Advance Access publication on September 12, 2007 doi:10.1093/comjnl/bxm063

Hyperion—Next-Generation Battlespace Information Services

Robert Ghanea-Hercock[1,*], E. Gelenbe[2], Nicholas R. Jennings[3], Oliver Smith[4], David N. Allsopp[5], Alex Healing[1], Hakan Duman[1], Simon Sparks[5], Nishan C. Karunatillake[3] and Perukrishnen Vytelingum[3]

[1]*Pervasive ICT Research Centre, BT, Ipswich, UK*
[2]*Intelligent Systems and Networks Group, Department of Electrical and Electronic Engineering, Imperial College, London SW7 2BT, UK*
[3]*School of Electronics and Computer Science, University of Southampton, Southampton SO1 7 1BJ, UK*
[4]*General Dynamics UK Ltd, East Sussex, UK*
[5]*QinetiQ, Malvern Technology Centre, St Andrews Road, Malvern, Worcestershire WR14 3PS, UK*
**Corresponding author: robert.ghanea-hercock@bt.com*

The future digital battlespace will be a fast-paced and frenetic environment that stresses information communication technology systems to the limit. The challenges are most acute in the tactical and operational domains where bandwidth is severely limited, security of information is paramount, the network is under physical and cyber attack and administrative support is minimal. Hyperion is a cluster of research projects designed to provide an automated and adaptive information management capability embedded in defence networks. The overall system architecture is designed to improve the situational awareness of field commanders by providing the ability to fuse and compose information services in real time. The key technologies adopted to enable this include: autonomous software agents, self-organizing middleware, a smart data filtering system and a 3-D battlespace simulation environment. This paper reviews some of the specific techniques under development within the Hyperion sub-projects and the results achieved to date.

Keywords: agents; autonomous systems; middleware; data visualization

Received 14 May 2007; revised 14 May 2007

1. INTRODUCTION

Future military forces require a high degree of agility to overcome unpredictable and rapidly changing threats: (as envisaged in the 3 Block War scenario). This will require intelligent and real-time reconfiguration of information communication technology (ICT) services to meet the user's needs within the constraints of the network hardware and overall capacity, to realize the future state of the UK's network-enabled capability (NEC) [1]. However, defence networks and ICT systems are severely constrained by the extreme nature of the operating environment, and the need for resilience, security and stealth capability. As a result, current tactical communication links are inflexible and have limited and fixed capacity, typically measured in kilobytes per second. The challenge to the network is compounded by the growing demand for high quality and timely information, to be made available to a wide spectrum of defence users. This information may be supplied by high-bandwidth sensors or unmanned aerial vehicles (UAVs) that produce volumes of data that could easily overwhelm the communications available.

In order to address these issues, the Hyperion technical objective is to create an adaptive agent-based architecture capable of significantly enhancing the functionality and resilience of information fusion processes. Specifically, this will be achieved by providing an adaptive and reconfigurable capability for battlespace communication and information services. The project also has a set of scientific objectives, which includes the investigation of novel algorithms for self-organizing network infrastructures in support of military requirements. The domains for research are: resilient service-oriented peer-to-peer (P2P) architectures, information retrieval and integration, policy management and control, agent negotiation protocols and data visualization methods for distributed service-oriented computing in battlespace environments.

1.1. Approach

Our approach to this problem is to utilize a set of distributed software agents, embedded within the NEC environment. The agents manage the applications via goal-driven service workflows, adaptive quality-of-service (QoS) metrics and brokering to optimize the availability of the information resources. The Hyperion agents are designed to respond to policy level statements that specify preferred configurations and prioritization requirements for each information service or communication channel. The goal is to allow new policy requests to be hot loaded at any time to dynamically reconfigure the agents' behaviours. A command-oriented interface system is being developed to enable high-level management of the system and re-tasking of network resources. The targeted organizational benefit is an agile tactical communications network that is able to support a high tempo of operations by providing information services that are dynamically configurable in accordance with commanders' changing priorities. Furthermore, it should deliver an order-of-magnitude reduction in ICT support requirements for command and control (C2) processes, through a reduction in administrative manpower and time to reconfigure ICT services.

1.2. Project organization and cluster integration

The following organizations are partners in the Hyperion cluster, each providing the component capabilities.

(i) BT is working on the P2P and agent-based middleware for service-oriented/network-centric information integration and fusion.
(ii) Southampton University is investigating and developing the basic mechanisms that enable collectives of software agents to self-organize, self-repair and self-optimize in response to dynamic NEC environments.
(iii) Imperial College is implementing a security mechanism for protection from distributed denial of service (DoS) attacks of the Hyperion resources, in order to assure specified bandwidth, latency and user prioritization.
(iv) QinetiQ is responsible for a front-end command interface tool for Hyperion. This enables re-tasking of the system with mission policy and requirements.
(v) Finally, general dynamics is working on the underlying military NEC scenarios and scenario visualization via a 3-D battlespace virtual reality system.

Hyperion is designed to be a disruptive technology as it addresses the most challenging aspect of NEC, i.e. the integration of heterogeneous networks and systems. It aims to demonstrate the type of adaptive functionality required to weld such large-scale ICT networks and services together. This approach is of high value, as MoD moves to adopt a service-oriented architecture (SOA) [3] philosophy for future ICT platforms. The need for this dynamic and proactive distribution of information is illustrated by the following quote from a senior British defence source:

> In Iraq, British forces rely on coalition assets to provide much of the ISTAR information required... However, there was no means to exchange ISTAR information across the coalition in a timely and effective manner.

Brigadier David Capewell, Assistant Chief of Staff (Operations) at the UK's Permanent Joint Headquarters (PJHQ), Shephard 'Vision' conference in London on 8th November 2005.

2. METHODOLOGY

The Hyperion project is a cluster of five sub-projects, which in combination aim to build an integrated solution to the NEC problem space outlined in Section 1. Each of the sub-projects is briefly discussed in the following section with a summary of their implementations and results to date.

2.1. Nexus II: adaptive P2P NEC service middleware

The first phase of the Nexus project [2] demonstrated the value of an agent-based P2P middleware for the discovery and fusion of NEC services. The Nexus middleware is based on three key paradigms: P2P computing, autonomous agents and SOA [3]; all of which have been identified as key components of future NEC network architectures.

Existing implementations of SOA, as applied in the civil domain, suffer from several issues that make them unsuitable for volatile environments. These include centralized service discovery and process orchestration, and fixed manually specified workflows. These factors lead to fragile, non-adaptive and difficult-to-maintain network applications.

The aim is to develop a hardened, agent-based SOA implementation that meets the strict reliability requirements of the NEC domain and accommodates the needs of network-centric information fusion applications. More specifically, the following capabilities are being developed either as a direct part of Nexus II middleware or by integrating technologies from other projects within the Hyperion cluster:

(i) Seamless and reliable service delivery in volatile environments.
(ii) Request prioritization and load balancing.
(iii) Resilience to volatility of the underlying network infrastructure: by adopting a P2P architecture Nexus maintains its operability even if a large subset of services or the network itself becomes unavailable.
(iv) Decentralized service discovery whereby networked resources are discovered based on their advertized properties and real-time information regarding their

dynamic attributes without reliance on a centralized repository.

(v) Semantic and adaptive service selection based on dynamically maintained QoS profiles.

(vi) Proactive monitoring and automated service substitution: the state of services is actively monitored and should a failure occur the failed resource is rapidly substituted with the closest alternative, preserving the overall capability.

(vii) Filtering of information services based on their semantic relevance to the user as well as imposing some structure at the messaging layer of the middleware allowing bandwidth to be conserved.

