GEMOM Middleware Self-healing and Fault-tolerance: a Highway Tolling Case Study

Federica Paganelli, Gianluca Vannuccini, David Parlanti,
National Interuniversity Consortium for Telecommunications
Firenze, Italy
Federica.paganelli@unifi.it

Dino Giuli[1], Paolo Cianchi[2]
[1]Dept. of Electronics and Telecommunications
Via S. Marta 3, Firenze, Italy
dino.giuli@unifi.it
[2] Negentis srl, Firenze, Italy
pcianchi@negentis.com

***Abstract*—Application of message-oriented communication in business critical systems has to cope with requirements for end-to-end intelligence, security, scalability, self-adaptation and fault-tolerance. To this extent, the Genetic Message-Oriented Middleware (GEMOM) European Research Project focused on the design and development of a fast-forwarding message oriented middleware, endowed with robustness, resilience, self-adaptability, and scalability capabilities. This paper reports on the design, development and testing results of a case study for the GEMOM middleware on highway toll data management and collection. The case study has a twofold objective: first, it offers a reference scenario that poses requirements challenging a specific set of self-healing and fault-tolerance GEMOM features and thus providing an application scenario suitable for features validation; second, it aims at representing a real-world application scenario and consequently at providing valuable insights on GEMOM exploitability in a specific market sector.**

Keywords-message-oriented middleware; self-healing; fault tolerance; toll data management.

I. INTRODUCTION

Message-Oriented Middleware (MOM) systems are considered as promising assets for supporting current challenges in the enterprise computing landscape [1]. These challenges are: the need for increasing support of sense-and-respond applications (i.e., applications endowed with massive sensing, analytics and control capabilities); the growing interconnection of enterprise systems over geographically distributed wide areas; the need to differentiate message traffic according to QoS-aware policies. Such challenges stress requirements for end-to-end intelligence, security, scalability, self-adaptation and fault-tolerance.

One of the most widely adopted approaches to support scalability and resilience in messaging infrastructures is based on hot standby brokers with instant switch over and no data loss. However, once switch-over is performed, usually these systems have no means to compensate for the reliability loss by automatically finding another source of redundancy. Also, they are relatively prone to the incidence of feed failures as they often do not take redundant feeds into account. It is often said that the existing state-of-the-art achieves arbitrary resilience by a brute-force approach. The state of the art is often outside of the reach of Small Medium Enterprises) (SMEs) and even of large companies. Moreover, self-healing is either rudimentary or non-existent, and when it is available, it requires high-level skills to be configured and managed [2].

The European Project for a Genetic Message Oriented Middleware (GEMOM [3]) was aimed at addressing the above issues, by researching, developing and deploying a prototype of a messaging platform endowed with robustness, resilience, self-adaptability and scalability capabilities.

According to their experience in messaging-systems and business areas of interest, the GEMOM partners were involved in the development of five case studies, with a twofold objective: first, each case study offers a reference scenario that poses requirements challenging a specific set of GEMOM features and thus providing an application scenario suitable for GEMOM key features validation; second, each case study represents a real-world application scenario and consequently provides valuable insights on GEMOM exploitability across a wide set of market sectors.

This paper reports on preliminary results in the design, development and testing of a GEMOM case study on a highway toll data management and collection scenario. The proposed case study aims at validating GEMOM capability in guaranteeing reliable message exchange across highway infrastructure nodes against different fault simulation scenarios.

The paper is structured as follows: Section II outlines the GEMOM middleware requirements definition, the corresponding GEMOM key features and system architecture. Section III describes the GEMOM experimentation in a toll collection management case study. Finally, Section IV sums up conclusions and future research directions.

II. THE GEMOM MIDDLEWARE

This section briefly introduces the GEMOM middleware by first presenting the adopted risk analysis methodology and design requirements, and then by describing main characteristics of the GEMOM architecture.

A. Risk Analysis and Requirement Definition

Risk analysis for the GEMOM infrastructure was derived by taking into account the assets of a MOM, the threats that may hang over such assets, the vulnerabilities that may be

 ISBN: 978-1-61208-166-3

exploited by the attacker and, finally, the impact of a specific attack on each asset.

