

Improving Federation Executions with Migrating HLA/RTI Central Runtime Components

Ricardo Lent

Department of Electrical and Electronic Engineering
Imperial College London, London SW7 2BT, U.K.
E-mail: r.lent@imperial.ac.uk

Abstract— Simulation tools are popular in the design and study of communication systems and networks. The implementation of analytic distributed simulations commonly require a central entity to support certain tasks (e.g., time management). The HLA/RTI uses a central runtime component (CRC) to this end. Because of network variability, the location of the CRC with respect to the federates may severely influence the federation execution performance. The paper proposes the optimal migration of a federation CRC to achieve better execution times. To achieve this goal, it applies an iterative search mechanism and proposes suitable cost functions to determine the optimal host. The process involves probing the network at selected nodes for quality-of-service metrics and deciding the search steps. Finally, a simulation study supports the investigation by quantifying the time required to complete a typical Monte Carlo experiment. Operational metrics of the search process are reported.

I. INTRODUCTION

The high level architecture (HLA) [14], [6] was developed by the U.S. Department of Defense (DoD) with the purpose of fostering interoperability and software reuse in the federation of computer simulators (federates). The IEEE 1516 is an improved HLA with support for standards, such as XML and Unicode [1]. A further revision of the IEEE 1516 standard is expected to be released (HLA-Evolved). These standards share the basic HLA functionality, which consists of Object Model Templates (to describe the objects that exist in a federation along with their attributes and interactions), HLA compliance rules, and the Runtime Infrastructure (RTI). The latter is the HLA concept that provides communication-related services to federations. These services fall into five main categories: federation management, declaration management, object management, ownership management and time management. As the name implies, federation management involves controlling the creation and destruction of federations. Declaration management allows federates to indicate to the RTI their desire to generate or receive object updates or interactions, so that it allows to implement a publish-subscribe system. The actual generation and reception of the information is implemented by the object management concept, which also provides the means to instantiate and delete objects. Ownership management allows federates to transfer the ownership of object attributes. Finally, time management coordinates the exchange of events in a federation. This collection of services that conform the RTI relays on a computer network for the transport of messages.

An RTI implementation typically has the following components: a RTI API, which provides the interface for federates to use RTI services, a RTI LRC (Local Run-time Component, also known as the Ambassador) that executes in the same host computer as the federate and provides local access to the RTI library and a RTI CRC (Central Run-time Component, or Gateway), which is a software daemon that interacts with the LRCs to coordinate the federation-wide operation. However, there is not an RTI reference implementation model so that different implementors may realize the internal RTI structures and functionality in different ways.

In a federation realization, the underlying network (e.g., the Internet) can strongly influence its execution performance. A poor performing network with high packet delays can severely extend the execution time of an event-driven federation or cause unwanted behaviors in some time-stepped simulations (e.g., real-time virtual environments). Because several RTI functions are supported by a constant message exchange among LRCs and between the CRC and LRCs, the relative location of these components could greatly improve or degrade a federation performance.

The use of migrating processes is a common topic in the area of mobile agents, which could help in improving a federation's performance. There exist several migration frameworks, some of which are able to transfer runtime state (strong or transparent migration) and some which are not (weak migration). Some examples of mobile agent frameworks are D'Agents [9], Aglets [16], and JAFMAS [4]. A comprehensive survey of mobile agents was done by Gray et al. [8] and Qu et al. [15].

Recent works address the use of process migration in federations but only in the context of federates. Federate migration was considered by Tan et al. [17], Cai et al. [3], and Boukerche et al. [2], who used federate migration as a load balancing mechanism to move federates from loaded to less loaded hosts. Li et al. [13] proposed a Service Oriented HLA RTI (SOHR) for federate migrations with a reduced control overhead on the migration process. An approach closely related to federate migration was studied by Eklöf et al. [5] who suggested a fault tolerant federation mechanism that uses check-points to restore a federation after a failure. The use of a grid as the execution and migration environment for a HLA-based federation was studied by Zajac et al. [18].