2.1.2. Implementation

In order to offer the necessary resilience Nexus adopts an entirely decentralized approach. At the lowest level, a P2P overlay is constructed connecting, either directly or indirectly, each of the nodes in the network running Nexus with each other. Similar to [4, 5], the overlay network is then coupled with component-model technologies which in our case offer a publish/subscribe (Pub/Sub) structured messaging layer from which higher level management of the network can be constructed.

Each Nexus node can host a number of services and these are made available through the middleware by means of advertizing their associated metadata on the messaging layer. Users of Nexus are required to connect to only a single node from where the middleware allows them to discover resources throughout the network and manage their view and usage of the information services according to their requirements.

Above the middleware we adopt an autonomic computing [6] paradigm which introduces self-* capabilities to allow Nexus to intelligently and autonomously handle the dynamic environment for which it is intended; including, changing requirements of users, unreliable service availability or a failure of the underlying physical network.

Nexus is entirely implemented in the Java programming language and relies on several open-source third party libraries. In particular, the current embodiment of Nexus builds on an open-source P2P implementation of Java message service (JMS) [7] to provide the majority of the functionality of the bottom two layers shown in Fig. 1. IP multicast is used to discover other Nexus peers and construct the overlay. JMS topics provide Pub/Sub functionality for the message-oriented component of Nexus and allow for information service advertisements to be structured in their transmission across the network. Each peer acts as a message broker and routes messages to peers that are subscribed to the topic on which the message was published. The topics can be structured into a hierarchy, allowing one to subscribe to only messages concerning a specific subset of services. To some degree, the semantics relating to the service descriptions in the resource layer can be exposed to the messaging structure in the layer below. The routing of messages throughout the overlay can therefore be linked to the semantic relevance of the resources that the messages describe to each peer.

There are numerous aspects of the architecture to which autonomic computing principles may be applied. For example, the driving of the aforementioned messaging structure by the service metadata may be an autonomous process. At the lowest level, the overlay network is self-organizing in that the changes to the topology are dealt with seamlessly allowing for new peers joining the network to be discovered by others as well as the overlay to adapt their routing when peers are removed from the network.

The focus of the autonomic capability, however, is at the upper levels of the system model. Agent-based approaches to service orchestration have been investigated as well as methods to enable self-healing to fulfil a certain user service requirement in the case of a service failure. These two aspects are related and both rely on the system understanding, to some degree, (i) what the user requirements are, (ii) what services are available and how they relate, (iii) the expected QoS services can deliver in a certain context. The following

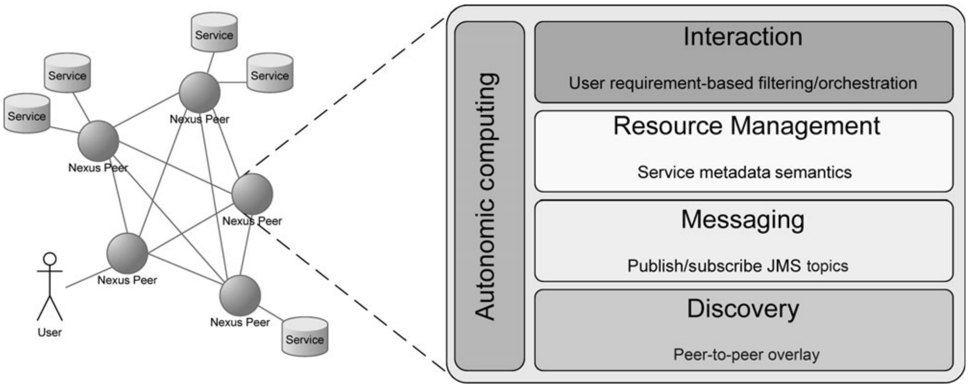

FIGURE 1. Nexus architecture.

section discusses one particular framework that has been developed as part of Nexus which addresses point (iii) and uses a multi-agent learning approach to model service quality and allows Nexus to autonomically select services to fulfil a certain requirement with the maximal QoS.

2.1.3. *Autonomic computing case study—Mercury adaptive service selection*

The Mercury framework [8] is designed for application within an SOA and as such assumes a network of interconnected devices, each capable of hosting a number of processes. The processes may adopt at least one of two roles: service provider or consumer. Service providers offer capabilities that other devices (consumers) can access and use. Mercury-based service selection takes place on the consumer side and assumes that for every device where there is a service consumer, a *selector agent* is hosted. Thus in Nexus, we envisage embedding a selector agent at each Nexus node.

Mercury relies on there being some service discovery mechanisms in the SOA in order to gain a list of functionally capable service providers for a particular task. This functional discovery is based on those attributes that the service providers advertise in their description and can be provided by other components of Nexus. The Mercury selector agents then use the list of capable services as a basis for further finer-grained, non-functional selection. This is achieved by aggregating QoS data for each of the providers through the consumer's experience of them and ranking them accordingly. The result is a model of selection learnt over time, which distinguishes those services that are best at performing the task in terms of the QoS they are expected to deliver.

The QoS data of providers are stored in a model local to each selector agent and is parameterized by the task, as well as the context. Context is defined as the set of attributes that are external to the task requirements but nevertheless may influence the performance of providers (e.g. performing differently at different times of day). A particular service selector therefore builds up a model of how suited each provider is at fulfilling each particular task in each context.

The main contribution is the design of an efficient distributed service selection framework and (collaborative) algorithms for its construction and real-time adaptation. Specifically, a decision function is employed (Fig. 2) to ensure that the probability of exploration (selection of services for which there is little or no prior data in the model) is linked to the relative improvement expected when exploration is pursued over exploitation (selection of those for which there is a large amount of data). An adaptive momentum mechanism for updating the model has been developed so that the incorporation of new data into the model is dependant on the amount and recency of the information already stored. The methods used allow a system of multiple agents to be adaptive to changes in the service environment improving the overall QoS of the system, and may be made more effective through introducing collaborative strategies.

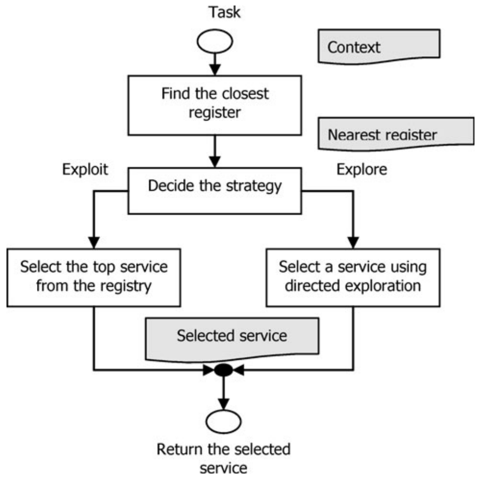
Task
Context
Find the closest register
Nearest register
Exploit
Decide the strategy
Explore
Select the top service from the registry
Select a service using directed exploration
Selected service
Return the selected service

FIGURE 2. Exploration–exploitation decision.

Two collaborative gossiping strategies have been investigated, which vary in the degree to which the selector agents share information. The first strategy, *anonymous gossiping*, involves only partial sharing of information and allows selector agents to gain a better estimation of the distribution of QoS attainable in the network on which the exploration–exploitation control is based. The second collaborative strategy, *full gossiping*, involves sharing detailed information about providers between selectors to speed up learning through exploration. The agents, although cooperative may, however, choose to be selective with the information about providers which they share with others so as not to create unfavourable competition on a subset of service providers, and hence undermine their own performance—*secretive full gossiping*.

