INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS VOL. 9, NO. 3, SEPTEMBER 2016

www.s2is.org
International Journal on Smart Sensing and Intelligent Systems
ISSN 1178-5608

MIDDLEWARE FOR SMART HETEROGENEOUS CRITICAL INFRASTRUCTURE NETWORKS INTERCOMMUNICATION

Titus Okathe, Shahram Shah Heydari, Vijay Sood, Orane Cole and Khalil El-Khatib

University of Ontario Institute of Technology

Oshawa, Ontario, Canada

Corresponding Email: shahram.heydari@uoit.ca

Submitted: June 1, 2016 ***Accepted: July 12, 2016*** ***Published: Sep. 1, 2016***

***Abstract*-Critical Infrastructures (CIs) are physical assets and organizations responsible for the production and distribution of society's vital goods and services. The increasing interconnection of CIs has resulted in interdependencies which might lead to propagation of failure from one infrastructure to another. Most of current critical infrastructures are equipped with data collection and communication capabilities that can be used to inform and warn other CIs about such events and alarms. In this paper, a publish/subscribe-based communication system among dissimilar (heterogeneous) CIs is presented. The proposed system improves the manageability of CIs by providing an exchange medium for status information and alerts. It achieves this via a uniform architecture, within and across infrastructure boundaries, that maintains data restrictions that reflect real life organizational, administrative, and policy boundaries. Finally, the proposed system is modeled using the OMNET++ simulation framework, and a network performance study investigating scalability is presented. Simulation results showed that system scalability depends on service time per packet, subscription density, and number of clients per router.**

Index terms*:* Critical infrastructure, Publish/Subscribe, Interdependency, Smart Utilities.

I. INTRODUCTION

The collection of physical assets, processes and organizations that are responsible for the production and distribution of daily essential goods and services are referred to as Critical Infrastructures (CIs) [1]. The term CI encompasses a number of vital public sectors including telecommunications; electric power generation and distribution systems; government services; banking and finance; water distribution systems and waste water management systems; transportation (railways, airways, waterways); and emergency services [2]. These sectors interact closely with one another and with their environment. In a modern city, such infrastructures are increasingly becoming interconnected and interdependent. These interactions lead to the emergence of complex behaviours that are characterized by dependencies and interdependencies between the CIs and their components.

Formally, one infrastructure is dependent on another if the state of the former is influenced by the state of the latter or is correlated with the state of the latter. In practice, dependency is often bidirectional. Such dependency may be classified into physical, cyber, geographic, and logical dependencies [2]. A physical dependency exists between infrastructures that rely on the flow of materials/information from one infrastructure to another, for instance between a gas power plant and the gas pipeline infrastructure. Cyber dependency arises when there is a reliance on information flow between infrastructures, for example between the supervisory control and data acquisition (SCADA) network of the electric grid and the electric grid itself. Geographic dependency emerges between infrastructures that are in close spatial proximity. This type of dependency can be found between road networks, natural gas pipelines, underground electricity cables, and or telecommunication fibers that usually follow the same path in a city or municipality. Finally, logical dependency is a type of dependency that does not arise as a result of any of the previously mentioned type of dependencies but as a result of human decisions and or policies. For example, the halting of commercial airlines in the US after the 911 terrorist attack that resulted in slow business leading to layoffs in the airline industries and even some airlines operators filing for bankruptcy [3].

Due to such dependencies, the effect of undesirable events on a particular CI may become compounded by its dependency on other CIs. Furthermore, as the CI in which the event occurs fails, it may cause its dependent CIs to fail or, at the very least, diminish the quality of their operation. This propagation of failure from one infrastructure to another interdependent infrastructure is termed a cascade failure. Cascade failure is a direct consequence of the interdependency between critical infrastructures [4-6]. However, the intricate nature of CI independencies makes it difficult to predict how a failure would propagate through the network of interconnected CIs. Further exacerbating this uncertainty is the limited information available to CI operators about the state of interdependent CIs. Therefore, CI-CI communication is seen as an important tool in improving CI resilience.

The main objective of the research work presented here is to take advantage of the communication and data collection capabilities in modern smart infrastructures to design a comprehensive middleware layer that could provide a mechanism to collect and exchange status and alarm data among multiple heterogeneous CIs. The design for the middleware layer is aimed at providing:

- Scalability: The system should maintain reasonable performance when the number of CIs, or components in CIs, increases.
- Data filtering: This refers to not just filtering in the traditional sense of matching data to specific criteria specified by the end user, but also to the ability to screen and process the data before delivering it. This is especially useful in cases where the end user is not authorized to have access to certain aspects of the data, for instance for security reasons.
- Extendibility: The system should be able to support new applications without a need for reconfiguration of devices or a system upgrade. In other words, it should be relatively trivial to add new applications on top of the existing system.

In this paper, we present a uniform architecture based on publish/subscribe middleware technology to facilitate the communication between distinct CIs. We call this architecture the Publish/Subscribe Middleware for CI Communication (PSMCC). Our main objective in this design is to apply the same principles for CI internal networks as well as the external networks interconnecting them. This helps to simplify the process of inter communication as new infrastructure or technologies need to be installed.

The proposed design and analysis in this paper include:

- Architecture: An architecture for CI communication that allows the communication of heterogeneous CI systems.
- Data format: It specifies a data model that is able to accommodate structured or binary data.
- Detailed cases/mechanisms: Description of mechanisms for the following operational cases - subscribing, publishing, subscription matching, and packet forwarding.
- Performance analysis: Analysis of an example network under varying conditions to investigate the parameters that have the highest effect on performance.

The rest of this paper is organized as follows: In Section II, a review of the previous attempts for CI data collection and monitoring are presented. Section III describes our proposed architecture in detail. The simulation model and analysis of results are presented in Section IV, and conclusions and future work are presented in Section V.

II. RELATED WORK

The term *critical infrastructure* covers a wide range of sectors. Each sector has different communication needs, and therefore has developed independent methods to meet these needs. For the purpose of discussing CI communication, two distinct classes of CIs are identified: these are Utility Infrastructures and Services Infrastructures. Utility CIs includes electricity, telecommunication, water, oil and gas, and transportation. In contrast Services Infrastructures include banking and finance, emergency services, and government services. This distinction is necessary as the members in each group have similar communication requirements. For instance in utility CIs the aim is usually to monitor the state of the equipment out in the field to ensure they are working optimally. In these types of systems there is also a need to take corrective action by means of some form of control system when undesirable events occur or the system is heading towards unstable operating conditions. This control is usually in the form of nudging a process back within operating limits. Furthermore, these type of CIs are more geographically disperse and contain a greater number of equipment than the services type. Also Utility Infrastructures tend to be more geographically and physically dependent. In contrast the services CIs communication needs are less about controlling a physical system but about the timely delivery of relevant information such as location, severity etc. of relevant events and or transmitting data usually to an end user which is usually a person. Therefore, data in these systems are designed around being human readable. Services style infrastructures usually employ existing commercial internet technologies and enterprise systems. There have been a number of projects whose goals are aligned with providing a communication architecture for CI-CI communication, especially in the context of sharing information for the purpose of improving resiliency and the effects of cascade failures. In the next subsections, each of the projects is highlighted with its contributions.

The Integrated Risk Reduction of Information-based Infrastructure Systems (IRRIIS) [7] was one of the earliest programs focused around developing models and tools for analyzing, simulating and managing dependent and interdependent CIs. It did this by introducing a language for describing risk as well as a set of middleware communication technologies to exchange this information among dependent CI systems.

