Landmark-based self-healing service recovery for distributed pub/sub services on disaster management and M2M systems

C.-S. Shih, H.-Y. Chen & L.-J. Chen
Embedded Systems and Wireless Networking Lab,
Graduate Institute of Networking and Multimedia,
Department of Computer Science and Information Engineering,
National Taiwan University, Taiwan

Abstract

How to exchange information between parties in a disaster management system is one of the fundamental challenges to support timely and efficient disaster response and relief. Specifically, the timeliness, scalability, and availability are three desirable features for information exchange. We call the framework to support information exchange with the three features an Open Information Gateway (OIGY). In this paper, we present the challenges of information gateway and the design of the communication protocols and landmark-based service recovery mechanism to support the aforementioned features. The mechanism aims at recovering the real-time publish and subscription services within an affected region. The developed method allows a landmark service node to monitor pub/sub service within a region, rather than the entire pub/sub network. To evaluate the performance of the service-recover mechanism, we measured its transmission and recovery overhead under different number of message brokers in the network.
Keywords: middleware, message publish and subscription, service recovery.

1 Introduction

Timely disaster response requires the collaboration from many parties including telecommunication service providers, web service providers, general public, rescue agencies, and rescue coordinators. When disaster occurs, how to exchange information among victims, rescuers, and decision makers is one of the most critical challenges. The goal of this work is to design and prototype a self-healing information gateway to enhance responsiveness and availability of information exchange for disaster response and machines-to-machines (M2M) communication.

In the last few decades, many attempts aimed at developing special communication devices and reserving specific communication channels for disaster rescue. Examples include satellite phones [1], IP-based 911 [2] and rescue radio [3]. However, the applicability of these new technologies was founded limited. Take satellite phones as an example. Satellite phones provide location-free communications no matter whether the users are located in mountainous area, metropolitan, or on the sea. It is extremely effective for rescuers in mountainous area and sailors on board. However, due to its high deployment cost, it is not possible to put a satellite phone in every emergency kit. During a typhoon, the sky could be blocked by thick cloud for several days. As a result, the rescue will be delayed until the sky is cleared [4].

The experience in last several disaster rescue efforts show that how to effectively make use of all available communication devices and services is the key to a successful rescue effort. During Haiti earthquake [5], the victims trapped in the damaged buildings sent text messages via their low-ended cell phones (or called feature phones) which allow the rescuers to locate them in the left-behind area and save more than 60 victims. During the 2009 Morakot flooding in Taiwan [4], the destroyed communication infrastructure prevented the victims from contacting their family members and rescue agencies. Fortunately, their family members posted messages on social network services such as plurk and twitter that the victims were not reachable, and marked their possible locations on online maps. The information was broadcast by phones and social network web services. Consequently, the rescue team was able to locate the victims, and provided food and water supply to the victims. This information exchange model was proved to be effective for disaster response. However, such a successful rescue requires both effective coordination and timely intelligent information, which were conducted by experienced rescuers and/or crowd sources.

We call an information exchange framework that are designing for collecting, fusing, and distributing information during disaster management the information system for disaster management, and refer to it as ISDM for short. An ISDM is capable of making diverse, multi-domain data and information generated by independently developed intelligent things and from people less fragmented and more trustworthy, delivering the information to independently developed disaster management applications and services with high availability and on a timely basis, and supporting different usages of the information for disaster preparedness, response and relief purposes and for research and planning in disaster reduction. The system can also adapt to needs, evolve and grow in capabilities with scientific and technological advances and can readily accommodate new information sources, applications and services as needed in response to unforeseen crisis situations.