The task processing cycle is illustrated in Fig. 3 whereby a task is dispatched to the selector agent and based on both the

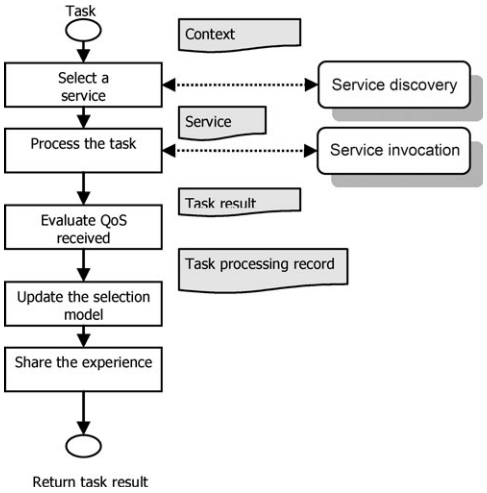

FIGURE 3. Task process cycle.

results of functional service discovery and the selection model built up so far, a service is selected to process the task. The QoS with relation to the task is calculated and used to either *augment* the model if the chosen service was not experienced in the past, or *adapt* the model in the case that there was past experience.

The selection model consists of *registers*, which represent clusters of experience for services used for particular tasks and contexts and are used to simplify the problem space. An important autonomous decision that the selector agents must make is whether to exploit their existing (usually incomplete) model and choose the service which they expect will act best or explore the service landscape further and either select a service for which there is a sub-optimal expectation of QoS or for which there is no prior experience. In Mercury, the calculation of the expected gain from exploration is distributed by making the agents collaboratively share their expected outcomes. This ensures that agents have a reliable understanding of the distribution of QoS achievable throughout the network of services, improving their decision-making ability of whether to explore or not. Further details of the Mercury framework including the structure of the model and algorithms involved with its construction and adaptation are discussed in depth in [8].

In order to quantitatively compare the main features of the Mercury framework, a simulation environment has been developed which can be populated with *n* providers of a single service and *m* service selector agents. We abstract away from the notion of consumers in this case and assume that both the task and the context parameters of the problem stay constant.

We were particularly interested in investigating the effectiveness of the system in the case where QoS of a particular service degrades depending on how many simultaneous connections there are to it at any one time. In this sense, there is competition for resources and in order to reach an optimal configuration of service selection, it is necessary for the selector agents to both be able to form relationships with certain providers while remaining adaptive to changes. In the simulation, the environment is dynamic in the sense that resultant QoS is non-deterministic from an individual selector's point of view due to competition and the distribution of QoS capability can be parameterized.

For all of our experiments, the simulation was set up with 30 service providers and five selector agents and consumers. The QoS capability distribution was set to uniformly increase such that the first provider had the minimum capability and the 30th provider had the maximum (zero and one, respectively). At each time step in the simulation, each selector agent chooses a provider to be invoked and receives the measure of QoS from the provider as a result. The internal selection model is built up through subsequent time steps and at the end of each time step, each selector agent may gossip with other selector agents, depending on their gossiping strategy. The results are averaged over 10 runs.

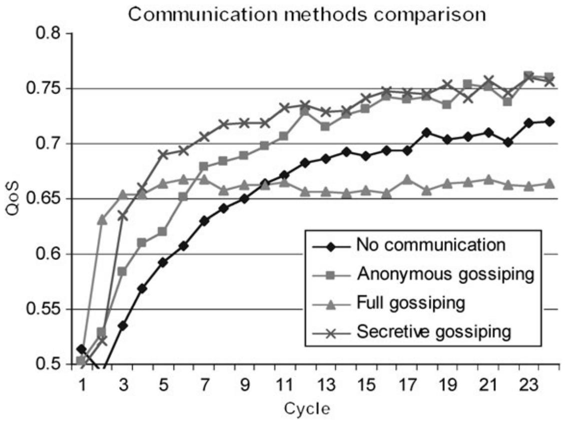

FIGURE 4. Effect of different selector agent collaboration strategies on resultant system QoS.

The first set of experiments was used to compare the different selector agent collaboration strategies on the resultant system (global) QoS attained (Fig. 4). It is clear that gossiping enables the QoS to be increased faster and rather unsurprisingly full gossiping produces the fastest rate of QoS increase through the initial stages. The full gossiping approach would be highly effective if at some point the service landscapes were to change dramatically. With little or no provider churn, though, full gossiping actually results in a lower QoS than if there was no communication. This demonstrates how, by sharing information about the 'best' provider with other agents results in unfavourable competition whereby relationships between a selector S_1 and a particular provider P becomes infected by another selector S_2, which has gained information about P from S_1 and so believes that such a relationship is best for it too. In this case, the global QoS actually decreases. Secretive full gossiping aims to counteract this effect by not sharing the 'best' providers between selector agents. Indeed, Fig. 4 indicates that the resultant QoS is highest when using the secretive full gossiping strategy. A slight lag compared to the full gossiping curve can be seen and this represents the trade-off of not sharing with other agents the top provider. The secretive full gossiping strategy also clearly performs best in the aggregate performance comparison (Table 1), which takes into account both the resulting level of QoS and the speed with which it is achieved.

TABLE 1. Average aggregate effect of different selector agent collaboration strategies on resultant system QoS derived by averaging each of the 25-cycle sequences in Fig. 7.

Collaboration strategy	Aggregate performance
No communication	0.65
Anonymous gossiping	0.69
Full gossiping	0.65
Secretive full gossiping	0.71

The anonymous gossiping strategy clearly also proved to be very good but elicits slower convergence which demonstrates that there is a case for sharing direct references to providers such as in the full and secretive strategies. Nevertheless, its effectiveness highlights the importance of collaborating to improve the data on which the exploration/exploitation decision is based.

The second set of experiments set out how the adaptive exploration probability mechanism employed by Mercury compared to a fixed strategy. For all the experiments, the secretive full gossiping strategy was used although the other strategies produced similar results when tested.

Figure 5 shows the results from this second experiment set and shows that the adaptive exploration mechanism is particularly useful in the initial stages where little is known about the services available. It also results in a level of QoS almost as good as the best fixed level of exploration found (0.2). Its main use, though, is the adaptivity, which it gives the system, allowing the selector agents to choose the appropriate amount of exploration given the conditions in the network and the accuracy of their selection models, rather than performing 'blindly' following a fixed probability of exploration or perhaps a pre-defined exploration–exploitation scheduling function.

The Mercury framework is a concrete illustration of how emergent properties can be leveraged to improve global system behaviour in SOAs, and particularly in P2P cases such as Nexus. The combination of local decision-making (exploration/exploitation strategy) with diffusion of QoS information (gossiping) allows a population of selectors with variable needs to collectively identify and converge towards a configuration that meets the requirements of a majority of participants. Moreover, this distributed problem-solving is largely implicit: the establishment of preferential relationships between selectors and providers incorporates any bias associated with initial conditions and/or the influence of the early history of the system. For instance, in the case that there is competition between two or more selectors for a contended resource, the progressive gain of momentum will ensure that random fluctuations are amplified to the point where only an adequate subset of all competing selectors keep their affiliation with the service. By forcing the 'losers' to identify an alternative provider, this process usually leads to improve global QoS, without any need for central planning or explicit negotiations between selectors. Furthermore, since QoS is constantly re-evaluated, Mercury is capable of detecting and adapting to change the circumstances, whether they affect the service consumer (e.g. new requirements), the provider (e.g. change of context) or the relationship between them (e.g. bandwidth shortage). This effectively means that a stable configuration may spontaneously become unstable when the existing web of selector–provider relationships is no longer adequate, allowing the system to self-organize into a new stable state reflecting the changing conditions. Depending on the severity of the perturbation and/or on the presence of strong coupling (e.g. intense competition for services), the process can lead to a cascading reorganization or, on the contrary, be confined to a small region of the system. Most critically though, this global plasticity is achieved without any modification to the selector agents' behavioural repertoire (there is no explicit 'emergency response mode'). So, by any practical definition, the spontaneous adjustment to changing conditions is an emergent property of the whole system, mediated exclusively by local decision-making.

