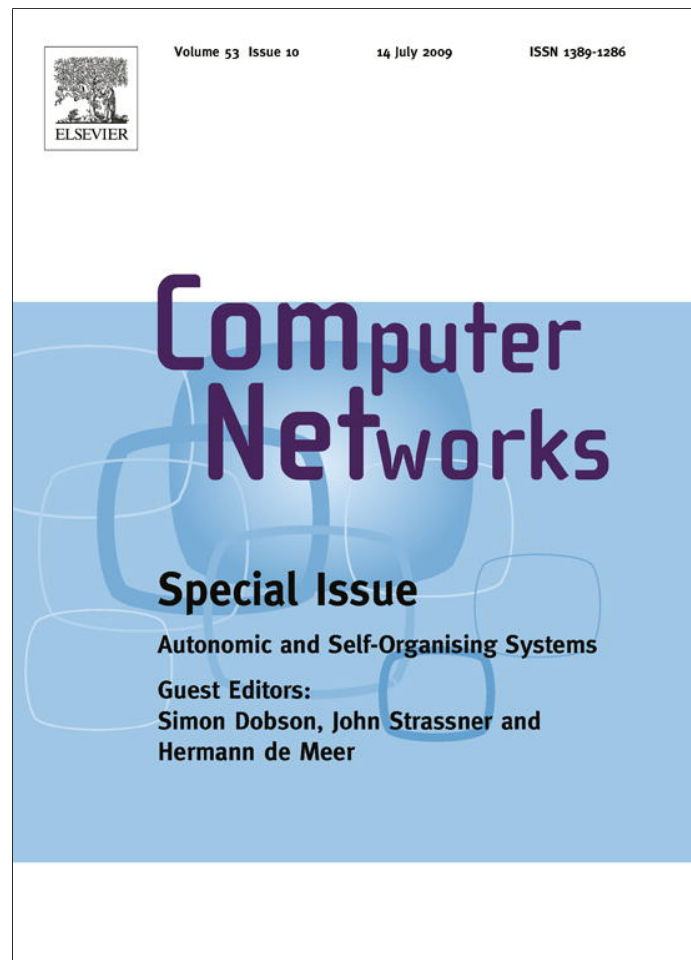


Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.



This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

<http://www.elsevier.com/copyright>



Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Policy-constrained bio-inspired processes for autonomic route management

Sasitharan Balasubramaniam, Dmitri Botvich, Brendan Jennings*, Steven Davy, William Donnelly, John Strassner

TSSG, Waterford Institute of Technology, Cork Road, Waterford, Ireland

ARTICLE INFO

Article history:
Available online 1 October 2008

Keywords:
Autonomic networking
Autonomic communications
Bio-inspired algorithms
Policy-based network management
Survivability
Routing

ABSTRACT

Autonomic networking systems must be designed to achieve an appropriate balance between the operation of decentralized algorithms and processes that seek to maintain optimal or near-optimal behavior in terms of global stability, improved performance and adaptability, robustness and security, with the requirement for top-down control of the system by humans to ensure business goals are met. Taking a communications networking survivability case study, we show how the operation of decentralized algorithms, inspired by the operation of biological systems, can be controlled and constrained through the deployment of management policies authored by network administrators. We present survivability-related routing algorithms (inspired by chemotaxis, reaction-diffusion and quorum sensing biological processes) which work together to effectively reconfigure network resources when transient link failures occur and demonstrate how these algorithms can be re-parameterized via policies to improve performance given prevailing network conditions. Simulation results show how the combined operation of these algorithms, as controlled by policies, allows the network to react well to survive link failure events.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

As communications networks have become more dynamic, more heterogeneous, less reliable and larger in scale, researchers and industry practitioners have started to actively investigate how best to engineer systems embodying autonomic principles. The aim is to simplify network management processes by automating and distributing the decision making processes involved in optimizing network operation. This allows expensive human attention to focus more on business logic and less on low-level device configuration processes, with consequent revenue savings and increased efficiency.

Much of the work to date on autonomic network management has focused on the deployment of decentralized

processes and algorithms into the network infrastructure that seek to maintain optimal or near-optimal behavior in terms of global stability, improved performance and adaptability, robustness and security. As described by Babaoglu et al. [1], many of these processes and algorithms can be profitably modeled on various biological processes found in the natural world. However, to ensure that they act in accordance with business goals, we argue that such processes and algorithms should themselves be modeled so that their operation can be automatically configured via appropriate management policies. This allows the human administrators of the network to explicitly constrain the behavior of the processes and algorithms via appropriate parameterization, so that the system operates in accordance with (changing) business goals and/or changing operational context. Integrating models of the algorithms and processes in system information models allows specification of the impact their parameterization has on

* Corresponding author. Tel.: +353 51302917.
E-mail address: bjennings@tssg.org (B. Jennings).

services, products, customers, and ultimately on the revenue generated by the system.

In this paper we discuss how policy-constrained bio-inspired algorithms and processes for routing can allow a network to gracefully react to transient link failures, which are described by Iannaccone et al. [2] as the primary cause of service degradation in current networks. More specifically, we specify routing algorithms and processes based on biological principles and describe how they can be integrated within a policy-based network management framework for autonomic networking. We show how the operation of the algorithms under the control of management policies allows network elements to react to link failures by cooperatively re-routing traffic in a manner that ensures prioritized services are impacted minimally.

The paper is structured as follows. In Section 2 we briefly summarize related work in the areas of autonomic computing and communications, bio-inspired algorithms, policy-based network management and network survivability. Section 3 introduces our policy-based network management architecture for control of bio-inspired routing processes, whilst Section 4 specifies the bio-inspired routing algorithms and processes whose operation is controlled and constrained via policies. Section 5 outlines how the operation of these algorithms is modeled in the context of a wider information model, which is then used to (semi-)automatically generate appropriate policy authoring and analysis tools. Section 6 describes how our prototype implementation was used to explore a case study of traffic re-routing following a link failure event. Finally, Section 7 summarizes the work and outlines areas for further exploration.

2. Related work

2.1. Autonomic computing and autonomic networking

The term *autonomic computing* was coined by IBM as an analogy of the autonomic nervous system, which maintains homeostasis (essentially maintaining equilibrium of various biological processes) in our bodies without the need for conscious direction. Autonomic computing attempts to manage the operation of individual pieces of IT infrastructure (such as servers in a data center), through introduction of an *autonomic manager* that implements an autonomic control loop in which the managed element and the environment in which it operates is monitored, collected data is analyzed, and actions are taken if the managed element is deemed to be in an undesired (sub-optimal) state [3]. The autonomic computing vision can be summarized as that of a “self-managing” IT infrastructure in which equipment will have software deployed on it that enables it to self-configure, self-optimize, self-heal and self-protect. That is, it will exhibit what has come to be known as “self-*” behavior. Clearly this is a powerful vision, albeit one that is acknowledged by Kephart [4] as requiring significant advances in the state-of-the-art over a number of years.

In recent years the networking community have extended the vision of autonomic computing from autonomic

management of individual elements to *autonomic networking* – the collective (self-)management of networks of communicating computing elements [5]. Autonomic networking is currently a burgeoning research area, which seeks to integrate results from disciplines ranging from telecommunications network management to artificial intelligence, and from biology to sociology. For a good summary of the state-of-the-art in this fast-moving area, the reader is referred to Dobson et al. [6].

Our previous work has focused on the application of autonomic computing and autonomic networking principles specifically to network management. In particular, we have introduced the FOCAL (Foundation – Observation – Comparison – Action – Learn – Reason) architecture [7,8], which implements two autonomic control loops: a *maintenance control loop* that is used when no anomalies are found (i.e., when either the current state is equal to the desired state, or when the current state of the managed element is moving towards its intended goal); and an *adjustment control loop* that is used when one or more policy reconfiguration actions must be performed, and/or new policies must be codified and deployed. This policy-driven approach contrasts with other published research works in the area of autonomic network management, which typically focus on the development of highly distributed algorithms that seek to optimize one or more aspects of network operation and/or performance, in essence aiming to provide various point solutions incorporating self-management capabilities. In this context, as we discuss in Section 2.2, many researchers are investigating the potential use of biologically inspired algorithms and processes.

2.2. Bio-inspired algorithms

As noted by Dobson et al. [6], complex biological systems “tend to exploit decentralized and uncoupled coordination models, relying primarily on environment-mediated local information transfer.” Examples include homeostasis (the tendency towards stable equilibrium between interdependent elements), chemotaxis (movement of a cell in a particular direction corresponding to a chemical gradient), and stigmergy (indirect communications between organisms through modification of their local environment), all of which have been used as inspiration for development of networking related algorithms. We will briefly describe some representative examples of this kind of work. The SECOAS project [9] developed a clustering algorithm for sensor networks deployed for monitoring environmental conditions in a seabed based on the biological quorum sensing process. Use of the quorum sensing like capability allows sensors with similar activity to form a localized cluster for the transmission of data. Shen et al. [10] employed the concept of hormone signaling in the development of CONRO robots, in which different modules perform different tasks but coordinate their macro-level behavior. Suzuki and Suda [11] applied the analogy of emergent behaviors exhibited by bee colonies in their bio-networking architecture, in which groups of “cyber-entities” mimic bee behaviors such as migration and reproduction. This architecture was later extended [12] to incorporate evolutionary capabilities for self-organizing

network services. Similar techniques have been applied to improve network security; for example Dressler [13] describes approaches based on cell behavior that serve to protect networks during virus attacks.