The Middleware Improved Technology (MIT), provides the communication technologies that supports the communication between CI systems of different types. MIT is the middleware backbone that allows the communication between multiple dissimilar CI systems. It consists of the MIT Communication Tool, Risk Estimator (RE); CRIsis management and Planning System (CRIPS) decision support tool; Tools for Extraction and functional status (TEFS); and the Incident Knowledge Analyzer (IKA) [7]. The MIT Communication Tool represents the communication backbone of the IRRIIS model. It supports the exchange of information between dependent CIs using the risk management language (RML) via Web services. The MIT communication backbone is designed to use current internet based technology. It uses the TCP (Transmission Control Protocol) for guaranteed exchange of information.

IRRIIS is a multi-faceted program and developed a number of outcomes including a federated simulator (SimCIP) for modeling interdependent CIs. To mitigate the problems of data format exchange and relevance across domains it employs the TEFs to extract the data from SCADA systems and uses the XML based RML as a format to exchange the extracted information. However, it did not directly use the raw data from the infrastructure; rather the Risk Estimator (RE) uses the input from the SimCIP simulator. Data from the infrastructure such as that from the SCADA system is transformed before it is used via the TEFS module.

The European Commission MICIE FP& ICT-SEC project [8] was aimed at developing a Critical Infrastructure Warning Information Network (CIWIN) for European Union member states. The project improved CI resilience by providing real time risk level that measures the likelihood that a given CI would be unable to provide its services with the required quality of service (QoS) as a result of undesired events in the reference CI and/or in its interdependent CIs. The MICIE system was designed to provide a method to discover distributed information relevant for the alerting system, overcome the disparity of this information from multiple CIs, and finally provide the means to exchange/share this information securely over the internet. The MICIE architecture is described in [8].

Some goals of the MICIE project were similar to our research objective; however, MICIE used a secure mediation gateway (SMGW) as the communication element between multiple CIs. The primary goals of the SMGW included: providing a secure cross-CI communication infrastructure; CIs critical event discovery and propagation of relevant information to trusted interdependent CIs; composing of CIs critical events and semantic inferences; and the extension of risk prediction from a single CIs to multiple interdependent CI [9]. The MICIE project tries to enhance the resilience of CIs by integrating status information into analytical tools that help predict the future states of a CI. MICIE differs from IRRIIS in that, instead of data from a simulator, it uses the raw data from the infrastructure [9]. Furthermore, the MICIE project has been tested on a real pilot system using the interdependencies between a telecommunication infrastructure and the electrical power infrastructure [8].

The projects considered so far have dealt with multiple CIs. The GridStat project [11] is different in that it tries to address the limitations of current SCADA systems by making it possible for any entity, within the network, to receive the data from any other entity on the network irrespective of location. It also differs from the previous projects in that it deals specifically with attempts to improve the communication and monitoring systems in the electric infrastructure and similar industrial-like utility systems. Its main contributions include: a reliable QoS managed middleware; hierarchically organized control entities called “*QoS brokers*”; and rate filtering mechanism. It uses the Common Object Request Broker Architecture (CORBA) protocol, a form of remote procedure call (RPC) for distributed object. This differs from the use of web services as in MICIE or IRRIIS, and provides lower latency. Although GridStat provides a means for distributing power system data, it does not provide a means to connect multiple infrastructures of different types. Therefore, in its current form, it cannot connect heterogeneous infrastructure types. However, it does pave the way for a more inclusive communication architecture design for

CIs that may be extended beyond the boundaries of single infrastructure to multiple, heterogeneous infrastructures.

It must also be noted that generic Machine-to-Machine (M2M) data dissemination systems based on the publish-subscribe model such as DDS [12] are shown to be inadequate for large scale CIs, in particular for their decentralized unbrokered architecture that limits the ability of CI administrators to control the access, quality and flow of information from one CI to another. Previous research has also raised questions about the scalability and resilience of such systems for deployment in large scale CIs [13]. The goal of our research was to create a system that extends the architecture of GridStat to provide for a unified and scalable message exchange between heterogeneous CIs.

III. SYSTEM ARCHITECTURE

The proposed publish-subscribe middleware for CI communication (PSMCC) uses a content based publish-subscribe with a type based system. It consists of four basic components: brokers, information routers, publishers, and subscribers. The brokers and routers together make-up the publish/subscribe middleware infrastructure, which is the message brokers. Together, these two entities provide the system clients with the necessary infrastructure backbone to exchange information.

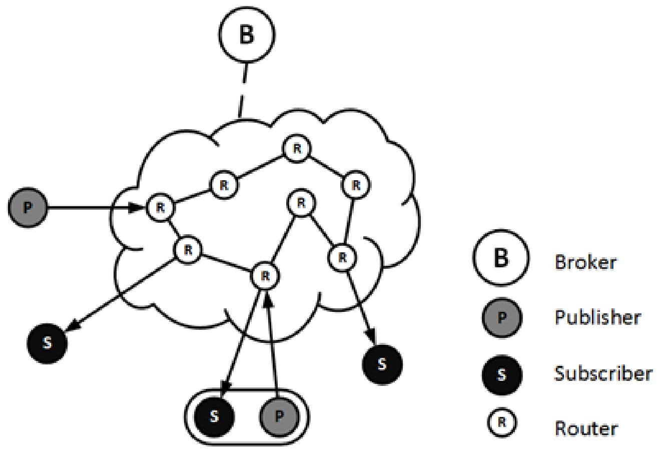


Figure 1 - System Architecture diagram

Figure 1 shows the basic system architecture. The broker and the network of routers represent the message broker infrastructure. Similarly, the publisher and subscriber entities represent the system clients. A client may also be both a publisher and a subscriber, as shown in the diagram.

A. System Components

Broker

The broker is the network controller. It authenticates the clients of the systems and allocates network resources. The broker allocates paths and controls how the routers forward data from publishers to subscribers. From a functional point of view, the broker may have some or all of the following components:

- Subscription Database (SD): This stores the information about active subscriptions on the network.
- Publication Database (PD): This stores the publication information for all active publishers on the network. This includes their address and information about the type of data they publish and how they publish that data.
- Security Manager (SM): This is responsible for authenticating and authorizing the clients of the system to either publish or subscribe the information via the routers. It may also implement security policies that limits the parts or kinds of data that may be visible outside its domain.
- Network Information Database (NID): This database stores the topological information of the underlying network. This information is necessary to allow the broker optimize the use of network resources.

Routers

The routers in Figure 1 represent application layer (overlay) routers. Each router includes a routing engine to forward data from publishers to subscribers and a matching engine to match the published data to specific subscriptions. The router is responsible for aggregating and scheduling data transfer to multiple subscribers who may need data at different rates. We assume, without loss of generality, that a standard shortest-path routing algorithm is used for path determination among the overlay routers.

Publishers

The publishers are those clients of the system that generate events or data that is of interest to the subscribers. A client may be both a publisher of data and a consumer of data. For example, a monitoring application may subscribe to sensor information and generate alarms (new events) when it observes an undesirable pattern. The alarms it generates may then be subscribed to by a control system or even a system administrator who may then take appropriate action.

In practice, publishing devices may be any device from sensors to enterprise servers. A single device may also contain more than one publisher application, each application being associated with a particular data type. In the model, applications are identified using a Global Unique IDentifier (GUID) as well as its parent device Internet Protocol (IP) address and a port number.