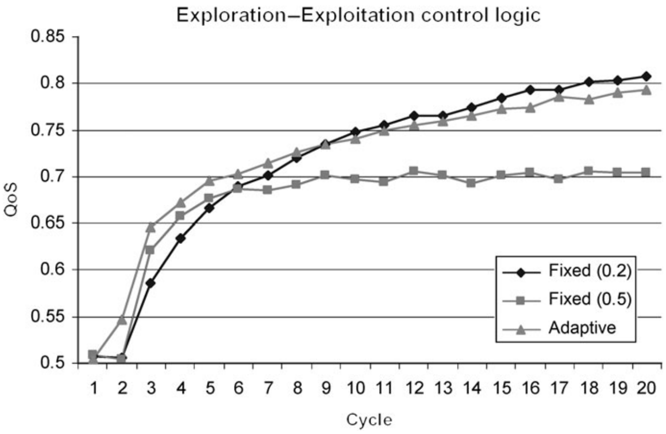

FIGURE 5. Comparison of fixed versus Mercury adaptive exploration probability mechanisms.

2.2. Self-organizing multi-agent systems

The primary aim of the project CASSANI (Complex Adaptive Self-organising Societies in Agents Network Infrastructures; Southampton branch of the Hyperion cluster project) is to investigate and develop the basic autonomic mechanisms needed to enable the Hyperion agents to self-organize, self-repair and self-optimize in response to dynamic environments. Such autonomic behaviour is required to ensure that agents are able to best achieve both their individual and collective objectives [6].

In battlespace-networked systems, the resources available (communication and computation) to the agents are severely constrained. In such cases, it is impossible for the *a priori* system design to continue to be maximally effective because many of its operational assumptions and parameters are changing. To this end, our aim in this project is to devise mechanisms and algorithms to formulate autonomic behaviour within such battlespace systems.

2.2.1. Self-organization

Self-organization refers to the process by which a system changes its internal organization in order to adapt to changes in its goals and environment without explicit external control. In particular, a self-organizing system functions through contextual local interactions. Without central control, these components within the system attempt to individually achieve simple tasks. The particularity of self-organized systems is their capacity to spontaneously (without external control) produce a new organization in case of environmental changes. These systems are particularly robust, because they adapt to these changes, and are able to ensure their own survivability. In some cases, self-organization is coupled with emergent behaviour, in the sense that although individual components carry out a simple task, as a whole they are able to carry out complex tasks emerging in a coherent way through the local interactions of the various components.

Self-organization can, in certain instances, result in emergent behaviour that may or may not be desirable. Therefore, one of the major challenges in devising novel techniques to formulate such self-organizing systems is to design mechanisms that encourage patterns of individual behaviour and interactions that do indeed enhance the performance of the system rather than degrade it. To achieve such desirable global emergent behaviour, local agent behaviours and interactions should comply with some behavioural framework dictated by a suitable theory of emergence. We can find inspiration for such theories within various disciplines. For instance, the theory of social science, nature (i.e. eco-systems, behaviour of social insects), biology, organization theory and economics provide a diverse array of examples of self-organization and emergence in everyday life [9–11].

2.2.2. *Self-optimization*

Rather than managing their dynamic behaviour through centralized control, there is a trend to design systems to self-organize and self-manage themselves to be autonomic. To improve their efficiency, these systems are further expected to self-optimize by allocating resources or services in an optimal manner. In this part of our work, we will investigate how we can self-optimize systems for autonomic behaviours.

Self-optimization can be achieved as an emergent behaviour of agents interacting cooperatively or competitively in the system. One disadvantage of a cooperative approach is that information must be shared for the best performance. However, in these large complex multi-agent systems, agents (which could be companies or computer nodes) have their own goals and when dealing with a highly uncertain context, assuming self-interest is the safest alternative (such that they keep their information/preferences about the resources or services they contribute or consume private). Thus, we advocate a competitive approach. In particular, we believe agents strategically compete in such systems to maximize their individual utility, and it is from this complex strategic behaviour that a self-optimized behaviour will emerge.

Thus, we intend to investigate the different decentralized approaches for self-optimization in systems where we assume that agents have their own aims. Indeed, we will study such techniques to evaluate their effectiveness within static and dynamic environments. Self-optimization is typically achieved by the efficient decentralized resource/service allocation, a subject that has long been studied in economics [12]. Thus, our study will be specifically concerned with using economic metaphors and tools to achieve self-optimization. Broadly speaking, such work can be non-priced based (employing a mechanism that does not involve price or payment for resources) or price based (using price as an economic motivator). The former is usually based on game-theoretic models (with selfish agents that seek to maximize their individual return) or techniques based on decentralized algorithms (with non-selfish agents that cooperate and have the individual aim of maximizing social welfare). The latter is a market-based approach with the ability to facilitate resource allocation based on very little (price) information. Indeed, the market minimizes the dimensionality of messages required to determine Pareto optimal allocations [13]. Furthermore, the market-based approach is more flexible than a non-price-based approach through its distributed nature and its reliance on local decision-making and is more dynamic through its ability to be robust and resilient in changing environments (by reacting effectively to changes). For these reasons, we believe a market-based approach is more appropriate to provide the self-optimizing behaviour required for an autonomic system.

2.3. Adaptive NEC network infrastructure

The aim of SHIELD (Imperial branch of the Hyperion cluster project) is to design an adaptive network architecture with enhanced functionality and resilience, for networks for battlespace communication and information services. Network security is of primary concern for such communication networks. In the future, cyberwars will constitute a significant part of warfare, and DoS attacks with the purpose of preventing legitimate users from using a specific network resource are already common. Our aim is (i) to evaluate overall infrastructure resilience and adaptability in the presence of dynamically varying service requests and possible network failures and threats, (ii) to develop detection schemes that monitor traffic continuously so as to respond to develope failures and attacks with a minimum number of false alarms and missed detections, (iii) to design and evaluate a comprehensive DoS response scheme.

Various commercial, governmental and organizational sites have been subject to major DoS attacks in the last decade, causing significant financial loss and halt of services. Since many services that are vital for the welfare of the public are becoming more and more dependent on network communications, the necessity of defence against DoS is becoming a fundamental issue in network security. The first step of any protection scheme against DoS is fast detection before destructive traffic build-up.

In our current research, we propose a detection scheme for DoS attacks using Bayesian classifiers and random neural networks (RNN). In the first step, the features to be used in the detectors are selected. Then using normal and attack traffic patterns, estimates of probability density functions for these features are determined off-line and likelihood ratios are computed. These likelihood ratios and normal and attack patterns are also used for the training of the RNN to discriminate between two types of traffic. In the detection phase, a likelihood value describing the possibility of an attack, for each feature of the incoming traffic is evaluated, then likelihood values for various features are combined with two different methods: (i) fusion of likelihoods with the mathematical averaging operation; (ii) fusion by the use of RNNs. Thus, a decision is given about the category of the incoming traffic, whether it is attack or normal traffic.

2.3.1. Selecting the input features

The selection of useful and information bearing input features is vital for successful detection of DoS. We have used six features in our scheme, namely: bitrate, change in bitrate (acceleration), entropy for bitrate, Hurst parameter, delay and delay rate.