Whilst work on development of bio-inspired and other algorithms enabling decentralized self-management is crucial, and noting that significant advances have been made, we believe that deployment of these algorithms in networking equipment will not in and of itself be sufficient. Equally important will be the flexible specification and enforcement of the goals these algorithms collectively seek to achieve – goals reflecting the need for the network to successfully deliver services to users. As described in Section 2.3, policy-based network management [14], is widely seen as an appropriate management paradigm to facilitate higher-level, human-specified cognitive decision making and, as such, can be leveraged to help realize the autonomic network management vision.

2.3. Policy-based network management

Policy-based management systems aim to separate the behavioral rules for network management from the code that realizes the functionality of given network devices. Policies are typically formulated as event-condition-actions tuples with the semantics of “on event(s), if condition(s), do action(s).” These rules can be specified by network administrators and deployed directly onto network devices (via configurations applied typically through command-line interfaces), or are maintained by independent “Policy Decision Points,” which specify actions to be enforced when notified of events by the network devices. This paradigm has been applied by numerous researchers and industry players who have developed many policy languages and policy-based management systems, typically targeting very specific application domains, for example access control security policies. Examples of well-known policy languages are Ponder [15], Rei [16], KAoS [17] and XACML [18]. Examples of policy systems include a policy-based multi-domain QoS management architecture for seamless mobility support in UMTS/WLAN environments [19] and the IBM PMAC system for policy-based management of autonomic computing environments [20].

Clearly, large communications networks are hugely complex, dynamic and heterogeneous in nature, thus large numbers of policies must be defined and continuously amended to ensure the network satisfies high-level goals. A major challenge is ensuring that these policies, which are defined by many different policy authors with many kinds of expertise, are consistent with each other and have the desired effect on the network. For this reason much of the ongoing research in policy-based network management focuses on development of effective and extensible algorithms and processes for policy translation/code generation, conflict analysis and policy enforcement. For example, a key concern is detection and resolution of policy conflicts. Lupu and Sloman [21] investigated conflicts of access control policies, which they defined as occurring when a set of simultaneously applicable policies result in multiple decisions being equally applicable. A simple case is when two policies of incompatible modality need to be

enforced in the same situation. Bandara et al. [22] described a formal model of policies based on event calculus that takes into account simple policy events and conditions. Policy conflicts, specified in Prolog, are analyzed and if two policies are deemed to breach a constraint then a conflict is detected. Information about the system is encoded into logical assertions derived from state machines. Broadly similar work on policy conflict analysis has been done in the context of firewall filtering [23] and security policies [24].

A major limitation of existing policy languages and systems is that they only support policy analysis to the degree to which the semantics of policy execution can be described in policy specifications themselves, possibly with the addition of particular application-specific constraint information. We argue that a better approach is to explicitly tie policy languages to system information models and ontologies that describe the structure, behavior and semantics of the managed network. As described in Section 5, policy languages and associated authoring tools can be generated from an information model and the information model and associated ontologies can be harnessed for policy analysis tasks including policy conflict analysis, policy refinement and policy relationship analysis.

In our work we use the DEN-ng information model (outlined in [25]), which is a comprehensive information model for telecommunications, capturing everything from business concepts (e.g., products, service level agreements, and customers) down to low-level device functionality (e.g., packet marking, forwarding, and queuing). It is designed so that it can be readily augmented with vendor specific information and data models; thus it provides a highly flexible and extensible modeling solution. A significant benefit of DEN-ng is its comprehensive policy model, which facilitates the specification of policy representation languages that are tightly coupled to the information model of the network that policies authored in those languages will govern (as discussed below this characteristic is particularly useful for policy analysis). Furthermore, we can apply the DEN-ng policy continuum [26], in which policies at different levels of abstraction are organized in stratified business, system, network, device and device instance views – mirroring the different constituencies of people that will work together to define and deploy the policies that provide a product or service.

Whilst policy-based network management has been widely researched, to our knowledge little work has been done to date on integrating distributed self-management algorithms with policy-based management. This step will allow policies to re-parameterize such algorithms to change their behavior in a manner that is better suited to changed business goals and/or operational context. We believe this step, a key focus of this paper, is essential to providing a solution that balances the need for explicit control over network operation with the benefits of highly efficient and robust self-management algorithms and processes.

2.4. Network survivability

Survivability of networks when unplanned outages occur has long been a challenging research topic, with

numerous proposed solutions. The various solutions that have been proposed are not suitable for future communication networks, which will face very dynamic and changing traffic behavior. The common approach towards supporting network survivability is through link-disjoint paths (e.g. primary and alternative paths) [45,27,28]. Using this technique, in the event of any failures, routes will be immediately diverted to backup secondary paths (in what is referred to as a protection switching process), which are pre-computed paths using spare capacity of neighbouring nodes. However, such solutions are not practical, due to two reasons: firstly, communication networks are usually not designed with excessive spare capacities to support all primary traffic paths during failures; and secondly, the backup capacities are computed based on the assumption of static traffic demand patterns. In the event this traffic demand behavior changes significantly, the backup paths would have to be re-computed. An example of this technique was proposed by Pan et al. [27], who used a pre-computed backup path for MPLS networks. This solution is not ideal for network traffic that changes frequently, since relatively small changes in traffic levels can trigger backup path computations. Also, the approach relies on availability of substantial spare network capacity that can accommodate secondary paths without affecting pre-existing primary paths. Ash et al. [28] employed a technique involving manipulation of existing MPLS path parameters in the event of link or node failures; however the approach involves significant signaling overhead.

Sa-Ngiamsak et al. [29] proposed the merging point of recovery optimization (MPO) approach for discovering merging point nodes for re-routing traffic in the event of link or node failures. The solution is based on broadcasting recovery messages to upstream nodes that will establish a local re-routing path around the failed region of the network (e.g. failed link or node). This proposed approach leads to reactive local re-routing. However, the disadvantage is the centralized computation required to ascertain the current load of all links to determine the optimum merging point. Another local re-routing approach was developed by Kvalbein et al. [46], who propose deployment of local re-routing rules to be enforced in the event of any anomaly that may be result from failures. The solution incorporates inferencing to detect any packets that are re-routed back to the original nodes, and performing any local re-routing to avoid failed regions of the network. Although the solution allows local re-routing, a centralized computation is initially required to determine all backup paths, which are then translated to local rules deployed onto each node.

Leibnitz et al. [30] propose a biologically inspired technique for self-adapting routing in overlay networks. Their approach initially computes all primary and secondary paths between source and destination. In the event of any changes detected on the primary path (e.g. link failures), the streaming will switch from the primary to the secondary path. However, the approach is based on global re-routing, which can lead to large delays in switching flows from the source itself.

3. Policy-based management architecture for control of bio-inspired routing processes

In this section we introduce a management architecture that facilitates coordination of decentralized, bio-inspired network algorithms and processes via management policies specified and maintained by network administrators. Fig. 1 depicts this architecture, which incorporates an autonomic control loop in which policies are executed in response to changes in the network in order to effect changes that will maintain the network in a desired state. This can be viewed as a simplified version of the FOCALe autonomic networking architecture [8], the main differences being that (1) FOCALe is fully distributed (all network elements contain their own policy sub-systems and thus must negotiate with each other to effect network-wide changes) and (2) FOCALe contains learning and reasoning capabilities (so that new policies can be automatically generated based on analysis of the results of previous policy invocations).

Central to the architecture is the presence of one or more system models that abstract the static structure, functionality, and dynamic behavior of the underlying network infrastructure, management functionality, offered services and, crucially, the operation of the bio-inspired algorithms and processes. Also modeled is the governance model of the system, which is realized as a policy continuum [25]. Implementation of the policy continuum enables the different constituencies who create policies for each layer, to manipulate sets of policy representations at a view appropriate to them, and to have those view-specific representations mapped to equivalent representations at views appropriate for manipulation by other constituencies.

System models are continuously updated by the knowledge analysis/generation component in response to the changing operational context of the network. This component gathers raw context information from managed resources (e.g., SNMP alarms) and using various analysis techniques with the system models as a knowledge base, quantifies the impact, or potential impact of this information (e.g., an SNMP alarm means that customer X is not being delivered the QoS level indicated in their SLA for service Y). If the system is deemed to be in an undesired state the relevant policy or policies in the policy continuum are triggered and the policy analysis/deployment component utilises knowledge embodied within the models to automatically generate the appropriate network device level re-configurations that will serve to bring the systems to as close as possible to its desired state. As described in Sections 4 and 5, these reconfigurations could involve re-parameterization of bio-inspired routing algorithms to prioritize specific traffic when re-routing flows following a link failure event.