The main assets under consideration were: end user (using an application that exploits the middleware), agent (such as applications, probes, effectors), agents acting as message publisher and/or subscriber, message sender and receiver, brokers, links (primary and backup), paths composed of multiple links and, finally, messages and message topics defined in the MOM. Such assets were assigned a risk level depending on the specific case study under consideration within the GEMOM project.

Other assets were considered as extremely relevant and highly-risky for the GEMOM system, being them the management layer (hereafter named "Managerial Nodes") of the overall GEMOM infrastructure.

Afterwards, the threats that could affect those assets were assessed. The value of a threat was estimated by considering how often an attacker could perform its attack, or how easily it could access the asset.

Examples of threats that were considered for the GEMOM project are message flooding or publishing from non-existent or un-authorised brokers, agent nodes registration/deregistration via spoofing, and replay attacks from malicious nodes. Also threats related to confidentiality and integrity corruption in the messaging path were considered. Examples of specific threats, that might be more relevant for the highway tolling messaging system, could be toll-gate power-supply interruptions (due, for instance, to flooding or other natural phenomena), as they are likely to affect the capability to exchange messages with the central toll collection station. Other threats, even if less likely to happen, being the network a totally dedicated infrastructure, could be related to tolling message tampering and sniffing.

Vulnerability analysis in the GEMOM infrastructure was conceived as a continuous run-time assessment process, addressed with a specific tool that can be activated in the GEMOM messaging layer. Vulnerability detection and the consequent adaptive security policies are out of the scope of this paper (details may be found in [2]). Nevertheless, common vulnerabilities derived from the OWASP [4] and SANS [5] top lists were considered as a first step.

Once the main assets, threats and vulnerabilities were considered for the GEMOM infrastructure, a further step was done in order to extend the concept of security risk, including also performance and QoS degradation that, as much as a security attack, affect the overall system performance and, as a consequence, the final quality of the delivered service. This wide-sense approach is conceptualised in GEMOM in the extended notion of "fault".

A fault may be seen as a status-change of a GEMOM asset between two different security risk-levels, and/or between two different SLAs. For instance, if a message, belonging to a guaranteed class-of-service, encounters a path with compromised QoS capabilities, which will only offer unreliable class-of-service, a corresponding fault may be triggered.

The capability of the GEMOM system to react to (and to prevent) these faults is referred to as Fault Tolerance.

Fault tolerance requirements specify the prevention actions that GEMOM should perform in order to avoid the fault, as well as the actions that the GEMOM infrastructure should launch in order to mitigate the impact of the occurred fault, and to establish a new reliable and performing steady state.

The GEMOM requirements gathering process was driven also by case study analysis. Table I shows a resume of the requirements that were selected for the Highway Tolling Data Collection and Management case study with the help of the highway operator representatives and their corresponding priority level for validation.

B. Gemom Key Features

According to the above-mentioned risk analysis and requirements definition, the GEMOM infrastructure was

TABLE I. GEMOM REQUIREMENTS FOR THE TARGET CASE STUDY

Requirement	Detailed description	Priority
1. Tolerance to Connectivity failures	GEMOM shall use traffic engineering techniques at networking layer to be tolerant to links failures. In case of detection of compromised connectivity to consumers, GEMOM routing algorithm shall select another alternate path (or more, for redundancy and load sharing) to message consumers.	LOW
2. Tolerance to hardware/software faults in nodes	GEMOM shall keep an updated topology database of the network of brokers, in order to be tolerant to failures in one specific node and to be able to fast-switch to other nodes in case of failure.	HIGH
3. Self-Healing	The system should be able to automatically create new redundancy in case of node faults. If one broker or namespace fails and redundant one takes over the function, system's resilience capabilities are diminished. GEMOM should be capable of restoring its resilience and security profiles if resources are available.	HIGH
4. No single-point of failure	The communication highway shall not introduce a single point of failure in node to node communication	LOW
5. Sudden reconfiguration	The system should allow for sudden re-configurations of the available resources (such as the allocation of more messaging paths to deal with peak traffic rates and resources required under emergency situations)	LOW
6. Self-protection	GEMOM shall implement load balancing, topic mirroring, and shall be able to implement switchover to redundant components, and to spawn new hot standby components	LOW
7. JMS API support	GEMOM shall offer messaging services to JMS-based client application via proper bridging components	HIGH
8. Plug&play rule assisted semantics	"Plug-and-play rule assisted semantics" refer to the system capability of altering message delivery according to application-specific needs. GEMOM shall allow to attach plug and play rules to the exchange of various individual topics or groups, and so enhance its handling of exchange of messages with content transformation rules, routing or security semantics.	HIGH
9. Multiple bindings support	GEMOM shall prove bindings for Java	HIGH