An effective ISDM replies on many ICT (Information and Communication Technologies) components such as data repositories, fusion of symbiotic information, and information exchange. In this paper, we are interested on real-time information exchange in ISDM. In particular, we focus on how to recover the services when parts of the message delivery services are not available due to the damage to the communication network and computing services. During disaster response, it is very likely that the information is published by various sources. Pro-active data are collected from a collection of pre-installed or quickly-deployed sensor devices, monitoring stations, satellite images, as well as civilian witness reports. Each of the data sources has its individual characteristics of physical properties (e.g. proximity of observation location), temporal properties (e.g. how often data are reported), numerical properties (e.g. sensitivity capability), and even rational properties (e.g. observations under human emotional stress). Reactive data are collected by human including victims, rescue teams, volunteers, etc. An ISDM must be able to select and integrate multiple data sources into a coherent information service. Hence, how to discover and compose information service in an efficient manner is a major challenge for ISDM.

There are two major issues in existing communication systems for ISDM. One is that most, if not all, of the messaging exchange systems rely on static communication services. During the disaster, it could take from few days to few months to restore communication infrastructure. The second is that the communication experts are required to deploy temporary recovery communication services. Hence, in this work, we focus on designing a self-healing messaging exchange systems for disaster management and M2M systems. The remaining of this paper illustrates the design and performance evaluation of messaging services and service recovery mechanism of an ISDM. In Section 2, we illustrate the system architecture of message delivery services for disaster management and related works. Section 3 illustrates the landmark-based service recovery algorithm for real-time publication and subscription services. Section 4 presents the performance evaluation results to illustrate the overhead and service delay caused by the developed algorithm. We summarize the work and discuss the work to be completed in the near future in Section 5.

2 Background and system architecture

2.1 Publish/Subscribe model for message exchange

Publish/Subscribe is a messaging model to support asynchronous and persistent message-oriented communication. In this model, there are three major components. One is publisher, which is the information producer to provide information to the other. The second one is the subscriber, which is the information consumer to receive information from publisher. The third is the middleware, which is responsible for delivering information from publishers to subscribers. In comparison with traditional messaging model, the major

characteristic of publish/subscribe is decoupling. The following are the decoupling properties in three dimensions [6].
• Space decoupling: the publisher does not need to know the address of the subscribers. It is the middleware that deliver the data to corresponding subscribers.
• Time decoupling: the subscriber does not need to be active when the publisher is sending data. If the subscriber is not active, the middleware temporarily stores the data until the subscriber is ready.
• Synchronization decoupling: The publisher and subscriber are not blocked when the message is being delivered. In other words, they are asynchronous. The publisher gives the messages to the middleware, and the middleware will be responsible for routing and buffering the messages.

These decoupling properties make publish/subscribe scalable and flexible. Hence, publish/subscribe is suitable for the disaster management system.

Advanced Message Queue Protocol (AMQP) [7] is an open standard for message oriented middleware (MOM) communication. AMQP grew out of the need for interaction between MOM systems both within, and between, corporate enterprises. Due to the proliferation of proprietary, closed-standard, messaging systems such integration is considered challenging. As such, the primary goal of AMQP is to enable better interoperability between MOM implementations. Since AMQPs inception, several, open-source, messaging software distributions have emerged. The Apache Qpid AMQP distribution is one of the widely used projects. It provides a Broker federation option that can de-centralize the architecture and share the workload among a group of Brokers linked with each other. Qpid also has built-in fault-tolerance features, in which the most critical one is High Availability Messaging Cluster. A cluster is a group of Qpid brokers with the same configurations for exchanges, queues, and other entities. The brokers in the cluster have to synchronize their event with each other. As a result, a cluster is a set of brokers in exactly the same state. With replicated states, the publish/subscribe service does not fail unless all the brokers in the cluster fail. Although cluster involves great amount of synchronization overhead, it is assumed to be sustainable. The reason is that Qpid is originally designed for message delivery in small area such like office building where the network connecting brokers is stable and fast enough. However, the assumption is not suitable for wide area network and disaster management. The other disadvantage is that cluster requires redundant computation resources which are usually not available in disaster management. Hence, we design a self-healing mechanism for Qpid which involves less overhead than cluster approach.

2.2 System architecture of OIGY

To support aforementioned desired features of information exchange services, we proposed to deploy an information exchange service framework, named Open Information Gateway (OIGY), to support distributed timely information exchange over heterogeneous networks. In this section, we present the methodology and advantages of each component in the system and how they interact with each other.