4. Bio-inspired routing processes

We apply a number of bio-inspired techniques to develop a robust reactive routing process that can react well to network failure events. The three biological processes we

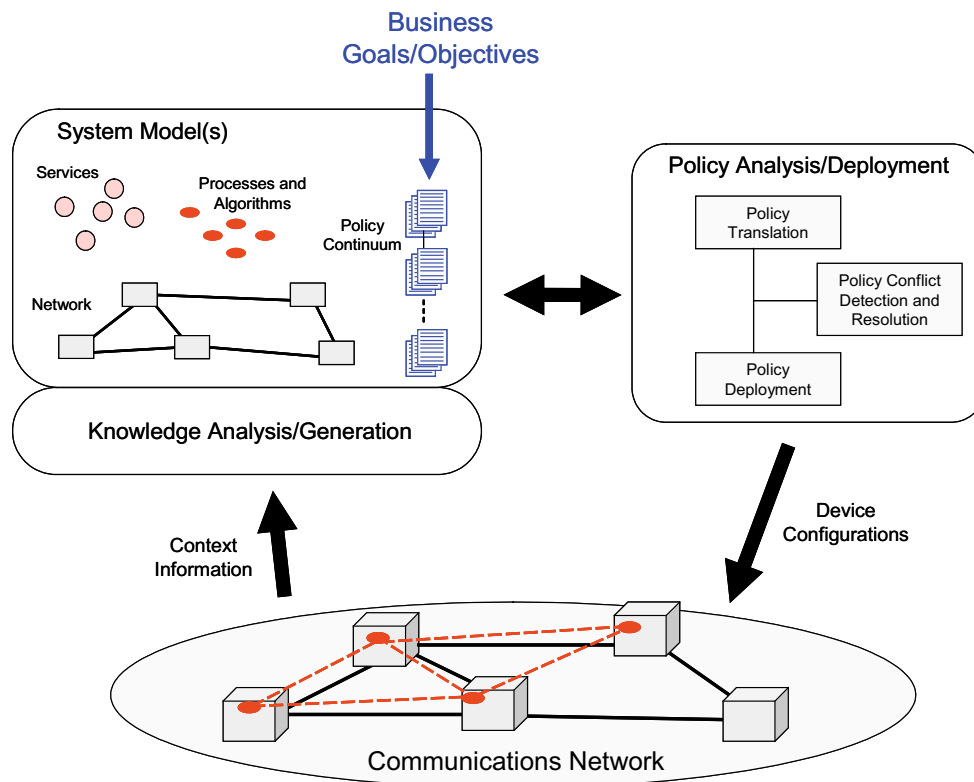


Fig. 1. Policy-based management architecture for control of bio-inspired network processes.

our techniques mimic – reaction-diffusion, chemotaxis and quorum sensing – each exhibit characteristics that are desirable in the design of a decentralized, self-organizing routing solution. We first briefly introduce these three biological and then describe how they are mapped to the design of a coordinated route management process.

4.1. Biological processes overview

4.1.1. Reaction-diffusion

The concept of reaction-diffusion was proposed by Turing [31] to describe the process of self-organization of biological cells. The concept is based on chemical diffusion from cells known as morphogens, where there are two morphogen types: activators and inhibitors. Based on the release of morphogens and non-linear interactions between the cells, a spatially distributed form of self-organization is exhibited by the cells during the formation of tissues.

4.1.2. Chemotaxis

Chemotaxis [32] is a process by which micro-organisms direct their movements according to the concentration of certain chemicals in their environment. The guided movement process of chemotaxis is based on the stimulus attraction between the micro-organisms and the environment, which results in the micro-organisms following paths corresponding to the gradients of the chemical(s) in question. There are two types of chemotaxis: (1) positive chemotaxis, where the chemical gradient produces a positive attraction for the micro-organisms to migrate towards

the source (e.g. a chemical gradient formed towards a food source) and (2) negative chemotaxis, where the chemical gradient produces a negative attraction leading the micro-organisms to migrate away from the source (e.g. a chemical gradient formed away from a poisonous source). The process of chemotaxis has inspired development of algorithms for a number of applications, including robotics [33] and network load balancing [34].

4.1.3. Quorum sensing

Quorum sensing is a process that allows groups of cells to use the size and density of their colony to initiate production of a specific type of substance [35]. Examples of quorum sensing include the behavior of colonies of the bacterium *Pseudomonas aeruginosa* [35], which is responsible for urinary tract infections in the human body and colonies of the bacterium *Vibrio fischeri*, which is responsible for production of light within a squid. In computer science, quorum sensing based algorithms have been proposed for various applications, including agents in mobile ad hoc networks [36] and underwater sensor networks [37].

4.2. Mapping of bio-inspired processes to route management

We will now describe the process of mapping the different biological processes described above to adaptive hop-by-hop routing in communications networks. Fig. 2 illustrates how algorithms modelling these processes can be coordinated to achieve adaptive routing as well as re-routing in the event of failures. In our solution, each node of the network nodes is assumed to store the hop count

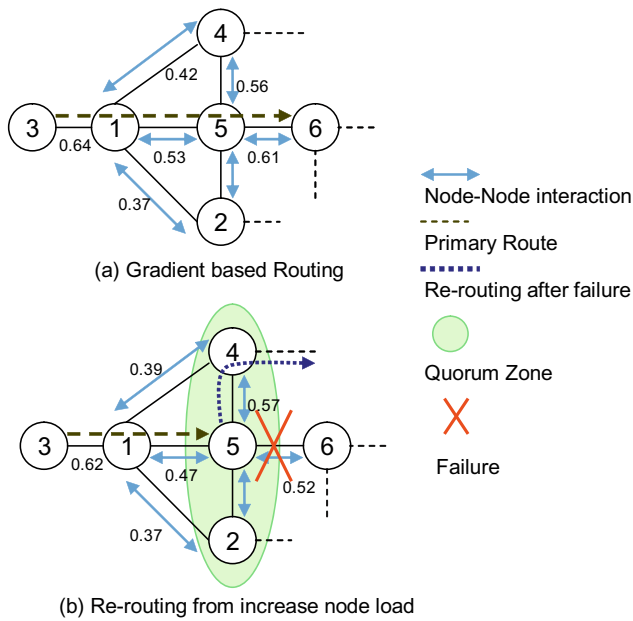


Fig. 2. Illustration of (a) hop-by-hop bio-inspired routing, and (b) re-routing after failure.

distance to each destination; these values are calculated initially when the topology is constructed and are automatically updated in the event that topology expansion/contraction occurs (e.g. new nodes are added to/removed from the network).

The chemotaxis/reaction-diffusion process involves nodes interacting with their neighbouring nodes to provide information on their own load. Transfer of load information to neighbour nodes is performed every time the load on a given node changes by an amount greater than or equal to a specified threshold amount. This information transfer results in the creation of a self-organized “chemical field” which is used for route discovery. To create the chemical field, each node uses the hop count values for specific destinations from the neighbouring node, the load of the immediate link connecting to the particular neighbour, and its own load information to calculate the relative gradient field values. The gradient field values are subsequently used for hop-by-hop route discovery from source to destination. An example of the gradient-based hop-by-hop routing is illustrated in Fig. 2a, where the initial path is through nodes 3 – 1 – 5 – 6. Fig. 3 illustrates an example of the gradient values of the links that led to this path selection. An example of gradient selection for the next hop can be seen for node 1, where link 1 – 5 is selected since the gradient value is higher in comparison to that of links 1 – 4 and 1 – 2. The reason that this link value is high is partly due to the low load of node 5, as is shown in Fig. 4.

The quorum sensing process is used by nodes in the event of node or link failures, in which case nodes surrounding the failure will cooperatively negotiate to determine the minimum amount of resources required to re-route the affected traffic. Where there are multiple traffic types with different associated revenue for the network operator, the re-routing algorithms can be configured

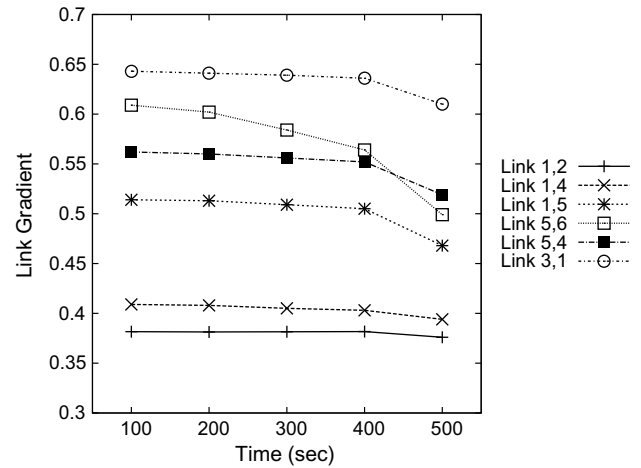


Fig. 3. Link gradients of Fig. 2 example.

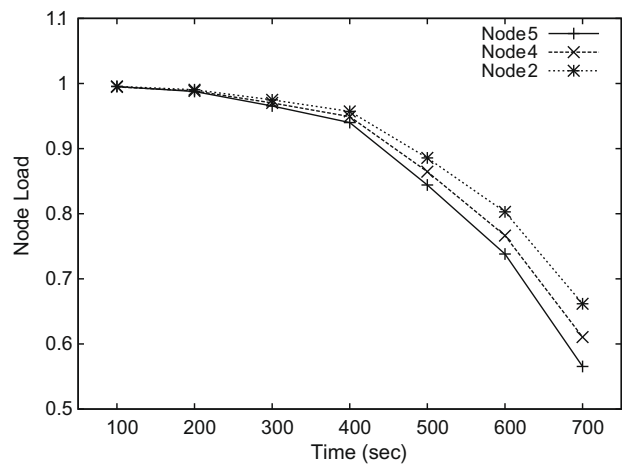


Fig. 4. Node load of Fig. 2 example.

