival rates in the x axis.

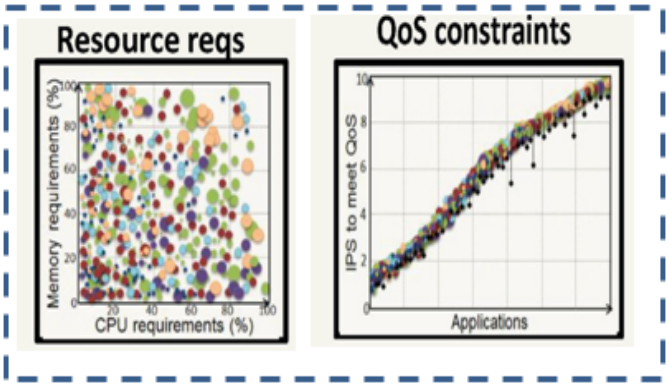
C. SDN with the Cognitive Packet Network

The Cognitive Packet Network (CPN) protocol has been incorporaed into SDN [71] for QoS driven routing. It uses



Overall Architecture of the Self-Aware Distributed Cloud Task Allocation Platform (TAP)

Fig. 4. Overall architecture and operation of the Task Allocation Platform which distributes tasks locally to a Cloud server, or distributes jobs remotely via the Internet to other Fog and Edge servers. On the tight-hand-side of the figure we see that tasks may be of different types and may have wide ranging requirements in terms of memory occupancy and task execution time, while their QoS requirements may also vary widely.



Task Resource Requests and QoS Constraints

Fig. 5. Tasks may be of different types and their Resource Requirements may vary in terms of memory occupancy and task execution time. Their QoS requirements and Service Level Agreements may also vary widely.

smart packets (SPs) which are sent out by each node e to the destination to measure the round-trip delay $D(f,e)$ from e to the destination node of the flow f. The packet loss rate $L(f,e)$ is measured by comparing the rate at which packets are forwarded for f and the rate at which ACK packets return. Similarly, at any node u of the network it is possible to measure the power consumption in watts π_u, as well as the total traffic rate at the node λ_u, in packets/second, so that the average energy consumption per packet at the node is $E_u = \frac{\pi_u}{\lambda_u}$. This data is also collected by SPs.

$E(f,e)$, the total average energy consumed by a packet of flow f from node e until the destination, is then the sum of the energy consumed at each node from e to the destinatin.

The CPN approach uses a *Goal Function*, which describes the objective that the self-aware system is trying to minimize, and whose value can be measured. At any node e, for any flow f travelling through it, the Goal Function $G(f,e)$ can include both the effect of delay $D(f,e)$, Energy Consumption $E(f,e)$

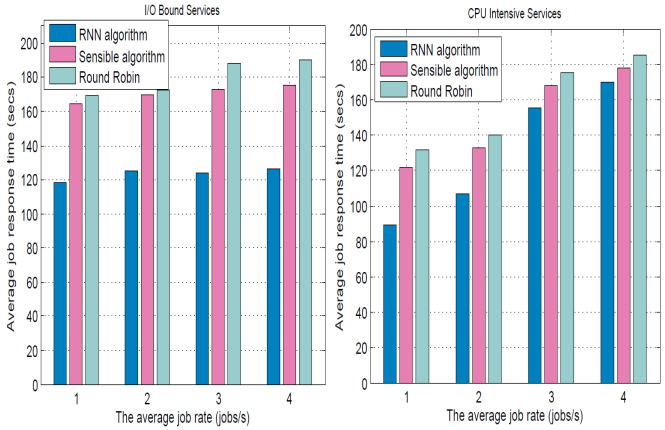


Fig. 6. Average response time for tasks that allocated to servers using RL and the RNN based algorithm (marked RNN) as compared to the Average Response Time observed by using Round-Robin scheduling and the Sensible algorithm. The x-axis indicates increasing load levels in number of tasks arriving to TAP per unit time. For all load levels, the self-aware system that uses the Cognitive Packet Network based on Average Response Time, provides the best performance. The "sensible" scheduler is the second best, while the Round-Robin heuristic offers the worst performance.

and packet loss rate $L(f,e)$:

$$\begin{aligned} G(f,e) &= [b.D(f,e)+(1-b).E(f,e)][1-L(f,e)] \\ &\quad +L(f,e)[bD(f,e)+(1-b)E(f,e), \qquad (3) \\ &= \frac{bD(f,e)+(1-b)E(f,e)}{1-L(f,e)}, \qquad (4) \end{aligned}$$

where $0 \leq b \leq 1$ and $(1-b)$ provide a relative weight to the importance given to Delay and Energy. Thus $G(f,e)$ is a composite Goal Function that the CPN tries to minimize as it decides how to forward packets.

From $G(f,e)$, the RL algorithm used by CPN computes the "reward" $R(f,e) = [G(f,e)]^{-1}$ at each node e using only locally available information that is collected by SPs, and carried back by the acknowledgement packet (ACK) that corresponds to each SP. Each time such an ACK arrives at e a new value $R_l(f,e)$ becomes available at e, where l is the integer describing the successive values of the reward. The RL algorithm will first update the quantity:

$$\theta_l = \alpha\theta_{l-1} + (1-\alpha)R_l, \ 0 \leq \alpha < 1, \qquad (5)$$

so that θ_l describes the historical behaviour of the reward, and tells how well the network has been doing, and a large value of θ_l represents "good" behaviour.

The core decision element is the RNN [107], where each neuron corresponds to a distinct outgoing link of a router. Note that hardware implementations of the RNN have also been suggested [108], [109], and such additional hardware can potentially be installed in a routing engine.

For the case of task allocation discussed in Section III-B, each neuron corresponds to a distinct server that may be assigned to a task. The RNN has three useful properties:

1) It is a "recurrent" model, so that each neuron is interconnected with all other neurons, and allowing the RNN to represent competition between neurons which

recommend the choice of using different outgoing links of the network.
2) For a given numerical input and a given set of weights, the RNN state exists and is unique [110], i.e. we are guaranteed to find a single solution to the state equations, which will be identical if the initial data is the same. This characteristic is very important for the reproducibility of the results, and for the explainability of experimental outcomes.
3) The RNN state is easily computed from a non-linear fixed-point computation.

The RL algorithm will then compute a set of RNN [110] connection weights as follows.

An N neuron RNN will be used, where N is the number of outgoing links for node e. With each outgoing link i from the node we associate te neuron i whose state is the "excitation probability" q_i. Let the RNN weights be the non-negative real numbers $W^+_{ij},\ W^-_{ij}\ \leq 0$ for $i,\ j \in \{1,\ \ldots\ , N\}$. From RNN theory [110] we know that:

$$q_i = \frac{\lambda^+_i + \sum_{j=1}^{N} q_j W^+_{ji}}{r_i + \lambda^-_i + \sum_{j=1}^{N} q_j W^-_{ji}}, \tag{6}$$

where $r_i = \sum_{j=1}^{N} [W^+_{ij} + W^-_{ij}]$ is the "total firing rate" of the neuron i, and $\lambda^+_i,\ \lambda^-_i$ are, respectively, the arrival rate of excitatory and inhibitory spikes to neuron i from outside the neuron i. These rate parameters are set so that when all connection weights are of equal value, all the neurons in the RNN have an excitation probability of $q_i = 0.5$ representing an equal choice among all outgoing links.

The RNN's weights are updated as follows:

$$\begin{aligned}
&If\ R_l \geq T_{l-1},\ then\ for\ j \neq k\\
&\forall i \neq k,\ W^+_{ik} \leftarrow W^+_{ik} + R_l,\ W^-_{ij} \leftarrow W^-_{ij} + \frac{R_l}{N-2},\\
&If\ R_l < T_{l-1},\ then\ j \neq k\\
&\forall i \neq k,\ W^-_{ik} \leftarrow W^-_{ik} + R_l,\ W^+_{ij} \leftarrow W^+_{ij} + \frac{R_l}{N-2},
\end{aligned}$$

where the division by $N-2$ is due to the fact that we are excluding the node I from which the SP initially arrived since we will not send the SP back, and also not increasing the inhibitory weights of the winner nodes when $R_l \geq \theta_{l-1}$, nor will we increase the excitatory weights of the loser nodes when $R_l < \theta_{l-1}$.

Then we also renormalize the weights to avoid having weights that indefinitely increase or decrease:

$$r^*_i \leftarrow \sum_{j=1}^{N} [W^+_{ij} + W^-_{ij}], \tag{7}$$

$$W^+_{ij} \leftarrow W^+_{ij} \frac{r_i}{r^*_i},\ W^-_{ij} \leftarrow W^-_{ij} \frac{r_i}{r^*_i}. \tag{8}$$

Finally we calculate all the q_i from equation (6), select the new output link for flow f at node e by selecting the new output link k^*. Note that the node from which a SP entered the current node (where the next-hop decision is being taken) will not be used as the next hop since a SP is not allowed to head backwards along its path, so that the decision at a given node will only cover $N-1$ other nodes. Thus if I is the incoming link of a packet to be forwarded, we will choose the new outgoing link (or next hop) as being k^* as follows:

$$k^* = \text{argmax}\{q_i :\ i \neq\ I,\ 1 \leq i \leq N\}. \tag{9}$$

IV. Self-Aware Network for QoS, Security and Energy Consumption

In this section we describe the example of a practical working system that embodies many of the concepts and techniques that were described in the previous sections, with the objective of showing how they can be used in a self-aware network, to optimize the QoE which was discussed in Section I-B, and combines in the same common a goal function three key components of QoE which are security, QoS and energy consumption.

