Design of a Mobile Agent-Based Adaptive Communication Middleware for Federations of Critical Infrastructure Simulations

Gökçe Görbil and Erol Gelenbe

Imperial College London
Department of Electrical and Electronic Engineering
Intelligent Systems and Networks Group
SW7 2BT, London, UK
{g.gorbil,e.gelenbe}@imperial.ac.uk

Abstract. The simulation of critical infrastructures (CI) can involve the use of diverse domain specific simulators that run on geographically distant sites. These diverse simulators must then be coordinated to run concurrently in order to evaluate the performance of critical infrastructures which influence each other, especially in emergency or resource-critical situations. We therefore describe the design of an adaptive communication middleware that provides reliable and real-time one-to-one and group communications for federations of CI simulators over a wide-area network (WAN). The proposed middleware is composed of mobile agent-based peer-to-peer (P2P) overlays, called virtual networks (VNets), to enable resilient, adaptive and real-time communications over unreliable and dynamic physical networks (PNets). The autonomous software agents comprising the communication middleware monitor their performance and the underlying PNet, and dynamically adapt the P2P overlay and migrate over the PNet in order to optimize communications according to the requirements of the federation and the current conditions of the PNet. Reliable communications is provided via redundancy within the communication middleware and intelligent migration of agents over the PNet. The proposed middleware integrates security methods in order to protect the communication infrastructure against attacks and provide privacy and anonymity to the participants of the federation. Experiments with an initial version of the communication middleware over a real-life networking testbed show that promising improvements can be obtained for unicast and group communications via the agent migration capability of our middleware.

Keywords: Communication middleware, mobile agents, peer-to-peer overlays, critical infrastructures, simulation federation.

1 Introduction

Critical infrastructures (CIs) are systems that are essential for the normal operation of society and of the economy; electricity and power generation and

E. Rome and R. Bloomfield (Eds.): CRITIS 2009, LNCS 6027, pp. 34–49, 2010.

distribution, telecommunications, transportation, and water supply and distribution systems are some of the assets that fall under the CI categorization. A disruption of any such system will have significant adverse effects, not only on the immediately affected infrastructure and its resident region or country but also to other infrastructures and regions that are dependent on the affected systems. Thus governments, owners and operators of CIs are especially interested in critical infrastructure protection (CIP), which relates to the preparedness and response to incidents that affect the normal functioning of CIs within a country or region. This increasing concern for CIP is highlighted by the recently adopted European Programme for Critical Infrastructure Protection (EPCIP), which was created to identify and protect CIs that, in case of fault, incident or attack, could seriously impact its hosting country and possibly at least one other European member state.

The difficulty of CIP research is exacerbated by the fact that many CIs are inter-dependent, both in terms of sectors and regions, meaning a fault in a CI in one sector (e.g. power) will affect CIs in other sectors within the same region (e.g. transportation and telecommunications) and possibly also CIs in other regions. Many of these dependencies may not explicitly be known beforehand and must be discovered through research and experimentation. The difficulty in conducting CI research lies in the fact that existing systems cannot be used or compromised for experimentation, and building and maintaining real-life CI systems merely for experimentation is time-consuming and extremely costly. These difficulties are especially apparent in CIP research which deals with multiple and interdependent CI systems. Thus a good solution to this problem is the use of simulation [1], which obviously will introduce other complications and challenges.

Current CI simulators tend to be built as monolithic and highly specialized software, and the development of a single simulator combining multiple CI sectors is a complex and expensive task. This approach also fails to reuse current software and expertise. Thus in order to evaluate the interconnected behaviour of systems of diverse CIs, a good approach is to use existing CI simulators in a distributed fashion rather than to develop a single simulation system for the aggregate, so that multiple simulators will coordinate and cooperate. Since each CI simulator is typically quite costly and contains domain specific knowledge, such simulators will often be proprietary, and will be stored, managed or updated and run in geographically distributed and distinct centres. Access to distinct simulators is then only available via a wide-area network (WAN), typically the Internet. Users who would like to organise and run a joint simulation would then need to form a federation of simulators for the simulation of multiple CI sectors.

In a federated simulation system, a group of modeling and simulation tools residing on different computers are networked over a WAN and jointly operate as a federation, collaborating and coordinating with each other to accomplish a common simulation task. This allows the reuse of existing simulation models and tools and reduces the cost of developing and maintaining simulation models. The distributed approach to CI simulation for the purposes of CIP is gaining

popularity [2,3] and the EU/FP7 research project DIESIS[1] [4] provides an instance for investigating this approach. DIESIS proposes the establishment of a European modeling and simulation e-infrastructure based upon open standards to support research on all aspects of CIs with a specific focus on their protection.