(via policies) to prioritise high revenue traffic classes over low revenue traffic classes. Fig. 2b illustrates this process in the event that link 5 – 6 fails at time 400 s. A quorum zone is established between nodes 5, 4, and 2 to negotiate the new resources as well as the re-routing of existing flows. After the negotiation is performed, node 5 selects the link to node 4 for re-routing. The reason this link is selected is shown in Fig. 3, where after the failure at time 400 s, the gradient of link 5 – 6 drops below that of link 5 – 4. We now provide detailed specifications and algorithms of the chemotaxis/reaction-diffusion routing process and the quorum sensing failure re-routing process.

4.2.1. Chemotaxis/reaction-diffusion based routing process

The algorithm for the chemotaxis/reaction-diffusion based routing is provided in Fig. 5. Initially, the algorithm performs the hop count calculation for each node n of a topology with N nodes. Each destination d diffuses an arbitrary (but larger than the maximum hop count between an source, destination pair in the network) hop count value (H_d) to its neighbours. As each node n receives the H_d , it de-

```

1 //Initialisation (when topology is formed / changes)
2 for all  $n \in N$ 
3   Each destination  $d$  diffuses  $H_d$  to each source  $s$ 
4   Each node  $n$  deducts 1 from received  $H_d$  values
5   Each node  $n$  receiving multiple  $H_d$  values select highest
6   value and passes it to next node
7   For all sources  $s$  receiving  $H_d$ :
8     back-propagate final  $H_{d,f}$  value
9     calculate normalized  $h_{d,n}$  value at each node  $n$  on path to  $d$ 
10
11 //Normal Operation
12 Each node  $n$  calculates load node  $\Phi_{n,new}$ 
13   and stores previous load node,  $\Phi_{n,old}$ 
14 if  $(\Phi_{n,new} - \Phi_{n,old}) / \Phi_{n,old} > TH_\Phi$ :
15    $n$  transmits  $\Phi_{n,new}$  to neighbouring nodes
16   neighbours of  $n$  calculate  $G_{n,d,i \rightarrow j}$  for all links  $i$ 
17 for node  $n$  routing flow  $f_{s,d}$  from  $s$  to  $d$ :
18    $n$  determines destination  $d$  of  $f_{s,d}$ 
19    $n$  selects link  $i$  with highest  $G_{n,d,i \rightarrow j}$  to forward  $f_{i,d}$  to node  $j$ 

```

Fig. 5. Algorithm for chemotaxis/reaction-diffusion based routing.

ducts 1 from the value and passes this to the neighbouring nodes. In the event that a node receives multiple values of H_d , the highest value is selected. Once the H_d value reaches the source (at which point we term it $H_{d,f}$), a back propagation process is performed, where the normalized $h_{d,n}$ value (in the range (0, 1)) is calculated for each node on the path to the destination. Details of the normalized value calculation can be found in [37,44]. In the event that a node detects its load changing by an amount greater than or equal to the load threshold ($TH_{\Phi,n}$), it transmits its new load value to its neighbouring nodes. This process mimics the reaction-diffusion process that cells use to self-organise. The calculation for the load node is based on the total ratio of used and free capacity on each connected link, as specified in [37].

Based on the neighbour's normalized hop count value, the load of the link to neighbour's node, as well as its own load, each node calculates the gradient value $G_{n,d,i \rightarrow j}$ for a link with respect to a specific destination d . (e.g. gradient of node n for link i connected to neighbouring node j for destination d). The equation for the gradient calculation is represented as follows:

$$G_{n,d,i \rightarrow j} = \alpha \Phi_j + \beta l_{n,i \rightarrow j} + \gamma h_{d,j},$$

where Φ_j represents the load on neighbouring node j ; $l_{n,i \rightarrow j}$ represents the link load for link i of node n to neighbouring node j ; and, $h_{d,j}$ represents the normalized hop count to destination d at node j . The α , β and γ represent normalized weights for each of the corresponding parameters and can be configured via policies. Policy-based configuration of these parameters provides the ability to flexibly change the operation of the process based on current state of the network. Specifically, the network administrators can author policies that change the values of these parameters based on the amount of load currently handled by the network as a whole. For example, in the event that the network is lightly loaded, preference may be given for route discovery to concentrate on shortest path (e.g. $\alpha = 0.3$, $\beta = 0.2$, and $\gamma = 0.5$). On the other hand, when the network is highly loaded, the priority for route behavior may be set to maximizing load balancing and stability of the network (e.g. $\alpha = 0.5$, $\beta = 0.2$, and $\gamma = 0.3$).

It is also possible to configure the value of the threshold value for node load changes (TH_Φ) which trigger communication of load information between nodes. This value indirectly influences the sensitivity gradient field calculation process described above – the lower it is, the faster the gradient field will react to changes in load conditions. When the network as a whole is highly loaded, the routing process is required to be highly reactive and sensitive to load conditions; therefore, the TH_Φ will be set to given high sensitivity. For example, in case the network is lightly loaded, this calculation process can be performed with low sensitivity (e.g. perform node load calculation for every 15% increase in node load). Authoring policies that result in this behaviour provide a straightforward means of giving the desired behaviour.

When a new flow $f_{s,d}$ is to be admitted into the network, the flow is routed hop-by-hop from source s to destination d . The routing process is performed by selecting the highest $G_{n,d,i \rightarrow j}$, until the flow reaches the destination. This process mimics the positive chemotaxis mechanism used by micro-organisms in response to a chemical gradient.

4.2.2. Quorum sensing based failure re-routing process

As described earlier, the quorum sensing process is used to reconfigure the resources between neighbouring nodes, when failures have been encountered in specific regions of the network topology. Fig. 6 illustrates the algorithm for the quorum zone formation; for simplicity we assume that the network carries two classes of traffic: data and multimedia (with multimedia having priority); however, the algorithm can be readily generalised to more than two traffic classes.

In the event that a link failure is detected by node m , a quorum zone request Q_R is transmitted to the immediate neighbours of m . The quorum zone is formed between node m and neighbouring nodes N_m . Each node of N_m replies with the spare capacity ($C_{S,T}$) available on their links, where node m will determine the ratio of multimedia and data streams to be re-routed along $C_{S,T}$ if the spare capacity is sufficient to handle the multimedia traffic and a minimum amount of data traffic. In the event that the spare capacity is unable to accommodate all the existing streams

```

1  if link failure detected on node  $m$ 
2   $m$  sends request for spare capacity  $Q_s$  to its neighbouring
   nodes  $j \in N_m$ 
3  Nodes  $j$  receiving  $Q_s$  reply with their spare capacity,  $C_{s,j}$ 
4   $m$  sets total spare capacity  $C_s^T = \sum_{j=1}^{N_m} C_{s,j}$ 
5  if  $C_s^T - TH_D < S_{M,FL}$ 
6   $m$  sends request for additional capacity  $Q_a$  to  $j \in N_m$ 
7   $j$  receiving  $Q_a$  responds with
    $C_{a,j} = \begin{cases} T_{D,j} - TH_D & \text{if } T_{D,j} - TH_D > 0 \\ 0 & \text{if } T_{D,j} - TH_D \leq 0 \end{cases}$ 
   where  $T_{D,j}$  is the total size of data flows currently
   routed via  $j$ 
8  Set total additional multimedia capacity  $C_{a,M}^T = \sum_{j=1}^{N_m} C_{a,M,j}$ 
9  Re-route data flows with total size not exceeding  $TH_D$ 
   and multimedia flows with total size not exceeding
    $C_s^T + C_{a,M}^T - TH_D$  via nodes  $j \in N_m$ , whilst ensuring that the
   total size of flows routed via node  $j$  does not exceed
    $C_{s,j} + C_{a,M,j}$ 
10 else
11  Re-route multimedia flows with a total size not exceeding
    $C_s^T - TH_D$ . Drop remaining multimedia flows. Use remaining
   capacity to re-route data flows and drop and remaining
   data flows.
12 for re-routing after failure
13  $m$  selects  $G_{n,d,i \rightarrow j}$  to re-route flow via node  $j \in N_m$ 

```

Fig. 6. Algorithm for quorum zone formation.

from the failed link (S_{FL}), node m begins negotiation with neighbouring nodes to free a certain amount of capacity to accommodate as much of these streams as possible. This will be performed by having the neighbouring nodes discard (to a certain degree – as described below) low revenue traffic in preference for higher revenue traffic.