(i) *Bitrate*: Observation of high-bitrate values is the most conspicuous property in flooding DoS attacks. A sustained high bitrate would be an important indicator of DoS, if not absolute proof, since flashcrowds may also create a similar effect in the traffic.

(ii) *Increase in bitrate*: Gradual or sudden increase of bitrate could be the sign of a flooding attack. In pulsing DoS attacks, bitrate would undergo increasing and decreasing periods consecutively.

(iii) *Entropy of bitrate*: Entropy is a measure of randomness or uncertainty of information. It has been reported in technical literature that the entropy of normal Internet traffic and traffic under DoS attack differ significantly and hence there have been some studies where it is used in discriminating between attack and non-attack traffic [14].

(iv) *Hurst parameter*: Self-similarity properties of normal and attack traffic are different. Hurst parameter is computed as an indicator of the self-similarity of the related process and has previously been suggested to be used in DoS detection [15]. We have used the (R/S) analysis to evaluate the Hurst parameter for the bitrate [15, 16].

(v) *Delay*: During a DoS attack, packet delay into the nodes is increased as congestion builds up. We have measured the round trip time (RTT) values of the victim nodes and observed a conspicuous increase, so we utilized RTT in our detection scheme.

(vi) *Delay rate*: In flooding or pulsing DoS attacks, high-delay rates are observed especially at the initial phases of the attack. So, we included delay rate as an input feature to provide additional information.

2.3.2. Statistical information gathering

For each input feature mentioned above, estimates of probability density functions for both normal and attack traffic are obtained. Thus, we computed $f_{\mathrm{feature}}(x|W_{\mathrm{N}})$ and $f_{\mathrm{feature}}(x|W_{\mathrm{A}})$ where 'feature' can be bitrate, bit acceleration, entropy, Hurst parameter, delay and delay rate, x the measured value of the feature from the available traffic data, W_{N} the normal traffic and W_{A} is attack traffic. We have used the histogram method in calculating the estimates of the probability density functions [17]. After obtaining the probability density function estimates for each input for both traffic types, we calculated likelihood ratios for each feature by

$$\frac{f_{\mathrm{feature}}(x|W_{\mathrm{A}})}{f_{\mathrm{feature}}(x|W_{\mathrm{N}})}.$$

2.3.3. Decision taking

Decision is taken in two consecutive steps. In the first level decision-taking step, the value of the each input feature is measured/calculated from the incoming traffic and the computed likelihood functions of the features are re-sorted to give a decision for each feature. Although the individual decisions taken for each feature by Bayesian classifiers would be optimal, false alarms and missed detections are inevitable, due to any imperfections in the data set used in information gathering and possibility of unconformity between the data set and actual traffic. Hence, a second level decision is obtained by the fusion of all first level decisions. We expect to provide a compensation for any possible errors and reinforcement of correct decisions, so that a low level of false alarms and missed detections are observed at the final decision. We have used two different methods for the fusion of information obtained from different features for second level decision taking; mathematical averaging of all six likelihood ratios and feedforward RNNs.

2.3.3.1. Averaging method. In order to fuse the individual first-order decisions, we have calculated the mathematical average of the six input features, to give a final decision, l_{final}, about the nature of the incoming traffic to the victim node (Normal traffic or DoS attack). l_{final} is a number between 0 and 1 and gives the likelihood of a DoS attack in the incoming traffic to the node.

2.3.3.2. RNN method. RNNs are computationally efficient structures and they represent a better approximation of the true functioning of a biophysical neural network, where the signals travel as spikes rather than analogue signals. They can be used in both feedforward and recurrent architectures. In our work, we have used a feedforward RNN, with six input neurons, one hidden layer with 12 neurons and two output neurons.

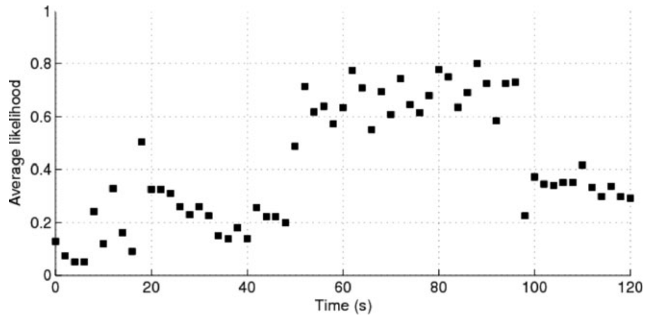

FIGURE 6. Detection results for trace1 attack traffic obtained by average likelihood.

2.3.4. *Experimental results*

We have carried out experiments to evaluate the performance of our detection scheme in our networking testbed. The testbed consists of 46 nodes connected with 100 MB links. We tested our scheme with four different traffic data sets, normal and attack traffic we have designed and two different attack traces (pulsing and increasing-rate attacks) extracted from an on-line repository of traces [18]. We have superposed the attack traffic onto normal traffic from $t = 50$ to 100 s. For each type of traffic, we calculated l_{final} and used the RNN with the individual likelihoods as inputs. To implement the RNN, we have utilized [19]. In all the experiments, we used the ratio of the output neurons of the RNN as the RNN output. Results smaller than 1 indicate normal traffic whereas a ratio greater than 1 can be interpreted as a signal of attack. Some representative results are shown in Figs 6 and 7.

For all of the data sets, we observed that the averaged likelihood detected the attack correctly and also the RNN output was calculated as greater than 1 throughout the duration of the attack, signalling the attack accurately.

In the next step of our research, we are going to design an integrated defence mechanism against DoS attacks, incorporating the Bayesian classifier-based detection mechanism with response approaches of prioritization and rate-limiting. We are also going to investigate failure mechanisms in networks by simulating intermittent failures in our testbed as a first step to understanding the nature of network resilience.

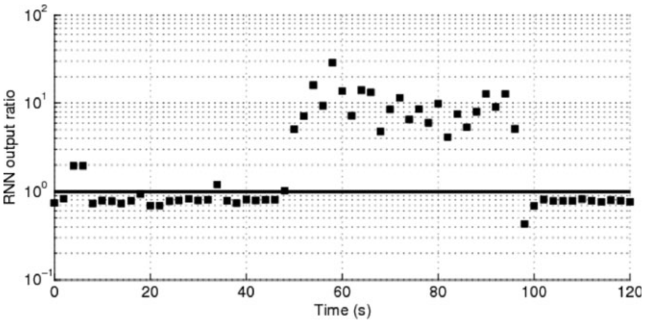

FIGURE 7. Detection results for trace1 attack traffic obtained by RNN.

2.4. Decision Desktop II: agile interfaces for agile services

Decision makers, particularly in the military, are faced with a world that is uncertain and dynamic, where events and courses of action unfold at ‘run-time’ and do not follow predetermined or predictable courses. However, the majority of the tools we use to influence and control this world are frozen at the initial ‘design-time’ stages of their development.

Such approaches, that assume a fixed threat, and provide a fixed defence, are out of touch with reality. Whatever fixed zone our systems are designed for (and limited to), an intelligent adversary will seek to push us out of that zone [20].

To address these challenges, the information systems that support decision makers need to be agile and flexible—reconfigurable at run-time, and with minimal built-in assumptions. Rather than specialized systems optimized for efficiency in one narrow problem space, we require more generalist systems providing greater effectiveness by keeping our option space as open and large as possible.

The goal of this work is thus to demonstrate that agility can be improved by shifting efforts from design time ‘optimization’ towards run-time reconfiguration, placing the tools for agility into the hands of military users in theatre, rather than service providers at home. This could actually reduce workload since the tools are malleable to suit the task at hand, rather than being designed for tasks envisaged some years previously. For example, decision makers would have control over the types of information they wish to see, the level of detail and the attributes of interest. The project is creating a flexible framework that facilitates the rapid integration of new functionality and new information types and sources, and minimizes the effort required for a developer to add these new capabilities. This also facilitates automated feeds of information between applications, rather than disjointed ‘swivel-chair’ interfaces requiring manual re-keying of information, as is often the case at present.