There are some limitations to federate migration. In some

situations, the location of the federates (and their LRCs) cannot be changed. In training simulations, some federates are human-controlled so that they cannot migrate. The situation also arises when federates require access to large databases, which are impractical to migrate or there are legal constraints on all or part of the software (e.g., determined by licenses of use).

However, even in situations where the federates cannot migrate, it should be possible to appropriately place the CRC, either dynamically (at runtime) or at the beginning of the execution, to take advantage of those nodes whose relative location to federates can offer a differential advantage to the federation execution. This host selection normally requires a number of nodes, other than the federate nodes, that would be willing to host the federation CRC.

Unlike previous works, which are focused on federate migration, this paper proposes the optimal migration of the central runtime component (CRC) of simulation federations with centralized RTI implementation. The optimal placement of the CRC could improve the response or running times of a simulation federation. To the best knowledge of the author, this problem has not been investigated by previous works.

II. SEARCH PROBLEM AND COST FUNCTIONS

Given a network represented by the graph $G = (V, E)$ where V is the set of vertices (nodes) and E the set of edges (links). Consider a number of federates running each on a node $n \in N$, such that $N \subseteq V$ and that use a central runtime component running on node $m \in V$.

Furthermore, assume that there exist a function $\Phi : \{N, m\} \rightarrow \mathfrak{R}$ (a cost function). The problem is to identify a node $m^* \in V$ such that $\Phi(N, m^*) \leq \Phi(N, m)$ for all $m \in V$ (i.e., the group communication cost is minimized).

Although G is in general a directed network, we can assume it is undirected graph because we would be mostly interested in optimizing message latency from and to the central runtime component to federates. Therefore, the main QoS metrics of interest would be the round-trip message latency or hop count, both of which can be measured from either side of a link. Other metrics may be side-dependent.

We propose two forms for the cost function Φ :

- 1) Φ as the average cost of the individual costs to each federate:

$$\Phi(s, N) = \frac{1}{N} \sum_{n \in N} c_n P_{m,n} \quad (1)$$

where $P_{m,n}$ is the measured metric of interest between nodes m and n . Constants c_i are introduced to weight the contribution of each node to the group communication. Equation 1 would make the search process to try to minimize the group communication “effort”. For example, if hop-count is used as metric, the search will try to find the host that minimizes the total number of packet forwardings.

- 2) Φ as the Euclidean metric:

$$\Phi(s, N) = \sqrt{\sum_{n \in N} c_n P_{m,n}^2} \quad (2)$$

Equation 2 will guide the search to try to return the centroid of the nodes involved, so that all communications would approximately be of the same cost, for example, with a similar number of hops or latency.

III. OPTIMAL LOCATION SEARCH

On very large networks, it may be impractical to carry out centralized optimization computations to determine the optimal host for the CRC federation because of the difficulties in collecting global network status (topology, traffic, etc.) We approach the problem with a local search metaheuristic and the use of a search agent. A search agent implements three basic functions (modules): probing, search and migration. An execution environment (daemon) runs at each participating node to let the search agents operate (Figure 1). The execution environment also allows the corresponding node to join the peer-to-peer overlay network, which consists of nodes that are able to host federates or RTI components. Participating nodes are organized on a peer-to-peer overlay for scalability purposes. This paper assumes that the overlay approximately follows the underlying network topology.

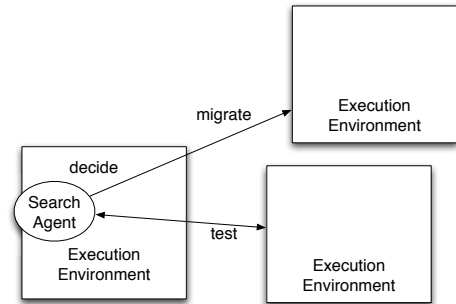


Fig. 1. Search agent

A. Network Probing Module

The purpose of the network probing module is to measure the metrics of interest from a given node to a set of destinations. These measurements can be later used to calculate the cost function from the given node to determine the next steps of the search process.