Thus, the work presented here is part of the research conducted by the Imperial College team in DIESIS, where we design the core of this federated simulation e-infrastructure as a middleware that enables interoperability between CI simulators and federations of CI simulations through the Internet.

In this paper, we present the design of the communications layer, which is part of the e-Infrastructure Interoperability Middleware (EIIM) of DIESIS project, depicted in Fig. 1. The EIIM provides components necessary to interoperate different simulation tools within a federation, such as simulation time synchronization, federation management and control, scenario configuration, a knowledge base system (KBS) together with ontologies, grouped into the federation logic layer which can be accessed by the federate through the federate/middleware interface. The bottom layer of the EIIM consists of the set of network communication facilities grouped into a communications layer (CL). Central to the design of the CL is the use of communication elements (CEs), which provide adaptive communication services [5] to federates and the middleware itself for the effective and QoS-aware [6,7,8] exchange of control and information messages.

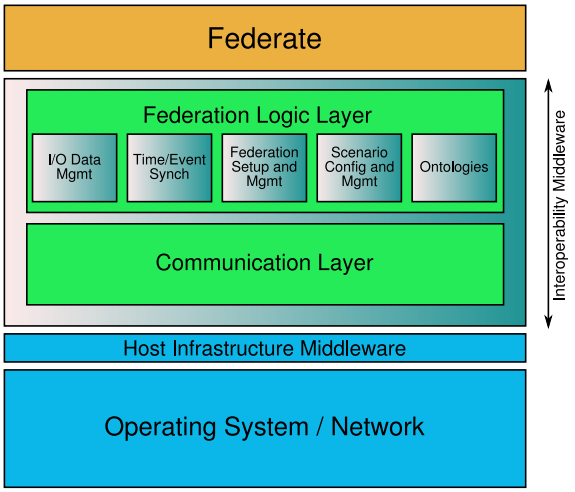


Fig. 1. The e-infrastructure interoperability middleware (EIIM)

The rest of this paper is organized as follows: Section 2 provides a brief overview of existing technologies and current research in the field of simulation federation. Section 3 presents the operational setting and requirements in

[1] DIESIS stands for "Design of an Interoperable European federated Simulation network for critical InfraStructures".

respect to federation communications. We present the design of the architecture for the CL in section 4 and discuss security issues in federation communications in Section 5. Experimental results are presented in Section 6; we conclude and discuss future work in Section 7.

2 Federating Simulations

Several standards have been proposed and used, mainly in the military domain, in order to enable distributed simulations and federations over communication networks. These standards include Distributed Interactive Simulation (DIS), Aggregate Level Simulation Protocol (ALSP), and the well known High Level Architecture (HLA).

DIS [9] is an open standard for conducting real-time platform-level wargaming across multiple host computers and is the basis for human-in-the-loop military training systems, such as the Combined Arms tactical Trainer (CATT). DIS provides no time coordination between independently running real-time simulators; interaction between simulations in DIS federations is achieved by the broadcasting of Protocol Data Units (PDUs), which are broadcast using the unreliable UDP protocol over a best-effort IP network, and therefore may be lost during transmission. Since DIS does not provide a time synchronization mechanism or communication facility other than broadcast over UDP, it is not suitable for federations which require explicit time management, message delivery guarantees and reliability.

ALSP [10] is a protocol and supporting software used extensively by the US military to interoperate analytic and training simulations; like DIS, it was superseded by HLA. Unlike the real-time simulators that participate in DIS simulations, ALSP caters for discrete-event simulators that explicitly manage their time. ALSP provides time management services to coordinate simulation times and preserve event causality across simulations. However, ALSP does not provide any support for real-time communications and although federation communications in ALSP are reliable, they suffer from performance issues, inhibiting the federation from proceeding in an efficient manner, especially in federations with participants dispersed over a wide area.

HLA [11] is a general-purpose architecture that enables distributed computer simulation systems. It seeks to generalize and build upon the results of DIS and ALSP and consists of three parts: (1) compliance rules, governing certain characteristics of HLA-compliant simulators, (2) object model templates, which define a basis for the exchange of data and events between simulators, and (3) run time infrastructure (RTI), which is a collection of software that provides commonly required services to simulation systems. The HLA RTI [12] acts as a distributed operating system for the federation, providing sufficient time management functions so that real-time, time-stepped, event-driven, and optimistic time warp simulations can all run in the same federation [13]. HLA supports these capabilities provided that federates adhere to certain rules that are necessary to realise each service. In order to participate in an HLA federation, a

federate needs to implement the HLA rules, which define the responsibilities of each federate and of the federation. The HLA RTI then provides the services required by federates during federation execution, such as communication and time management services.