In the first phase of the Decision Desktop project, we developed a proof-of-concept agile information tool that can be used to acquire, visualize and manipulate diverse and dynamic battlespace information, then create flexible visualizations according to the immediate military imperative. It thus enables decision makers to obtain the information they want, when they want it, in a form that makes sense for the task at hand. This tool was initially conceived in the DARPA Coalition Agents Experiment [21], which used agent-based computing approaches to address run-time interoperability of heterogeneous systems.

Our design philosophy was to minimize design-time assumptions in order to maximize the run-time flexibility of the system. In practical terms, we needed to avoid placing arbitrary limitations on, for example:

(i) where information comes from;
(ii) what information is required;
(iii) how the information will be displayed.

The system is based upon an architecture that seeks to provide extensibility at all stages of the system life cycle (design time, assemble time and run time [22]). We adopted a technical approach with four strands: flexible visualizations, ‘plug-and-play’ components, ontologies and software agents and services. These elements are described further below, and illustrated in Fig. 8.

Flexible, generic visualizations: Visualizations (such as geographical maps or Gantt charts) need to be able to usefully display diverse types of object, in a variety of ways. They need to be able to show multiple possible values (where information is conflicting or changing), and show objects at different levels of detail or granularity. Objects displayed in views do not form a static ‘picture’ presented to a passive user as a finished artefact, but are interactive; the user can ‘drill down’ to further information about an object, or transfer it to another view to gain different perspectives on it (Fig. 9).

Plug-and-play components: To enable a system to be reconfigured over short timescales, and to evolve over longer timescales, it should be assembled at run-time from a ‘toolbox’ of components that can be plugged together to select and visualize the desired data. The majority of the system’s functionality is provided by plugging components into a minimal core framework. The design allows new components to be plugged in on-the-fly while the system is running, and also provides for the addition of entirely new classes of component with negligible effort. This allows new, unanticipated information sources or visualizations to be connected to the system without affecting existing components. For example, a plug-in has been created to enable Decision Desktop to discover, and communicate with, nodes on the Nexus middleware (discussed earlier). Components are loosely coupled, with minimal interdependencie, and communicate by event passing. They can discover one another via a registry mechanism.

Ontologies: In many software tools, the domain models are usually implicit and hidden, hard-coded within the software and thus very hard to change. Adapting such systems to new situations can be almost impossible without completely re-writing them. Ontologies enable explicit, portable and interchangeable domain models that are easier to change and verify for new situations, and facilitate the merging of diverse information from multiple, unanticipated sources. They enable the software to be developed in a domain-independent manner, increasing its adaptability. Decision Desktop employs the resource description framework (RDF) and web ontology language semantic web specifications.

Software agents and services: Rather than accessing a static, predetermined structure of databases and servers, flexible tools should be able to dynamically access information providers as required by current operations [21, 23, 24]. With appropriate support, Decision Desktop can dynamically discover services and agents on a network in order to access diverse and unexpected information sources and functionality.

Recent work has focussed on improving the architecture and implementation via the introduction of simple, flexible event-based communication between components (including

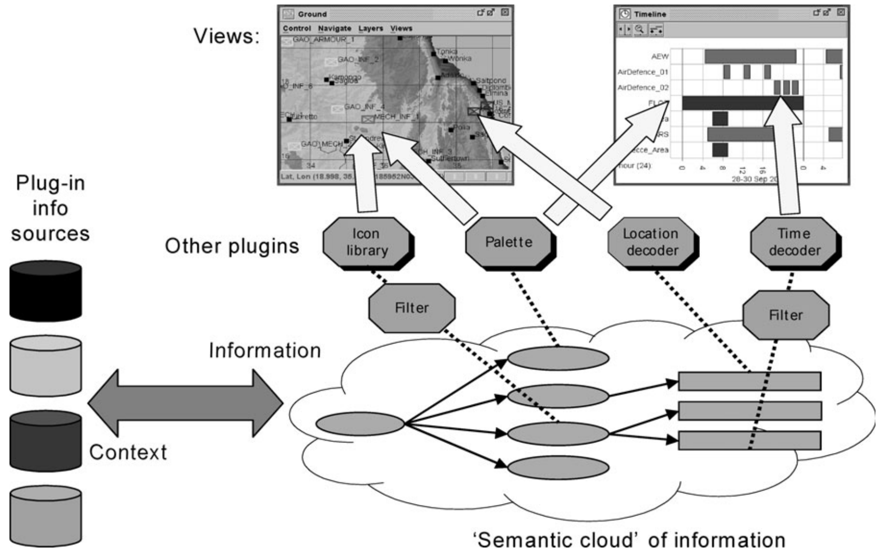

FIGURE 8. Outline of Decision Desktop architecture. Information sources feed into a knowledge base of interlinked, semantically marked-up information, from which plug-in components extract information, and map it to visual elements that are rendered in the user interface by further components.

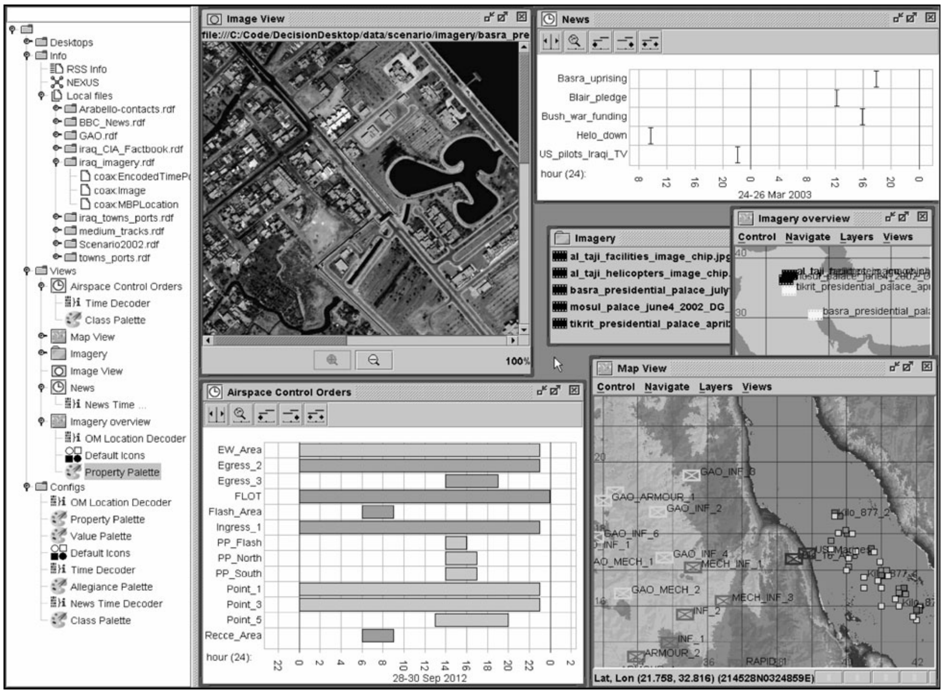

FIGURE 9. A Decision Desktop user interface. The tree-view ('overview panel') on the left shows the available information sources and the components loaded into the system, whereas the central area contains a number of views onto the information, such as maps, timelines (Gantt charts) and images.

core components). The event model is richer and more flexible than standard Java events, yet requires less 'boilerplate' code for keeping track of event subscribers. Subscriptions can be explicitly created between components, or can be implicitly and automatically set up when a component is created.