OIGY services will be executed on various types of devices in the network, including computationally weak devices such as cell phones and POS (Point-of-Services) in convenient stores, and computationally powerful devices such as weather forecast service on cloud servers and data repository server in data center. Figure 1 illustrates the system and software architecture of OIGY. As discussed earlier, numbers of individuals, news agencies, government agencies, and rescue agencies form the disaster management system. Each of them can be the providers and consumers of the information. For instance, data center in government agency can subscribe information from sensors in disaster area and publishes the fused/verified data to the subscribers including first responders, victims, news agencies, and general public. Victims in disaster area can also publish messages to the others and can subscribe information from their family members and government agencies. To achieve scalability, OIGY will be deployed as a middleware software component in the network. For victims, an OIGY widget can be installed on their cell phones; for weather forecast agency, an OIGY service can be installed on their powerful servers. In addition, OIGY can also be installed on Point-of-Service devices, which are located in supermarkets and convenient stores to publish and subscribe information.

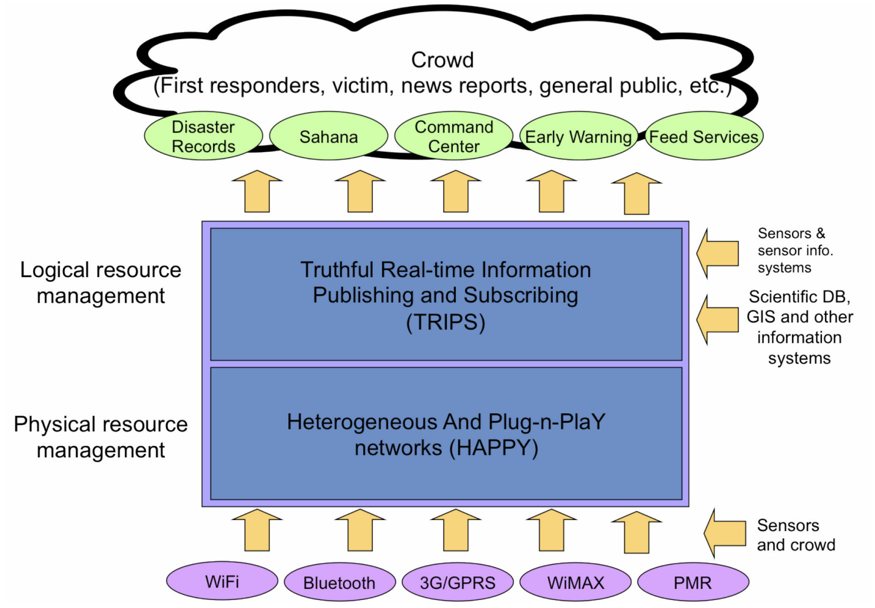
Crowd
(First responders, victim, news reports, general public, etc.)
Disaster Records
Sahana
Command Center
Early Warning
Feed Services
Logical resource management
Truthful Real-time Information Publishing and Subscribing (TRIPS)
Sensors & sensor info. systems
Scientific DB, GIS and other information systems
Physical resource management
Heterogeneous And Plug-n-PlaY networks (HAPPY)
Sensors and crowd
WiFi
Bluetooth
3G/GPRS
WiMAX
PMR

Figure 1: System architecture of Open Information Gateway (OIGY).

Open Information Gateway (OIGY) consists of two major components: one is the distributed Truthful Real-time Information Publishing and Subscribing (TRIPS), and the other one is Heterogeneous And Plug-n-PlaY networks (HAPPY). The objective of HAPPY is to interconnect all network-capable devices using all kinds of possible manners, which is comprised of heterogeneous network access technologies (e.g., WiFi, Bluetooth, Professional Mobile Radio (PMR), and 3G/GPRS) using different approaches (e.g., Infrastructure-based networks, wireless mesh networks, mobile ad hoc networks, and opportunistic networks). The objective of TRIPS is to support distributed real-time publish and subscription (P/S) services. Note that a device/service in the system is not limited to be either information publishers or subscribers. Usually, resource limited device/service act as information publishers only. The sensors to detect mud-slide, to measure rainfall and water level in river, and to monitor earthquake are examples. Most of the other services such as weather forecast services and GIS system will subscribe information from sensors, GIS databases, and other information sources and publish their information to the disaster management system and general public. While requesting for P/S service, the application specifies its Quality of Service or Class of Service of its P/S service including timeliness, description, resource requirements, etc. TRIPS will register and announce the P/S service. Hence, one capability of TRIPS is to manage the declarations, automatically establish connections between publishers and subscribers for matching topics and dynamically detect new status in the system. When the subscribed message arrives or is sent, TRIPS delivers the message to its subscribers.