HLA aims to enable various types of federations and facilitates for real-time and as-fast-as-possible (AFAP) execution of distributed simulations. However, there are almost no HLA-compliant CI simulators. Furthermore practical experience with the HLA RTI shows that it is not suited for federating highly dynamic simulation systems [14]. Since communication services offered by HLA RTI are not adaptive, they are not suitable for operation in highly dynamic and failure-prone environments. HLA RTI's support for real-time communications is also minimal and insufficient, especially in stress conditions. These shortcomings of HLA have been recognized and partially addressed, either as proposed add-ons or modifications to the RTI [15, 16, 17, 18] or as completely new frameworks [19, 20]. Unfortunately, these proposals have not yet become part of the standard. Moreover, most of the proposals fail to adequately address at least one of the communication requirements of reliability, security, and efficiency (e.g. QoS guarantees) by either restricting their simulation types of interest (e.g. by considering only analytic federations of discrete-event simulations) or the operating conditions that are assumed about the network, e.g. a high-speed local network with no failures.

In order to support reliable and real-time federation communications over dynamic and failure-prone networks, the communication infrastructure must adapt to changes in the physical network, it must be fault-tolerant and efficient. To decrease the need for administrative support, including manual setup and configuration, the communication infrastructure should also be self-configuring. For particular federations, data privacy and network security are also important requirements. Since these requirements are particularly important when the simulations are being used in preparation or in response to a real and critical incident which may be a result of natural or man-made causes, our proposed CL, described in section 4, aims to provide a communication architecture that will meet these important challenges.

3 Requirements of Federation Communications

The EIIM, our interoperability middleware [21] for federations must allow each individual federate to freely join and leave a federation without full knowledge of or by its peer federates. It should allow quick and easy assembly of independently developed simulation models and offer maximum independence to each simulation model. The user interested in the results of the joint simulation should be shielded from how or where these tasks are executed and also from the implementation details of the simulators and of the CL middleware. The communication infrastructure must also enable federates to join and leave the system without interruption to the operation of ongoing federations, and provide mechanisms for the discovery of services offered within the system.

EIIM aims to interconnect a wide variety of modeling and simulation tools into a heterogeneous federation, where federates may differ in their simulation domain, execution environment, and time management mechanisms. A federate may be implemented in one or more physical machines (e.g. with a computing cluster or grid). Various federation types are supported, not only in the nature of the participating federates (i.e. homogeneous vs. heterogeneous federation) but also in the purposes of the federation.

Distributed virtual environments (DVEs) aim to create networked interactive environments, such as virtual worlds, which are useful for training or the observation of the simulated systems in real-time. DVEs typically execute in actual or scaled real-time with a wallclock-driven simulation progress and place strict real-time constraints on the communication middleware. DVEs may tolerate some message loss but they have austere message delivery deadlines for federation messages. While DVEs require QoS guarantees, analytic simulations and decision tools, characterized by the AFAP execution pace model, require reliable delivery of messages. Low communication latency is not strictly required but the federation would not perform well if latency is high. Correspondingly, the communication middleware needs to concurrently accommodate different, and possibly conflicting, communication needs, types of federates and federations.

In order to meet these different requirements, the CL provides multiple communication services, such as reliable and real-time message delivery, under dynamic federation and network conditions. To achieve these objectives, the CL must be flexible and adaptive and self-aware [22,8]. In order to increase usability and applicability, the communication middleware must run using standard network protocols. The use of the Internet and the IP protocol stack as the main communication means for federations offers great flexibility to reach a large number of participants. However, it also creates challenges to the design of the communications architecture because the Internet essentially offers a *best-effort service* that is generally unsuitable for most federation communications.

Thus, in the next section, we propose an intelligent and adaptive communication middleware solution that provides reliable and real-time one-to-one and group communications for heterogeneous federations running over a best-effort IP network, and specifically over the Internet.