Many networks use low-end devices which may be battery operated. Some of them rely on sources of intermittent harvested energy. Thus low energy consumption is a significant issue [111]. Furthermore some attacks, such as "denial of sleep" and battery attacks, directly aim at depleting rapidly the energy stored in battery operated devices to disable the sensors and their networks [112], [113]. In addition, the overall energy consumption load due to networks, routers [114] and Clouds [28] are also a critical issues.

A. System Architecture

The System Architecture that we consider is shown in Figure 7 where we show several Smart Forwarding Elements or Routers (SFE) connected to each other. Each SFE can be connected to several fixed or mobile devices, or to gateways that themselves are connected to several devices (as at the bottom of the figure). Some of the SFEs will typically be connected to Cloud Servers (as at the top right-hand side of the figure) which may also consist of one or more Fog servers. Some SFEs can be connected to an IoT or other device which comes under attack (as at the bottom right-hand-side of the figure), and some SFEs may be connected to Honeypots (H in the middle of the figure) which collect and analyse attack data. Attacks are also detected by software at the SFEs and IoT gateways that carry out attack detection schemes similar to the work presented in [115].

The QoS, Energy and Security data is being constantly collected throughout the network about the gateways and SFEs via Smart Packets (SP), and is provided to the "Controller and Routing Engine" (top of the figure), operating as a standard OpenFlow SDN Controller to compute paths for the active, using the CPN algorithm. We call it the Smart Routing Engine (SRE) whose is to update paths for the flows that are being carried in the network from source-destination pairs so as to minimize the Goal Function, and then communicate these paths to the SFEs using Openflow [116], [117].

Our SDN Controller uses CPN routing [65] described in Section III-C, that is implemented with the RNN [107] and RL. Though such techniques appeared to be excessively advanced in the past, they are now attracting serious attention from industry [118]. The SDN Controller is an extension of a standard SDN network, whose principal components are:

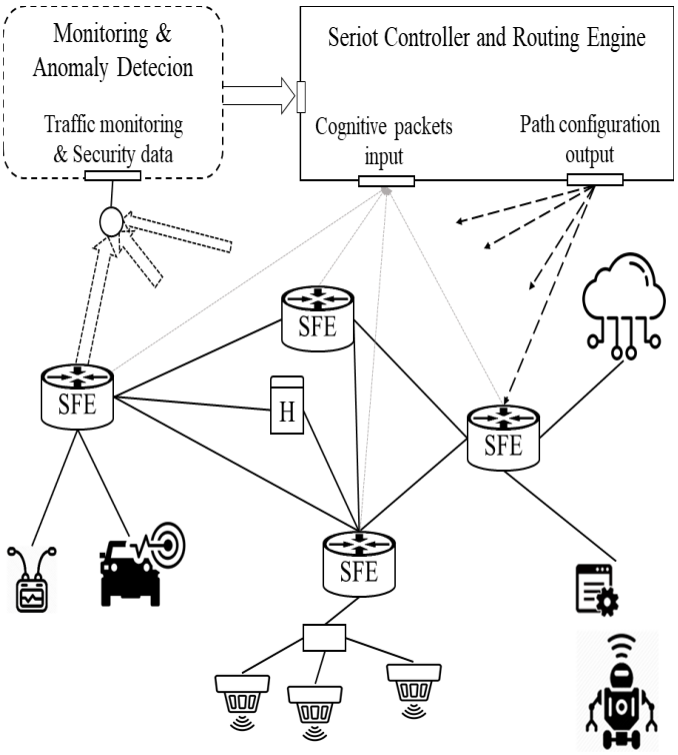


Fig. 7. Overview of the networked architecture.

1) The Smart Forwarding Elements (SFE), extending the concept of a SDN forwarder (or switch).
2) The Smart Controller(s) or Routing Elements (SRE), built upon a standard SDN controller which is an open-source, professional grade SDN Controller ONOS [67], as shown in Figure 8.
3) Attack Detectors [119]–[122] that are designed to detect attacks at edge devices and SFEs, and Honeypots [123] that are also installed at SFEs, or edge devices and servers, and used to attract potential attacks and to inform the Controller by sending SPs to the SRE. The SPs will contain an attack detection probability that is included in the Goal Function that the SRE uses for routing control.

The heart of SRE shown in Figure 8 is the Cognitive Routing Module (CRM) which implements the decision making by RNNs [110], [124], responsible for path selection according to the current QoS, security and energy consumption conditions in the network. The algorithm used by the CRM was described in Section III-C. The SPs, traveling from SFE to SFE, gather time-stamps used for QoS evaluation, the current energy consumption at the SFEs (and possibly at edge devices when this information is available), and security data. The Network State DB combines Trust obtained from Anomaly Detection and forwarded in SPs to the SRE by SFEs, delays obtained via SPs, and Energy Per Packet coming from energy consumption and traffic data also carried by SPs. The Path Translation (PT) module of the SRE translates the RNN decisions into path configuration commands, which are subsequently processed by the SRE and sent to specific SFEs using OpenFlow commands.

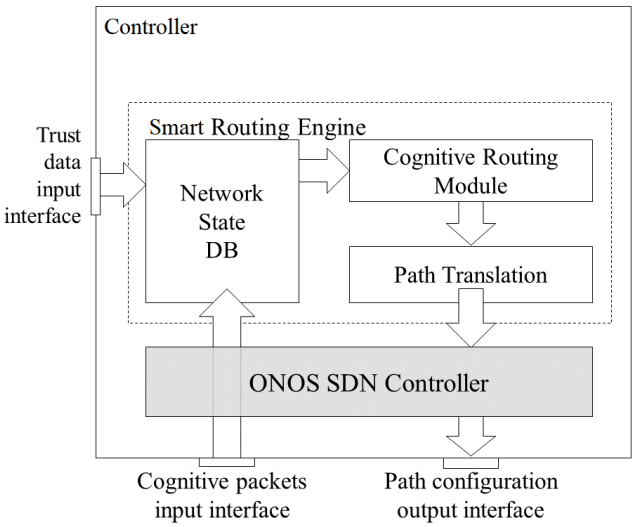


Fig. 8. The Smart Routing Engine SRE acting as an SDN Controller that takes decisions for Goal Based Routing using the RNN and RL.

Each of SFE forwards the network's flows according to Openflow instructions that are sent to it by the SRE. SFEs are also able to collect data in the network with Smart Packets (SPs) which gather security, QoS and energy usage data. On the other hand, SPs are routed by each SFE's internal Cognitive Packet Agent (CPA). The agent unpacks the SP, adds its own data to the list stored inside it, packs it again and forwards it to the next node based on the path set up by the SRE. After the SP has travelled a complete path, it carries the information given it by all SFEs along the path. When a CPA recognizes that the SP it receives has reached the end of its path, it encapsulates the SP and sends it up to a specific data module, the Network State Database (NetStatDB) in the SRE. On the other hand, payload packets travel from SFE to adjacent SFE following a path that was set up by the SRE according to the SDN rules. Thus SFEs handle both user packets and SPs.

Each SFE also sends standard network monitoring data (packet counters, byte counters) to the SRE, and an edge SFE will have client devices connected to it, such as IoT devices or edge servers.

B. Incorporating Security in the Goal Function

To show how new self-aware functionalities can be introduced into the system we have described, we illustrate the case of being able to recognize security issues, and react to them in a timely manner. So suppose that attack detectors placed at SFE's or at edge devices can generate security allerts that are then conveyed by SPs to the SRE. In that case, the SRE needs to be able to act upon these alerts. Thus the Goal Function $G(f, e)$ must be extended and the expression in (3) should be modified to include security issues.

For some SFE or node e, and flow f, we define the *Trust Level* $T(f, e)$, as a non-negative number that says how much we can trust f when it flows through e, or reciprocally, how much f itself can trust e. This quantity can be provided either

by the attack detector or by a honeypot, or it can be based on some *a priori* knowledge concerning the flow f, for instance related to its source and destination nodes. Note that even when a flow is in one direction from source to destination, the source node can still be compromised by the flow via acknowledgement or other control packets that move back from intermediate or destination nodes towards the source or by other flows that pass enter that node .

Similarly, we define $S(f,e)$ the *Sensitivity* of e when it is carrying f. This is a metric which says how tolerant the node e may be to events that are interpreted as a sign of insecurity in f, and $S(f,e)$ is again a non-negative number.

