Abstract

The Internet is presently being used to support increasingly complex interaction models as a result of more and more applications, services and frameworks becoming network centric. Efficient utilization of network and networked-resources is of paramount importance. Network performance gathering is a precursor to any scheme that seeks to provide adaptive routing capabilities for interactions. Furthermore, hardware based performance based routing solutions are hard to manage since they require a knowledge about the underlying hardware structure of the network, as well as access privileges to the network related hardware devices.

In this thesis we present de-centralized, scalable, and hardware independent performance based dynamic routing system which consists of three parts; performance monitoring, aggregation and dynamic routing. We present a network performance aggregation framework that is extensible and appropriate for distributed messaging systems that span multiple realms, disparate communication protocols and support different applications. We propose using publish/subscribe messaging scheme for efficient propagation of monitored performance values in the network system. We present the test results regarding the difference aspects of our system for different network settings.

A Framework for Network Monitoring and Performance Based Routing in Distributed Middleware Systems

by

GURHAN GUNDUZ

B.S., Ege University, Turkey, 1996

M.S., Syracuse University, USA, 1999

DISSERTATION

Submitted in partial fulfillment of requirements for the degree
of Doctoral of Philosophy in Computer and Information Science
in the Graduate School of Syracuse University

December 2006

Approved _____
Professor Geoffrey C. Fox

Date _____

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to my advisor, Professor Geoffrey C. Fox, for his guidance throughout my research and for his wise and acute observations on how to improve my work. I especially want to express my genuine thanks to Dr. Shrideep Pallickara for his advising and help throughout my research.

I gratefully acknowledge Dr. Geoffrey C. Fox, Dr. Shrideep Pallickara, Dr. Ehat Ercanli, Dr. Jim Fawcett, Dr. Dan Pease and Dr. Yildiray Yildirim for serving on my defense committee.

I thank my wife, Melike Gunduz for her patience during the course of my work toward the Ph.D. degree. I also thank my parents, Sait and Sultan Gunduz, for their continuous moral support.

I would like to express my appreciation to many friends for their company, moral support and suggestions during my studies.

CHAPTER 1

Introduction

1.1 Problem Statement - motivation

The predominantly passive interaction model, with interactions initiated by clients and driven by web servers, though still accounting for a large share of the data being routed to clients has found its share reduce considerably. The Internet is presently being used to support increasingly complex interaction models as a result of more and more applications, services and frameworks becoming network centric. The entities, with which applications and services need to interact, span a very wide spectrum that includes desktops, PDAs and other handheld devices, appliances, and other networked resources. Furthermore, supported clients have transient connection semantics and are themselves originators of voluminous content.

Services are usually implemented on a distributed network of routing nodes, with each routing node making decisions regarding the route requests or responses take

en route to their final destination(s). The processing and servicing of requests is itself a distributed problem that involves several nodes and the links that connect them. As the scale of the system increases effective interactions between clients and services, in these settings, is dictated not just by the processing power of the nodes hosting a specific service but also by the network cycles expended during these interactions. Network usage has generally not been metered, at least not in terms of network accesses and the amount of bytes that have been sent over communication channels. Service providers relying on faster processors have ignored premiums associated with network utilizations. Under conditions of high load, visible degradations in response times are attributed to network clogging (a feature out of hands-on control) and it is expected that these response-times would improve either due to an attenuation of network traffic or due to advances (long-term) in networking technology.

Moore's law while ensuring that the nodes in the network fabric would be more and more powerful provides no such solace as far as network usage goes. Bandwidths, though they will improve (at comparatively lower rates), would always be outstripped by demand. Furthermore, depending on the popularity of the service, there is seldom any attenuation in the network traffic directly controlled by the service. Poor solutions to network utilizations lead to buffer overflows, queuing delays, network clogging and other related problems that add up considerably over a period of time. Straitjacketing clients by allowing a fixed set of accesses, preventing certain types of interactions, limiting the type of content that is routed to them or even restricting accesses to a fixed number of clients is not the solution.

Although multicasting and bandwidth reservation protocols such as RSVP [1] and ST-II [2] can help in better utilizing the network, they require support at the router level. It is impossible to re-route traffic using the routers since routers will only have information regarding the local traffic. There needs to be a conceited effort at higher levels to ensure the efficient utilization of networks and networked resources.