Figure 2 illustrates the architecture of PubSub network over HAPPY network. In PubSub network, P/S brokers are responsible for storing and forwarding

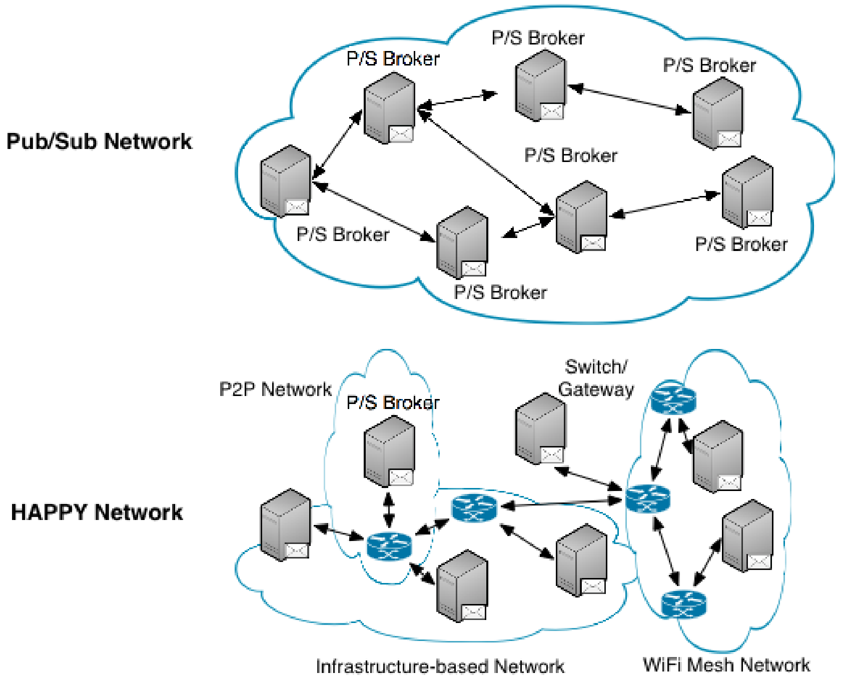

Figure 2: Pub/Sub Network over IP Network.

messages to the brokers and subscribers. The brokers are connected by the HAPPY networks and the messages are routed over the switches and gateways in HAPPY networks. In Qpid network, a link between two P/S brokers could be a route of multiple IP links and a shorter path on P/S networks do not necessarily be a route with shorter message transmission delay. Within HAPPY networks, HAPPY agents will be deployed on the nodes having multiple communication interfaces to bridge the communication between different physical communication networks.

3 Landmark-based service recovery

In the landmark-based service recovery framework, the PubSub network is divided into several disjointed subnets and each subset has a server to management the broker services. Figure 3 illustrates the architecture of the PubSub network within the subnet. Brokers are grouped into subnets according to their network identifications, for instance, combining IP address and port number can provide a unique identification within a subnet. Each subnet has one landmark monitor, at least one backbone node, and several brokers. Landmark monitor is a service process that is reachable from all the broker nodes in the subnet and is responsible for monitoring and recovering the service in case of node failure. The directed lines represent the links and data flow between broker nodes, backbone node, and landmark node. The dashed lines represent the flows of control messages; the solid lines represent the flows of pub/sub messages. Landmark monitor is usually deployed on a reliable node so that it has a very low failure rate or it can be recovered automatically or manually without disrupting its services. During runtime, the landmark listens to the messages sent