Subscribers

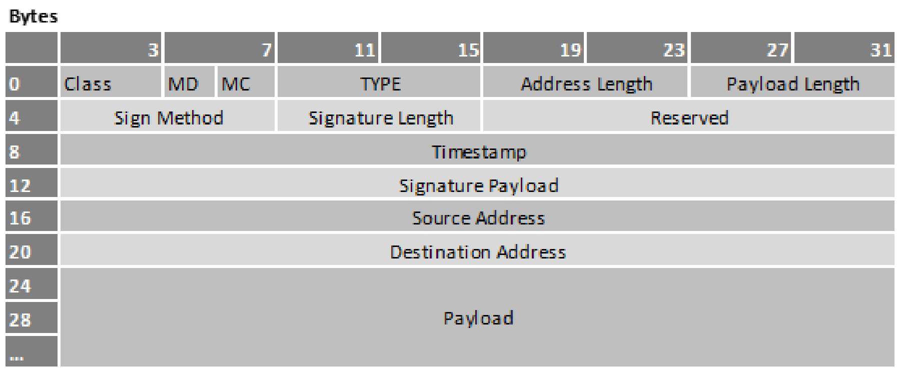
Subscribers represent applications or systems that are interested in receiving the information about a set of publications. They express this interest by sending a subscription request to the broker through a leaf router. The leaf router is the network router to which the subscribers are connected to. The leaf router is not a simple IP router but is also capable of processing the content of the packets flowing through it. Similar to publishers, subscribers may be small devices or large enterprise servers.

B. Network Architecture

In CI systems, various communication technologies may be used within the same organisation. However, by employing an overlay network, the system can offer a uniform interface to its clients without requiring a major change to existing communication infrastructure. Furthermore, the TCP/IP communication stack is already widely available in many applications in CI systems. Therefore, building an abstraction on top of the TCP layer seems a logical choice.

Message Structure

Figure 2 shows the general packet format used in this model. The source and destination addresses are abstract overlay addresses. While a typical 32-bit IP address has been assumed in Figure 2, in our model the address can be a tuple of the applications identifier, the IP address of its host and its listening port number (with appropriate fields added to the message format).



Bytes	3	7	11	15	19	23	27	31
0	Class	MD MC	TYPE		Address Length		Payload Length	
4	Sign Method		Signature Length		Reserved			
8	Timestamp							
12	Signature Payload							
16	Source Address							
20	Destination Address							
24	Payload							
28								
...								

Figure 2 - Message Format

The Mode field (MD) is used to model request-reply message interactions. The mode flag is set if the datagram is a reply and unset if the packet is a request. This allows the system to model client-server interactions for control messages. Control messages are used by the message brokers to change the behavior of the system. They may also be sent by clients to register intent to use the system, using the TCP protocol for transport since control messages should be guaranteed delivery. The Class field is used to specify if the message is a control or data message. Messages can either be unicast or multicast. The Multicast flag (MC) models this difference. This flag is set for multicast messages and unset for unicast messages.

The Type field (TYPE) is used to specify the type of control message. The model defines the following control messages:

- Forward Packet: This type of control message is sent by the broker to the routers under its control to modify their forwarding table. It carries a forwarding rule payload. The forwarding rule consists of a subscription predicate and the next hop address.
- Ping Packet: This type of control message is sent by the system clients to locate the nearest router to connect to. This is how the system clients join the network.
- Publication Packet: This type of control message originates from publishing clients and carries the publication definition. The publication definition specifies the type of data published by a publisher, including how it intends to publish the data. This information is used by the brokers to match subscriptions against publications at subscription time.

- Subscription Packet: This type of packet carries the subscription information. It is used by the subscribing clients to register subscriptions on the network.
- Query Packet: This type of packet is used by the brokers to query other brokers for publications matching a local subscription. When a broker receives a subscription request from a local subscriber, it also queries other networks for matching publications. This also carries subscription information.

The signature and time stamp fields are used by the security manager module, which is described in more detail in the next sections.

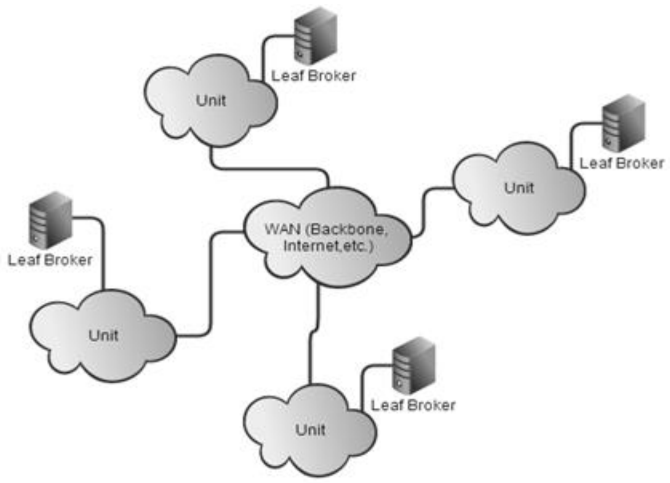


Figure 3.a - Flat Topology

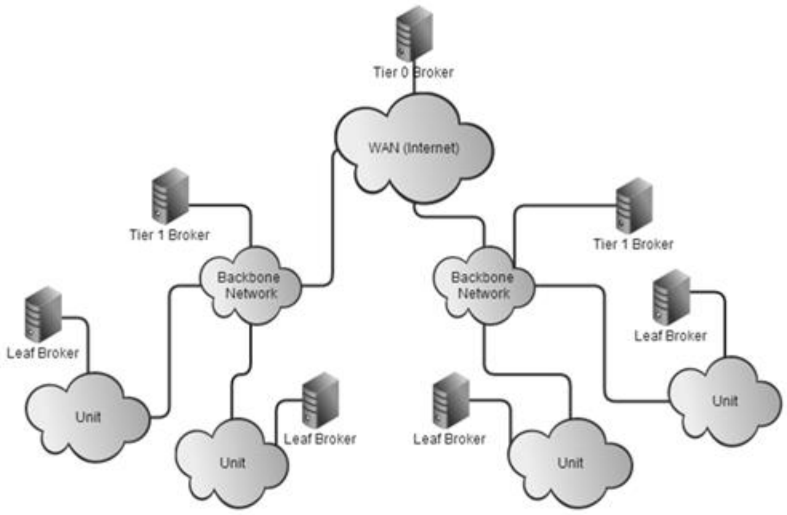


Figure 3.b - Hierarchical Topology

The proposed network architecture may be connected either as a flat or hierarchical topology. In Figure 3, examples of flat and hierarchical topologies are shown. This hierarchical topology allows more general rules to be set higher up the hierarchy. It also allows the system to reflect geographic and administrative boundaries. In this arrangement, when a subscription request is received by a leaf broker, it contacts its tier broker for matching publications in other units under the administration of the tier broker. The tier broker then acts as a trusted middle man between the leaf brokers. The leaf brokers determine the kind of data that subscribers from other units can access or subscribe to. For instance, if the data contains sensitive parts, this may be removed before being sent to a subscriber outside the leaf broker’s domain. This can be achieved by the routers on the network.

In a flat topology, instead, the brokers operate as peers. There is no tier broker to act as a middle man. Therefore, when a leaf broker receives a subscription request for data outside its domain it contacts its peer leaf brokers for matching publications. Any peer with a matching publication replies to the leaf broker where the request originated and a path is created for matching publication from the source unit to the destination unit.

The topologies introduced so far provide flexibility to the system. The hierarchical topology may be suitable in situations where a central controller is required with many sub controllers. In contrast, a peer-wise broker system may be suitable for simpler systems or at the top of a hierarchy where no central authority exists.

Start up and Service Discovery Phases

An important part of any publish/subscribe system is how the system clients discover publications and how they connect to the network. One approach is to broadcast presence information on a predefined port on a local network. Clients supporting the required service would listen on this port and respond to incoming requests. This is a simple but rather inefficient method if all the clients on the network do not support or need the required service. An alternative is to have a predefined multicast group and port through which clients can discover the services supported by network connected devices. This approach is used in the Simple Service Discovery Protocol (SSDP), a part of the Universal Plug and Play protocol. Another technique for service discovery presented in RFC6763 is the Domain Name Service-Service Discovery (DNS-SD) that uses DNS packets to advertise services available on a network. The advantage of a DNS-SD is that it is not limited to a link local address.