 ISBN: 978-1-61208-166-3

designed to implement a fast-forwarding message-oriented middleware endowed with end-to-end resilience, security, scalability and self-adaptation capabilities.

GEMOM key features, and related research challenges, may be listed as follows [2]: a) system scalability in handling variable messaging volumes and clients cardinality; b) context-aware adaptive security via policy-based authorization, authentication and confidentiality techniques; c) new techniques and tools for pre-emptive and automated checking vulnerabilities to faults, oversights and attacks; d) message delivery reliability to message broker mirroring and workload distribution techniques; e) extensibility for accommodating application-specific requirements (e.g., content-based message filtering, JMS API support, message traceability).

Features d) and e) are particularly relevant to this paper, as discussed in the following section. For an extensive description of the other features, and the discussion of GEMOM contribution with respect to the state of the art, for the sake of brevity, we refer the reader to [2].

Above mentioned GEMOM key features are supported by the following specific research contributions:

- the architecture of an externalized system to support resilience and anomaly detection for MOM resilience and protection [2] [6].
- The design and implementation of a resource allocation mechanism for balancing brokers' workload [6].
- The integration of a mechanism for anomaly detection. Examples of target anomalies are high message rates, degradation of broker performance in the context of Denial of Service (DoS) and anomalous message content [7].
- Design of adaptive security mechanisms and security metrics for a distributed messaging system based on threat and vulnerability analysis and security requirements [8].

C. GEMOM Architecture

The GEMOM system architecture was modelled as a set of communicating nodes, distinguished into operational and managerial nodes.

The *Operational nodes* are those responsible for executing basic operational tasks according to a specific behaviour, and message exchange. Examples include Message Brokers and Clients (either message publisher or subscribers) and modules providing security and fault detection capabilities, i.e., the Authentication and Authorization Modules.

Managerial nodes are modules that, based on the system context awareness, take decisions about possible run-time adjustments of Operational nodes behaviour. Examples include modules responsible for elaborating adjustments for the broker topology and workload (Overlay and Resilience Managers) and modules responsible for adapting security policies (Adaptive Security Managers).

Therefore, GEMOM infrastructure can be devised as a network of GEMOM brokers (Gbroker) configured, protected, monitored and optimised by an overlay of Managerial nodes, as sketched in Fig.1.

A GEMOM Broker is designed in order to keep the message routing process as simple and fast as possible. To this extent, topic names follow schemes similar to those used in variables or class definitions in programming languages, while topic values are simply key-value pairs. Message brokers and API then add metadata to the stream of routed topics.

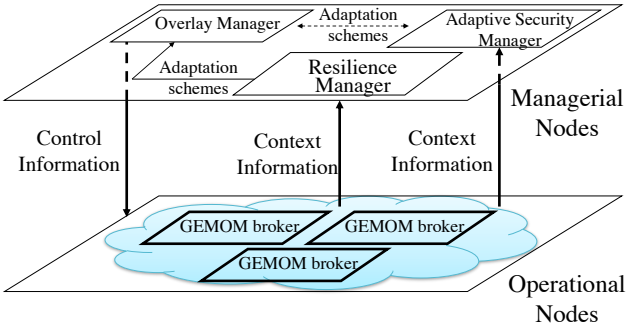

Figure 1. GEMOM Architecture

In addition to this simple messaging layer, the Overlay Manager is responsible for a range of functions to improve performance and resilience. It is external to the message forwarding system and receives data pertaining to security and QoS from a range of sensors that monitor the core messaging system. It then evaluates such data and performs the consequent actions using effectors deployed within the Operational Nodes, and the contextual information gathered by multiple nodes both at the Managerial and Operational Layers (e.g., Adaptive Security Manager, Vulnerability assessment tools, Monitoring Tools, Gbrokers collecting internal data, etc.).