One of the motivations for this thesis is the Tango[109] experience in Syracuse University. Tango is a distance education program which sometimes showed poor performance due to network problems. Often there were non-optimal routing choices which resulted in packages being sent on bad performance links as well as over long distances. Even though there were clearly better routing solutions, it was impossible to re-route the traffic as one has no control over the hardware routers. The problem could have been easily solved if there was a high level routing solutions which allow you to easily select the best possible routes without interacting with routers. Further this better routing could be found automatically using the strategy developed in this thesis.

The onus of providing an efficient service that takes network conditions into account rests with the middleware. The middleware thus has to ensure that its routing strategy adapts to changing network conditions, while eschewing decisions that lead to congestion and concomitant problems. To arrive at such intelligent decisions what is needed is the ability to snapshot the state of middleware network fabric, which is a precursor to empowering routing solutions.

Distributed messaging/brokering systems which provide a scalable infrastructure for several applications involve a set of nodes (called brokers)

communicating with each other. Clients connected to any of these brokers can communicate with each other.

Effective utilization of networks and responsiveness to changing network conditions are essential to mitigate network utilization problems associated with distributed messaging systems. This is predicated on four essential issues.

First, there needs to be a monitoring of the communication links hosted at a broker node. This will involve monitoring the links to other brokers and of course to clients connected to the broker in question.

Second, brokers should be able to able to respond to changing local and remote conditions. To respond to changing local conditions, a broker may deploy different transport protocols. For example, if there is a high concentration of clients from a given geographic location, a broker may choose to deploy multicast (if possible) for communications with these nodes. To enable a broker to respond to changing remote conditions there should be a scheme for aggregating network performance from individual broker nodes. These aggregator nodes thus take the snapshot of the state of the network domain within which it aggregates performance.

Third, it should be possible to view the aggregated network performance data and specify constraints to detect network thresholds under which remedial measures might need to be initiated. These measures could involve creation/purging of connections, migration of underlying transports for communications and forcing connected clients to connect to another broker.

Finally, in a truly dynamic system, the detections and responses would be initiated dynamically. Such self healing systems can respond, in real-time, to changing

local and remote network conditions. These systems are responsive to changing client concentrations and also optimize bandwidth utilizations associated with the interactions that they route. However, it should be noted that a truly dynamic system would still need performance monitoring at a broker node. This requires the broker to respond to changing local and remote conditions based on performance accumulated at an aggregator node.

We enumerate issues to be addressed within the distributed middleware pertaining to gathering this network state:

- Ability to measure performance metrics on communication links originating from a node.
- Information should be gathered on each communication link in such a way that the measurements are not intrusive.
- The communication links should not be constrained to work with any specific transport protocol.
- A node should be able to expose all the performance metrics accumulated over all the communication links originating from that node.
- Performance aggregator instances would aggregate performance information from multiple nodes (usually from a certain domain) and thus provide information regarding the state of the domain.
- The entire distributed middleware would possibly span multiple domains and realms controlled by different administrators. Routing within the realm would then be done by metrics aggregated within that realm.

- Ability to integrate information from other network monitoring services is also needed. Different networking monitoring services measure different metrics, which could also be used in arriving at better decisions.
- Aggregated performance information should be amenable to discovering information that would aid routing algorithms.
- The routes should be dynamically updated to minimize the cost.
- Network administrators should be able to specify thresholds on specific or a combination of performance metrics. A reporting scheme would then alert administrators when these thresholds have been reached.

This research was conducted within the context of the NaradaBrokering[20] system. NaradaBrokering is a distributed messaging infrastructure, and enables the efficient routing of messages within the system. NaradaBrokering also incorporates a flexible transport framework and supports a wide variety of transport protocols.

1.2 Research Issues

In this thesis, we investigate the issue of developing a scalable, de-centralized, and hardware independent performance based dynamic routing system, which consists of three parts; performance monitoring, aggregation and dynamic routing. We propose using publish/subscribe schema to dynamically update the costs associated with the links within the network for efficiency, performance and scalability. Publish/subscribe systems[66, 69, 73] have evolved from earlier forms of distributed messaging systems to provide loosely coupled messaging systems to provide richer interactions among communicating entities. (They provide an asynchronous and

scalable communication infrastructure which is suitable for performance based dynamic routing systems since the size of the network system could change from few nodes to several hundreds of nodes.) We use the NaradaBrokering[20] distributed broker network, since it is an open source project and provides scalable architecture and many other additional features like flexible transport mechanism, scalable messaging infrastructure

We propose having a hardware and transport protocol independent scalable performance monitoring system. Here we also investigate the performance issues caused by the performance monitoring due to the extra processing on each node. We also investigate the effect of frequency of the performance measurements in the system.