Recent standards such as the SPARQL query language for RDF have enabled a standardized and concise approach for flexibly subscribing to information. Future work will need to provide user-friendly mechanisms for constructing queries, but the underlying mechanism avoids the need to laboriously construct each new type of query programmatically.

2.4.1. *Results*

Earlier work has been demonstrated within the DARPA Coalition Agents Experiment [21] and the Shared Tactical Ground Picture coalition programme. In the former, unexpected data from an underwater sensor grid (provided by a new coalition partner) were integrated in a matter of hours. The data could then be visualized in a number of ways as outlined in Fig. 10. In the latter, information using a new coordinate format was integrated via the addition of a simple plug-in created in one day.

More recent work has demonstrated basic interoperability with the Nexus middleware discussed earlier, enabling Decision Desktop to discover and display the services available on the network.

The work to date has shown that it is possible to construct a system that provides agile information visualization, avoiding hard-coded domain assumptions. The core architecture and implementation enable components (plug-ins) to be loaded and configured dynamically. These components can interact in order to collect, process and present diverse information. The improvements over previous versions of Decision Desktop are expected to enable significantly easier and richer integration with the other Hyperion partners.

Future work will need to extend the architecture and implementation for tasking the underlying layers of the system, as developed by the other project partners (pushing out information and instructions as well as pulling in information). Another area for future investigation is the management of groups of information objects, to support the organization and aggregation of information.

2.5. NEC scenario and BLaDE integration

General dynamics UK's contribution to Hyperion includes the provision of a relevant military context and a scenario within which the capabilities of the Hyperion projects could be demonstrated. An initial set of military vignettes have been drafted as the focus for engagement with the Hyperion research teams to establish their detailed research requirements. The vignettes are:

- *Collaborative planning*: A brigade commander wants an *ad hoc* teleconference and collaborative planning session

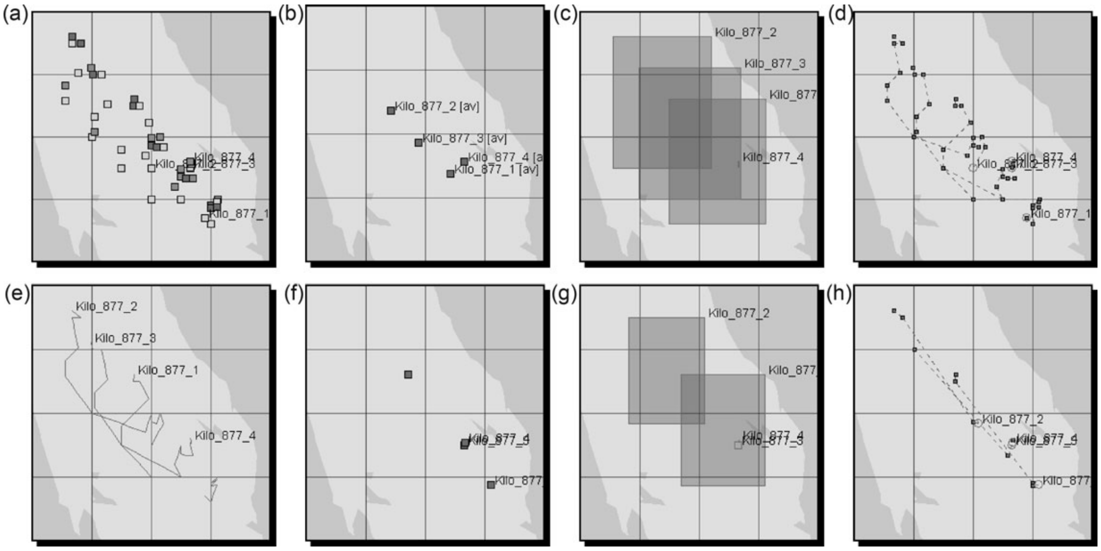

FIGURE 10. Given a single set of data, a user can switch in seconds between a number of different interpretations of that data (or maintain different interpretations in separate views). (a) Several sets of submarine detection points, colour coded by confidence values. (b) The average of each set. (c) The bounding box of each set. (d) Tracks generated from the sets by sorting them into time order. (e) The points joined up into lines. (f) Points filtered by confidence value. (g) Bounding boxes of filtered points. (h) Tracks generated from filtered points. A user can change between these interpretations in seconds.

with deployed subordinate commanders and key staff at brigade HQ.

- *Capability resilience*: Emergency change of control for a battle group HQ that has come under attack.
- *ISTAR support*: Provision of UAV imagery (via a remote viewing terminal at brigade HQ) to a deployed battle group commander to support a key decision.
- *Network attack*: DoS attack.

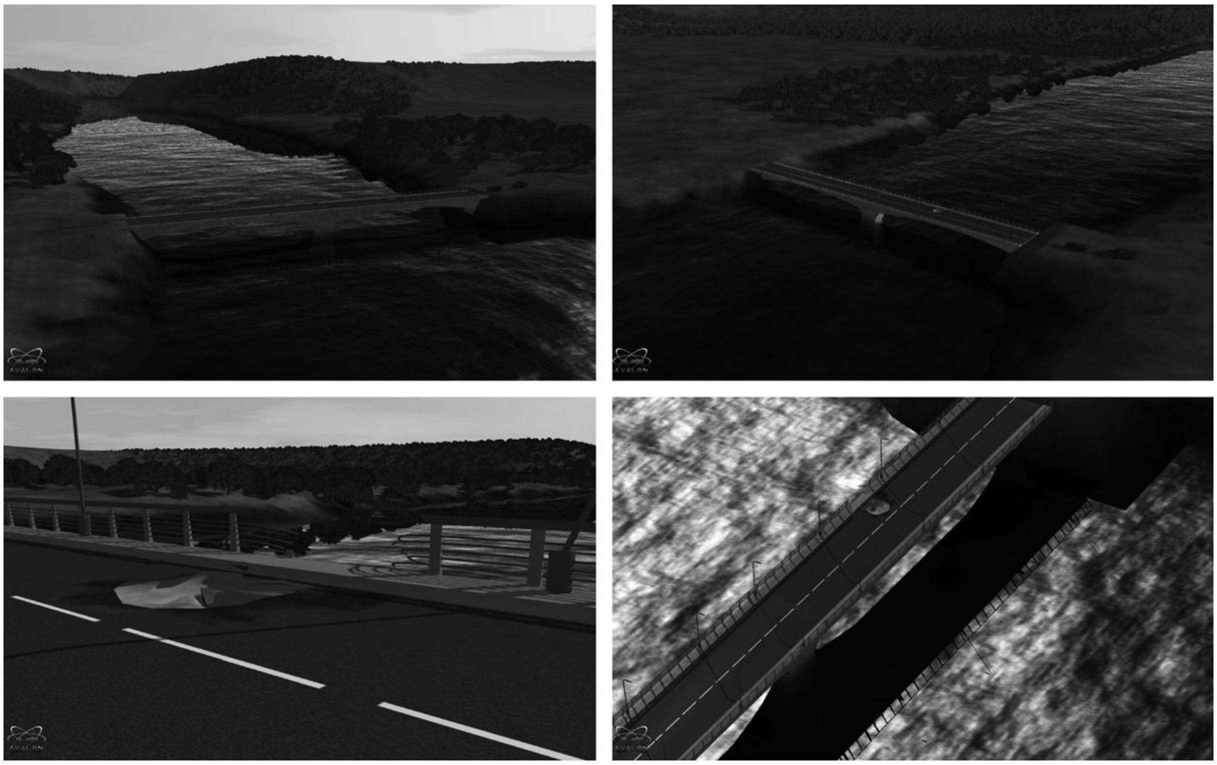

FIGURE 11. Computer-generated imagery of a bomb-damaged bridge.