In other words, the above actions are triggered by «fault» events that are detected by active and passive monitoring the QoS and Security parameters, according to specific SLAs. When a violation in the committed service guarantees occurs, GEMOM must react by executing a suitable series of actions.

Examples of actions suggested by the Overlay Manager to the Operational Nodes layer, that are also specifically relevant to the requirements described in the previous section for the Toll Collection scenario, are: rebalancing existing load, adding new GBrokers to the system and re-routing the traffic on some namespaces or individual topics. These actions are the basic mechanisms for realizing Gemom Broker Mirroring and Self-Healing capabilities.

Operationally, if there is a severe failure in the primary Gbroker, then message handling is passed to the mirror, which is re-labelled as primary (broker mirroring), and a new mirror for the primary found. This mechanism allows to automatically re-establish the required resource redundancy also after a fault occurrence (self-healing). The same happens if the failure is related to a link between two Gbrokers, or during a path between publishers and subscribers. Note that a failure could also concern the chosen QoS SLA profile. The following figure shows how the

 ISBN: 978-1-61208-166-3

GEMOM system reacts to a fault through broker switchover and how the self-healing capability is achieved by spawning a new broker acting as a mirror (Fig.2).

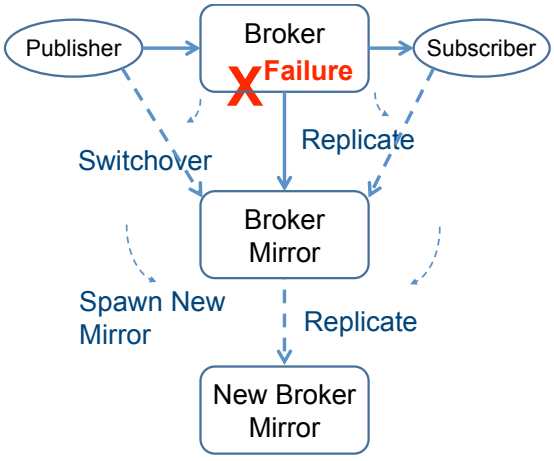

Figure 2. Broker Mirroring and Self ealing through new broker spawning acting as mirror

III. GEMOM EXPERIMENTATION IN A TOLL COLLECTION AND MANAGEMENT SCENARIO

The GEMOM middleware was conceived, developed, deployed and tested within the project lifetime by GEMOM partners.

The research challenges addressed and documented within the GEMOM project were experimented by carrying out suitable testcases in different real-life scenarios, each related to the major expertise of the corresponding GEMOM partner.

In particular, the case study reported in this paper had the twofold objective of: a) evaluating how GEMOM message-oriented infrastructure could be conveniently applied to cope with information distribution needs of highway operators b) validating a subset of GEMOM features, which were chosen according to the case study application requirements. The case study was designed with the collaboration of an Italian highway operator. More specifically, requirements were collected through face-to-face unstructured interviews with the operator representatives.

From these interviews it emerged that Toll collection management is an application scenario that is strategically relevant to the highway operator's purposes as well as potentially challenging for GEMOM validation. As a matter of fact, as already argued by Clark et al. [4], a wide-scale tolling system should cope with several requirements, including system reliability and availability, which are strongly required as money is involved.

Toll collection and management deals with the tools, techniques and processes involved in collecting revenue from a vehicle user for the use of road-space through road-use pricing [9].

Toll messages represent a significant volume of data exchanged within the target highway operator network as well as with external information systems of neighbour highway operators.

These issues motivate the need for a uniform, reliable, self-optimising, well-structured, extensible architecture for application-level communication and integration. Moreover, a uniform approach for data exchange based on message-oriented paradigm may facilitate the adoption of efficient and cost-effective system maintenance strategies.