In PSMCC architecture, we assume that each network device is connected to at least one information router on the local network. This leaf router represents the clients access point to the publish/subscribe infrastructure. When a publisher or subscriber connects to a local network, a packet is sent to broadcast its presence and discover the leaf router as shown in Figure 4. When a router supporting the service receives the packet, it responds with its full address which, in this case, is a global unique id (GUID) and IP address, and a connection port. After receiving information to connect to the publish/subscribe network. The next step differs slightly depending on whether the system client is a publisher or a subscriber. Publishers send out an advertisement packet. This contains the information about what kind of data they will be publishing and how they would publish the data.

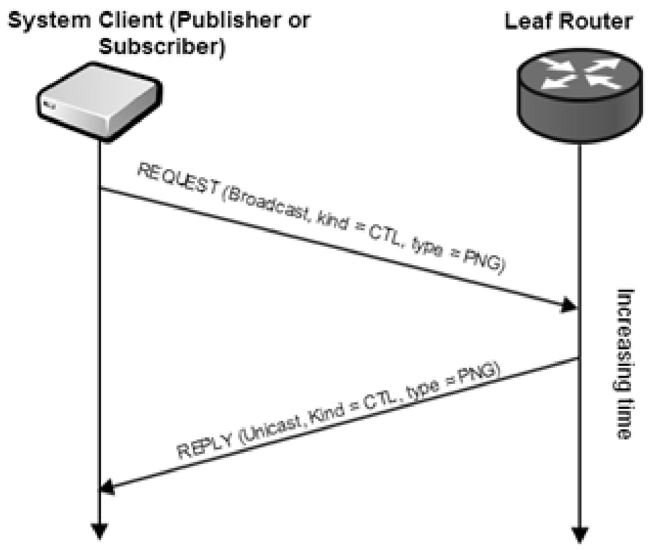

System Client (Publisher or Subscriber)
Leaf Router
REQUEST (Broadcast, kind = CTL, type = PNG)
Increasing time
REPLY (Unicast, Kind = CTL, type = PNG)

Figure 4 - System client start up timing diagram

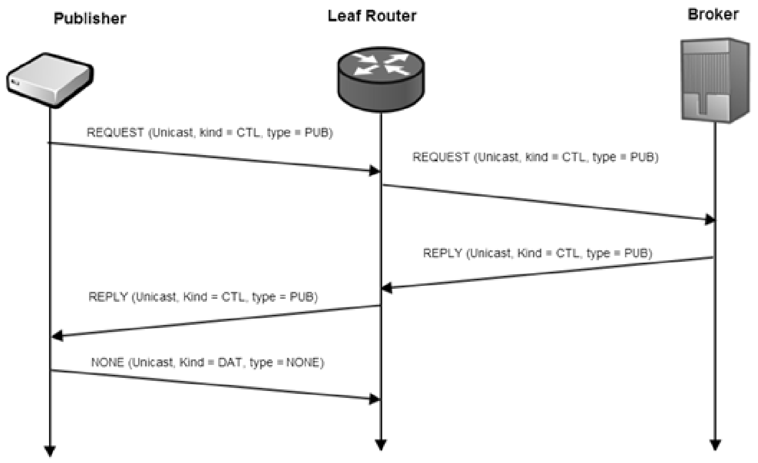

Publisher
Leaf Router
Broker
REQUEST (Unicast, kind = CTL, type = PUB)
REQUEST (Unicast, kind = CTL, type = PUB)
REPLY (Unicast, Kind = CTL, type = PUB)
REPLY (Unicast, Kind = CTL, type = PUB)
NONE (Unicast, Kind = DAT, type = NONE)

Figure 5 - Publisher timing diagram

In this model, the publisher can either publish in a periodic manner or sporadically. The publisher sends out a request that is forwarded to the broker. The broker responds by granting the publisher permission to publish data on the network, after storing the details of the new publication. On receiving the broker's reply, the publisher can then start sending its data packets.

The process for a subscriber is more involved as it depends on whether the system has hierarchical or flat topology of brokers. After obtaining the application address of the leaf router, the subscriber sends a subscription request, via the router to the broker. The broker queries its publication database to find local publishers that are publishing data matching the new subscription. This includes not just the data but how the data is published. For instance, say a subscription is received for the topic '*.*.water.level' matching publications 'xyz.abc.water.level' and 'xyz.rtc.water.level'. In addition, the subscription specifies a minimum interval time between readings. For example, if our subscription requires a minimum interval time of 200 milliseconds but 'xyz.abc.water.level' only publishes every second, then the subscription cannot be satisfied by this publication even though the topic matches. If no local subscriptions are found or the subscriptions topic is not in the local domain, the search is expanded to other brokers. If matching publications are found, a reply is sent to the subscribing and network paths setup to allow the matching publications reach the subscriber. This process is summarized in Figure 6.

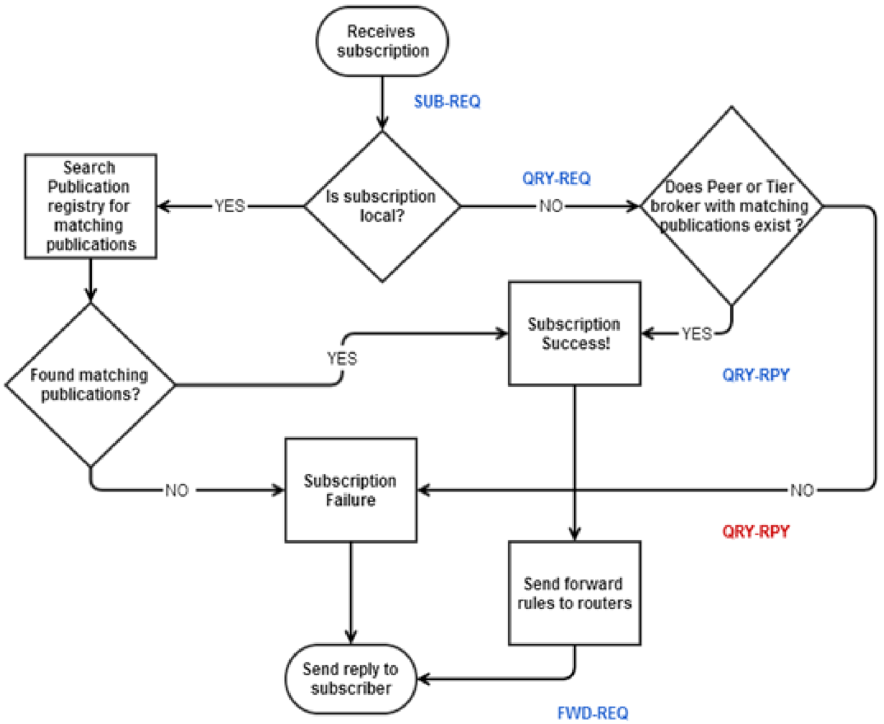

Receives subscription
SUB-REQ
Is subscription local?
YES
Search Publication registry for matching publications
QRY-REQ
NO
Does Peer or Tier broker with matching publications exist ?
Found matching publications?
YES
Subscription Success!
YES
QRY-RPY
NO
Subscription Failure
NO
QRY-RPY
Send forward rules to routers
Send reply to subscriber
FWD-REQ

Figure 6 - Subscription Flowchart

When a subscription request is made, the broker attempts to match the subscription request with active publications; this process is in three parts as shown in Figure 7. First, it matches the subscription object type with the publication object type. Second, it matches the subscription topic with the publication topic. Finally, it confirms that the publication QoS can satisfy the subscription QoS requirements. After a subscription is matched, the broker sets up paths from the publishers leaf routers to the subscriber leaf router. This is done using a forward packet, which is one of the control packets.