- *Agile grouping*: The brigade commander needs to create an unplanned force grouping around a specialist reinforcement unit.

Immediate work is focusing on vignette 3—ISTAR Support—and using it to prototype an interface between general dynamics UK's BLaDE synthetic environment and BT's Nexus middleware.

Vignette 3 addresses the agile provision of support from a UAV across the tactical communications network. The advance of a battle group is impeded by the explosion of a car bomb on a bridge on the convoy's designated route. The battle group commander, stuck further back in the convoy, needs to remotely survey the situation, consult with his engineering officer and his superiors and make a decision on how to proceed. A UAV is tasked to fly over the bridge and the resultant imagery is to be made available to the battle group (Fig. 11).

A computer-generated forces simulation suite will be used to create synthetic sensor reports about the status of vehicles in the convoy and the UAV. A detailed 3-D model will then be used to generate computer imagery to use in ISTAR reports from the UAV. Extensible messaging and presence protocol (XMPP) has been chosen as the method of disseminating these synthetic sensor reports to the Nexus middleware, as it can be easily extended to carry custom XML payloads, and it has the ability to easily cross network boundaries so research partners can work together from their own networks with minimal network reconfiguration.

Future work will include developing the Hyperion protocol on top of XMPP [25] to integrate more tightly with the Nexus architecture and allow more interaction between Nexus nodes and the synthetic environment. A crucial part of the development of the synthetic environment will be the modelling of data links and background network traffic to accurately and realistically emulate the stresses on the tactical communications network infrastructure. Agents hosted on the Nexus middleware can then be supplied with real-time communications metrics, giving them the situational awareness they will need to be able to adapt to changes in the network environment. The vignettes will be developed further and eventually integrated into a single consistent story that can be represented in a complete, end-to-end demonstration of Hyperion's capabilities.

3. CONCLUSION

The application of autonomic computing and agent technology has significant potential to resolve a number of emerging requirements for UK defence networks. In particular, it addresses the need for agility, responsiveness and resilience. For example, the use of agents as service brokers and negotiators for C2 systems will support the flexible interaction of multiple commanders to achieve shared situational awareness.

As illustrated in the results of the work to date, automation, agents and SOA can make a significant contribution to the requirements for resilience, adaptability and ease of administration in future defence ICT systems. A pivotal issue however is the question of what bandwidth will be available in battlespace networks by 2015. If it remains constricted then the focus of research now needs to be on bandwidth optimization and pre-processing of data at source. If a richer network capacity will exist then more effort can be concentrated on knowledge extraction and visualization. In this work, we have assumed that the former will be the case, but we have also developed demonstration capabilities that adapt to situations where greater bandwidth is available.

REFERENCES

[1] Alston, A. (2003) Network-enabled capability—the concept. *J. Def. Sci.*, **8**, 108–116.

[2] Kaveh, N. and Ghanea-Hercock, R. (2004) NEXUS: resilient intelligent middleware. *BT Technol. J.*, **22**, 209–215.

[3] Huhns, M.N. and Singh, M.P. (2005) Service-oriented computing: key concepts and principles. *IEEE Internet Comput.*, **9**, 75–81.

[4] Van Roy, P. *et al.* (2005) Self-management of large-scale distributed systems by combining peer-to-peer networks and components. CoreGRID Technical Report TR-0018.

[5] Mondejar, R. *et al.* (2006) Enabling Wide-Area Service Oriented Architecture through the p2pWeb Model. *Proc. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 89–94.

[6] Kephart, J.O. and Chess, D.M. (2003) The vision of autonomic computing. *IEEE Comput.*, **36**, 41–50.

[7] Mantaray. http://www.mantamq.org.

[8] Jakob, M., Healing, A. and Saffre, F. (2007) Mercury: multi-agent adaptive service selection based on non-functional attributes. *Proc. 2nd Int. Workshop on Engineering Emergence in Decentralised Autonomic Systems (EEDAS)*, 22–31.

[9] Serugendo, G.D.M., Gleizes, M.-P. and Karageorgos, A. (2005) Self-organisation in MAS. *Knowl. Eng. Rev.*, **20**, 165–189.

[10] Mano, J.-P., Bourjot, C., Lopardo, G. and Glize, P. (2006) Bio-inspired mechanisms for artificial self-organised systems. *Informatica*, **30**, 55–62.

[11] Hassas, S., Serugendo, G.D.M., Karageorgos, A. and Castelfranchi, C. (2006) Self-organising mechanisms from social and business/economics approaches. *Informatica*, **30**, 55–62.

[12] Mas-Collel, A., Whinston, W. and Green, J. (1995) *Microeconomic Theory*. Oxford University Press.

[13] Jordan, J.S. (1982) The competitive allocation process is informationally efficient uniquely. *J. Econ. Theory*, **28**, 1–18.

[14] Feinstein, L., Schnackenberg, D., Balupari, R. and Kindred, D. (2003) Statistical approaches to DDoS attack detection and response. *Proc. DARPA Information Survivability Conference and Exposition (DISCEX)*, 303–314.

[15] Xiang, Y., Lin, Y., Lei, W.L. and Huang, S.J. (2004) Detecting DDOS attack based on network self-similarity. *IEE Proc. Commun.*, **151**, 292–295.

[16] Cajueiro, D.O. and Tabak, B.M. (2004) The Hurst exponenet over time: testing the assertion that emerging markets are becoming more efficient. *Physica A*, **336**, 521–537.

[17] Duda, R.O., Hart, P.E. and Stork, D.G. (2001) *Pattern Classification*. John-Wiley and Sons.

[18] UCLA CSD packet traces: http://www.lasr.cs.ucla.edu/ddos/traces/public/usc/trace1/exp2/udp/

[19] Abdelbaki, M. (2007) Matlab simulator for the RNN, http://www/cs/ucf.edu/~ahossam/rnnsim

[20] Allsopp, D.N. (2006) Mechanisms for agility. *Eleventh Int. Command and Control Research and Technology Symp.*, Cambridge, UK, September 26–28.

[21] Allsopp, D.N., Beautement, P., Bradshaw, J.M., Durfee, E.H., Kirton, M., Knoblock, C.A., Suri, N., Tate, A. and Thompson, C.W. (2002) Coalition agents experiment: multiagent cooperation in international coalitions. *IEEE Intell. Syst.*, **17**, 26–35.

[22] Beautement, P. (2006) Agile and adaptive coalition operations—leveraging the power of complex environments. *Eleventh Int. Command and Control Research and Technology Symp.*, Cambridge, UK, September 26–28.

[23] Beautement, P., Allsopp, D.N., Greaves, M., Goldsmith, S., Spires, S., Thompson, S.G. and Janicke, H. (2006) Autonomous Agents and Multi-Agent Systems (AAMAS) for the Military—Issues and Challenges. In: Thompson, S.G. and Ghanea-Hercock, R. (eds), Defence Applications of Multi-Agent Systems *2005*. Lecture Notes on Artificial Intelligence, **vol. 3890**, 1–13. Springer-Verlag (ISBN 3-540-32832-7).

[24] Allsopp, D.N. Armed Services: Challenges for Military Distributed Systems, In Thompson, S.G. and Ghanea-Hercock, R. (eds), Defence Applications of Multi-Agent Systems *2005*. Lecture Notes on Artificial Intelligence, **vol. 3890**, 1–13. Springer-Verlag (ISBN 3-540-32832-7).

[25] XMPP: www.xmpp.org