A basic representation of a toll data collection system includes the following entities:

Highway Toll Central System. This system collects toll data from the infrastructure and performs toll data archiving, validation and processing for end users' accounting and monetary compensation with external operators.

Station Systems. Highway Stations may group lanes of both types: manual (i.e., with on-site payment) and electronic lanes.

Electronic Lanes. Electronic Lanes are equipped with RFID readers and sensing devices. This infrastructure is used to detect the transit of a vehicle equipped with an RFID transponder. The transit event (both in entrance and in exit) triggers the generation of a message (Electronic Toll message) which is sent to the Highway Toll Central System.

Manual Lanes. Entrance manual lanes provide drivers with a paper-based token registering the vehicle transit details. At destination, the driver shows the token at the exit lane. The lane system calculates the road fare, depending on the adopted pricing models, and the driver pays on-site. For each entrance and exit event, a message is created by the lane system and sent to the Highway toll central system. Updates on tolling policies are notified to Lanes via messages delivered by the Highway Toll Central System.

External toll systems. A target Highway Toll Central System should interact also with Toll Systems of external operators (e.g., for monetary compensation). Exchanged data include aggregated electronic toll messages and tolling policies update.

The case study scenario focuses on the distribution of two message types:

a) automatic toll payment data, which are data collected at toll lanes and transmitted periodically to the central control room for performing billing operations. Message size is limited. Data loss is not tolerated, while timing constraints are not hard real-time (Many-to-One message delivery).

b) tolling policy update records: update of tolling policy is performed once in a while and have to be communicated to all the peripheral nodes (i.e., toll lanes and stations) within a limited time interval. The system does not tolerate data loss. As regards timing constraints, the system is not specifically sensitive to single message delays. (One-to-Many message delivery)

 ISBN: 978-1-61208-166-3

These data represent a strategic and relevant information

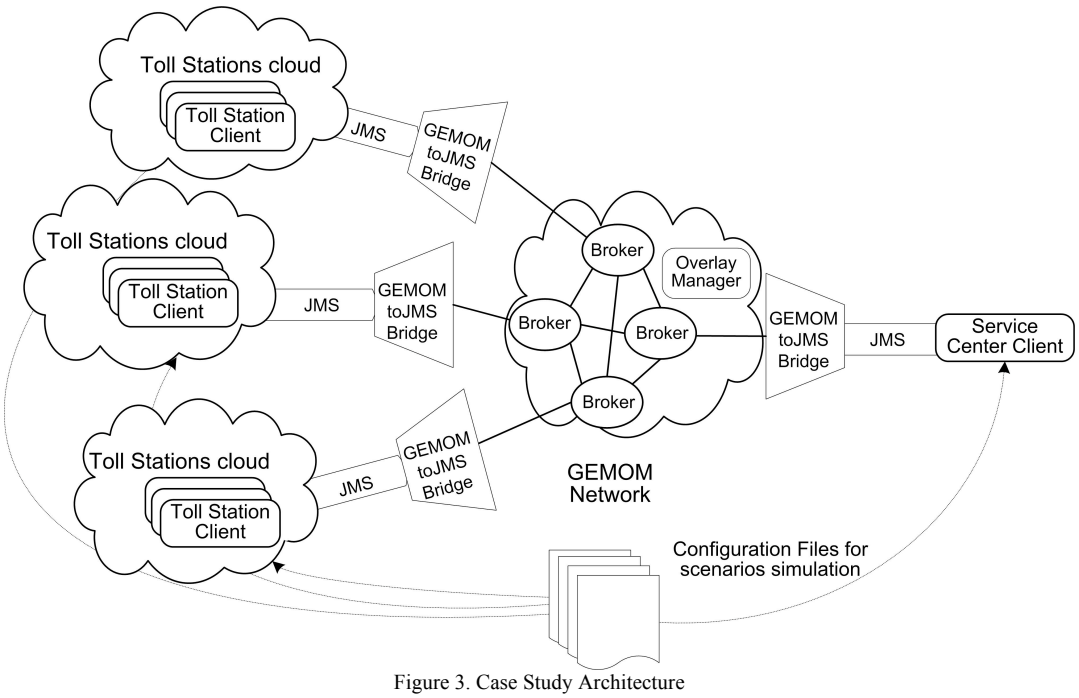

Figure 3. Case Study Architecture

asset for a highway operator and no information loss is tolerated.