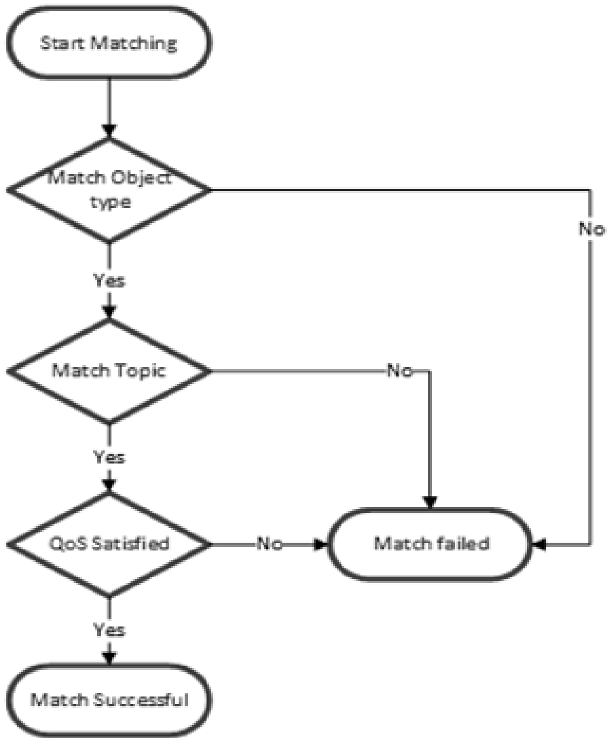


Figure 7 - Subscription Matching flowchart

C. System Data Dissemination Model

Data model

Our proposed data model implements a mixture of a hierarchical topic-based model and a type-based publish-subscribe model. In this case, the topic is yet another property of an object, while preserving the extensibility provided by type-based publish/subscribe. For instance it becomes possible to have object types that define a specific data with a topic that reflects the specific CI, such as Alert class with a topic such as ‘abc.genco.area1.disruption’ and ‘cvh.watercorp.zone1.pressure.flunctuation’.

The model also defines an advertisement data structure that is used by publishers to register their data types with the broker. This is encapsulated by the Advertisement object. The data structure consists of a specification field that defines the characteristics of the data. The qualities here refer to additional information provided by the publishing client such as whether it intends to publish periodically or otherwise, packet size descriptions, publication interval if applicable etc. The sample data field is used by the publisher to provide a dummy object to the publish/subscribe system. This provides a means for the publish/subscribe system to acquire the type information for the messages that will be sent through the system.

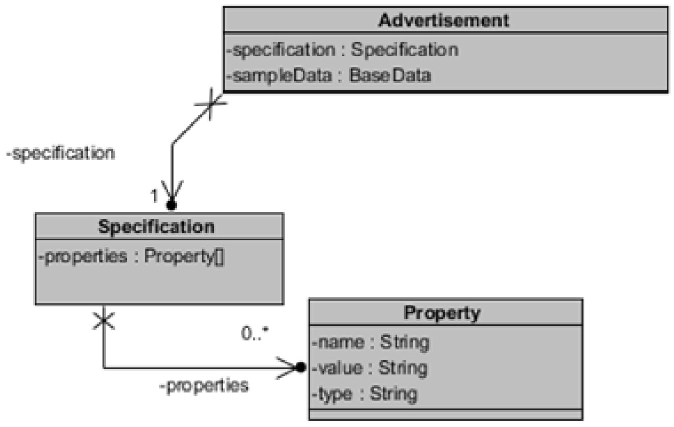


Figure 8 - Data Advertisement UML class diagram

Subscription Language Model

The subscription model uses a template class and predicates to subscribe to data. This leverages the flexibility of type based publish/subscribe. Since the data model are based on programming language types the subscription are based on selecting a class type to subscribe to. Next, the subscription contains a number of predicates which are Boolean expressions against the public members (properties) of the selected data type. The subscription model, shown in Figure 9, has two properties:

1. The requirements provide information about how this subscriber intends to receive matching data. For instance, it may want to receive every matching data at a fixed rate. Furthermore, the requirements field may also specify latency and delay requirements.
2. The getPredicate() method is an abstract method which allows the subscriber to define it on implementation. For instance the PeriodicSubscription class embodies a subscription to a

PeriodicData (data published at regular intervals). It also uses a PeriodicPredicate as the predicate type. The PeriodicPredicate is explained in more detail in the next section.

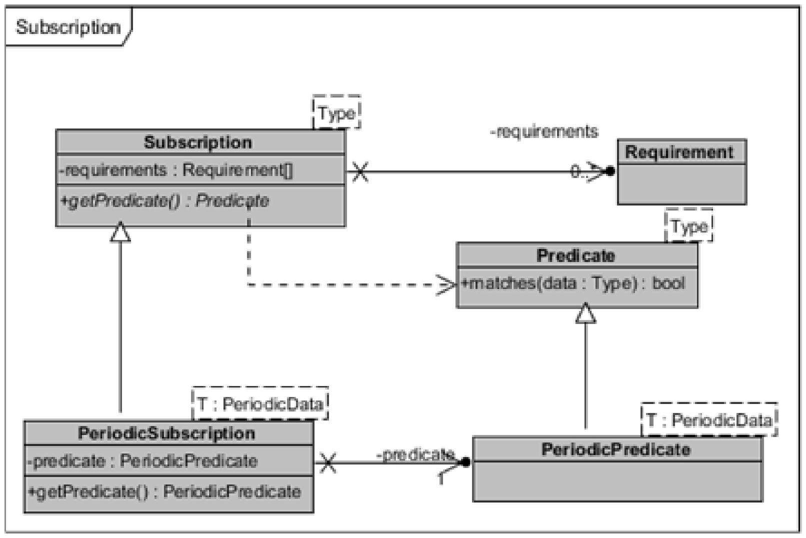

Subscription
Type
Subscription
-requirements : Requirement[]
+getPredicate() : Predicate
-requirements
0..*
Requirement
Type
Predicate
+matches(data : Type) : bool
T : PeriodicData
PeriodicSubscription
-predicate : PeriodicPredicate
+getPredicate() : PeriodicPredicate
-predicate
1
T : PeriodicData
PeriodicPredicate

Figure 9 - Subscription Class UML diagram

Matching/Filtering Events

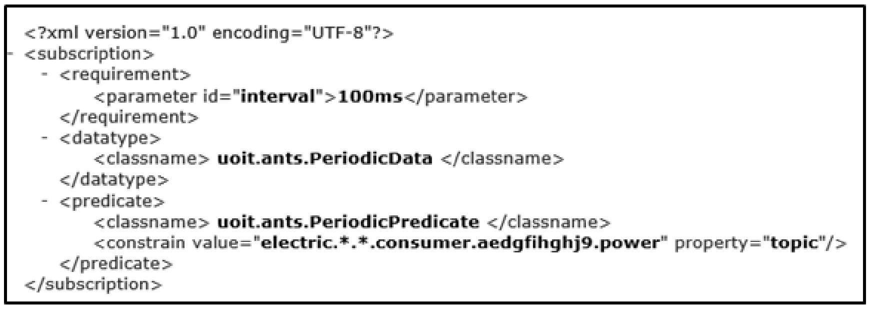
The Predicate base class defines the mechanisms for matching publications to subscription. Depending on the implementation, matching may be stateful or stateless. Stateful matching refers to a matching where the current match depends on previous matches. For instance, a predicate that tracks changes in a value requires the previous states of the value in question to compute the change in value. In other words, whether the current message matches or not depends not just on its value but the value of the previous messages. Stateless matching refers to matching in which the previous messages have no effect on whether the current message matches or not.