The negotiation process is based on satisfying a number of threshold constraints – where these thresholds constrain the degree to which nodes can discard low revenue traffic. For example, for multimedia streams, if the amount

of spare capacity ($C_{S,T}$) indicated by neighbour nodes is not able to accommodate the current multimedia streams from the failed links ($S_{M,FL}$), the data streams on neighbouring nodes will be discarded until enough capacity is obtained to support the multimedia streams; however, neighbouring nodes must maintain pre-existing data streams whose aggregated bandwidth is equal to the minimum threshold set for data traffic, TH_D . The minimum threshold values for different traffic classes can be set via policies, based on a business level decision as to how the trade-off between

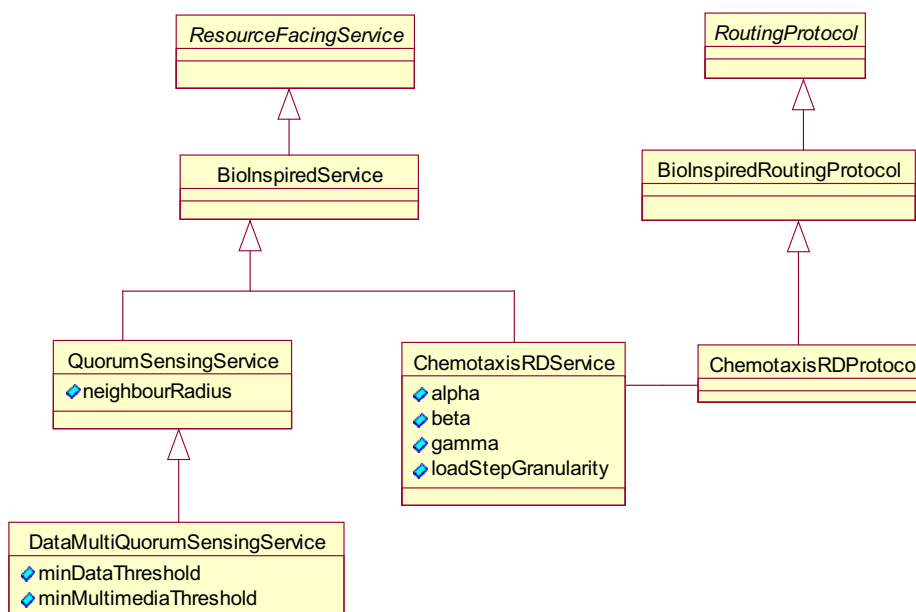


Fig. 7. New ResourceFacingServices for bio-inspired routing processes.

maximizing revenue (setting low threshold values for low revenue traffic so that higher revenue traffic is prioritized) versus fairness between traffic classes (setting higher minimum threshold values for all traffic classes).

5. Prototype implementation

In this section we describe our prototypical implementation of the policy-based network management architecture introduced in Section 3. This prototype consists of two main sub-parts: a policy authoring/execution environment that is loosely-coupled to a network simulation execution environment. As described in Section 6, the simulation model is used to analyze the operation and performance of the bio-inspired routing processes, which are parameterized via policies authored by network administrators. Before describing the policy authoring/execution and network simulation environments, we will specify extensions to the DEN-ng information model that model the operation of the bio-inspired routing algorithms specified in Section 4.

5.1. Extended DEN-ng information model

In order to enable the tools and languages developed using the information model to be cognisant of the bio-inspired routing algorithms, we must introduce new classes and relationships into the information model. This subsection describes these extensions to the DEN-ng information model. Fig. 7 depicts the new classes that describe bio-inspired services, where the services embody the previously specified bio-inspired routing algorithms. The *BioInspiredService* class is defined as a subclass of *ResourceFacingService* from DEN-ng. In DEN-ng, service types are split into two general categories, those of *Cus-*

tomerFacingService and of *ResourceFacingService*; *ResourceFacingServices* (RFS) are services that are not directly exposed to customers, instead they are used to provide functional support for customer facing services. Two new services, that subclass the *BioInspiredService* are *QuorumSensingService* and *ChemotaxisRDService* (where RD indicates the use of reaction-diffusion in the chemotaxis/reaction-diffusion routing process). *QuorumSensingService* is extended to *DMQuorumSensingService* to reflect the usage of the algorithm in differentiating between data and multimedia traffic types.

Each service describes a set of manageable attributes that reflect the current configuration of the services. For example, *ChemotaxisRDService* has attributes describing the α , β and γ parameters of the routing algorithm. These attributes can be set via polices to influence and constrain the behavior of the routing algorithm. Similarly, attributes for other bio-inspired services can be described and used.

As the new classes derive from RFS, they inherit all public and protected attributes and more importantly associations. In DEN-ng the RFS is associated to *LogicalResources* and *Protocols* indicating the usage of the particular service [25]. Taking the *ChemotaxisRDService* for example, we can extend the pre-existing association to *RoutingProtocol* to create a specialized routing protocol called *ChemotaxisRDProtocol*. This new protocol class can then describe the characteristics of the chemotaxis routing protocol (for example the reaction-diffusion exchange of load information between neighbouring nodes), and it is again manageable by policies.

In the next section we describe the policy tools and languages we built based on the additions to DEN-ng to support bio-inspired services and offer example policies that can be defined to manage these services.

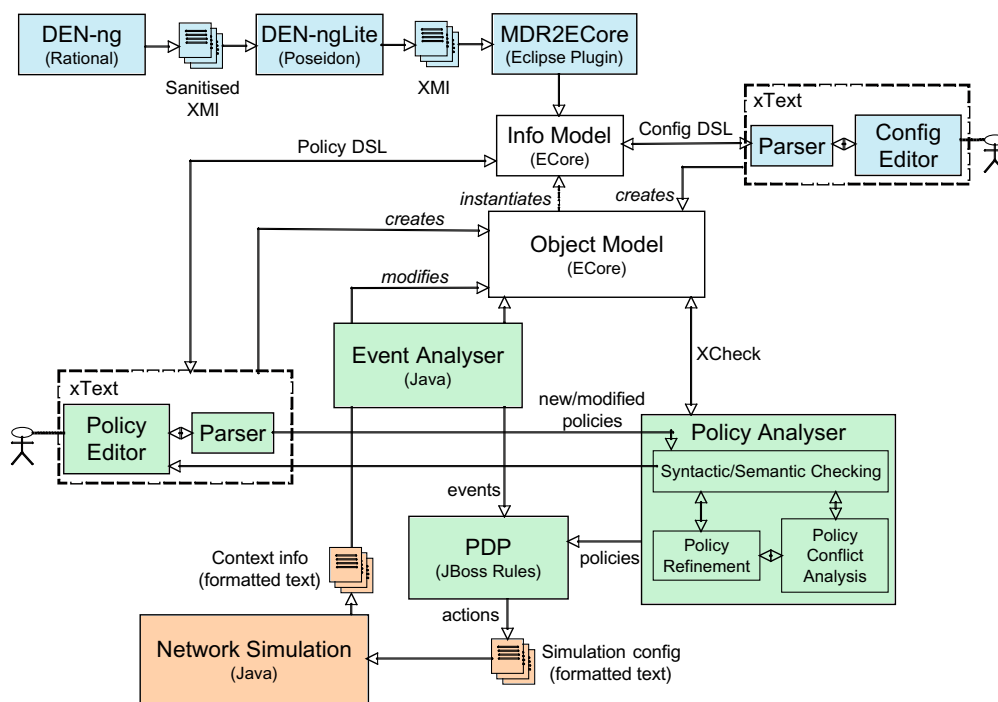


Fig. 8. Policy-based network management system prototype.

5.2. Policy authoring/execution environment

The structure of our prototype is depicted in Fig. 8; a full description of the steps involved in building it are provided in Barrett et al. [38]. At the core of the prototype is the object model, which represents the current state of the simulated network, the customers and services it supports, and the policies deployed in it. To initialize this model we created a configuration domain specific language (DSL) and editor that enables the creation of information model instances (i.e., object models) to represent the structure and initial state of the managed network. To do this we used the textual DSL framework called *xtext*, created by open Architecture Ware [39] (available through Eclipse's Generative Modelling Technologies (GMT) initiative [40]). The configuration DSL is generated from the model elements in the DEN-ng model that are marked as relevant for describing the structure of our particular managed network – this offline process is depicted at the top of Fig. 8. The same technique used to create the configuration DSL and its editor were used to generate an event-condition-action (ECA) policy DSL that is based on the policy representation entities in the DEN-ng model. The ECA policy DSL has the following policy execution semantics: on the occurrence of one of a set of events, if the condition clause evaluates to true, then execute the action clause (s). The policy DSL and its editor allow the network administrator to create, modify, or withdraw policies during system operation.

The policy analyser uses the object model to build a more complete understanding of the characteristics and behavior of policies and how they affect managed resources. Analysis of newly created or modified policies prior to their deployment is essential to ensuring that the system remains in a consistent state. For example, it is important to assess whether newly authored policies potentially conflict with policies already deployed in the system. For a description of our ongoing work on the use of information models for policy conflict analysis see Davy et al. [41]. The policy analyser translates policies into a format suitable for deployment on a rule inference engine based Policy Decision Point (PDP). We use JBoss Rules

(based on the Drools rule engine [43]), which uses a tailored object-oriented form of the Rete algorithm [42], called Rete-OO, for evaluating the rules. The Rete algorithm efficiently stores rules in memory in the form of a network so that it can take advantage of rule patterns to reduce the number of conditions that need to be evaluated. Policies are particularly amenable to Rete-based rule engines, as sets of deployed policies often share event, condition, and even action components.

To demonstrate the result of generating a policy language specifically designed to take advantage of the extensions to the DEN-ng information model, we will now walk through specification of some example policies. The nature of the policy language enforces a specific structure where events, conditions and actions are specified individually and these specifications are combined to form policy rules. Fig. 9 demonstrates the use of the *ChemotaxisRDService* defined in a PolicyAction. The action is used when a policy must reconfigure the routing behaviour of the network. The configurable attributes are α , β and γ ; as described in Section 4.2.1 different values of these parameters suit normal and heavy load conditions. Constraints can be added via the policy authoring tool to ensure that the actual constraints associated to the attributes are always maintained (for example α , β and γ must always sum to 1). This policy action may be used to configure network routing depending on network context. An example of a condition based on network context is depicted in Fig. 10; in this the global load on the network is used to ascertain whether the policy action in Fig. 9 should be enforced – thus once the global load passes a threshold the value of α , β and γ are changed to the “high load” set of values. A full specification of description of the policies for configuration of the bio-inspired routing algorithms is included in Appendix I.