With these concepts in mind, we have to indicate, how we take actions based on these metrics, and we define the *Insecurity Factor* $I(f,e)$ that "separates" e from f and vice-versa:

$$I(f,e) = \begin{cases} 0 & \text{if S(f,e)} \leq \text{T(f,e)}, \\ S(f,e) - TF(f,e) & \text{S(f,e)} > \text{T(f,e)}, \end{cases}$$

and using the notation $[X]^+ = X$ if $X > 0$, and $[X]^+ = 0$ if $X < 0$, we write:

$$I(f,e) = 100.[S(f,e) - T(f,e)]^+, \tag{10}$$

where the multiplicand 100 is a factor used to scale insecurity in comparison to the values of the QoS and Energy Consumption in the goal function. Clearly, if security is a critical issue this scaling factor will be set to a much larger value than the usually measured values of energy consumption per packet, and of delay per packet.

The parameters $S(f,e)$ and $T(f,e)$ can be easily set by the arrival to the SRE from a node e that is equipped with an attack detection mechanism such as the ones described in [122], [125]. The SP will bring to the SRE the probability of an attack on node e, $P_A^N(e)$. Similarly the flow attack probability $P_A^F(f)$ can be defined for some flow f via the probabilities that the nodes u which are on the path f, including the source and destination, have been attacked or compromised, via the expression:

$$P_A^F(f) = 1 - \Pi_{u \in f}[1 - P_A^N(u)]. \tag{11}$$

Let us illustrate the manner in which this may be used with two examples:

- The flow f has a sensitivity to attacks of the order of 20%, i.e. if the probability of an attack being detected is larger than 0.2 then the flow f feels uncomfortable about using e. In this case, we set $S(f,e) = 20$. Now suppose that the attack detector reports an attack probability $P_A^N(e) = 0.90$ – then we set $T(f,e) = 90$ and this results in $I(f,e) = 100$.
- The node e has a sensitivity to attacks of the order of 20%. Again, we set $S(f,e) = 20$, and if the attack detection system provides a probability $P_A^F(f) = 0.5$, then we obtain again $I(f,e) = 100$.

Thus $S(f,e)$ acts as a threshold above which an attack becomes of concern. Note that the non-linearity (10) can be modified to offer a more graduated, rather than step-function, response to security alarms. The *Goal Function* that will be used in our RL [126] based routing scheme is extended from (3) to become:

$$G(f,P) = \frac{bD(f,e) + cE(f,e) + (1-b-c)I(f,e)}{1 - L(f,e)}, \tag{12}$$

where $0 \leq b + c < 1$ are constants that allow us to weigh the relative importance of QoS, Energy and Security.

C. Experimental Setting

To illustrate these ideas, laboratory test-bed was set up with several SFEs that are implemented in lightweight Linux boxes with a quad-core ARMv8 processor running at 1,4 GHz, 4 Gigabit Ethernet interfaces and 2.4GHz and a 5GHz 802.11b/g/n/ac WiFi interface. SFEs were configured to use Ethernet ports as data plane interfaces, and WiFi as a management, monitoring and SRE (Controller) communication interface.

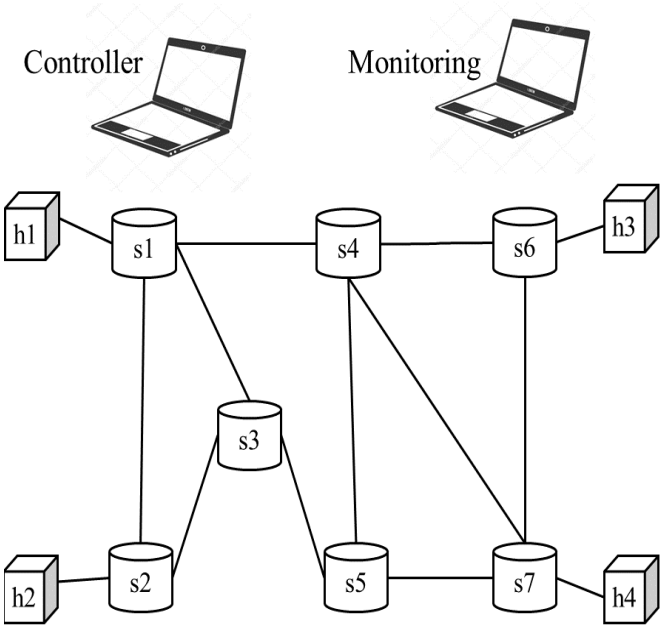


Fig. 9. Topology of the test-bed.

The data plane connections are represented schematically in Figure 9 where:

- The symbols $s1,\ s2,\ ..\ , s7$ denote SFEs.
- The symbols $h1,\ h2,\ h3,\ h4$ denote terminal devices which are each emulated by 633MHz MIPS processors, a 100Mb/s Ethernet port, and 2.4Mhz WiFi connection that is used as a management port.
- The Controller (SRE) is installed on a separate workstation that is connected to the test-bed via WiFi.

We have run many experiments, and can emulate a variety of attacks, such as the effect of a worm which drops packets at random from the queue of a given SFE, the effect of DDoS attacks, or the detection of an intrusion which would result in an increase of the numerical value of the trust metric $TF(.)$, indicating greater danger or mistrust, associated with a flow and a node. We can also represent QoS degradation by sudden

increases in network traffic rates, or by introducing artificial delays or even complete stoppages (e.g. failures) of the SFEs.

However due to space limitations, we focus on examples of the self-aware SDN's, i.e. the SRE's, adaptive reactions to two types of effects: the degradation of QoS due to a significant increase in source to destination packet forwarding delay, and the degradation of security by an increase in the trust metric associated with a given path. The experiments we describe in this section are summarised in the real trace of events shown in Figure 13.

The traffic in the network during the experiments was as follows:

- Every *distinct pair* of clients from the set $\{h1,\ h2,\ h3,\ h4\}$ connected to the network generated the same flow of 20 packets per second each, i.e. of the order of $20-40$ Kb/sec. Thus the network had 12 ongoing connections. Each individual flow's packet rate is compatible (and even quite high) with respect to IoT connections that may be monitoring physical conditions such as temperature of devices or rooms, water flow in pipes, etc.
- Additional traffic came from SPs generated by every edge node at 10 packets per second, and the SRE's management traffic including OpenFlow commands, link discovery, topology discovery, and traffic statistics. We observed that the number of management packets passing through a SFE (router or forwarder), as observed using the Wireshark packet analyser, was about four to five times higher than the SP traffic.
- The measurements we present only concern one of the 12 end-user flows.

The experiments we report illustrate *the ability of the network to adapt as a self-aware system*. In particular:

1) We evaluate the reaction time of the network as a whole, to changes in the observed end-to-end delay of flows, so as to measure the network's reaction delays to sudden deterioration of QoS.
2) We measure the reaction time of the SRE itself to changes in the end-to-end delay of flows.
3) We track the changes to the important parameters of the RL algorithm: R_l and θ_{l-1} defined in (5), measured at successive steps $l = 1,\ 2,\ ....$
4) We measure the reaction time of the SRE to changes of the security conditions represented by the level of trust.
5) We examine the behaviour of the SRE in conditions that combine both the changes in path delay and the changes in the level of trust regarding the flows.

The SRE was programmed to update the network paths every 5 seconds. On the other hand, the measured SRE response times combine the delay related to the RL algorithm, the delay related to the RNN's computation, plus the network delay to receive data via SPs, together with the controller's own decision cycle. The resulting measurement does not give a clear picture of the speed at which the SRE works, but it does provide a realistic view of the overall reaction times perceived from the viewpoint of the network user.

D. System Reaction Times

As indicated in the literature [127], SDN routers introduce delays in decision making and network changes due to the need for the collection of data from the base routers, followed by the computation of a decision, followed in turn by the transfer of the decision to the base routers using the OpenFlow protocol. Thus it was important in this work to actually measure the resulting effect when we use the self-aware routing algorithm.

We measured the packet round-trip time between hosts $h1$ and $h4$ of Figure 9. The best path found by the controller was $\{s1, s4, s7\}$. After 5s, the delay on link $s4 \rightarrow s7$ was changed in three distinct sets of experiments to the artificial values $100ms$, $200ms$ and $300ms$, and each trial was repeated independently 20 times.

We measured the time between the change in the value of the link delay (resulting immediately in an increase in the packet Round-Trip Delay), and the installation of a new path, whose effect is observed by the return of the round trip delay to a lower value, since the new path does not use the link with the longer delay. The results are shown in Figure 10.

Figure 10 summarizes the resulting distribution of the network reaction time, measured with an accuracy of 1 second, over 20 trials for each of the colour-coded delay values (100, 200 and 300 ms). Very interestingly, and as expected, we see that the reaction time is substantially better when the link delay is higher. Since we are considering the networked system as a whole, the reaction times are higher than those observed in Figure 11.