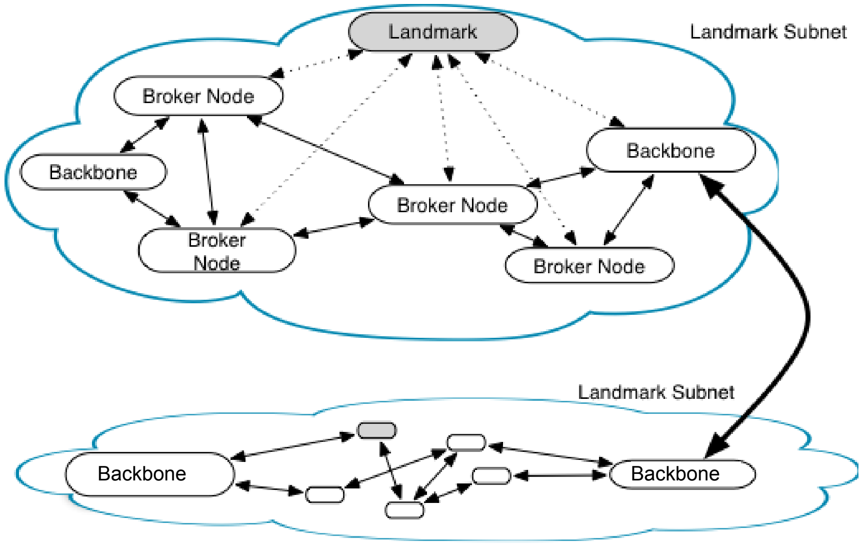

Figure 3: Pub/Sub networks with landmark and backbone nodes.

by each broker node, represented by dash lines in the figure, so as to learn the aliveness of each node. (The monitoring process between landmark monitors and broker nodes will be described later in the section.) In addition to service monitoring, the landmark monitor also duplicate the service objects on each broker node to its local repository. The duplication is conducted when the landmark monitor starts and a new broker service is detected. QPID/AMQP and several other Pub/Sub framework supports this functionality. Backbone node is a service node that is responsible for transmitting messages among subnets. In practice, we may deploy Landmark and Backbone nodes on single physical node for the sake of simplicity.

Landmark-based service recovery takes advantage of the reliability and reachability of landmark monitors to monitor and recover the service network. When a broker service is active and health, every messaging activity conducted by the broker service is transmitted over the PubSub network; change on queue configuration and queue state are forwarded to the landmark monitor by a QMF monitor on broker service. In addition, when a broker service learns that a link connected to or from itself is down, a control message is sent to the landmark monitor to report the failure broker service. Consequently, without actively probing the broker nodes, the landmark monitor can keep track of the activities of the broker services in the subnet and learns if a broker service fails.

When the landmark learns that a broker service is failed, it starts the recovery process. The first step of the recovery process is to update the distance vector table among broker services. The distance vector table keeps tracks of route length, which is represented by average transmission delay or average bandwidth between every pair of broker nodes in the subnet. To avoid injecting great amount of probing traffic, the table is updated when a service recovery process starts. Figure 4 illustrates the process to update distance vector table. To learn route length between nodes, each node executes ping daemon. To update the distance vector table, landmark monitor sends distance update request, i.e., Step 1 in Figure 4, to brokers. The daemon sends and reacts to ping requests from other broker nodes. When the daemon receives a distance update request from landmark monitor, it sends a ping message on application layer to the corresponding broker node, i.e., Step 2 in Figure 4. When the daemon on the corresponding broker node receives a ping request, it echoes a null message back to the sender, represented by Step 3 in Figure 4. After receiving the echo message, the sender calculates the distance based on the remained TTL in the packet header, represented by Step 4 in Figure 4. When the landmark has the remaining TTLs from a pair of broker nodes, it can calculate the number of hops between them and estimate the route length. Then, the landmark selects a node with shortest distance or shortest response time according to the distance vector table. Note that there is no need to update the distance vector table for all pairs of broker nodes in the subnet and only the distance of the routes connected to the neighbors of the failed node require updates. After the candidate node is selected, the second step is to re-create the service on the selected node, according to the local repository of landmark monitor. The landmark monitor also recreates the links to the neighbor nodes of the failed broker node.