The network simulation environment is loosely coupled to the other system components via formatted text files that are periodically written to or read from the various components. One set of files contain the bio-inspired routing algorithm parameters, which are updated by executing policies. On the other hand the simulation writes to a set of files indicating aspects of the current operational status of the simulated network. These files are polled by the event

```
PolicyConditionAtomic
{
    Name "GlobalLoadAboveThreshold"
    ThePolicyStatement
    {
        Description "GlobalNetworkLoad >= GlobalThreshold"
    }
}
```

Fig. 9. Policy action configuring the *ChemotaxisRDService*.

```
PolicyActionComposite
{
    Name "SetNetworkMode_HeavyLoad"
    ThePolicyAction
    Ref {Path "Net1.ChemotaxisRDService.SetAlphaTo .3"}
    Ref {Path "Net1.ChemotaxisRDService.SetBetaTo .2"}
    Ref {Path "Net1.ChemotaxisRDService.SetGammaTo .5"}
}
```

Fig. 10. Policy condition testing global network load.

analyser, a Java process, which parses them and, using information from the object model, ascertains whether changes correspond to events that should be forwarded to the PDP for processing. In our prototype, generation of global load update events is the responsibility of the event analyser, which calculates the mean of the link loads in the simulated network. In a real network this value could be estimated through analysis of management information collected via SNMP, or through direct interaction with network devices.

The network simulator deployed in the prototype is a bespoke simulator written in Java. The simulator is developed to incorporate a number of bio-inspired techniques. In this paper, the simulation study focuses on using these techniques for re-routing in the event of link failures in core network infrastructures. The resource negotiations are also incorporated into the simulator to support prioritized traffic negotiations after the failure. Similar to a conventional network simulation, it allows specifications of topology size, as well as probability models specifying traffic behavior for two types of traffic (multimedia and data), network admission behavior, as well as service duration.

6. Case study

In this section we present a case study of the use of the prototype to illustrate the concepts of policy-constrained bio-inspired network survivability. First we illustrate the path restoration behaviour that our combination of chemotaxis/reaction-diffusion routing and quorum sensing failure recovery algorithms are designed to exhibit. Next we present results generated via our prototype that demonstrate how policy re-configuration can change the behaviour of the bio-inspired adaptive routing to suit the objective of the network operator.

As described in Section 2, current approaches for failure recovery concentrate largely on proactive solutions that are not suitable for future communication systems, which require reactive techniques that maximize the operator's revenue. This is particularly true for transient network failures, which require swift local re-routing to minimize dis-

ruptions. Current proactive re-routing solutions are reliant on the static traffic behavior models used to derive pre-computed backup routes for each destination; when traffic behavior deviates significantly these backup routes need to be re-calculated. However, unlike the pre-computed backup paths which only consider network level re-routing of traffic, the adaptive routing solution presented in this paper is load sensitive and re-routes traffic depending on the condition of the load of neighbouring nodes. At the same time, the solution is also based on a systems perspective that considers a number of criteria such as prioritization of traffic types in the event of failures. Based on this, the solution ensures that adaptive routing can be performed in a distributed fashion while ensuring that network recovery is performed with minimum disturbance to the rest of the topology.

We evaluate our solution using two topologies, which are illustrated in Fig. 11: Topology 1, which has 8 nodes and 13 edges (Fig. 11a), and Topology 2, which has 20 nodes and 74 edges (Fig. 11b). Our intention is to simulate our solution for different topology sizes and to see the effects of policy changes for re-routing in response to transient network failure. The scenario addressed is a policy administrator who monitors the network performance during transient link failure, and subsequently decides to perform changes to parameters through a set of policies to improve the performance of the network in the event a similar failure occurs for that particular traffic behavior. Transient failures, are short period failures (with duration on the order of minutes) that happen relatively frequently. The parameters used for the simulation are shown in Tables 1–3: Table 1 shows the time parameters for the simulation; Table 2 shows the Poisson process arrival rate

Table 1

Simulation time parameters

Simulation time	Quantity (s)
Total duration	900
Failure time	450

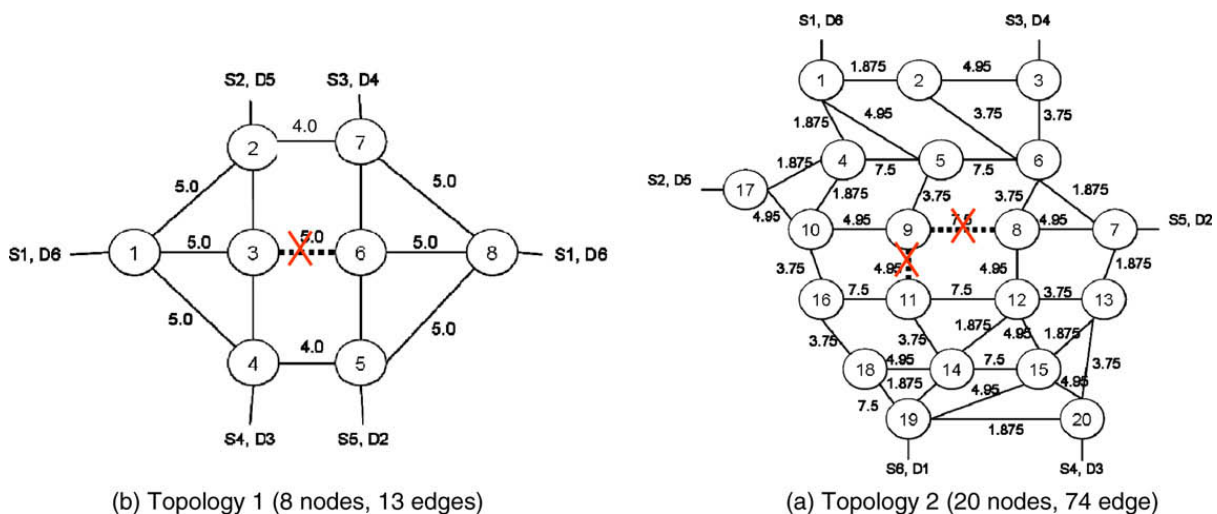


Fig. 11. Simulated topologies.

Table 2

Arrival rate per traffic type

Traffic type	Arrival rate (λ)
HTTP	30.0
VoIP	1.0
VoD	0.09

Table 3

Service time and flow quantity per traffic type

Traffic type	Distribution for service time (average – seconds)	Distribution for flow quantity (average – Kbps)
HTTP	Uniform (2–8)	Uniform (20–60)
VoIP	Uniform (60–180)	Uniform (54–74)
VoD	Uniform (200–400)	Uniform (200–400)

(λ) for each traffic type; and Table 3 shows the service time and flow quantity for each traffic type. For both simulated topologies, the evaluation includes average node load of the entire topology, average path lengths of all routes, total multimedia and data throughput, the number of diverted flows for re-routing, and prioritization of traffic flows for diversion. The evaluation of average node load and path length is for load balancing tests to be evaluated by the policy administrator. The mechanism for calculating the average node load is to determine the difference between the load of each node against the average node load for the whole topology (therefore, a value of node load closest to 1 indicates a well balanced network).

6.1. Topology 1

For Topology 1, we simulate a failure occurring on link 3 – 6 (illustrated in Fig. 11a as dashed line with cross). Initially, the administrator had set the following values for the gradient route calculations: $\alpha = 0.2$, $\beta = 0.4$ and $\gamma = 0.4$, where a greater weighting is given to link loads and hop count in order to focus on achieving shortest paths where possible. Given the transient failure on link 3 – 6 at time 450 s, the multimedia and data throughput, average

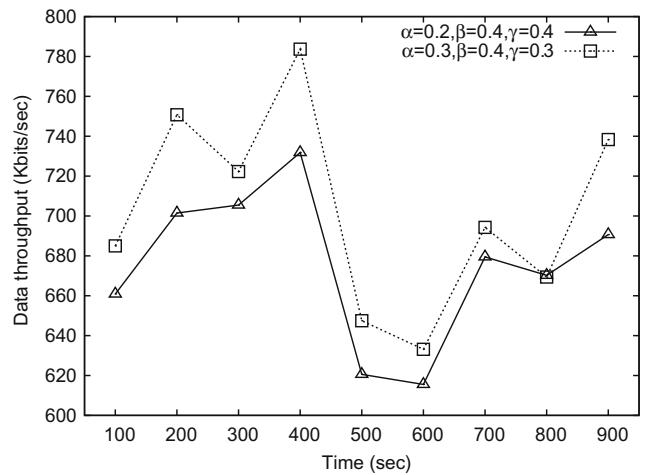


Fig. 13. Data throughput for Topology 1.