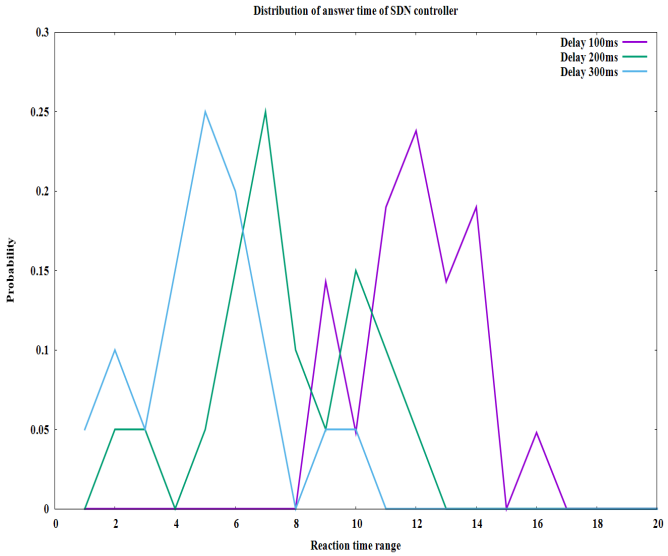


Fig. 10. Measured Probability Distribution of the Reaction Time of the system as a whole, including the network and the SRE, as viewed by the end user.

1) The SRE's Reaction Time to a Large Increase in Link Delay: The next experiments show the performance of the RNN based algorithm in the Routing Engine (including the effect of the SPs which convey the QoS information), measured in a way that eliminates the impact of delays introduced by the SRE. The reaction time here is measured from the instant the

link delay increases to the time when the RNN decision is made within the SRE, but excludes the SDN time needed to change paths and inform the SFE. The relevant probability distributions, each resulting from 20 trials, and for the three values of link delay ($100ms$, $200ms$, $300ms$) are presented in Figure 11. Again we see that the largest change in link delay (dotted green) results in the fastest reaction times, as would be expected. As expected, all times are shorter than those seen in Figure 11 for the routing engine by itself.

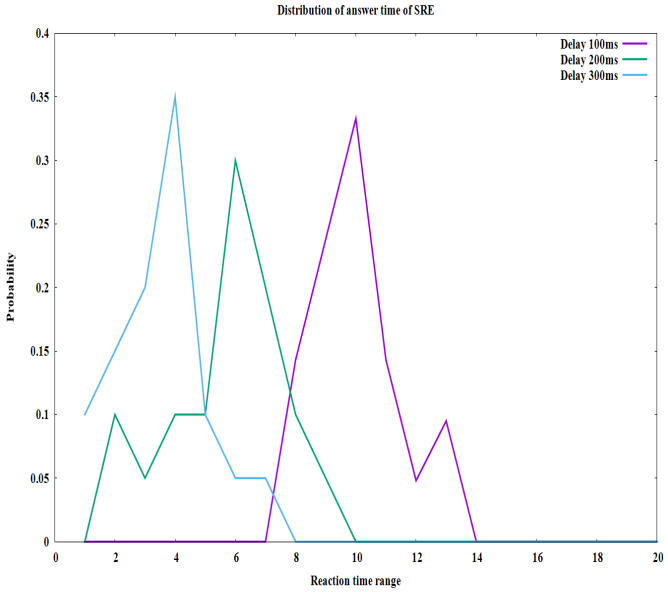


Fig. 11. Distribution of the Routing Engine's (SRE) Reaction Time, including the arrival of SPs to the SRE, and the effect of the RNN based RL algorithm.

E. Plotting the Values of the Reward

The values of R_l and θ_l are plotted versus the index l of successive events in Figure 12. We see that that after each update of θ_l its value follows R_l, but the speed at which θ_l follows R_l will be affected by α in (5). In the present case we used $\alpha = 0.4$. The instants when the two variables *differ* significantly is when the RNN-based algorithm "recommends" a path change to the SRE, which in turn will use the SDN protocol to forward the new paths to the SFEs.

Note also from Figure 12 that in these measurements the system is starting from an empty state, hence the rise in the values of θ_l and R_l, but we see that the increase levels off at the right-hand-side of the figure.

F. Reaction to Changes in Network Delay and Trust Level

The impact of a change in the Trust $T(.,.)$ level is shown in Figure 13, where we present the probability distribution of the time it takes the SRE to respond to a large increase of 100 indicating some form of attack, in the value of $T(f, e)$ for a given e on the path being currently used. We note the roughly one second average delay showing the SRE's capacity to react very rapidly to attacks.

Finally, in Figure 14 we show the observed delay across the network's response to changing delay (i.e. QoS in this case)

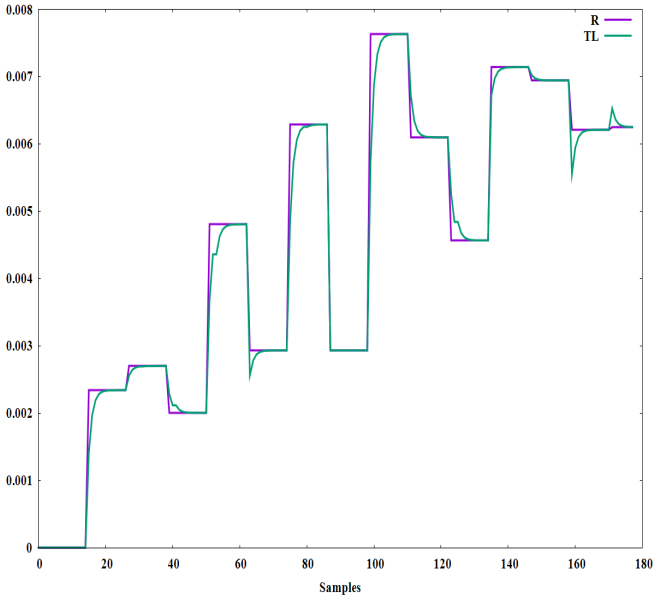


Fig. 12. Values of R_l and θ_l versus l the index of the sample.

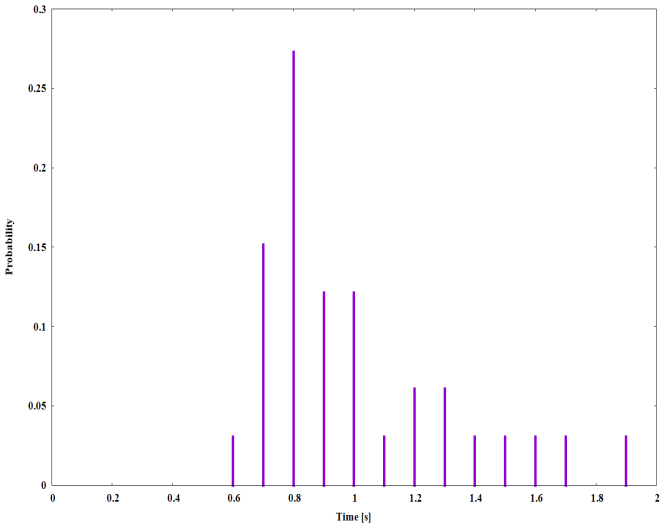


Fig. 13. Distribution of Routing Engine Reaction Time to a sudden substantial change in the Trust Metric $T(.,.)$.

and security conditions. At the first change point starting at the left of the figure, we see that when a very high delay of $300ms$ appeared on the link, the system as a whole reacts in *circa* $2s$, finding a new path that does not use the deteriorated link. Then, as the $T(.,.)$ value rises significantly as an indication of the lack of security of the path that is being used, the network reroutes the traffic via a safe path, which includes a link with high delay. However, a small change in delay, results in a rapid path change to an insecure path with good QoS, and rapidly back. After *circa* 50 seconds the system manages to find a longer path (in number of nodes) which has a relatively low path delay and a good security metric.

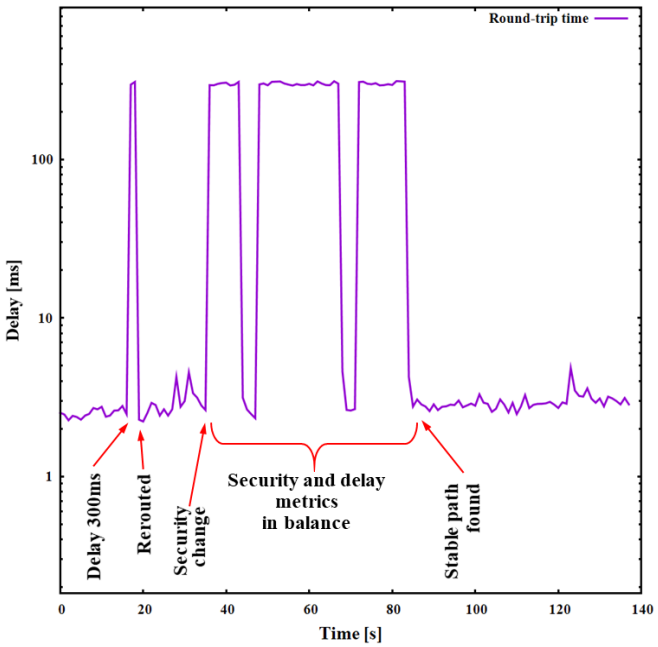


Fig. 14. System reactions to changes in delay and security. Note that the x-axis describes the course of real events in seconds, while the y-axis plots the reaction times of the system as a whole, in milliseconds, to the events that occur along the x-axis.