This case study is thus particularly significant for testing GEMOM messaging service's continuous availability and robustness achieved via mirroring and self-healing features. As a matter of fact, GEMOM structural replication capabilities (self-healing) should assure robustness of the messaging infrastructure even under high volumes of traffic.

Moreover, in order to enable the interoperation of the GEMOM capabilities in the target operator technological environment, where Java-based standard and enterprise technologies are widely adopted, the case study exploits also the developed full-fledged java bindings to GEMOM C++ native interfaces. In order to facilitate interoperation with widely-diffused commercial messaging platforms, the case study architecture is based on the adoption of a component providing GEMOM-to-JMS bridging capabilities, as Java Message Service (JMS) [10] is a wide adopted specification for messaging services API.

A. Case Study Architecture

The case study architecture is composed of the following functional components:

- Application clients that publish/subscribe for toll and tolling policies data. Clients have been developed against JMS messaging interfaces.
- a JMS-GEMOM bridge, interfacing a JMS bus with GEMOM. The JMS-GEMOM bridge is responsible for transmission/receipt of messages over GEMOM. Bridging has been realized by mapping JMS topics onto GEMOM ones.
- A network of GEMOM brokers responsible for message exchange.

Applications clients are configured in order to simulate the behaviour of toll stations and the Service Centre. Toll station clients are spread on a set of virtual machines to resemble the highway operator physical wide area network. They may be configured in order to act as message producers (to simulate the delivery of electronic toll message) and as message consumers (to simulate the reception of tolling policy updates). Analogously, the Service Centre has been modelled as a JMS client capable of listening for toll data coming as JMS messages transferred by GEMOM and sending tolling policy updates.

Toll station clients simulate the production of Electronic Lane messages over a target time period and deliver the produced messages to the messaging system. Each toll station client may be configured in order to simulate different message traffic scenarios, resembling real-life message passing statistics during ordinary days. Toll gate working time is divided into time intervals whose starting time and duration can be configured; toll gate data are generated for each time interval according to Poisson distributions with different average values in order to simulate traffic flow at different hours over a day. It is possible to configure on each host the number of gates that have to be simulated and the desired message distribution over a target time interval.

The Service Centre simulates the generation of tolling policy updates. According to real practices, this event may be modelled as a one-shot event. Analogously to Toll station clients, the message generation process may be defined in a configuration file.

Figure 3 shows the proposed case study configuration for simulating the behaviour of the Highway Infrastructure toll stations network.

 ISBN: 978-1-61208-166-3

The toll station clients are grouped in set of toll stations clouds. Each toll station cloud represents the traffic to/from toll stations covering a specific geographical area. According to the characteristics of the target highway operator network, the case study will include at least three toll station clouds, one for each of devised geographical areas (north-, central, south Italy). To each area we can assign a given number of toll gates and/or toll stations, in order to reach an order of magnitude comparable to that of the real highway operator network.

As depicted in the figure, the application clients deliver and consume messages to/from a JMS MOM (i.e., Apache ActiveMQ). Messages are transferred to GEMOM network via the GEMOMtoJMS Bridge.

The GEMOM Broker Network is composed of a variable set of GEMOM Broker Agents and an Overlay Manager component is responsible for the overall network management and adaptation, as described in Section II.C.

B. Testing activities and results

The scenario analysis was carried out in collaboration with the Highway Infrastructure representatives. Details provided by the Highway Infrastructure on the current approach for message transfer handling and on typical message volumes have driven the design of the case study architecture and the configuration of the overall system for the demonstration activities.

Given the above-mentioned flexible configuration capabilities of the implemented case study, we were able to simulate different traffic scenarios by varying message size and number of toll gates involved, according to statistics data and requirements gathered during the meetings with the Highway Infrastructure representatives.