4 Architecture of the Communication Middleware

In order to address the communication requirements discussed in section 3, we propose an agent-based, intelligent, adaptive and resilient communication middleware, to provision the required smart, reliable and real-time communication services. The middleware consists of multiple **virtual overlay networks (VNets)** running over a *physical communication network (PNet)*. Each VNet is a *peer-to-peer* (P2P) network of *autonomous, mobile, goal-based software agents*, called **communication elements (CEs)**. Each VNet offers a set of communication services to federates and the EIIM, and all communications are assured by the CEs. This overlay approach does not require any modifications to the

PNet components or protocols, and all the logical functionality is embedded in the CEs which use the capabilities of the physical nodes (PNs) to achieve their goals. The proposed solution based on the CL provides the following capabilities:

- Seamless and reliable communication services in dynamic and unreliable environments: federates can communicate even when the underlying PNs are mobile and subject to failures.
- Resilience to volatility in the underlying network infrastructure: by adopting a self-organizing and distributed P2P architecture, VNets are able to operate even if a subset of the PNs is unavailable due to failures or other incidents.
- Decentralized service discovery: networked resources such as the available simulators, simulation models and data, are discovered in a distributed fashion over the P2P architecture, without reliance on a centralized directory facilitator.
- Self-monitoring and real-time service provisioning: each CE proactively monitors its own performance and if it observes that it cannot fulfill its users' communication requirements according to desired levels on its current host, then it may decide to move to another host where it may perform better. This decision is taken autonomously and CE mobility is transparent to the federates.

We assume that the operating environment of the system is dynamic and volatile, with potential failures in the PNs and the links in the PNet, and possible network disconnects because of mobility of PNs. Network load, communication patterns and requirements of federates may change at any time, as well as the set of PNs and federates in the system. All components within the system can be heterogeneous and the PNet may consist of different types of nodes, e.g. with different capabilities and operating platforms, and different types of networks. The layered approach has the advantage of insulating users of the communication middleware from the dynamic changes that occur in the environment, with the VNets providing all the access functions. VNets have no control over the mobility of PNs, and the CEs optimize VNet services through migration over the PNet and dynamic adaptation of the P2P overlay. Figure 2 presents an example federation and the system components, which are described in the rest of this section.

One major consideration for the communication middleware is portability over different operating systems and platforms. Thus the CL favours the use of services from a *host infrastructure middleware* (as presented in Figure 1) to access network and inter-process communication (IPC) facilities of the underlying operating system. Higher-level standard middleware services are avoided (i.e. distribution middleware), because federated communications are mainly message oriented and as such, they have limited need for remote functions or for direct access to remote objects. The components within the CL communicate asynchronously. IPC, which is much more efficient and faster than network communication, is employed when the communicating entities are resident on the same PN. Otherwise, network communication is used.

The federate is distinct from the middleware, and it is the user of all services provided by the EIIM. A PN may host several federates and a simulator may

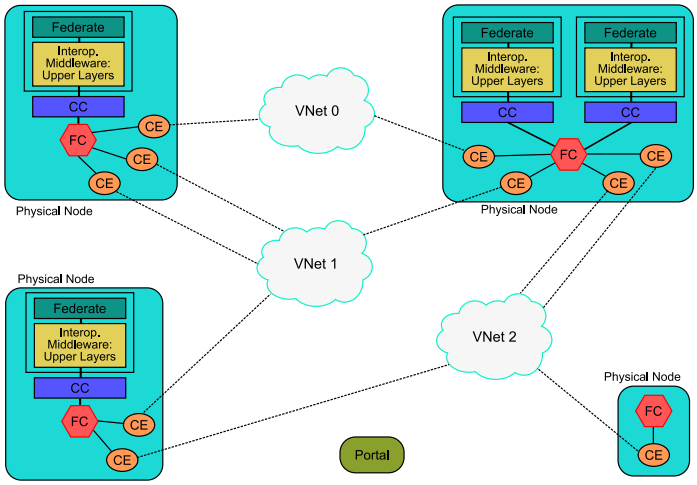


Fig. 2. A federation of simulators, connected through VNets

participate in multiple federations at the same time. Each participation of a simulator is considered as a separate federate by the EIIM. When a federate is instantiated at a PN, a local communication controller (CC) is created to handle all its communication requests. The CC is always located at the same PN as the federate and acts as the intermediary between the federate and the CEs.

Communication Controller. The communication controller acts as the communication endpoint for its associated federate. All communication requests of a federate are first handled by its CC, which selects the best VNet (and therefore CE) to use for the requested communication type. VNets that have been created to handle the communications of a federation are always registered at the CCs of the federates, either at CC or VNet creation time. For each type of VNet, a CE is registered with each CC to handle the communications of the corresponding federate. Since registered CEs may be local or remote, the CE list within a CC includes the current location (e.g. address of the host PN) of each CE and this list is updated by the migration management module of the federation controller (FC) when CE migrations occur. Once the CE that will handle the communication request is determined by the CC, the request is passed on to that CE, using IPC when the CE and CC are co-located, or via the network when the CE is remote.