The basic element of the probing module is an end-to-end tester that has the ability of measuring message latency, jitter, loss, hop count, or effective bandwidth (throughput) between any two nodes. Typically, only one of these metrics is of major interest for a specific application (e.g., throughput for data distribution applications, latency for interactive applications, etc.) Therefore, only the metric of interest needs to be measured. To test a connection, the search module sends one or more messages to a pre-defined port(s) at the remote end, which is controlled by the execution environment. Standard tests can only be conducted (ICMP).

Message latency can be measured from its sending and receiving times. The sending time can be inserted by the origin into the message so that the destination can calculate message latency. However, both the origin and destination systems require to have a clock synchronization, which can be achieved for example by requiring both systems to run the Network Time Protocol (NTP). Similarly, round-trip latencies can be calculated with a timestamp at the origin, but not time synchronization is required. In most cases, it is possible to reuse the Ping servers that are available in standard TCP/IP implementations to measure round-trip message latencies (therefore, the module only needs to implement an ICMP echo client).

Similarly, hop count can be measured by implementing an ICMP echo client by increasing the time-to-live field in the packet to obtain ICMP time exceeded replies from the intermediate routes along the path (exactly in the same way the program *traceroute* works). When the underlying network is provided by a CPN, each origin node may obtain the hop-count information directly from the CPN [7].

End-to-end throughput and message loss can be measured by making the probing module to generate a pre-determine packet flow between the probing nodes and comparing it with the arriving packet flow.

Most network probes return average values (either absolute or exponential averages), so that more than one test message will be sent to check the network status. The exception is hop count for TCP/IP networks given that it rarely changes.

B. Search Module

The purpose of the Search Module is to select the most suitable host for the CRC. To this end, we have adopted two optimization metaheuristics: Hill Climbing and Simulated Annealing [10], both of which are simple to implement and have provided good results in our tests. The basic idea in both cases is to iteratively compare the cost function, as evaluated at the current hosting node, to a number of alternatives in the *search set*. The search set consists of some or all of the current node's neighbors (depending on the algorithm being used). If a better cost can be found in one of the neighbors, the algorithm moves the search agent to that node and repeats the process iteratively until no further improvement can be found (which gives the solution to the search).

In the basic Hill Climbing, the search set consists of all node's neighbors, so that the neighbor's providing the lowest cost is selected and the search always move to the local best next step. The process works well for convex cost functions, but it may get stuck in a local minima for more generic cost functions. Moreover, it may produce a large overhead (messages) in networks with nodes with high degree (20 or more neighbors). However, the problem can be reduced by a introducing stochastic behavior in the search. With Random Hill Climbing, the search set consists of one neighbor's node randomly selected, so that the search moves at each step to a better cost but not necessarily the best local cost. Another solution to the local-minima problem is given by Simulated Annealing (SA), which allows the search to move to a worse

cost state with a given probability, which is a function of the two cost difference (δ) and the elapsed search time (t_s): $\exp\{-\delta/T_e\}$. T_e is the temperature parameter that gradually decreases in proportion to t_s .

The discrete nature and structure of the search problem allows to apply these search metaheuristics almost without modification to solve our problem. However, to help in reducing same-host probes, search agents carry the list of already visited nodes, which is used to exclude them for further move decisions (similar to Tabu search). Furthermore, all test results are stored in the nodes up to a "time-to-live" period, so that they can be reused by later searches.

C. Migration Module

Agents operate within the execution environment, which provides access to the communication ports of the hosting node. As a result of a search decision, a search agent may migrate to a different host to continue the probe-and-move process. At the end of the process, the search agent returns to the origin node and informs the federate manager of the best location that was found and its associated cost. The federation manager then may evaluate a CRC migration based on the cost advantage offered by the new host.

IV. FEDERATION EXECUTION MODEL

The following model attempts to capture the core functionality of a HLA/RTI federation from the point of view of the main network flows that exist during its execution. It is assumed the use of a conservative time management mechanism. The model consist of two main components: a central runtime component and a federate (which includes the RTI local runtime component).