According to the risk and requirement analysis derived in Section III and to the architecture specification described above, the case study had the objective of functionally validating the following GEMOM capabilities:

1. offering a reliable messaging service via broker mirroring techniques (see req. 2 in Table I).
2. readjusting the structure of running nodes in order create new redundancy in response to failure-type events (req. 3 in Table I), as depicted in Fig. 2.
3. allowing clients to subscribe to topics and specify transformation rules (e.g., encoded in an XSLT file) in order to receive filtered/aggregated data (see req. 8 in Table I).
4. offering messaging services to JMS-compliant Java-based clients via proper bridging components (req. 7 and 8 in Table I).

For each toll station cloud, 150 toll lane clients were instantiated. We deployed a network of three brokers, as it is the minimum number of broker required to support GEMOM mirroring and self-healing features. Each machine was deployed on a separate host. All components were on the same LAN network.

We defined a set of test cases in order to test the system in different working conditions. Test cases are defined by varying the toll station clients configuration in order to resemble real-life road traffic scenarios.

A low-traffic scenario models the nightly traffic (with an average of 100 message per hour produced by single lane clients).

A medium-traffic scenario models the average traffic on an ordinary working day (four millions of messages per day).

Finally, a third scenario is defined in order to stress the system in a heavy traffic scenarios (even if unlikely to occur in real-life scenarios) characterized by an average of 1000 messages per hour produced by each lane client.Moreover, a set of messages related to price list updates were sent once for each test case in the opposite direction (from the Service Center to lane clients).

For each target GEMOM features (see list above), we ran each test case ten times. Table II summarizes the outcomes of the functional tests that were carried out in the testbed, with the corresponding most relevant issues and comments, representing the lessons learned from the experimental validation, and, hence, a sort of todolist for the next steps of the research activity.

Vertical and horizontal scalability were systematically tested in other GEMOM case studies [11].

For what concerns the testcase presented in this paper, the overall percentage of correctly received messages was 99,5%, while GEMOM highest measured throughput was 5000 msg/sec.

IV. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This paper reported on the design, development and testing results of a case study aiming at validating a set of GEMOM middleware features in a highway toll data management and collection scenario. In order to cope with the application scenario requirements, this work was mainly focused on the experimentation of mirroring and self-healing capabilities of the GEMOM system. We also tested interoperability with JMS API and the capability of configuring content transformation rules.

With respect to the related work on the GEMOM project, the remaining set of GEMOM features (e.g., adaptive security and authorization), were specifically addressed within the project lifetime in other case studies [11][12].

With respect to the related work in evaluation frameworks for MOM dependability and QoS [1] [7][13][14], this paper was based on requirements gathered from industry experts of highway infrastructures, where secure and reliable MOMs can be effectively applied, and it aimed at validating such requirements by means of experimental tests. However, given the mission-critical profile of the considered Highway operator infrastructure, it was not feasible – during the project lifetime - to validate the GEMOM middleware directly into the real operating messaging network. Further investigations could be focused on the deployment of GEMOM modules (especially those related to reliability and self-healing) into subsets of the real Highway Infrastructure and on testing and validation activities in more complex scenarios.

 ISBN: 978-1-61208-166-3

TABLE II. CASE STUDY TESTING RESULTS

GEMOM Feature	Test Description	Issues/Comments
Broker mirroring	At least two GEMOM brokers are running. Broker A is a master broker and Broker B is a mirror broker for a group of topics. After a blocking fault in Broker A was caused, we observed that messages have continued to flow from publishers to subscribers with no data loss, while the OverlayManager has correctly reported the re-instantiation of the Broker A	We simulated faults in a master broker by killing the corresponding process. Future tests could include the simulation of different faults (e.g., Distributed DoS, faults related to performance degradation).
Self Healing through broker spawning	The objective of the trial consisted in verifying that in case of failure of a master broker, the mirror broker will act as a master broker and a new mirror broker is spawned. We checked that messages continued to flow from publishers to subscribers.	This test was performed with a GEMOM network made by up to four brokers. Future test could be performed by increasing the GEMOM broker network size.
Plug-and-play rule assisted semantics	Toll station clients subscribe to the Price Listing topic and specify an XSLT transformation script file in order to receive transformed data. We checked that transformed messages were correctly received (via XML Schema validation).	Future tests could simulate the exchange of price listings with external operators' systems.
JMS API Support and Java bindings	The trialling activities verified that the message traffic is correctly handled by a system deployment made of the Java client applications compliant with the JMS API, the GemomToJMS bridging component and the GEMOM broker network (Fig. 4)	At present the GemomToJMS bridging component has been tested with the ActiveMQ messaging system. First testing iterations were useful to find bugs in the first releases of the Java-binding implementation. Future tests could include alternative JMS-compliant MOMs.