When messages destined for the federate arrive, they are processed according to their priorities and types by the CC. Message processing includes ordering of messages (e.g. for timestamp-order message reception) and scheduling to prioritize messages (e.g. give processing precedence to real-time messages with closer deadlines), which are done within the CC. The dropping of messages, which have missed their deadlines and therefore are of no further use to the federate, is also done within this component.

Communication Elements. Each communication element (CE) is an autonomous, mobile, goal-based software agent that resides on a PN at any time. PNs provide physical communication and computation capabilities to CEs and a CE uses the PN it currently resides for all its physical needs. A PN may host different CEs during its lifetime and it may host multiple CEs, as well as multiple federates, concurrently. The PNet handles the physical routing of messages between nodes and we assume that the PNet provides a best-effort communication service.

CEs form a distributed P2P overlay network on top of the PNet and each CE (peer) provides one or more communication services. The goals of a CE may be dictated by the VNet it belongs to, and CEs operate and migrate in order to accomplish their goals, where these goals are quality-of-service (QoS) requirements such as minimum end-to-end latency, real-time delivery within a deadline, etc. The mobility of CEs is autonomous and they move to explore better options for their activities. A CE acts as a router for other CEs in the system, forwarding messages on behalf of others. The CL consists of multiple VNets, each VNet providing a different set of communication services to federates based on QoS criteria. While CEs within the same VNet interact and collaborate with each other in order to provide communication services, CEs across different VNets do not normally interact with each other.

The CL operates through VNets, which are self-organizing P2P overlays, and therefore knowledge of the underlying PNet is normally not explicitly needed by the CL. However, CEs continuously monitor their own performance and adapt their behaviour (e.g. the multicast tree or their host PN) in order to offer smart and efficient communication services in dynamic and failure-prone networks. While some types of CEs will be stationary, other CEs may migrate several times during their lifetimes for maintenance, resilience, reliability, security or optimization purposes.

Each CE monitors its performance continuously, and when it detects a significant decline in QoS such as an increase of end-to-end message latency, it probes its peers in the VNet and/or the PNet (based on the performance metric of the CE) in order to learn the current state of the PNet and identify candidate hosts from where it could provide better service. Once a better PN is identified, the CE signals the FC of its host and requests migration. The local FC contacts the remote FC and if the CE's migration request is accepted, then the CE's jobs and services are stopped, it is serialized, and transmitted over the network to its new PN. The FC at the recipient site creates the CE object from the received stream and starts the CE's jobs and services. The initiating FC is also responsible for buffering messages that may arrive for the CE during its migration and ensuring that the CE gets these messages after migration is complete.

Within the CL, each CE is implemented as one or more threads controlled by the CE itself. In order to enable broad applicability and ease of modification, a modular and extensible approach was taken for the design of CEs: CE behaviour is composed of *jobs*, which are reusable agent behaviours and a CE may combine multiple jobs to compose its overall behaviour. Jobs are owned by the CE and the

owner CE has complete control over their execution. Each job runs in a single thread and job threads are protected from outside access (e.g. a CE cannot manipulate another agent through controlling its job threads). A job has access to the internal structures of its CE and can communicate with its CE and other jobs in the CE through asynchronous message passing, using IPC facilities.

Federation Controller. The federation controller (FC) is an operating system resident process (e.g. a UNIX daemon or Microsoft Windows service) that manages the participation of a PN in the PNet and the EIIM. Only one FC runs per PN and the FC on a PN enables the execution and migration of VNs, acting as the "agent world". The FC provides access to the PN's communication facilities for the CCs and CEs on the same PN and implements the host infrastructure middleware to improve portability. The FC provides at least three communication sockets: UDP, TCP, and reliable UDP (R-UDP). Additional sockets may be provided depending on the PN's capabilities. Each type of socket is controlled by a *communication module*. Addition of other sockets as required by the federation and available through the PNet can easily be accomplished due to the modular design of the FC: since all logical functionality is grouped in other components, a new socket may be provided by the addition of a new communication module to the FC. The FC keeps track of the CCs and CEs on its PN in the CC and CE lists. Note that the CE list within the FC is different from the CE list within the CCs, as the FC's CE list contains all CEs on its PN whereas a CC's CE list contains all CEs (local and remote) that have been associated with that CC and corresponding federate.

The FC contains the *network statistics repository*, which is a common database that is shared by CCs and CEs and contains observations of the network, such as measured end-to-end delays to peer PNs. The purpose of this repository is to offer an up-to-date information about the network to CEs to allow them make better configuration decisions (e.g. for migration). The database will typically be updated by CEs as they continuously use the network and evaluate its performance. The location of the database is in the FC since network observations are specific to the PN where the observations are made. The agent management module within the FC manages the execution of the CEs. Since CEs operate autonomously, the FC does not control CE execution by itself but rather according to the requests made by the CEs themselves. The migration management module controls CE migration to and from this PN.