Consider a simulation federation (federates $n \in N$). Because we are interested mainly on the traffic generated during a federation execution, we exclude the setup steps from the model (i.e., federation and declaration management). Therefore, federates start with prior knowledge of the addresses of all other federates and the central element. The federation proceeds by requiring all federates to send a *time advance request* message (TAR) to the CRC. Once the CRC receives a TAR from each federate, it will reply to the one who sent the TAR with a timestamp nearer but greater than the current time with a *time advance grant* (TAG) message. This process is represented in the model by assigning to each federate a selection probability p_n , ($n \in N$, $\sum_{n \in N} p_n = 1$) according to the discrete probability distribution $P(X)$. Function $P(X)$ can be tailored to match different federation situations, from balanced simulations (with a uniform distribution) to unbalanced cases where one or more federates execute at a higher rate and therefore, generate more traffic.

The federate receiving the TAG message from the CRC executes for a random time. During the execution time, the federate generates d messages (object updates and interactions) towards other federates. d is an exponential random variable of parameter D , which is fixed for all the federation to simplify. The d messages are sent at a rate r , which is also

TABLE I
SIMULATION PARAMETERS

Parameter	value	comments
p_n	1/L	federate selection probability
q_n	0.5	data routing probability
d	5 msg	average number of messages to send
r	100.0 msg/sec	message sending rate
F	1000 steps	steps to complete before stop
TAR/TAG size	16 bytes	RTI message size

an exponential random variable with parameter R . Both d and r model the traffic generated at one federated and directed to other nodes. On the other hand, each of such messages is sent to federate n with probability q_n (routing probability), which models the data distribution function. Once the processing time is over, the federates issues a new TAR message to the CRC. The process continues iteratively until the stop condition is met. We assume that the stop condition is reached once the CRC grant F time advances.

V. EVALUATION

We implemented the federation model just described as a simulation module for INES [12] (so, we will be simulating a simulation federation). Our main interest is in evaluating the expected running time of a given simulation experiment (event-driven based) and compare the resulting time length with the one obtained from an optimized system. As a benchmark experiment, consider a Monte Carlo simulation that needs to calculate 10 averages of some metric of interest (e.g., to produce a plot of these values). Each average can be obtained by repeating the same simulation 100 times, therefore, 1000 federation runs are needed to complete the experiment.

The simulated federation runs on a grid network topology of 25×25 nodes, where any of these 625 nodes is able to host a federate or central element. The nodes are connected by links of equal characteristics with a transmission rate of 256 Kbps and a propagation delay of 1 ms. The overlay network is formed by a one-to-one mapping of nodes and links. At the beginning of each simulation run, L nodes (federation size) are randomly selected from any of the grid edges (consisting of 96 nodes) to host each one of the federates. The first node selected is also assigned to run the CRC of the federation, which is kept in the unoptimized system. Federates start with prior knowledge of the addresses of all other federates and the central element. The federation executes as described in the previous section until the stop condition is met. The parameters used in the simulations are indicated in Table I.

The simulator models all layers of the protocol stack, so the actual TAR or TAG messages on the link are the values indicated on the table plus lower layers overhead (i.e, UDP, IP headers).

On the optimized case, a search process is executed immediately prior to the start of the federation and the CRC is migrated and started at the resulting location. Given that the nodes have a low degree, we have opted for Hill Climbing for

this evaluation study. The metric used to evaluate the cost functions was hop count, which should provide reasonable results for latency as well given the structure of the test network. Figure 2 depicts the average running times for the Monte Carlo experiment for a range of federate number selections. The curves show the points that represent the average of about 3000 executions each. When indicated, the curves show the 95% confidence interval. The results show that the system with optimized CRC migration was able to finish all the simulation tasks in 10 to 30 % faster on average than the original system. The proposed Euclidean metric (Equation 2) performed slightly better than the average cost metric (Equation 1).