ACKNOWLEDGEMENTS

The Authors gratefully acknowledge the cooperation of "Autostrade per l'Italia" S.P.A and Dr. Eng. Paolo Tonani for his contribution to the requirements analysis and case study demonstration. They also thank Mr. Luca Capannesi for his technical support.

REFERENCES

[1] H. Yang, M. Kim, K. Karenos, F. Ye, and H. Lei, "Message-Oriented Middleware with QoS Awareness", in the Proceedings of the 7th International Joint Conference on Service-Oriented Computing (ICSOC-ServiceWave '09), Springer-Verlag, Heidelberg, pp. 331-345, 2009.

[2] A. Habtamu, R.M. Savola,., J. Bigham, I. Dattani, D. Rotondi,G. Da Bormida, "Self Healing and Secure Adaptive Messaging Middleware for Business Critical Systems", International Journal on Advances on Security, pp. 34-51, vol. 1&2, 2010.

[3] GEMOM Research Project Web Site, http://www.gemom.eu (last access date: July 31st, 2011)

[4] http://www.owasp.org The Open Web Application Security Project – Top Ten Web Application Security Risks (last access date: July 31st, 2011)

[5] http://www.sans.org The SANS (SysAdmin, Audit, Network, Security) Institute (last access date: July 31st, 2011)

[6] J. Wang, J. Bigham, B. Murciano, "Towards a Resilient Message Oriented Middleware for Mission Critical Applications", Proc. of the Second International Conference on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE 2010), Lisbon, Portugal, 2010, pp. 46-51.

[7] J. Wang and J. Bigham, "Anomaly detection in the case of message oriented middleware", in Proc. of the 2008 Workshop on Middleware Security (MidSec '08). ACM, New York, NY, USA, pp. 40-42, 2008.

[8] R. Savola,H. Abie, Development of Measurable Security for a Distributed Messaging System, International Journal on Advances in Security, vol. 2, no. 4, 2009, pp. 358-380.

[9] C.J. Clark, P.T. Blythe, A.Rourke, "Design considerations for road-use pricing and automatic toll-collection systems" Electronics in Managing the Demand for Road Capacity, IEE Colloquium on, pp. 5/1-5/11, 5 Nov 1993

[10] Java Message Service (JMS), Official Web Page, http://www.oracle.com/technetwork/java/index-jsp-142945.html (last access date: July 31st, 2011)

[11] P. Ristau, S. Topham, F. Paganelli, L. Blasi, "GEMOM Platform Prototype Validation through Case Studies - Main Results and Viewpoints to Exploitation", in the Proc. of 30th IEEE Int. Conf. on Distributed Computing Systems Workshops (ICDCSW), pp. 290-291, 21-25 June 2010

[12] L. Blasi, R. Savola, H. Abie, and D. Rotondi. 2010, "Applicability of security metrics for adaptive security management in a universal banking hub system", in Proc. of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10), Carlos E. Cuesta (Ed.). ACM, New York, NY, USA, pp. 197-204, 2010.

[13] N. Looker and J. Xu, "Dependability Assessment of Grid Middleware", 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), pp. 125-130, 2007.

[14] S. Chen, P.Greenfield, "QoS evaluation of JMS: An empirical approach", in: HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences HICSS'04 - Track 9, IEEE Computer Society, Washington, DC, USA, pp. 1-10, 2004.

 ISBN: 978-1-61208-166-3