The *message dispatcher* forms a central part of the FC, handling all message communications from/to the local CCs and CEs and de-multiplexing messages to their intended destinations. The message dispatcher uses the CC and CE lists and the message parameters to determine if a message's destination is local, i.e. on the same PN. If the message is local, then IPC is used to forward the message to its destination; otherwise, one of the communication modules (i.e. sockets) is chosen according to the message type to dispatch it through the network. The message dispatcher may incorporate a temporary queue and a scheduler to prioritize among messages.

5 Security

Security is an important aspect of communications, particularly when data confidentiality is required between participants in a federation. In order to provide secure communications, the CL can incorporate standard security methods, such as encryption/decryption, authentication, verified signatures, etc. The design of the CL is modular due to the use of VNets and additional VNets employing standard security methods may be created that provide secure communications between federates. In addition to the three standard sockets provided, the FC may also offer access to native secure sockets via the PNet. When native secure sockets are not available from the network, secure communications may be provided by the use of SSL or TLS over IP. CEs (and potentially federates) may migrate in order to increase security within the network. For example, a CE may migrate from an insecure PN to one that is more secure (e.g. because it hosts native security methods), thereby increasing the security of its communication services.

The CL uses mobile agents to provide adaptive and efficient communication services. It should be noted that our middleware only supports agents (i.e. CEs) developed as part of the CL, and agents developed by other parties cannot migrate and execute on the PNs. Therefore, there are no security issues related to accepting and executing foreign and unknown mobile software. However, there is still the possibility of malicious hosts (i.e. PNs) trying to influence and affect other PNs by corrupting their resident mobile agents. The agents within the CL are designed so that they operate autonomously, so no other entities other than themselves can change their data or execution state. A malicious PN can still block all agent operations on itself, which would be synonymous to that PN having a failure, in which case other agents would ignore that PN and adapt their operations according to the observed failure.

We also need to think about threats that come directly from the network, such as from compromised hosts and denial-of-service (DoS) attacks which are a common form of cyber-warfare and a threat to network security. A typical DoS attack is often distributed (DDos), where the attacker takes control of a number of lightly-protected or compromised computers and uses them to send a large number of packets to the victim PNs. The PNs and links in the vicinity of the target(s) become overloaded and legitimate clients are unable to access or use the targeted PN(s).

DoS defence requires the detection of the attack via the classification of network traffic as normal and DoS, and response to the attack [23,24]. The first step requires fast detection before destructive traffic actually builds up. In [25], a detection scheme using Bayesian classifiers and random neural networks (RNN) is proposed. As a first step, the features to be used in detection are selected. Then, normal and attack traffic patterns are used to train the RNN to discriminate between two types of traffic. In the detection phase, a decision is given about the category of the incoming traffic (i.e. normal or attack) using the trained RNN. Using these approaches, an integrated defence mechanism against DoS attacks, incorporating the Bayesian classifier-based detection mechanism with response approaches of prioritization and rate-limiting, will be developed using the

approach in [25]. CEs may also integrate several different DoS response methods for each aspect of DoS defence.

6 Experimental Results

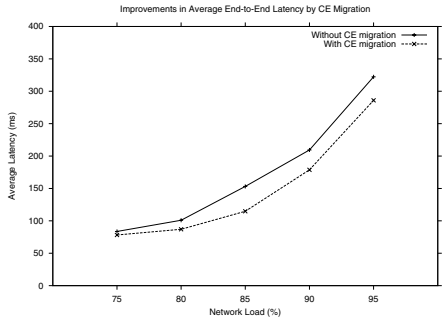
An initial version of the proposed communication middleware has been implemented in C++ with a multi-thread model for efficiency and decreased response time in operations. We have elected to use the Boost C++ libraries [26] to provide the host infrastructure middleware facilities. The network testbed we use consists of 16 physical routers connected in a static real-life topology, running IPv4 and using least-hop routing at the physical layer.

In order to emulate realistic traffic that may be generated by CI simulations, we used an existing wireless mobile ad hoc network simulation on the OMNeT++ discrete-event network simulator [27] and stored all the simulation events in a file. A federation of simulators was emulated by sub-dividing the network's simulated area into four equal quadrants and a federate was then associated with each quadrant[2]. The interactions that take place between these four areas then correspond to interactions between the federates that are installed in the testbed nodes. Since the communication protocol stack is simulated down to the link layer (i.e. MAC protocol level), the federates generate a significant amount of network traffic. Due to the broadcast nature of wireless communications simulated in OMNeT++, federates create both one-to-one and one-to-many communications, which are handled by the communication middleware.