V. Conclusions and Future Work

The widely distributed and highly interconnected structure of information processing systems, and the ever-changing nature of the underlying infrastructures makes it very difficult to control such systems in a top down hierarchical way. Thus it is important to investigate means to manage and run such systems autonomously, based on self-measurement, local decisions, and self-optimisation. The need for scalability imposes additional constraints, and the multiple of objectives of QoE, including QoS, Energy Savings and System Security offer further challenges. This took us to the concept of self-awareness, which allows a system to monitor itself and take decisions based on objectives that it pursues and observations regarding its own state.

Thus in this paper, we have started with a survey of approaches that incorporate self-awareness into networked systems. We have discussed the premises from cognitive science, and also described early attempts related to self-observation in packet networks which resulted in widely used Ethernet-like systems. We have then discussed an early attempt to bring intelligence and adaptation into networks through the attractive concept of Active Networks and discussed some of the ideas that have then carried over into modern networking research.

Our focus has then turned to more recent work where self-awareness has been implemented in overlay networks and task management systems in the Cloud for optimising system performance. We have reviewed several proposals regarding the combination of recent advances in Software Defined Networks that create programmable controllers to dynamically manage paths in the network, to achieve significant performance improvements. These discussions have been completed with experimental results that show the significant performance improvements that can be obtained when real-time self-aware adaptive management is implemented in an overlay network and in a Cloud.

Then we have presented an approach to introducing self-awareness into SDN through a Cognitive Packet Network (CPN) which has specific performance goals it pursues, and which is implemented through a RL algorithm that is incorporated into SDN controllers. It was illustrated then via a specific imlementation in a multi-hop network test-bed where the SDN controller aims at optimising QoE including QoS, Security and Energy. Experimental results have been shown concerning the responsiveness of the system and its ability to react rapidly to sudden degradation in network delay or in security levels.

This broad panorama, going from historical premizes all the way to a system demonstrator, is meant to entice the reader's curiosity and encourage the reader to investigate this area further. Many things remain yet to be done.

The integration of such techniques into hierarchical structures is a worthwhile direction of research. We have incorporated QoS, Energy and Security together, and these areas are not distinct since they strongly interact: cyberattacks degrade QoS and increase energy consumption, while the optimisation of QoS will itself increase energy consumption through additional computation and communication. Similarly, by reducing energy consumption we may also slow down the system as a whole. Thus all these interactions are quite complex and will require further work.

Similarly, it will be useful to study the best possible machine learning techniques that may be used. For instance, dynamic system management based, not just on short term observation through RL, but also using long-term experience and big data, may be an alternative way to approach these interesting problems. Future work should also make use of predictions from performance models to improve and optimize the design of such networked systems [128], [129].

Acknowledgements

We have benefited greatly from the recommendations of the referees and editors, to whom we are deeply indebted, and from the support of the European Commission under the H2020-IOT-2016-2017 Program, Grant Agreement 780139, for the SerIoT Research and Innovation Action.

References

[1] Juniper-Networks, "Expel complexity with a self-driving network: Soon, your network will adaptively meet your business goals all by itself," 2020. [Online]. Available: https://www.juniper.net/us/en/dm/the-self-driving-network/

[2] J. Apostolos, "Improving networks with artificial intelligence," June 2019. [Online]. Available: https://blogs.cisco.com/networking/improving-networks-with-ai

[3] R. Kompany, "Huawei's 'autonomous driving' mobile networks strategy aims to increase automation and reduce costs," *Knowledge Centre*, December 2018. [Online]. Available: https://www.analysysmason.com/Research/Content/Comments/Huawei-autonomous-network-RMA18/

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning - an Introduction*, ser. Adaptive computation and machine learning. MIT Press, 1998. [Online]. Available: http://www.worldcat.org/oclc/37293240

[5] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. Morgan Kaufmann, 1993, pp. 671–678.

[6] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *Computer Communication Review*, vol. 26, p. 5–18, 1996.

[7] E. Gelenbe, Z. Xu, and E. Seref, "Cognitive packet networks," in *11th IEEE International Conference on Tools with Artificial Intelligence, ICTAI '99, Chicago, Illinois, USA, November 8-10, 1999*. IEEE Computer Society, 1999, pp. 47–54. [Online]. Available: https://doi.org/10.1109/TAI.1999.809765

[8] S. Duval and R. A. Wicklund, *A theory of objective self awareness*. Academic Press, 1972.

[9] R. Descartes, *Discours de la Méthode*. I. Maire, Printer, 1637.

[10] I. Brinck and P. Gärdenfors, "Representation and self-awareness in intentional agents," *Synthese*, vol. 118, no. 1, pp. 89–104, 1999. [Online]. Available: https://doi.org/10.1023/A:1005109414345

[11] K. Bard, "Self-awareness in human and chimpanzee infants: What is measured and what is meant by the mark and mirror test?" *Infancy*, vol. 9, no. 2, p. 191–219, 2006.

[12] P. Richat, "Five levels of self-awareness as they unfold early in life," *Consciousness and Cognition*, no. 4, p. 717–731, 2003.

[13] K. M. Heilman, A. M. Barrett, and J. C. Adair, "Possible mechanisms of anosognosia: a defect in self–awareness," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 353, no. 1377, p. 1903–1909, 1998.

[14] J. L. Bermúdez, "Bodily self-awareness and the will: Reply to power," *Minds and Machines*, vol. 11, no. 1, pp. 139–142, 2001. [Online]. Available: https://doi.org/10.1023/A:1011239215879

[15] G. Harman, "The intrinsic quality of experience," in *Philosophical Perspectives: Action Theory and Philosophy of Mind*, vol. 4. Ridgeview Publishing Company, 1990, pp. 31–52. [Online]. Available: http://www.jstor.org/stable/2214186

[16] N. Seufert et al., "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[17] C. E. Cramer and E. Gelenbe, "Video quality and traffic qos in learning-based subsampled and receiver-interpolated video sequences," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 2, pp. 150–167, 2000.

[18] C. Aracena, S. Basterrech, V. Snásel, and J. D. Velásquez, "Neural networks for emotion recognition based on eye tracking data," pp. 2632–2637, 2015. [Online]. Available: https://doi.org/10.1109/SMC.2015.460

[19] S. Jelassi and G. Rubino, "A study of artificial speech quality assessors of voip calls subject to limited bursty packet losses," *EURASIP J. Image and Video Processing*, vol. 2011, p. 9, 2011. [Online]. Available: https://doi.org/10.1186/1687-5281-2011-9

[20] ——, "A perception-oriented markov model of loss incidents observed over voip networks," *Comput. Commun.*, vol. 128, pp. 80–94, 2018. [Online]. Available: https://doi.org/10.1016/j.comcom.2018.06.009

[21] N. Torres-Cruz, M. E. Rivero-Angeles, G. Rubino, R. Menchaca-Mendez, and R. Menchaca-Méndez, "A window-based, server-assisted P2P network for vod services with qoe guarantees," *Mobile Information Systems*, vol. 2017, pp. 2 084 684:1–2 084 684:18, 2017. [Online]. Available: https://doi.org/10.1155/2017/2084684

[22] E. Gelenbe and Y. Caseau, "The impact of information technology on energy consumption and carbon emissions," *Ubiquity*, vol. 2015, no. June, pp. 1–15, 2015.

[23] C. Lange, D. Kosiankowski, R. Weidmann, and A. Gladisch, "Energy consumption of telecommunication networks and related improvement options."

[24] 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.

[25] E. Gelenbe and T. Mahmoodi, "Energy-aware routing in the cognitive packet network," *Energy*, pp. 7–12, 2011.

[26] E. Gelenbe, "Energy packet networks: adaptive energy management for the cloud," in *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*. ACM, 2012, p. 1.

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

[28] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The computer journal*, vol. 53, no. 7, pp. 1045–1051, 2010.

[29] J. P. G. Sterbenz et al., "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010. [Online]. Available: https://doi.org/10.1016/j.comnet.2010.03.005

[30] E. Gelenbe and G. Loukas, "A self-aware approach to denial of service defence," *Computer Networks*, vol. 51, no. 5, pp. 1299–1314, 2007.

[31] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. 18th ACM Symp. Oper. Syst. Principles (SOSP'01)*, 2001, p. 131–145.

[32] E. Gelenbe and S. Tucci, "Performances d'un système informatique dupliqué," *Comptes rendus de l'Académie des sciences. Série 2, Mécanique, Physique, Chimie, Sciences de l'univers, Sciences de la Terre*, vol. 312, no. 1, pp. 27–30, 1991.