The PeriodicSubscription class follow the stateful matching paradigm (Figure 10), as only message matching a given time-series is delivered to the subscriber. The MultiPredicate is a compound predicate that is made up of two simple or compound predicates to build more complex predicates. For example, consider the subscription defined by the statement:

“topic=electric.*.*.consumer.aedgfihghj9.power,type=uoit::ants::PeriodicData,rate=100"

The topic represents the data object topic to which this subscriber wants to subscribe. Furthermore, it is represented as a wild card notation. Consequently, all messages whose topic match the wildcard will be a match for this subscription request.

The ***type*** represents the data type in which this subscriber is interested. The subscription type must have the property topic otherwise the subscription is not valid. This is because not all data types have a topic. For instance, the XmlData type has no topic associated with it. Next, the rate defines how often the subscriber wants to receive data, in this case 100 data points per second. Although the subscription is shown here as text, it is only for illustration purposes and to make generating the objects during simulation simpler.



```
<?xml version="1.0" encoding="UTF-8"?>
- <subscription>
  - <requirement>
      <parameter id="interval">100ms</parameter>
    </requirement>
  - <datatype>
      <classname> uoit.ants.PeriodicData </classname>
    </datatype>
  - <predicate>
      <classname> uoit.ants.PeriodicPredicate </classname>
      <constrain value="electric.*.*.consumer.aedgfihghj9.power" property="topic"/>
    </predicate>
</subscription>
```

Figure 10 - Simple Subscription XML Message

D. D. Security Manager

To satisfy the security requirements of the publish/subscribe architecture presented, a security model as shown in Figure 11 has been proposed that addresses the security needs of the participating CIs. The proposed model adopts the security principles discussed in [14] and [15] and provides support for disparate technologies. To meet the security demands of the publishing and subscribing nodes, their respective security policies must be enforced by the Security Manager at the broker for each CI, as shown in Figure 11.

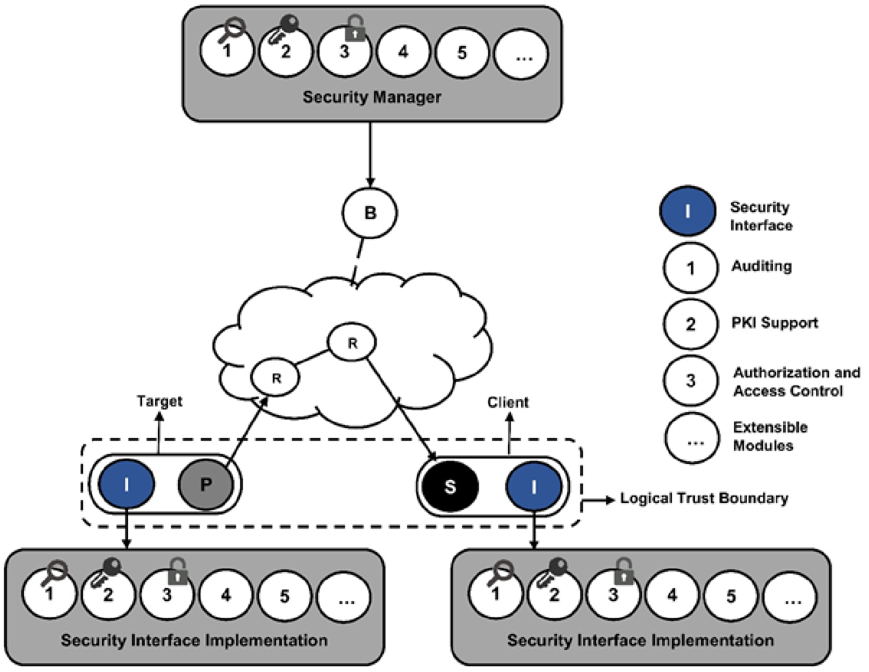


Figure 11 - Security Manager

The services covered by this framework are:

- Identification and authentication
- Authorization and access control
- Auditing
- Confidentiality and Integrity
- Non-repudiation
- Security Administration

Invocation of secure-aware nodes requires identification and authentication rules to be established by the publisher which are enforced by the Security Manager. A secure means of establishing trust between distributed nodes can be satisfied through Kerberos v5 authentication mechanism presented in Figure 12 due to its maturity and wide adoption. In this authentication scheme, trust boundaries are established via the CORBA framework-Level 2 [14] which implements the Generic Security Services Application Program Interface (GSS-API) to provide seamless integration with Kerberos services. The GSS-API provides a layer of abstraction and is extensible to include other authentication mechanisms such as PKI or other desirable modes of authentication.

A trust relationship between subscriber and publisher are not always required in the event of public message exchange. It should be noted, however, that there may still be a requirement to establish a trust relationship between subscriber and broker which does this through mutual authentication.

A general overview of the adapted CORBA GSS-API framework in Figure 8 is presented below:

- Each node, subscriber and publisher, acquires credentials explicitly, if credentials have not been acquired automatically;
- The subscriber initiates a security context and the publisher accepts it.
- The subscriber applies security protection to the message. This phase is optional as confidentiality may not be desired, authentication will be the default property of this scheme as the initiation phase establishes a security context between participating nodes;
- The publisher decrypts the message if needed and verifies it if appropriate;
- (Optional) Mutual authentication may be achieved through the response of the publisher by sending an identifier that will be decrypted by the subscriber; and
- Shared security context must be deleted once communication is complete.