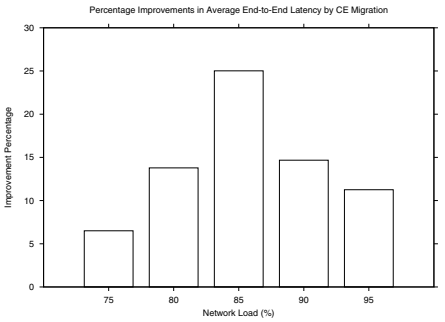
Each data point in the presented results consist of five runs of a federation, where the four federates were placed randomly within the physical network. The bandwidths of the physical links were artificially limited and for each run, random cross-traffic was introduced into the network to create dynamic network conditions. We then evaluated the performance of the communication middleware, with and without agent migration, under different network load conditions. The performance parameter of interest was selected as end-to-end message delivery latency; in the case of one-to-many communications, each message was considered separately. After an initial setup phase of 5 seconds, a federation was executed until simulation completion (about 120 seconds), during which cross-traffic patterns in the physical network were changed every 10 seconds.

Figure 3 summarises the results of these experiments. We observe that CE migration reduces message delivery latency under all network load conditions due to the smart repositioning of CEs over the PNet. CE migration adapts to changes in the PNet, and CEs relocate themselves when traffic conditions change,

[2] This is an example of a single-domain federation where only a single type of simulator is used. While the fidelity of this example application can be considered low when we consider all aspects of CI federation, including federation control and management, this paper focuses on the design of the CL, which does not use the actual data created by different types of federates for its operation but instead uses the delivery requirements for such data. Therefore, our example application is suitable to observe the performance of the CL as it covers all delivery requirements for federates.



(a) Latency values



(b) Percentage improvement

Fig. 3. Improvements in average end-to-end latency by CE migration

as shown in the experiments. Improvements in end-to-end latency with CE migration range from 6% to 25% as shown in Fig. 3. CE migration is especially effective when the network is moderately loaded (e.g. 80%-85%); under lighter load, CE migration does not provide a significant improvement as the PNet can route messages with low latency without the need for dynamic adaptation. CE migration is also effective under heavy load (90%-95%), but we observe a diminished improvement due to the fact that although CEs migrate to better locations over the PNet, the whole network is heavily loaded, increasing optimal latencies.

It is worth noting here that the current middleware does not incorporate adaptive overlay methods for efficient group communications, such as the creation and use of application-layer multicast trees which combine dynamic overlay adaptation (i.e. changing connections between tree nodes) and CE migration. We expect that the addition of these methods will improve group communication efficiency greatly.

7 Conclusions and Future Work

This paper has shown that in order to provide highly reliable, real-time and smart communication services for the federation of CI simulators, the communication

infrastructure needs to address the mobility, unreliability, protection/defence, and dynamic aspects of physical networks. In order to offer smart, reliable, and real-time services to users in mobile, dynamic, and unreliable environments, we propose the use of a mobile agent-based, adaptive and resilient communication middleware. Our proposed middleware consists of virtual overlay networks (VNets) running over the physical communication network (PNet), and each VNet is a self-organizing P2P network of autonomous, mobile, goal-based software agents, called communication elements (CEs). Each CE offers a set of communication services to federates and all communication functions are assured by the CEs. VNets adapt to changing conditions in the PNet through self-monitoring, communication, and migration of their component CEs. The VNet approach allows the easy addition of different services to the system, such as group and secure communications.

Our preliminary experiments with the proposed communication middleware on a real-life networking test-bed, which enables us to test the performance of the VNet approach in real-life situations using physical machines with different capabilities and operating systems, show that agent migration is a viable option to increase communication efficiency of CI federations. Our results indicate that CE migration improves end-to-end latency of message delivery for both one-to-one and group communications. The next step in the development of the middleware is the full implementation of communication modules and the inclusion of reliable and adaptive group communications, such as application-layer multicast protocols, in order to improve group communications for federations of CI simulations. We will also conduct experiments using a heterogeneous and dynamic network with failures to investigate the effects of network failures on the communication and coordination of CI federations using EIIM in a networked environment.