We also propose aggregation of the monitored network metrics for each link for future analysis and usage. We investigate the types of databases that could be used for this purpose.

Finally, we propose a dynamic routing framework that dynamically updates the costs associated with various links within the network and recalculate the best routes to reach the other nodes in the system. We investigate the issues related to updating these new costs in the system and recalculating the new routes based on these new values.

1.3 Organization of the Thesis

In the next chapter, we evaluate some of the existing performance monitoring systems. We also give an overview of NaradaBrokering[20-28] distributed brokering system which we use to implement our framework.

In the 3rd chapter we give an overview of our proposed architecture of Performance based dynamic routing system. We provide an overview of performance monitoring, aggregation and dynamic routing frameworks.

4th chapter explains the first part of the performance based routing system, which is the Performance monitoring system. Here, we give an overview of performance monitoring framework and explain the components of this framework.

In the 5th chapter we give an overview of the aggregation framework of the performance based dynamic routing system. We explain how the measured network performance metrics are aggregated and stored. We also investigate the question of how the aggregated data could be used for better network performance.

6th chapter presents the dynamic routing part of our system. We explain how the monitored performance of the links are calculated and disseminated in the system to have an up-to-date status of the network and change the routes based on these new costs.

In the 7th chapter, we have conducted several tests to see how our system does in real network environments. We also did tests to prove dynamic routing works in controlled environments. Real world network tests include brokers in countries such as Korea and England beside the United States.

Finally, in chapter 8th, we provide concluding remarks and future directions.

CHAPTER 2

Related Work

2.1 Information About Network Monitoring and Existing Network Monitoring Systems

Almost all of the existing networks monitoring schemes tend to use well known measurement engines like (Ping end-to-end reporting (PingER) [79] , IPERF[80], UDPmon[81] , Traceroute RENO(TRENO)[82]). Each implements a context specific framework for measurements and tends to have a context-specific visualization front end, and all speak different languages. Even though they are successful in the context that they have been created, as a general statement, none of these are designed to be joined up nor can they be used as a core set of low level services to provide information into different higher level views.

A complete end-to-end monitoring framework should have the following;

- Network Monitoring
    - Facilities to monitor network performance
- Data archive
    - Monitoring data to be used to establish a baseline for comparisons and predictions.
- Data analysis and tuning
    - Analyze archived data to for specific purposes
- Data Publication
    - Provide access to necessary data for allowed people
- Troubleshooting and fault tolerance
    - Should be able to detect failures and recover

Current measurement infrastructures use two different methods;

2.1.1 Active Measurement

Create and inject a packet into a network to do the measurement. This is usually used to make 2 point measurements, particularly in relation to response time, loss, bandwidth and availability. Active measurements architecture is usually based on variations of the ping and traceroute programs.

One of the problem with the active measurement is that it could be intrusive, which means that it creates extra traffic on the network. Another one is that the synthetic traffic it generates to make measurements might behave differently than the real user traffic. Systems that use active measurements include:

- Pathrate [84]

- Pchar [83]
- bprobe/cprobe[8,9]
- TRENO
- IPERF
- PingER.

2.1.2 Passive measurement

In passive measurements, existing traffic measurements are recorded and analyzed. These systems then run state machines which perform initial level of filtering to trace the interested traffic and use pattern-matching to find particular events. The data can be collected by interested parties using pull (Self-configuring network monitoring (SNMP)[78]) or push techniques. Passive measurement is highly accurate. However, it may not be able to keep pace if the traffic volume is too high.

Passive measurements are useful for 1-point measurements but not suitable for 2-point measurements due to the difficulty of correlating data collected at two distinct places. The list of few tools that perform this function include; National Internet Measurement Infrastructure (NIMI)[85], Real-time Traffic Flow Measurement system (RTFM)[86] (collects and publishes the SNMP results from the routers.), NetFlow[87], CoralReef[88].

A representative but not complete list of monitoring services in the European Union;

- DataGrid EDG[76] project—site-to-site monitoring and publication to MDS[11] and Relational grid monitoring architecture(R-GMA)[92]

- UK e-science monitoring infrastructure(GridMon)[89]—aggregate traffic statistics available on an ad hoc basis form core providers.
- Geant/TERENA[90] initiatives in core performance monitoring and diagnostic authority chains.