[33] G. Oke, G. Loukas, and E. Gelenbe, "Detecting denial of service attacks with bayesian classifiers and the random neural network," in *2007 IEEE International Fuzzy Systems Conference*. IEEE, 2007, pp. 1–6.

[34] R. Lent, G. Sakellari, and G. Loukas, "Strengthening the security of cognitive packet networks," *Int. J. Adv. Intell. Paradigms*, vol. 6, no. 1, pp. 14–27, 2014. [Online]. Available: https://doi.org/10.1504/IJAIP.2014.059584

[35] E. Gelenbe, "Review of experiments in self-aware networks," in *Computer and Information Sciences - ISCIS 2003, 18th International Symposium, Antalya, Turkey, November 3-5, 2003, Proceedings*, ser. Lecture Notes in Computer Science, A. Yazici and C. Sener, Eds., vol. 2869. Springer, 2003, pp. 1–8. [Online]. Available: https://doi.org/10.1007/978-3-540-39737-3_1

[36] E. Gelenbe and A. Núñez, "Self-aware networks and quality of service," in *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003, Joint International Conference ICANN/ICONIP 2003, Istanbul, Turkey, June 26-29, 2003, Proceedings*, ser. Lecture Notes in Computer Science, O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, Eds., vol. 2714. Springer, 2003, pp. 901–908. [Online]. Available: https://doi.org/10.1007/3-540-44989-2_107

[37] L. Massoulié, A. Kermarrec, and A. J. Ganesh, "Network awareness and failure resilience in self-organising overlay networks," in *22nd Symposium on Reliable Distributed Systems (SRDS 2003), 6-8 October 2003, Florence, Italy*. IEEE Computer Society, 2003, pp. 47–55. [Online]. Available: https://doi.org/10.1109/RELDIS.2003.1238054

[38] K. Paul and D. Westhoff, "Context aware detection of selfish nodes in DSR based ad-hoc networks," in *Proceedings of the Global Telecommunications Conference, 2002. GLOBECOM '02, Taipei, Taiwan, 17-21 November, 2002*. IEEE, 2002, pp. 178–182. [Online]. Available: https://doi.org/10.1109/GLOCOM.2002.1188065

[39] A. Bronstein, J. Das, M. Duro, R. Friedrich, G. Kleyner, M. Mueller, S. Singhal, and I. Cohen, "Self-aware services: Using bayesian networks for detecting anomalies in internet-based services," in *2001 IEEE/IFIP International Symposium on Integrated Network Management, IM 2001, Seattle, USA, May 14-18, 2001. Proceedings*, G. Pavlou, N. Anerousis, and A. Liotta, Eds. IEEE, 2001, pp. 623–638. [Online]. Available: https://doi.org/10.1109/INM.2001.918070

[40] S. Li, "Leveraging a web-aware self-organization map tool for clustering and visualization," in *Web Intelligence: Research and Development, First Asia-Pacific Conference, WI 2001, Maebashi City, Japan, October 23-26, 2001, Proceedings*, ser. Lecture Notes in Computer Science, N. Zhong, Y. Yao, J. Liu, and S. Ohsuga, Eds., vol. 2198. Springer, 2001, pp. 579–583. [Online]. Available: https://doi.org/10.1007/3-540-45490-X_75

[41] J. Mittola and J. G. Q. Maguire, "Cognitive radio: making software radios more personal," *IEEE Personal Communications Magazine*, vol. 6, no. 4, p. 13–18, 1999.

[42] J. Mittola, *Cognitive Radio – An Integrated Agent Architecture for Software Defined Radio*. Diva (Ph.D. Dissertation), Kista, Sweden: KTH Royal Institute of Technology, 1990.

[43] C. Stevenson, G. Chouinard, Z. Lei, W. Hu, S. Shellhammer, and W. Caldwell, "Ieee 802.22: The first cognitive radio wireless regional area network standard," *IEEE Communications Magazine*, vol. 47, p. 130–138, 2009.

[44] E. Gelenbe, R. Lent, and Z. Xu, "Networks with cognitive packets," in *MASCOTS 2000, Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 29 August - 1 September 2000, San Francisco, California, USA*. IEEE Computer Society, 2000, pp. 3–10. [Online]. Available: https://doi.org/10.1109/MASCOT.2000.876422

[45] E. Gelenbe, "Cognitive packet network," *US Patent 09/680184*, October 2004.

[46] E. Gelenbe, P. Liu, and J. Lainé, "Genetic algorithms for route discovery," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 6, pp. 1247–1254, 2006.

[47] N. Abramson, "The aloha system - another alternative for computer communications," in *Proc. 1970 Fall Joint Computer Conference. AFIPS Press*, 1970, pp. 281–285. [Online]. Available: https://people.eecs.berkeley.edu/~pister/290Q/Papers/MAC%20protocols/ALOHA%20abramson%201970.pdf

[48] R. Metcalfe and D. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Communications of the ACM*, vol. 19, no. 7, pp. 395–405, July 1976.

[49] E. Gelenbe, J. Geyl, O. Gibergues, and E. Horlait, "Rśeau local à diffusion de pacquets sur fibre optique: le système xantos," *Rapport de Recherche LRI*, no. 63, 1979.

[50] D. Sagar, "The privatization of inmarsat," in *Proceedings of the IISL 41st Colloquium on the Law of Outer Space*, 1998.

[51] G. Fayolle, E. Gelenbe, and J. Labetoulle, "Stability and optimal control of the packet switching broadcast channel," *J. ACM*, vol. 24, no. 3, pp. 375–386, 1977. [Online]. Available: https://doi.org/10.1145/322017.322019

[52] B. T. An and E. Gelenbe, "Near optimal behaviour of the packet switching broadcast channel," in *The IEEE Computer Society's Second International Computer Software and Applications Conference, COMPSAC 1978, 13-16 November, 1978, Chicago, Illinois, USA*. IEEE, 1978, pp. 728–734. [Online]. Available: https://doi.org/10.1109/CMPSAC.1978.810557

[53] L. Kleinrock, "Packet switching in radio channels: Part i –carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Transactions on Communications*, vol. 23, no. 12, p. 1400–1416, 1975.

[54] E. Gelenbe and I. Mitrani, "Control policies in CSMA local area networks: Ethernet controls," *SIGMETRICS Performance Evaluation Review*, vol. 11, no. 4, pp. 233–240, 1982. [Online]. Available: https://doi.org/10.1145/1035332.1035328

[55] A. T. Campbell, H. G. de Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 29, p. 7–23, 1999.

[56] K. Psounis, "Active networks: applications, security, safety, and architectures," *IEEE Communications Surveys Tutorials*, vol. 2, no. 1, p. 2–16, 1999.

[57] A. Galis, S. Denazis, C. Brou, and C. Klein, *Programmable Networks for IP Service Deployment*. Artech House Inc., 2004.

[58] J. Rubio-Loyola, A. Astorga, J. Serrat, W. K. Chai, L. Mamatas, A. Galis, S. Clayman, A. Cheniour, L. Lefevre, O. Mornard, A. Fischer, A. Paler, and H. D. Meer, "Platforms and software systems for an autonomic internet," in *2010 IEEE Global Telecommunications Conference GLOBECOM*. IEEE, 2010.

[59] M. W. Hicks, J. T. Moore, D. S. Alexander, C. A. Gunter, and S. Nettles, "Planet: An active internetwork," in *Proceedings IEEE INFOCOM '99, The Conference on Computer Communications, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, The Future Is Now, New York, NY, USA, March 21-25, 1999*. IEEE Computer Society, 1999, pp. 1124–1133. [Online]. Available: https://doi.org/10.1109/INFCOM.1999.751668

[60] D. S. Decasper, G. Parulkar, S. Choi, J. Dehart, T. Wolf, and B. Plattner, "A scalable high-performance active network node," *IEEE Network*, vol. 13, no. 1, p. 8–19, 1999.

[61] A. Dollas, D. Pnevmatikatos, and N. Aslanides et al., "Architecture and application of plato, a reconfigurable active network platform," in *Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '01)*, 2001, p. 101–110.

[62] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, 1997.

[63] D. S. Alexander, W. A. Arbaugh, A. D. Keromytis, and J. M. Smith, "Security in active networks," in *Secure Internet Programming, Security Issues for Mobile and Distributed Objects*, ser. Lecture Notes in Computer Science, J. Vitek and C. D. Jensen, Eds., vol. 1603. Springer, 1999, pp. 433–451.

[64] D. S. Alexander, P. Menage, A. D. Keromytis, W. A. Arbaugh, K. G. Anagnostakis, and J. M. Smith, "The price of safety in an active network," *Journal of Communications and Networks*, vol. 3, no. 1, pp. 4–18, 2001. [Online]. Available: https://doi.org/10.1109/JCN.2001.6596875

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

[66] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, August 2012, p. 13–16.