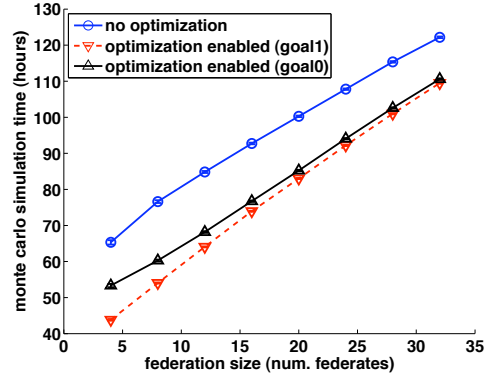


Fig. 2. Expected running time of Monte Carlo experiment

It is interesting to observe that the improvement factor is proportionally smaller when the federation size increases. This is explained by the fact that the results show the average of the combinatorial placement of federates and central element on a limited set of nodes. With a higher number of nodes there is a better probability to choose by chance better nodes for the CRC for the federation than with fewer nodes.

The average search time is depicted in Figure 3, which was under 10 s in all cases. It can be noticed that the search time decelerate with larger federations, which occurs because on the selected topology, with larger federations there is less need to optimize the location of the CRC. This reasoning is supported by figures 4 and 6, which show that the number of nodes involved in the search tends to saturate with larger federations as well as the ratio of the optimized system to unoptimized system costs (Φ).

VI. CONCLUSIONS

The paper has investigated the optimal CRC federation migration for improved execution performance in HLA/RTI federations. To this end, the paper proposed the application of local search metaheuristics on the federation network, along with cost functions and network probes to assess the suitability of network hosts for the task and guide the search process. The search process is iterative and scales well to large networks. It consists in exploring the network status with a search agent that moves towards the optimal location from a quality-of-service point of view for the group communication CRC to

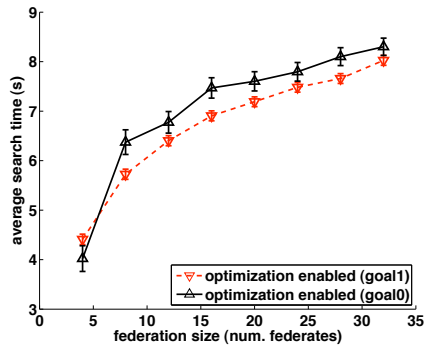


Fig. 3. Average search time vs. federation size

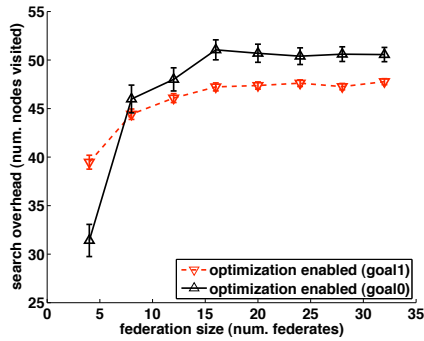


Fig. 4. Average search overhead (number of nodes probed by the search agent) vs. federation size

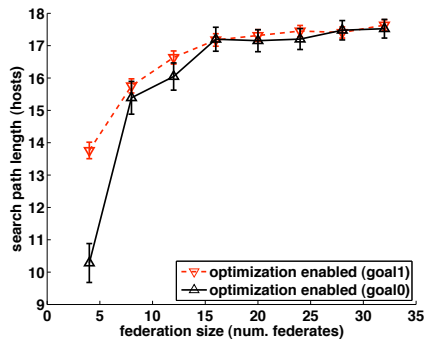


Fig. 5. Average search path length (number of nodes visited by the search agent) vs. federation size