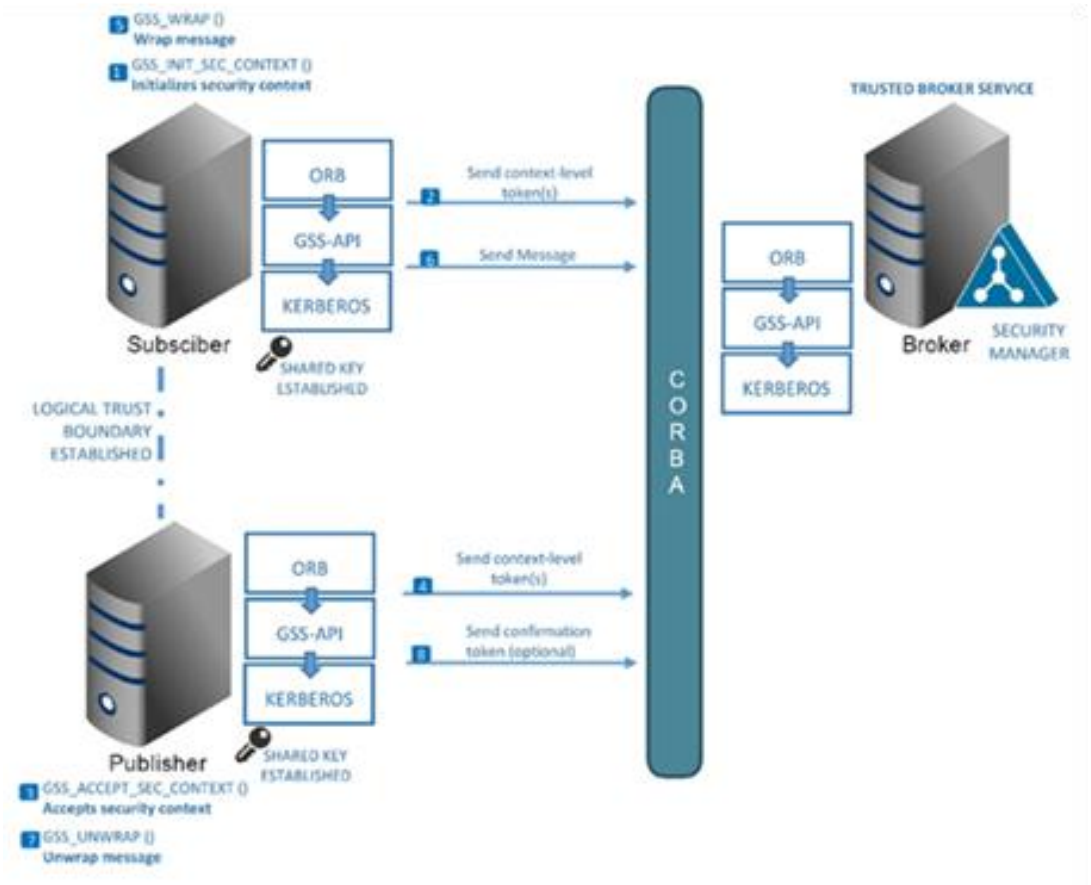

5 GSS_WRAP ()
Wrap message
1 GSS_INIT_SEC_CONTEXT ()
Initializes security context
TRUSTED BROKER SERVICE
ORB
GSS-API
KERBEROS
2 Send context-level token(s)
6 Send Message
Subsciber
SHARED KEY ESTABLISHED
LOGICAL TRUST BOUNDARY ESTABLISHED
C O R B A
Broker
SECURITY MANAGER
4 Send context-level token(s)
8 Send confirmation token (optional)
Publisher
3 GSS_ACCEPT_SEC_CONTEXT ()
Accepts security context
7 GSS_UNWRAP ()
Unwrap message

Figure 12 - GSS-API Kerberos v5

IV. PERFORMANCE RESULTS

The PSMCC architecture, including modules, network infrastructure and packets, were implemented in the OMNET++ simulator. In simulating the behaviour of the network, a simple network was chosen with two units. Each unit consisted of five routers, a broker, and an equal number of publishers and subscribers. This was to model the case where every subscriber is also a publisher. The router network was generated using the BRITE Topology generator and Waxman random topology model [17]. The security module was not implemented in this simulation, but we believe its impact on response time to be minimal and mainly limited to the start-up time. We simulated networks of 20, 40 and 80 end nodes per router. The experiment was conducted using an average packet size (1024 bytes).

Figure 13 shows the end-to-end latency between publishers and subscribers versus the message service time, i.e. the amount of time required for a node (router) to process a packet and send it to the egress queue to be put on the line. In a type-based publish/subscribe system, this would typically be the amount of time to de-serialize the packet, make a routing decision, serialize the packet, and send it to the output interface. If this time is larger than the average packet inter-arrival time then the ingress queue tends to grow out of bound and the network might experience congestion or packet loss. The end-to-end latency includes the propagation time, service time, queuing delay and transmission time. As our results indicated, in this network, as long as the service time per packet is less than the 400 microsecond threshold, the end to end delay was fairly negligible. However, there is a sharp, exponential rise as the service times increase and queue lengths grow, resulting in congestion. This scenario shows the effect of a slow packet processor on the system performance.

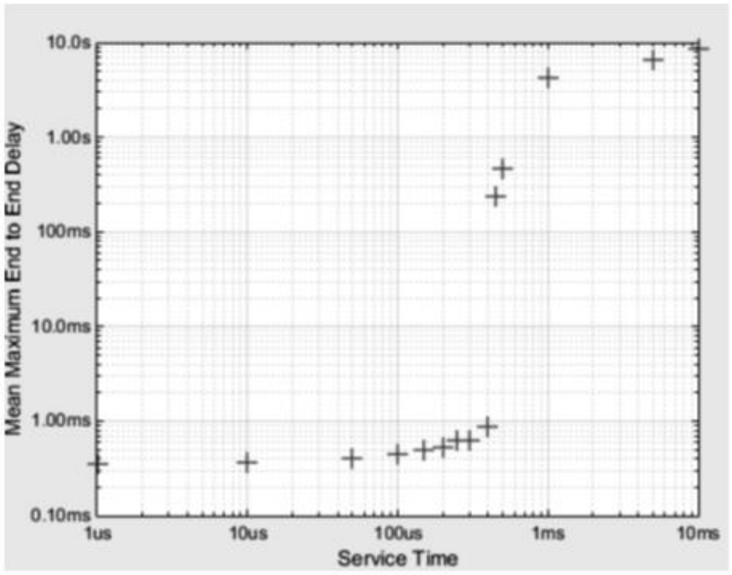


Figure 13 - Mean maximum end to end delay versus service time

Table 1 shows the average queuing time for data packets for each network. With 160 clients per router, the queuing time is increased to more than one second. This is for a system with an unlimited buffer size. If buffer sizes are small, packet loss could happen and some subscribers may not receive their data.

Table 1- Average queuing time for data packets

Clients/ Router	Low (ms)	Medium (ms)	High (ms)
20	0.0024	0.0043	0.0049
40	0.0058	0.0091	0.0067
80	0.0250	0.0273	0.0209
160	211.97	> 1s	> 1s

The data inter-arrival time represents the interval of time that elapses between each successive data packet received by the subscribers. This was an important parameter to record as it shows whether the system can meet the QoS requirement imposed by subscribers. In this case, this is the minimum interval between each notification. As an example, say a subscriber subscribes to a topic, “abc.efg.123.*” at a minimum interval of 100 ms between data points, that is 10 notifications per second. If the number of clients being served is few, does the system meet this requirement, and how does the performance degrade as the number of clients served increases? This parameter attempts to measure this drift from the requested subscription rate. Table 2 shows, as a percentage of total subscribers, the number of subscribers whose data inter-arrival time falls outside the requested interval by one percent and five percent. It is clear that an increase in the number of subscribers per router degrades this performance.

We also considered the scenarios of routers with limited resources. We conducted comparative simulation experiments in scenarios where each overlay router had a limited buffer size of 20 packets versus the case of unlimited buffer size, in order to study the impact of such limitation.

The maximum end-to-end delay at the subscribers and the data inter-arrival time were collected to compare what happens as the service time grew with a limited queue size. Figure 14 shows the data inter-arrival time requested by the subscribers overlaid by the mean data inter-arrival time recorded. It is clear that at 10 µs service time, the performance is unaffected as the maximum queue size of 20 packets is not reached if the packets are processed quickly enough. A performance difference becomes apparent at higher service times as the queue capacity is exceeded and the system starts dropping less important packets. At higher service times, newer packets are being favoured over older packets. Hence, we can see that by limiting the buffer size the system actually performs better albeit with some data loss.