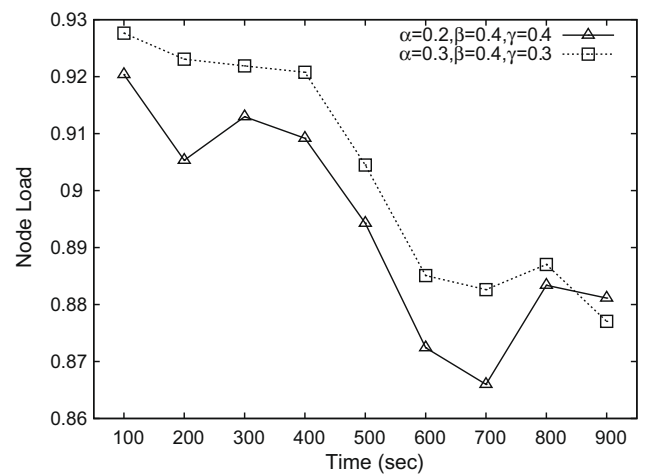


Fig. 14. Average node load for Topology 1.

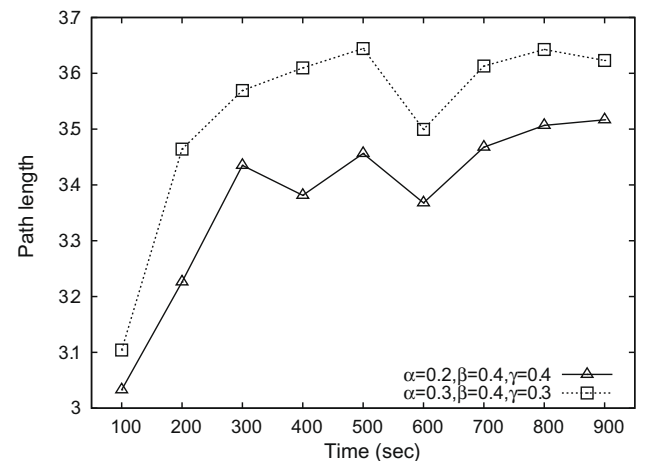


Fig. 15. Average path length for Topology 1.

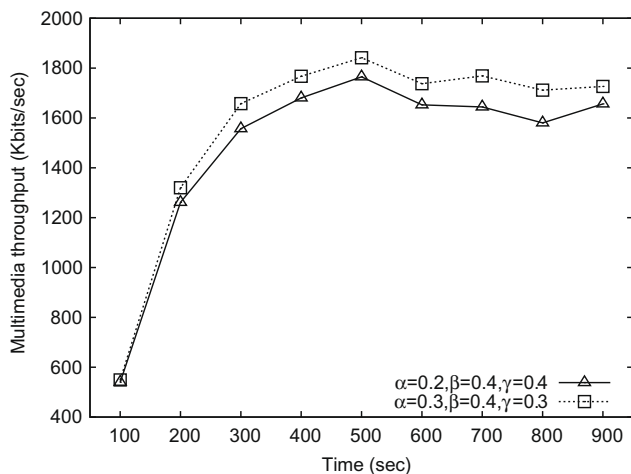


Fig. 12. Multimedia throughput for Topology 1.

node load and path length for the entire duration is shown in Figs. 12–15.

Based on the average node load in Fig. 14, the administrator noticed that the average load was reasonably far from the optimum value. This indicates that during the

Table 4
Flows diverted before and after parameter change for Topology 1

	Flows to be re-routed	Flows re-routed (S_{FL})	Data flows re-routed ($S_{D,FL}$)	Multimedia flows re-routed ($S_{M,FL}$)
$\alpha:0.2, \beta:0.4, \gamma:0.4$ ($R_D: 0.8, R_M: 0.2, TH_D: 5 \text{ Mb}, TH_M: 10 \text{ Mb}$)	64	43	28	15
$\alpha:0.3, \beta:0.4, \gamma:0.3$ ($R_D: 0.4, R_M: 0.6, TH_D: 4.5 \text{ Mb}, TH_M: 10 \text{ Mb}$)	61	48	23	25

failure, a certain amount of traffic flow to be re-routed was dropped due to the high congestion surrounding the region of failure. Therefore, to increase the number of re-routed multimedia and data flows, the administrator decides to re-parameterize to place greater emphasis on increasing load balancing in the network; this results in an increase of the load on part of the network, which results from no longer using shortest available paths. Specifically, the administrator changes the routing parameter values to $\alpha = 0.3, \beta = 0.4$ and $\gamma = 0.3$. Compared to the old parameters, the new value increases the weighting for the node load and reduces the value for the hop count. Based on

the changes, the node load improved as shown in Fig. 12, where the overall load for the nodes is closer to optimal balancing (including before the failure). The changes also improved the overall throughput of multimedia and data flows as shown in Figs. 10 and 11. Table 4 shows a comparison of the diverted flows before and after the parameter changes. Firstly, the results in Table 4 shows the total number of flows to be re-routed after the parameter change is lower (e.g. 61 compared to 64). This change is mainly due to the improved balancing of the network that reduced the number of flows along the original path. At the same time, the improved balancing of the network also allows

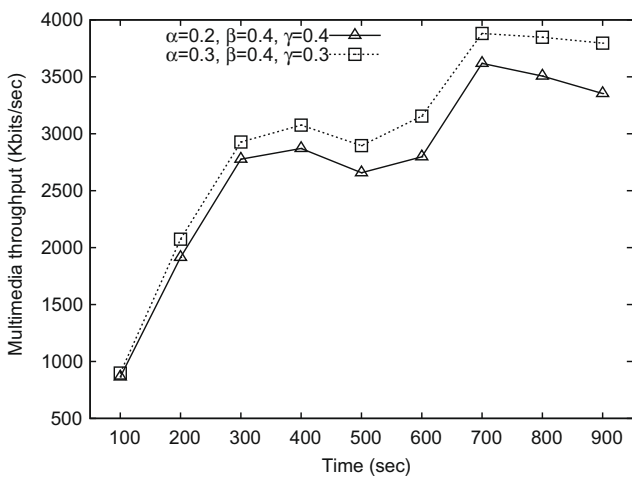


Fig. 16. Multimedia throughput for Topology 2.

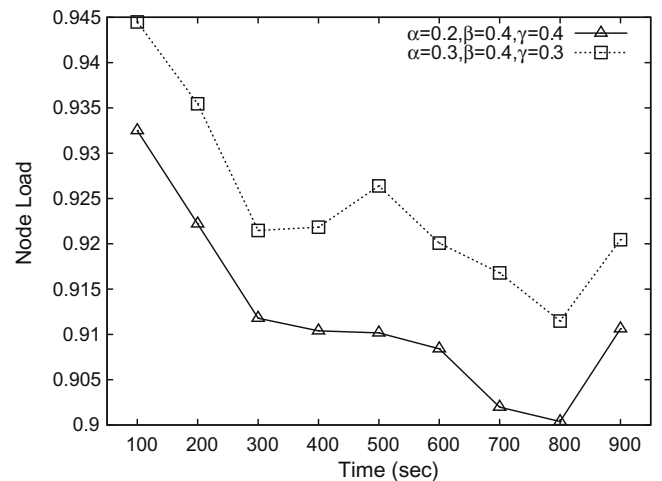


Fig. 18. Average node load for Topology 2.

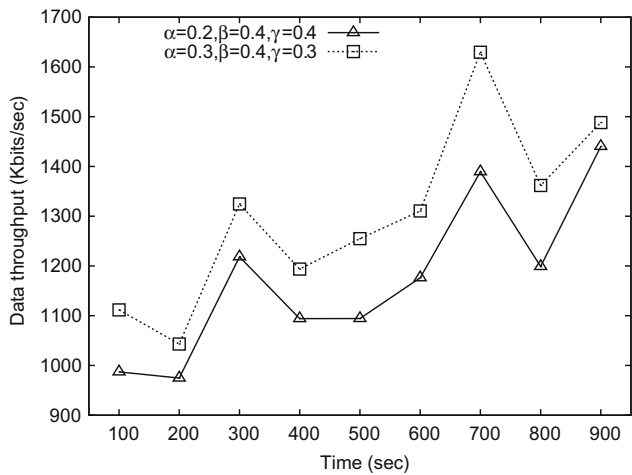


Fig. 17. Data throughput for Topology 2.

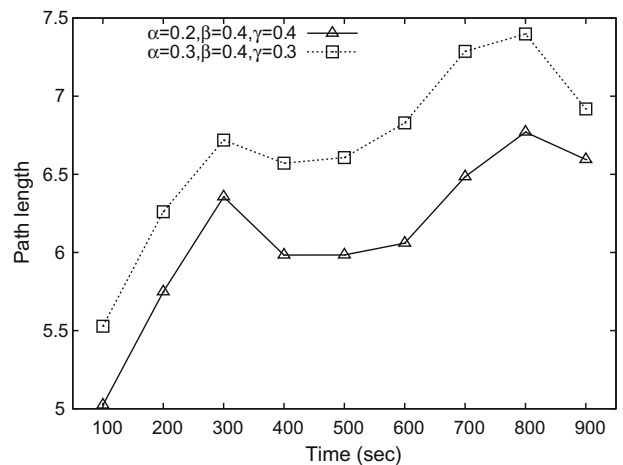


Fig. 19. Average path length for Topology 2.