[67] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "Onos: Towards an open, distributed sdn os," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/2620728.2620744

[68] R. Masoudi and A. Ghaffari, "Software defined networks: A survey," *J. Netw. Comput. Appl.*, vol. 67, pp. 1–25, 2016. [Online]. Available: https://doi.org/10.1016/j.jnca.2016.03.016

[69] M. Channegowda, R. Nejabati, and D. Simeonidou, "Software-defined optical networks technology and infrastructure: Enabling software-defined optical network operations," *J. Opt. Commun. Netw.*, vol. 5, no. 10, pp. A274–A282, Oct 2013. [Online]. Available: http://jocn.osa.org/abstract.cfm?URI=jocn-5-10-A274

[70] P. Fröohlich, E. Gelenbe, and M. P. Nowak, "Smart sdn management of fog services," in *Accepted for publication, Proceedings of the Global IoT Summit (GIoTS2020), 3-5 June 2020*. IEEEXPress.

[71] F. Francois and E. Gelenbe, "Towards a cognitive routing engine for software defined networks," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[72] T. Czachórski, E. Gelenbe, G. S. Kuaban, and D. Marek, "Transient behaviour of a network router," in *Submitted for Publication*, March 2020.

[73] F. Francois and E. Gelenbe, "Optimizing secure sdn-enabled inter-data centre overlay networks through cognitive routing," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2016 IEEE 24th International Symposium on*. IEEE, 2016, pp. 283–288.

[74] S. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *IEEE International Conference on Services Computing, SCC 2016, San Francisco, CA, USA, June 27 - July 2, 2016*, 2016, pp. 25–33. [Online]. Available: https://doi.org/10.1109/SCC.2016.12

[75] C. Qiu, S. Cui, H. Yao, F. Xu, F. R. Yu, and C. Zhao, "A novel qos-enabled load scheduling algorithm based on reinforcement learning in software-defined energy internet," *Future Generation Comp. Syst.*, vol. 92, pp. 43–51, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2018.09.023

[76] E. Gelenbe, J. Domanska, T. Czachórski, A. Drosou, and D. Tzovaras, "Security for internet of things: The seriot project," in *2018 International Symposium on Networks, Computers and Communications, ISNCC 2018, Rome, Italy, June 19-21, 2018*. IEEEXplore, 2018, pp. 1–5. [Online]. Available: https://doi.org/10.1109/ISNCC.2018.8531004

[77] M. Nowak, S. Nowak, J. Domanska, and T. Czachorski, "Cognitive packet networks for the secure internet of things," in *Global IoT Summit*, 2019.

[78] L. Tello-Oquendo, S. Lin, I. F. A. dplb, and V. Pla, "Software-defined architecture for qos-aware iot deployments in 5g systems," *Ad Hoc Networks*, vol. 93, 2019. [Online]. Available: https://doi.org/10.1016/j.adhoc.2019.101911

[79] F. J. Hauck, E. Meier, U. Becker, M. Geier, U. Rastofer, and M. Steckermeier, "A middleware architecture for scalable, qos-aware, and self-organizing global services," in *Trends in Distributed Systems: Towards a Universal Service Market, Third International IFIP/GI Working Conference, USM 2000, Munich, Germany, September 12-14, 2000, Proceedings*, ser. Lecture Notes in Computer Science, C. Linnhoff-Popien and H. Hegering, Eds., vol. 1890. Springer, 2000, pp. 214–229. [Online]. Available: https://doi.org/10.1007/10722515_18

[80] C. K. Toh, *Ad Hoc Mobile Wireless Networks*. Prentice Hall Publishers, 2002.

[81] R. Gotzhein, *Real-time Communication Protocols for Multi-hop Ad-hoc Networks - Wireless Networking in Production and Control Systems*, ser. Computer Communications and Networks. Springer, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-33319-5

[82] E. Gelenbe, "Quality of service in ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 3, p. 203, 2004. [Online]. Available: https://doi.org/10.1016/j.adhoc.2004.04.004

[83] Q. Liang, "Designing power aware self-reconfiguring topology for mobile wireless personal area networks using fuzzy logic," *IEEE Trans. Systems, Man, and Cybernetics, Part C*, vol. 33, no. 3, pp.

390–394, 2003. [Online]. Available: https://doi.org/10.1109/TSMCC.2003.817356

[84] E. Gelenbe and R. Lent, "Link quality-aware routing," in *Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, PE-WASUN 2004, Venezia, Italy, October 4, 2004*, M. Ould-Khaoua and F. Zambonelli, Eds. ACM, 2004, pp. 87–90. [Online]. Available: https://doi.org/10.1145/1023756.1023772

[85] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, p. 575–583, March 2016.

[86] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of internet path selection," *SIGCOMM Comput. Commun.Rev.*, vol. 29, no. 4, p. 289–299, August 1999.

[87] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, April 2005.

[88] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proc. ACM SIGCOMM Workshop on Future Directions Netw. Architecture, Portland, OR, USA, Aug. 30–Sep. 3*, 2004, p. 1–8. [Online]. Available: http://conferences.sigcomm.org/sigcomm/2004/workshop_papers/fdna01-feamster1.pdf

[89] E. Gelenbe, X. Mang, and R. Önvural, "Diffusion based statistical call admission control in atm," *Performance evaluation*, vol. 27, pp. 411–436, 1996.

[90] E. Gelenbe, "Réseaux neuronaux aléatoires stables," *Comptes rendus de l'Académie des sciences. Série 2, Mécanique, Physique, Chimie, Sciences de l'Univers, Sciences de la Terre*, vol. 310, no. 3, pp. 177–180, 1990.

[91] S. Timotheou, "The random neural network: A survey," *Comput. J.*, vol. 53, no. 3, pp. 251–267, 2010. [Online]. Available: https://doi.org/10.1093/comjnl/bxp032

[92] J. Snijders, "The nlnog ring," 2019. [Online]. Available: https://ring.nlnog.net

[93] Amazon and EC2, "Ec2: Amazon elastic cloud," 2019. [Online]. Available: https://www.amazonaws.cn/en/ec2/

[94] A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *Journal ACM*, vol. 32, no. 2, p. 445–465, 1985.

[95] C. Kim and H. Kameda, "An algorithm for optimal static load balancing in distributed computer systems," *IEEE Trans. Computers*, vol. 41, no. 3, p. 381–384, March 1992.

[96] H. Topcuouglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distributed Systems*, vol. 13, no. 3, p. 260–274, March 2002.

[97] X. Zhu, X. Qin, and M. Qiu, "Qos-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters," *IEEE Trans. Computers*, vol. 60, no. 6, p. 800–812, June 2011.

[98] W. Tian, Y. Zhao, Y. Zhong, M. Xu, and C. Jing, "A dynamic and integrated load-balancing scheduling algorithm for cloud datacenters," in *Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst.*, 2011, p. 311–315.

[99] Z. Zhang and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation," in *Proc. 2nd Int. Conf. Industrial Mechatronics Automation*, vol. 2, 2010, p. 240–243.

[100] E. Gelenbe and K. Sevcik, "Analysis of update synchronization for multiple copy data-bases," in *3rd Berkeley Workshop on Distributed Data and Computer Networks*, 1978, pp. 69–90.

[101] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. 5th Int. Joint Conf. INC, IMS IDC*, 2009, p. 44–51.

[102] R. Buyya et al., "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Computing Surveys*, vol. 51, no. 5, pp. 105:1–105:38, 2019. [Online]. Available: https://doi.org/10.1145/3241737

[103] L. Wang, O. Brun, and E. Gelenbe, "Adaptive workload distribution for local and remote clouds," in *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016, Budapest, Hungary, October 9-12, 2016*, 2016, pp. 3984–3988. [Online]. Available: https://doi.org/10.1109/SMC.2016.7844856

[104] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 33–45, 2018.

[105] H. Kameda, J. Li, C. Kim, and Y. Zhang, *Optimal Load Balancing in Distributed Computer Systems*. Springer, 2011.

[106] E. Gelenbe, "Sensible decisions based on qos," *Computational Management Science*, vol. 1, no. 1, pp. 1–14, 2003.

[107] ——, "Random neural networks with negative and positive signals and product form solution," *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.

[108] H. Abdelbaki, E. Gelenbe, and S. E. El-Khamy, "Analog hardware implementation of the random neural network model," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 4. IEEE, July 2007, pp. 197–201.

[109] M. Badaroglu, U. Halici, I. Aybay, and C. Cerkez, "A cascadable random neural network chip with reconfigurable topology," *Comput. J.*, vol. 53, no. 3, pp. 289–303, 2010. [Online]. Available: https://doi.org/10.1093/comjnl/bxp036

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

[111] A. J. Jara, P. Lopez, D. Fernandez, M. A. Zamora, A. F. Skarmeta, and L. Marin, "Evaluation of bluetooth low energy capabilities for tele-mobile monitoring in home-care," *Journal of Universal Computer Science*, vol. 19, no. 9, pp. 1219–1241, 2013.

[112] E. Gelenbe and Y. M. Kadioglu, "Energy life-time of wireless nodes with and without energy harvesting under network attacks," in *Advances in Cyber-Security: An ISCIS International Workshop*. Springer, 2018.