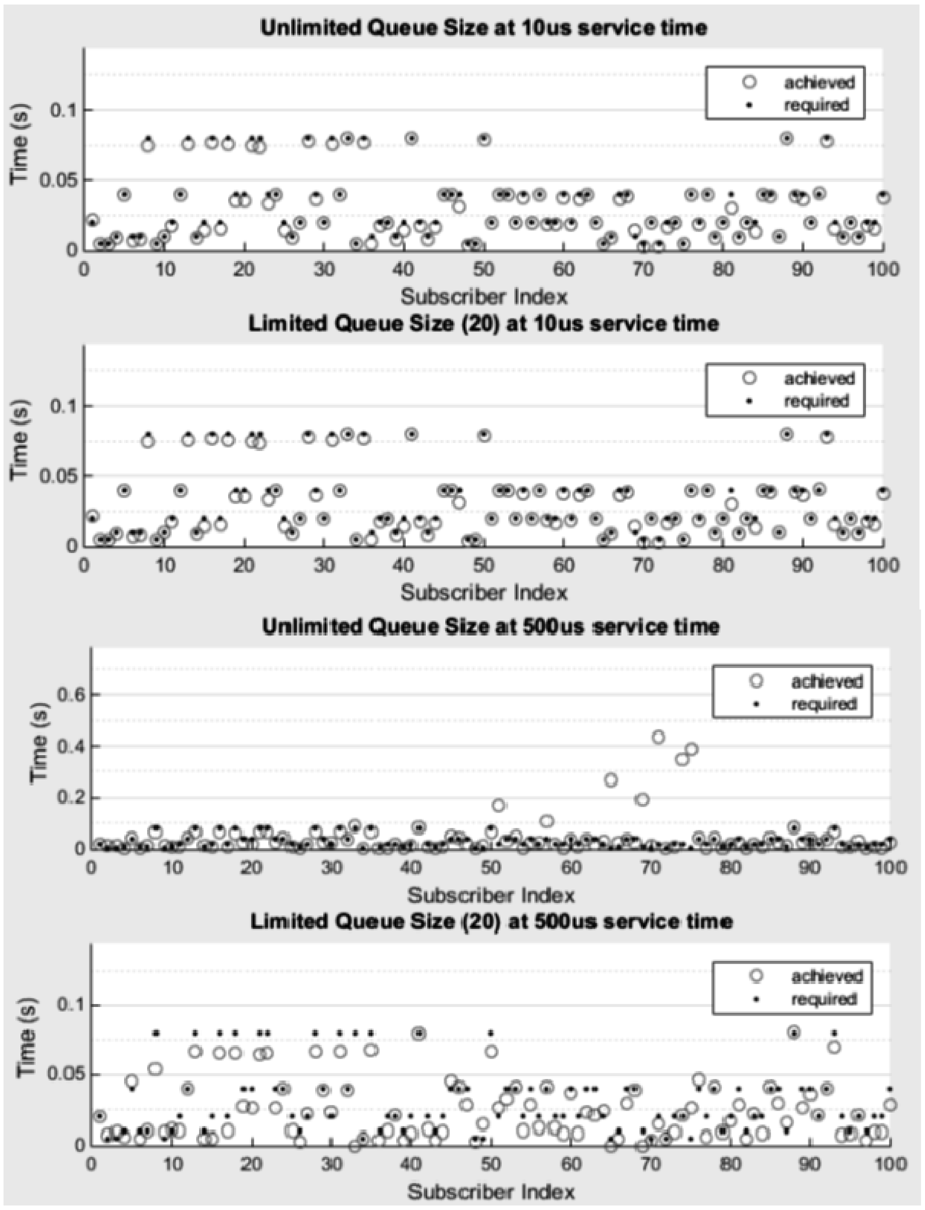


Figure 14 - Data Inter-arrival rate comparison for low and high service times

Table 2- Data Inter-arrival rate performance

	Clients/ Router	Low (%)	Medium (%)	Heavy (%)
Less than 1 %	20	75.00	56.00	31.00
	40	40.50	27.50	13.00
	80	28.00	21.75	1.50
	160	18.75	22.00	2.00
Less than 5 %	20	75.00	56.00	31.00
	40	41.00	29.00	13.00
	80	28.25	22.00	17.00
	160	18.75	22.00	2.00

V. CONCLUSION AND FUTURE WORK

A communication architecture based on the publish/subscribe communication paradigm for the exchange of information between heterogeneous CI systems is presented. The proposed architecture was simulated for the purposes of analyzing the network performance of the model. The network performance study showed a strong correlation between network performance and the number of clients serviced by each router. Furthermore, the network performance degraded heavily when the service time per packet was high. There was also a marginal improvement when the queue capacity was limited and the queue optimized to favour newer packets over older ones.

With regard to the future work, data prioritization in inter-CI communication is probably the most important step toward improving the performance. In our simulations, we considered the age-based priority; i.e. selectively dropping older packets. This is particularly relevant in case of monitoring data as the most current information has priority over older data. However, information such as alerts may selectively have higher priority than, say, a simple notification. Hence, the priority bit available in the message header may be used for this purpose in future implementations. If implemented over an IP based network, the IPv6 differentiated services of RFC2474 may be incorporated to achieve this behaviour.

The current system only provides one parameter for QoS namely the data inter-arrival rate. However, other QoS guarantees could be incorporated to improve the number of options available to subscribers, and consequently provide more parameters to optimize.

Also since the system is supposed to provide a backbone for the communication between CIs, it is important to investigate ways to ensure that the system itself is resilient to failures and available. One important piece in the system architecture is the broker who acts as the network controller. If it becomes offline, new subscriptions may not be processed and publications would fail. Also, at the router level, what happens when a router goes offline within a network? These are areas to investigate and implement a recovery mechanism.

REFERENCES

[1] J. D. Moteff, "Critical infrastructures: Background, policy, and implementation" (DIANE Publishing, 2010)

[2] S. M. Rinaldi, J. P. Peerenboom and T. K. Kelly. "Identifying, understanding, and analyzing critical infrastructure interdependencies." Control Systems, IEEE 21.6 (2001), pp. 11-25.

[3] P. Pederson et al. "Critical infrastructure interdependency modeling: a survey of US and international research." Idaho National Laboratory (2006), pp. 1-20.

[4] S. V. Buldyrev et al. "Catastrophic cascade of failures in interdependent networks." Nature 464.7291 (2010), pp. 1025-1028.

[5] I. Dobson et al. "Complex systems analysis of series of blackouts: Cascading failure, critical points, and self-organization." Chaos: An Interdisciplinary Journal of Nonlinear Science 17.2 (2007), pp. 026103.

[6] L. Duenas-Osorio and S. M. Vemuru. "Cascading failures in complex infrastructure systems." Structural safety 31.2 (2009), pp. 157-167.

[7] R. Klein, "Information Modelling and Simulation in Large Dependent Critical Infrastructures–An Overview on the European Integrated Project IRRIIS." Critical Information Infrastructure Security. Springer Berlin Heidelberg, 2009, pp. 131-143.

[8] P. Capodieci et al. "Improving resilience of interdependent critical infrastructures via an on-line alerting system." Complexity in Engineering (COMPENG'10), IEEE, 2010, pp 88-90.

[9] F. Caldeira et al. "Secure mediation gateway architecture enabling the communication among critical infrastructures." Future Network and Mobile Summit, IEEE, 2010, pp. 1-8.

[10] J. W. Morentz, "Unified incident command and decision support (UICDS): a Department of Homeland Security initiative in information sharing." Technologies for Homeland Security, 2008 IEEE Conference on. IEEE, 2008, pp. 321-326.

[11] H. Gjermundrod et al. "GridStat: A flexible QoS-managed data dissemination framework for the power grid." Power Delivery, IEEE Transactions on 24.1 (2009), pp. 136-143..

[12] G. Pardo-Castellote, "OMG data-distribution service: Architectural overview." Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on. IEEE, 2003, pp. 200-206.

[13] M. Cinque, C. Di Martino and C. Esposito. "On data dissemination for large-scale complex critical infrastructures." Computer Networks 56.4 (2012), pp. 1215-1235.

[14] S. Cherukuwada, "Distributed Object Technology: Security Perspective", SANS Institute white paper, 2002, http://www.sans.org/reading-room/whitepapers/application/distributed-object-technology-security-perspective-17, retrieved on June 1, 2015.

[15] "CORBA Security Service Specifications v1.8", The Object Management Group, 2002.

[16] J. Linn, "Generic Security Service Application Program Interface, Version 2", RFC 2078, 1997, https://tools.ietf.org/html/rfc2078, retrieved on June 13, 2015.

[17] A. Medina et al. "BRITE: An approach to universal topology generation." Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on. IEEE, 2001, pp. 346-353.W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.