Table 5

Flows diverted before and after parameter change for Topology 2

	Flows to be re-routed	Flows re-routed (S_{FL})	Data flows re-routed ($S_{D,FL}$)	Multimedia flows re-routed ($S_{M,FL}$)
$\alpha:0.2, \beta:0.4, \gamma:0.4$ ($R_D: 0.8, R_M: 0.2, TH_D: 5$ Mb, $TH_M: 10$ Mb)	48	35	25	10
$\alpha:0.3, \beta:0.4, \gamma:0.3$ ($R_D: 0.4, R_M: 0.6, TH_D: 2.5$ Mb, $TH_M: 10$ Mb)	50	41	16	25

a higher number of flows to be re-routed (48 flows compared to 43), as the neighbouring nodes of the failed node are less loaded, indicating lighter loads on their links. At the same time, the administrator also monitored the parameters for the quorum zone (R_D, R_M, TH_D, TH_M). The original ratio set for R_D was over-provisioned leading to a large number of multimedia streams being dropped. Therefore, the administrator decides to change the ratio, and this leads to an increase in multimedia flows (since multimedia flows have higher revenue). Even though after the parameter change there was an increase in multimedia flows, not all multimedia flows were admitted, since this exceeded the minimum data threshold (TH_D) that was set.

6.2. Topology 2

The evaluation tests performed on Topology 1 were also performed on Topology 2. Figs. 16–19 illustrate the multimedia throughput, data throughput, average node load, and average path length, respectively for the tests on Topology 2. In the simulation scenario of Topology 2, we failed two links: link 9 – 11 and link 8 – 9 (as shown as in Fig. 11b). Similar to Topology 1, there were also improvements when the parameters were changed for Topology 2, where the average node load balancing improved resulting in higher multimedia and data throughput. The main difference between the results of Topologies 1 and 2, is the increase in path length and improvement in node load balancing. After the α, β and γ change, the average path length for Topology 2 was 6.7 compared to 3.52 in Topology 1. The reason was the increase is the larger number of nodes and larger freedom of path movement. As described earlier, the mechanism of route discovery is through hop-by-hop discovery of the highest gradient for the destination. This will lead the routes to discover the lighter parts of the network when higher weighting is set for α . At the same time, the difference of average node load before and after the α, β and γ change is higher than in Topology 1. Table 5 shows the effects of the diverted flows when the quorum zone parameters (R_D, R_M, TH_D, TH_M) are changed.

As shown in Table 5, the effects of the quorum zone parameter change improves the overall throughput of the diverted flows. The total number of flows to be re-routed has increased, due to the improved load balancing of the network. In comparison to Topology 1, the large scale of Topology 2 has led to increased path discovery. This has led to a slight increase in total flows re-routed compared to Topology 1 (82% compared to 78.6%). Similar to Topology 1, Table 5 has also shown that by changing the R_D

and R_M ratios, the number of multimedia flows has increased dramatically to improve the overall revenue.

7. Summary and conclusions

The grand vision of autonomic computing and autonomic networking is that of a completely self-managing infrastructure that can itself access, or generate, the knowledge it needs to allow the system to optimally react to changing operational context. Although this vision is very attractive, it is unlikely to ever be fully realized, especially in the context of the hugely complex and dynamic global communications infrastructure. In this paper we have taken a more pragmatic approach: rather than seeking to entirely eliminate human intervention from the management process, we seek to minimize it and to focus it more on constraining and controlling the operation of decentralized algorithms that provide various forms of self-management functionality.

In particular, we discussed how centralized policy based management systems, having a global view of the operational context of the network can be integrated with distributed bio-inspired routing processes. We show how policies can be specified that re-parameterize these algorithms according to the current state of the network (for example, if it is normally or highly-loaded) or based on the business level decisions (for example whether to prioritize maximization of generated revenue during failure scenarios or alternatively to treat different traffic classes equitably). We believe that the ability for humans to control the operation of self-management algorithms in this manner would be of significant benefit to the network operator.

The paper proposed a routing process modeled partly on the chemotaxis and reaction-diffusion processes found in biological systems. In addition, we proposed a quorum sensing inspired process that facilitated optimal re-routing of traffic under link failure conditions. Taken together these processes greatly help in the realization of a survivable autonomic networking infrastructure.

Acknowledgements

We wish to acknowledge the valuable insights provided by Nazim Agoulmine, and the work on prototype design and implementation carried out by Keara Barrett, Alan Davy, and Elyes Lehtihet. Special thanks also to Julien Mineraud for performing the simulation work. This work has received support from Science Foundation Ireland under the ‘‘Autonomic Management of Communications Networks and Services’’ award (Grant No. 04/IN3/I404C).

Autonomous Agents and Multiagent Systems, Hokkaido, Japan, 2006.

- [37] S. Balasubramaniam, D. Botvich, W. Donnelly, M. Ó Foghlú, J. Strassner, Biologically inspired self-governance and self-organisation for autonomic networks, in: Proceedings of the 1st International Conference on Bio-inspired Models of Networks, Information, and Computing Systems (Bionetics 2006), Cavalese, Italy, 2006.
- [38] K. Barrett, S. Davy, J. Strassner, B. Jennings, S. van der Meer, W. Donnelly, A model based approach for policy tool generation and policy analysis, in: Proceedings of the 1st IEEE International Global Information Infrastructure Symposium 2007 (GIIS 2007), 2007, pp. 99–106.
- [39] Open Architecture Ware (oAW), Available from: <<http://www.openarchitectureware.org>> (accessed 13.12.2007).
- [40] The Eclipse Modeling Framework, Available from: <<http://www.eclipse.org/emf>> (accessed 13.12.2007).
- [41] S. Davy, B. Jennings, J. Strassner, Application domain independent policy conflict analysis using information models, in: Proceedings of the 2008 IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), 2008, pp. 17–24.
- [42] C.L. Forgy, Rete: a fast algorithm for the many pattern/many object pattern match problem, in: P.G. Raeth (Ed.), Expert Systems: A Software Methodology For Modern Applications, IEEE Computer Society Reprint Collection, pp. 324–341.
- [43] Drools: Business Rule Management System, Available from: <<http://labs.jboss.com/drools/>> (accessed 13.12.2007).
- [44] S. Balasubramaniam, D. Botvich, W. Donnelly, M. Ó Foghlú, J. Strassner, Bio-inspired framework for autonomic communication systems, in: Advances in Biologically Inspired Information Systems: Models, Methods, and Tools, Studies in Computational Intelligence, Springer-Verlag, 2007.
- [45] G. Agrawal, D. Huang, D. Mehdi, Network protection design for MPLS networks, in: Proceedings of the 5th International Workshop on Design of Reliable Communication Networks, Island of Ischia, Italy, 2005.
- [46] A. Kvalbein, A.F. Hansen, T. Cicic, S. Gjessing, O. Lysne, Fast IP network recovery using multiple routing configurations, in: Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006), Barcelona, Spain, 2006.



Sasitharan Balasubramaniam is currently a senior investigator at the Telecommunication Software and Systems Group, Waterford Institute of Technology. He received his Bachelors of Engineering (Electrical and Electronic) degree from the University of Queensland in 1998, Masters of Engineering Science (Computer and Communication Engineering) degree from Queensland University of Technology in 1999, and Ph.D. (Computer Science) from the University of Queensland in 2005. His research interests

include bio-inspired autonomic network management, sensor and ad hoc networking, and pervasive computing.



Dmitri Botvich is currently a principal investigator at the Telecommunication Software and Systems Group, Waterford Institute of Technology. He received his Bachelors (Mathematics) degree and Ph.D. (Mathematics) from Moscow State University, Faculty of Mechanics and Mathematics (Russia), in 1980 and 1984, respectively. His research interests include bio-inspired autonomic network management, security, trust management, sensor and ad hoc networking, queuing theory, and mathematical physics.



Brendan Jennings was awarded BEng in Electronic Engineering and Ph.D. degrees from Dublin City University, Ireland, in 1993 and 2001 respectively. He is a Senior Research Fellow at Waterford Institute of Technology, Ireland, working in the areas of policy-based network management, management of composed communications services, network monitoring and planning, and accountability processes in digital ecosystems. He has published over 50 papers in international journals and conference proceedings and has participated

in the work of standards bodies ETSI, FIPA, TM Forum, and ACF. He serves on the TPCs of a number of network management related conferences and has served as the TPC co-chair of MMNS 2006 and MACE 2007.



Steven Davy was awarded a B.Sc. in Computer Science from Trinity College Dublin, Ireland in 2003 and in 2008 completed a Ph.D. at Waterford Institute of Technology, Ireland. His current research focus is on policy based management for communications networks, in particular looking at efficient and scalable algorithms for policy conflict analysis. He has published over 10 papers in journals and international conference proceedings.



William Donnelly is director of the Telecommunications Software and Systems Group at Waterford Institute of Technology, Ireland. He has over 15 years experience of research and development of telecommunications network management systems, both in industry and academia. He is a Science Foundation Ireland Principal Investigator researching autonomic network management. His research interests are in the areas of bio-inspired network management solutions and management solutions for next generation

Internet based electronic media.



John Strassner is Director of Telecommunications with the Telecommunications Software and Systems Group at Waterford Institute of Technology, Ireland. Until 2008 he was a Fellow of the Technical Staff at Motorola, where he directed Motorola's autonomic networking research. He is a past Fellow of Cisco Systems and Chief Strategy Officer at IntelliDEN. His research interests include policy management and knowledge engineering, including ontologies, machine learning and reasoning. He has been awarded the

Daniel Stokesbury memorial award for excellence in network management. He is also a Distinguished Fellow of the TeleManagement Forum. He is the chairman of the Autonomic Communications Forum, and also the vice-chairman of WG6 (Reconfigurability and 233 Autonomics) in the WWRF. He has published two books and over 150 papers.