[113] K. Kalkan and S. Zeadally, "Securing internet of things (iot) with software defined networking (sdn)," *IEEE Communications Magazine*, November 2017.

[114] G. Sakellari, C. Morfopoulou, and E. Gelenbe, "Investigating the tradeoffs between power consumption and quality of service in a backbone network," *Future Internet*, vol. 5, no. 2, pp. 268–281, 2013.

[115] G. Öke, G. Loukas, and E. Gelenbe, "Detecting denial of service attacks with bayesian classifiers and the random neural network," in *FUZZ-IEEE 2007, IEEE International Conference on Fuzzy Systems, Imperial College, London, UK, 23-26 July, 2007, Proceedings*. IEEE, 2007, pp. 1–6. [Online]. Available: https://doi.org/10.1109/FUZZY.2007.4295666

[116] "Openflow switch specification, version 1.3.5," *Open Networking Foundation*, March 2015. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.5.pdf

[117] K. Phemius and M. Bouet, "Monitoring latency with openflow," in *Network and Service Management (CNSM), 2013 9th International Conference on*, October 2013, p. 122–125.

[118] S. Salam, "Weight initialization for random neural network reinforcement learning, appl. no. 15/718,901," *United States Patent Application US2019/097912AI*, March 2019.

[119] J. Du, C. Jiang, E. Gelenbe, L. Xu, J. Li, and Y. Ren, "Distributed data privacy preservation in iot applications," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 68–76, 2018. [Online]. Available: https://doi.org/10.1109/MWC.2017.1800094

[120] J. Augusto-Gonzalez et al., "From internet of threats to internet of things: A cyber security architecture for smart homes," in *24th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD 2019, Limassol, Cyprus, September 11-13, 2019*. IEEE, 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CAMAD.2019.8858493

[121] P. Natsiavas et al., "Comprehensive user requirements engineering methodology for secure and interoperable health data exchange," *BMC Med. Inf. & Decision Making*, vol. 18, no. 1, pp. 85:1–85:16, 2018. [Online]. Available: https://doi.org/10.1186/s12911-018-0664-0

[122] O. Brun, Y. Yin, and E. Gelenbe, "Deep learning with dense random neural network for detecting attacks against iot-connected home environments," *Procedia Computer Science*, vol. 134, pp. 458–463, 2018.

[123] L. Delosieres and D. Garcia, "Infrastructure for detecting android malware," in *Proc. 28th Int. Symp. on Computer and Information Sciences (ISCIS'13), volume 264 of Lecture Notes in Electrical Engineering*, E. Gelenbe, Ed. Springer, Cham, October 2013.

[124] E. Gelenbe and A. Stafylopatis, "Global behavior of homogeneous random neural systems," *Applied mathematical modelling*, vol. 15, no. 10, pp. 534–541, 1991.

[125] L. Delosieres and A. Sanchez, "Droidcollector: A honeypot for collecting and classifying android applications," in *Information Sciences and Systems 2014 - Proc. 29th International Symposium on Computer and Information Sciences (ISCIS'14)*, T. Czachórski, E. Gelenbe, and R. Lent, Eds. Springer, Cham, October 2014. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-09465-6_19

[126] U. Halici, "Reinforcement learning with internal expectation for the random neural network," *Eur. J. Oper. Res.*, vol. 126, no. 2, pp. 288–307, 2000. [Online]. Available: https://doi.org/10.1016/S0377-2217(99)00479-8

[127] A. Nguyen-Ngoc, S. Lange, S. Geissler, T. Zinner, and P. Tran-Gia, "Estimating the flow rule installation time of sdn switches when facing control plane delay," in *Measurement, Modelling and Evaluation of Computing Systems, 19th International GI/ITG Conference, MMB 2018, Proceedings*. LNCS, Springer, 2018.

[128] T. Czachorski, A. Domanski, J. Domanska, M. Pagano, and A. Rataj, "Delays in ip routers, a markov model," in *Computer and Information Sciences ISCIS 2016*, vol. 659. Springer CICIS, 2016, pp. 185–192.

[129] A. Domanski, J. Domanska, M. Pagano, and T. Czachorski, "The fluid flow approximation of the tcp vegas and reno congestion control mechanism," in *Computer and Information Sciences ISCIS 2016*, vol. 659. Springer CICIS, 2016, pp. 193–200.

Erol Gelenbe (F'86), Fellow of ACM, IFIP, RSS, IET (London), was born in Istanbul, graduated from TED Ankara Koleji, and was awarded the BS by METU (Ankara), the MS & PhD by Polytechnic Institute of NYU, and the D. ès Sci. degree by Sorbonne Université. Professor at the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, his previous positions include Chaired Professorships at Imperial College, UCF, Duke University, Université Paris-Descartes and Paris-Orsay, and Université de Liège. Renowned for pioneering mathematical models of computer systems and networks, inventing G-Networks and Random Neural Networks, he graduated over 90 PhDs, including 24 women. He has contributed to industry by developments that include the QNAP performance modeling package, the XANTOS fiber-optics local area network, the SYCOMORE distributed voice-packet switch, the manufacturing simulator FLEXSIM, and the C2Agents software. Awards include "honoris causa" degrees from Università di Roma II, Boğaziçi University (Istanbul), and Université de Liège (Belgium), the Parlar Foundation Science Award, Grand Prix France Télécom, French Acad. of Sci., ACM-SIGMETRICS Life-Time Achievement Award, IET Oliver Lodge Medal, "In Memoriam Dennis Gabor Prize", Hungarian Acad. of Sci.. Elected to Academia Europaea, the French National Acad. of Technologies, the Sci. Acad. of Turkey, the Royal Acad. of Belgium, the Hungarian and Polish Acad. of Sci., he is a Chevalier de la Légion d'Honneur, Chevalier des Palmes Académiques, Commandeur du Mérite of France, Commendatore al Merito and Grande Ufficiale dell'Ordine della Stella d'Italia of Italy. Associate Editor of *IEEE Trans. Cloud Comp., Acta Informatica, Performance Eval., Co-Editor-in-Chief of SN Computer Science*, his consultancies include INRIA, France-Telecom, Bull, IBM, Thomson CSF, Bell Labs, BT, General Dynamics UK, Huawei. PI in many EU research projects, coordinator of FP7 NEMESYS and H2020 SerIoT, he also won several grants from NSF and EPSRC.

Joanna Domanska obtained a Master of Engineering degree from the Silesian University of Technology, Gliwice, Poland, in 1994. She received her PhD in Computer Science in January 2005 from the Scientific Council of the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice, and defended the habilitation dissertation in Computer Science in 2015 at the Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology. She has taken part in seven projects for the National Science Centre, Poland, and was project leader in two of them. She is currently an Associate Professor at the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (IITIS-PAN) and member of the Computer Systems Modelling and Performance Evaluation Group since 1994, serving as leader of this group since 2018. She is a member of the SerIoT and the SDK4ED research teams, under the EU H2020 program. Her main areas of research include performance modelling methods for computer networks, particularly the modelling of network traffic intensity and the QoS related problems.

Piotr Frohlich received the Bachelor of Engineering degree from the Silesian University of Technology in 2020, and in 2018 he had already started research at the Institute of Theoretical and Applied Informatics of the Polish Academy of Sciences (IITIS-PAN), in Gliwice, Poland, on network security and performance as part of the EU H2020 Research and Innovation project SerIoT. His field of interest revolves around artificial intelligence, network security and network efficency, with emphasis on the connections between these fields. Currently, he is working towards his Master of Engineering degree at the Silesian University of Technology.

Mateusz P. Nowak was born in 1972. In 1996, he received MSc degree in Computer Science from the Silesian University of Technology in Gliwice. He then worked for as a software developer for Polish and German companies. In 2006 he defended his PhD in the field of Computer Networks at the Institute of Theoretical and Applied Computer Science of the Polish Academy of Sciences (IITIS-PAN). Then he joined the team on Systems Modeling and Performance Evaluation of Computer System at IITIS-PAN as an Assistant Professor, and has participated as a researcher or Principal Investigator in several scientific and industry R & D projects. His research interests have covered issues rellated to distributed programming, event-driven simulation and novel solutions for the Future Internet. His current research interests include various aspects of devices and communication technologies in the Internet Things, with an emphasis on network security.

Sławomir Nowak was born on November 9th, 1974 in Zabrze city, Silesia, Poland. He received his MSc diploma from the Silesian University of Technology, Faculty of Computer Science, in the field of Computer Systems and Networks in 1999. In 2000 he joined the System Modeling and Performance Evaluation of Computer Systems Group at the Institute of Theoretical and Applied Informatics, Polish Academy of Science (IITIS-PAN). In 2005 he completed his Ph.D. dissertation in Computer Science, and joined ITIS-PAN as an Assistant Professor. Since then, his research interests lie in the area of discrete event simulation, sensor networks, the Future Internet, and the IoT. He has collaborated actively with commercial companies in R&D projects in many different areas regarding the application of computer networks and information systems. He is currently active in the EU H2020 Research and Innovation Project SerIoT.