References

1. Rinaldi, S.M.: Modeling and simulating critical infrastructures and their interdependencies. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (January 2004)
2. Duflos, S., Diallo, A.A., Grand, G.L.: An overlay simulator for interdependent critical information infrastructures. In: Proceedings of the 2nd International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX 2007), June 2007, pp. 27–34 (2007)
3. Casalicchio, E., Galli, E., Tucci, S.: Federated agent-based modeling and simulation approach to study interdependencies in IT critical infrastructures. In: Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT 2007), October 2007, pp. 182–189 (2007)
4. DIESIS Project Official Web Page (2009), `http://www.diesis-project.eu/` (Accessed 2009)
5. Gelenbe, E.: Users and services in intelligent networks. In: IEE Proceedings of Intelligent Transport Systems, vol. 153, pp. 213–220 (2006)
6. Gelenbe, E.: Sensible decisions based on QoS. Computational Management Science 1(1), 1–14 (2004)

7. Gelenbe, E., Lent, R., Nunez, A.: Self-aware networks and QoS. Proceedings of the IEEE 92(9), 1478–1489 (2004)
8. Gelenbe, E.: Steps towards self-aware networks. Communications of the ACM 52(7), 66–75 (2009)
9. IEEE Standard 1278, Standard for Distributed Interactive Simulation
10. Weatherly, R., Seidel, D., Weissman, J.: Aggregate level simulation protocol. In: Proceedings of the 1991 Summer Computer Simulation Conference (1991)
11. IEEE Standard 1516, Standard for Modeling and Simulation High Level Architecture
12. Calvin, J.O., Weatherly, R.: An introduction to the high level architecture (HLA) runtime infrastructure (RTI). In: Proceedings of the 14th DIS Workshop on Standards for the Interoperability of Defense Simulations, pp. 705–715 (1996)
13. Fujimoto, R.M., Weatherly, R.M.: Time management in the DoD high level architecture. In: Proceedings of the 10th Workshop on Parallel and Distributed Simulation, pp. 60–67 (1996)
14. Bononi, L., Bracuto, M., D'Angelo, G., Donatiello, L.: Scalable and efficient parallel and distributed simulation of complex, dynamic and mobile systems. In: Proceedings of the 2005 Workshop on Techniques, Methodologies and Tools for Performance Evaluation of Complex Systems (FIRB-Perf 2005), September 2005, pp. 136–145 (2005)
15. Zhao, H., Georganas, N.D.: HLA real-time extension: Research articles. Concurrency and Computation: Practice and Experience 16(15), 1503–1525 (2004)
16. Chen, D., Turner, S.J., Cai, W.: A framework for robust HLA-based distributed simulations. In: PADS 2006: Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation, pp. 183–192 (2006)
17. McLean, T., Fujimoto, R., Fitzgibbons, B.: Middleware for real-time distributed simulations. Concurrency and Computation: Practice and Experience 16(15), 1483–1501 (2004)
18. Lent, R.: Improving federation executions with migrating HLA/RTI central runtime components. In: Proceedings of the 14th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks CAMAD 2009 (June 2009)
19. Boukerche, A., Zhang, M.: Towards peer-to-peer based distributed simulations on a grid infrastructure. In: Proceedings of the 41st Annual Simulation Symposium (ANSS 2008), April 2008, pp. 212–219 (2008)
20. Riley, G.F., Ammar, M.H., Fujimoto, R.M., Park, A., Perumalla, K., Xu, D.: A federated approach to distributed network simulation. ACM Transactions on Modeling and Computer Simulation (TOMACS) 14(2), 116–148 (2004)
21. Ghanea-Hercock, R., Gelenbe, E., Jennings, N.R., Smith, O., Allsopp, D.N., Healing, A., Duman, H., Sparks, S., Karunatillake, N.C., Vytelingum, P.: Hyperion - next-generation battlespace information services. The Computer Journal 50(6), 632–645 (2007)
22. Dobson, S., Denazis, S., Fernandez, A., Gaiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A survey of autonomic communications. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 1(2), 223–259 (2006)
23. Gelenbe, E., Loukas, G.: A self-aware approach to denial of service defence. Computer Networks 51(5), 1299–1314 (2007)

24. Gelenbe, E., Gellman, M., Loukas, G.: An autonomic approach to denial of service defence. In: Proceedings of the 6th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM 2005), June 2005, pp. 537–541 (2005)
25. Oke, G., Loukas, G., Gelenbe, E.: Detecting denial of service attacks with bayesian classifiers and the random neural network. In: Proceedings of the IEEE International Fuzzy Systems Conference 2007, July 2007, pp. 1–6 (2007)
26. Boost C++ Libraries (2009), `http://www.boost.org` (Accessed 2009)
27. OMNeT++ Community Site (2009), `http://www.omnetpp.org/` (Accessed 2009)