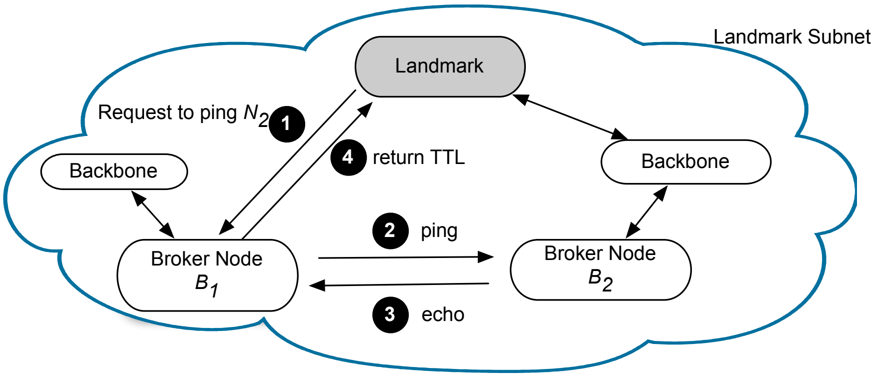

Figure 4: Procedure for ping daemon to collect route distance.

4 Performance evaluation

To evaluate the service recovery algorithms, we conduct extensive experiments to evaluate the network traffic overhead and recover latency for landmark-based service recovery mechanism.

The broker networks consist of 60 nodes, including broker nodes, gateway, and switch nodes, to simulate a broker network within a subnet. Client applications are ignored in this topology. Note that each broker node can support tens or hundreds publishers and subscribers. Hence, the network is designed to represent a network consisting of thousands of computing devices in the network. A subnet in this size represents a network to service the population in a metropolitan area. The simulated network consists of 60 computing devices. Among them, there is one landmark monitor, seven communication gateways, and the others are Qpid brokers.

In the experiments, we measure two metrics: (1) the amount of transmitted data for failure detection and failure recovery, and (2) time delay for failure detection and failure recovery. Figure 5 shows the network traffic on landmark monitor when the number of broker nodes (B_N) are 10, 20, 30, and 40. Note that the figure only shows the network traffic for changes on configuration and state. Pub/Sub messages are not sent to landmark monitor. When the landmark monitor starts at time 0, it queries reachable broker nodes in the subnet and receives great amount of network traffic. The length of initialization phase depends on the number of broker nodes in the subnet. When there are forty broker nodes, it takes less than 200 seconds to learn the configuration of broker nodes. After the initialization phase, the landmark monitor starts to listen to the control messages sent by broker nodes. Note that after the initialization phase, the control messages are sent only when the configuration or state of the broker node changes. When the number of broker nodes increase, the network traffic increase accordingly. The measurement shows that there are at most 400 bytes per second when there are forty broker nodes, which are negligible in bandwidth limited network.