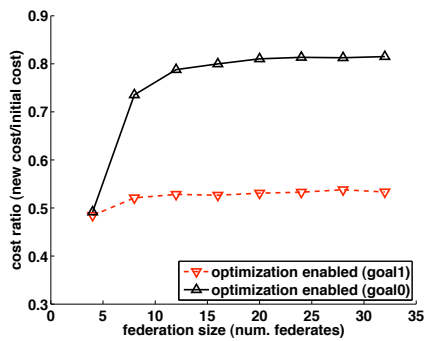


Fig. 6. Cost ratio (new to initial cost) vs. federation size

LRC (federates). The proposal was evaluated on a simulated setting with federations of various sizes. The results confirmed that the optimal migration of the simulation federation CRC can bring substantial reductions in the execution time of analytic simulations (in the range of 10% to 30% in the selected test case). Although the paper only focused on the use of local search on overlays that follow the structure of the underlying network, the author's on-going work tackles the problem of using more general overlays as well as the evaluation of the proposal on a network testbed with a real federation and a HLA/RTI implementation.

REFERENCES

- [1] IEEE 1516.3-2003 - recommended practice for high level architecture federation development and execution process (fedep).
- [2] BOUKERCHE, A., AND GRANDE, R. E. D. Optimized federate migration for large-scale hla-based simulations. In *DS-RT '08: Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 227–235.
- [3] CAI, W., YUAN, Z., LOW, M. Y. H., AND TURNER, S. J. Federate migration in hla-based simulation. *Future Gener. Comput. Syst.* 21, 1 (2005), 87–95.
- [4] CHAUHAN, D., AND BAKER, A. D. Jafmas: a multiagent application development system. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents* (New York, NY, USA, 1998), ACM, pp. 100–107.
- [5] EKLÖF, M., MORADI, F., AND AYANI, R. A framework for fault-tolerance in hla-based distributed simulations. In *WSC '05: Proceedings of the 37th conference on Winter simulation* (2005), Winter Simulation Conference, pp. 1182–1189.
- [6] FUJIMOTO, R. M. Parallel simulation: distributed simulation systems. In *WSC '03: Proceedings of the 35th conference on Winter simulation* (2003), Winter Simulation Conference, pp. 124–134.
- [7] GELENBE, E., LENT, R., AND XU, Z. Measurement and performance of cognitive packet networks. *J. Computer Networks* 37 (2001), 691–701.
- [8] GRAY, R., KOTZ, D., CYBENKO, G., AND RUS, D. Mobile agents: Motivations and state-of-the-art systems. Tech. rep., Handbook of Agent, 2000.
- [9] GRAY, R. S., CYBENKO, G., KOTZ, D., PETERSON, R. A., AND RUS, D. D'Agents: Applications and performance of a mobile-agent system. *Software—Practice and Experience* 32, 6 (May 2002), 543–573.
- [10] KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by simulated annealing. *Science* 220 (1983), 671–680.
- [11] LENT, R. Improving the location of mobile agents on self-aware networks. In *ICEIS (4)* (2008), pp. 239–242.
- [12] LENT, R. Ines: Network simulations on virtual environments. In *Proc. of 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems* (2008).
- [13] LI, Z., CAI, W., TURNER, S. J., AND PAN, K. Federate migration in a service oriented hla rti. In *DS-RT '07: Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 113–121.
- [14] MODELING, D., AND DMSO, S. O. High level architecture run-time infrastructure rti, 1.3ng programmer's guide, version 5, 1999.
- [15] QU, W., SHEN, H., AND DEFAGO, X. A survey of mobile agent-based fault-tolerant technology. *Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005. Sixth International Conference on* (Dec. 2005), 446–450.
- [16] TAI, H., AND KOSAKA, K. The aglets project. *Commun. ACM* 42, 3 (1999), 100–101.
- [17] TAN, G., PERSSON, A., AND AYANI, R. Hla federate migration. In *ANSS '05: Proceedings of the 38th annual Symposium on Simulation* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 243–250.
- [18] ZAJAC, K., BUBAK, M., MALAWSKI, M., AND SLOOT, P. Towards a grid management system for hla-based interactive simulations. In *DS-RT '03: Proceedings of the Seventh IEEE International Symposium on Distributed Simulation and Real-Time Applications* (Washington, DC, USA, 2003), IEEE Computer Society, p. 4.