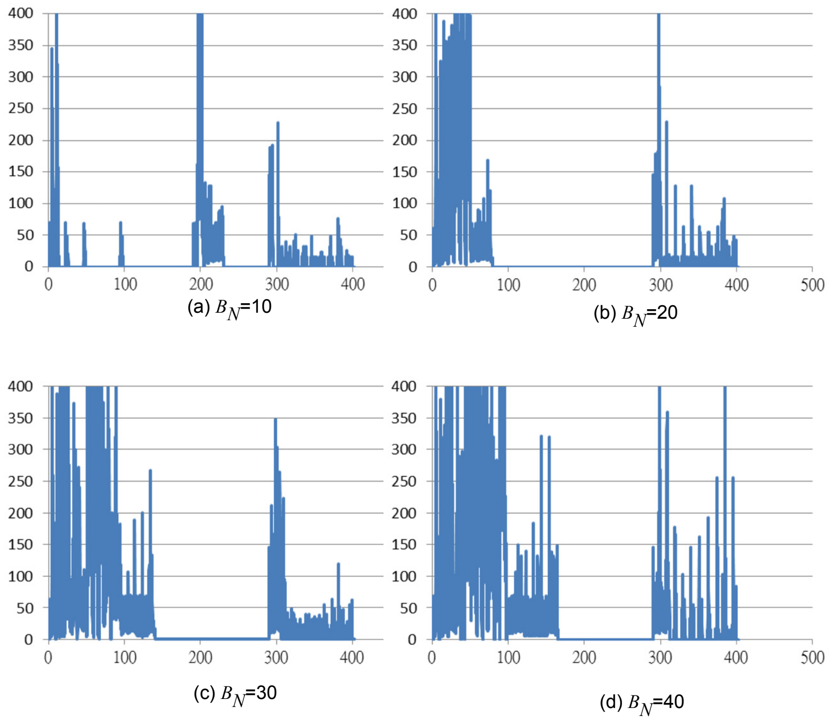

Figure 5: Traffic on landmark monitor.

Table 1 shows the average data rate on landmark monitor when a failure occurs and is detected. To recover a failure, the landmark monitor needs to transmit configuration data to a new node, reconfigure the broker networks, and fetch the messages from other brokers to the new broker node. Hence, the data rate during recovery is much greater than that during monitoring. The data rate increases as the number of brokers increase because the network traffic of updating distance vector table increases when the number of broker nodes increases. Table 2 shows the time overhead for recovering failed brokers. The results show that it takes 1.5 seconds in average to recover a failed service node. The time and network traffic overhead do not increase when the number of broker nodes increase. This is because recovery overhead mainly consists of

Table 1: Average data rate during fault detection and recovery.

Performance Metrics	B_N=10	B_N=20	B_N=30	B_N=40
Average Data Rate (KB/s)	23.97	42.86	68.92	89.49
Standard Deviation	1.67	2.97	3.27	6.68

Table 2: Recovery overhead of landmark-based service recovery.

Performance Metrics	B_N=10	B_N=20	B_N=30	B_N=40
Recovery Delay (s)	1.550	1.593	1.554	1.554
Recovery Traffic (KB)	200.54	321.26	249.07	246.96

transmitting configuration data and reconstructing the new broker service. Additional broker nodes in the subnet do not increase the recovery delay and recover traffic.

5 Summary

It has been shown that robust message exchange is the key to the success of disaster rescue. A self-healing service discovery mechanism can reduce human efforts during the disaster and shorten the blackout time of communication. Although landmark-based service recovery requires on a central server to collect service configuration and monitor the states of broker services, the proposed approach relies the QMF module on broker services to actively report the change on configuration and the monitor module on broker services to reactively report a failed service. As a result, the landmark node will not be overloaded by monitoring workload. The experiment results also confirm the aforementioned properties for a publish/subscribe network in a reasonable size. In the future, we will continue to develop a distributed service recovery mechanism to support large size publish/subscribe networks.

References

[1] Chiu, W.T., Arnold, J., Shih, Y.T., Hsiung, K.H., Chi, H.Y., Chiu, C.H., Tsai, W.C. and Huang, W.C., A Survey of International Urban Search-and-rescue Teams following the Ji Ji Earthquake. Disasters, 26(1), pp. 85–94, 2002.

[2] U.S.D. of Transportation, Next generation 9-1-1. Last accessed at January 2012.

[3] U.S. Coast Guard, Rescue 21. Last accessed at January 2012.

[4] Chanson, H., The Impact of Typhoon Morakot on the Southern Taiwan Coast. Shore & Beach, 78(2), pp. 33–37, 2011.

[5] News services, N., Rescue crews pull 2 more from Haitian market. Last accessed on January 17th, 2010.

[6] Eugster, P.T., Felber, P.A., Guerraoui, R. and Kermarrec, A.-M., The many faces of publish/subscribe.

[7] Vinoski, S., Advanced message queuing protocol. Internet Computing, IEEE, 10(6), pp. 87 –89, 2006.