In the United States, efforts related to network monitoring services includes;

- Network Weather Service (NWS)[6,7]
- SLAC based IP provider metrics(IPPM)[77] (subgroup of IETF's Bench Marking Working Group)
- Internet-2 end-2-end performance initiative[75]
- DOE National Collaboratories projects[91]

2.1.3 Network Weather Service(NWS)

The Network Weather Service[6,7] is a distributed system that periodically monitors and dynamically forecasts the performance that various network and computational resources can deliver over a given time interval. The service operates a distributed set of performance sensors (network monitors, CPU monitors, etc.) from which it gathers readings of the instantaneous conditions. It then uses numerical models to generate forecasts of what the conditions will be for a given time frame…

The goal of NWS is to dynamically characterize and provide accurate short-term forecasts of dynamically changing performance characteristics of a network based on historical performance measurement.

There are four different component processes in NWS. Persistent state, name server, sensor and forecaster. These components communicate with other processes through strongly typed messages.

Sensors gather the performance measurements from the resources. NWS network sensor is capable of measuring small-message round trip time, large-message throughput and TCP socket connect-disconnect time. Sensors contact persistent state processes to store the information. At the initialization of the sensor, the location of the persistent state process is specified.

The Persistent State process stores and retrieves measurements from persistent storage. It provides a simple text string storage and retrieval service, and allows each string to be associated with an optional timestamp. Each file is managed by a circle queue, so that the data is not kept indefinitely.

The Name Server process implements a directory capability to bind data and process names with low-level contact information. NWS processes register their name-location bindings with the Name Server.

The Forecaster process predicts the performance of a system during a specified time frame for a specified resource. Forecaster gets the relevant measurement history from a persistent state process. Since the measurements have timestamps associated with them, they can be ordered as a time series. The NWS then uses these time series of performance measurements to generate forecasts of future measurement values.

The current implementation of NWS supports measuring of the fraction of CPU time available for new processes, TCP connection time, end-to-end TCP network latency and bandwidth.

Our framework supports multiple transport protocols unlike NWS which only measures TCP performance. NWS uses strongly typed messages for communication, whereas our system uses publish/subscribe messaging infrastructure for that purpose. Also NWS only monitors the network performance and does not have a framework for dynamic routing.

2.1.4 Self-Configuring Network Monitor Project(SCNM)

There are very few tools to aid application developers to develop distributed applications that effectively utilize the network. Most of the tools are generally accessible by a network engineer and do not provide information regarding the entire network path. This leads to problems of not being able to identify and diagnose problems within the network.

This project aims to provide accurate application-to-application monitoring capabilities throughout the interconnection network domains. The intent is to design a self configuring monitoring system that uses request packets to automatically activate monitoring along the network path between the communication endpoints. It is designed to allow network engineers and application developers to access monitoring data from inside the network.

SCNM is a passive monitoring system. It has a hardware infrastructure, called SCNM monitoring host, which has to be installed between the site border router and the ISP router. Monitoring hosts does not allow sending its own traffic. Therefore host administration and transmission of monitoring results is performed through a separate network connection. Applications send special request packets to automatically activate monitoring along the network path between communication endpoints.

Software components of SCNM are listed below. The first three components run on the application host, the other two components run on the monitoring host.

- Activation packet generator
- Data collection daemon
- Network analysis program
- Packet capture daemon
- Capture configuration/data forwarding module

The main difference between our system and SCNM is that SCNM uses hardware devices to monitor network performance.

2.1.5 Grid Monitoring Architecture(GMA)

GMA[93] is an architecture designed to provide interoperability among various monitoring systems. It consists of three types of components;

- *Directory Service (lookup service):* Keep information about the consumer and producer.
- *Producer:* Makes performance data available, collects them from Monitoring Sevices.
- *Consumer:* Receives performance data. It can archive the information or do real-time analysis.

GMA uses event publication and subscription services to handle performance data. This system does not have monitoring services of its own. It collects performance measurements from other monitoring services.

2.1.6 Monitoring Agents using a Large Integrated Services Architecture(MonALISA)

MonALISA[94] is designed for physics collaborations for monitoring global grid systems. It is based on a globally scalable Dynamic Distributed Services Architecture (DDSA)[95, 96] . It uses JINI/Java[101] and Web Services Definition Language(WSDL)/Simple Object Access Protocol(SOAP)[102, 103] technologies.

It provides the following information in real time:

- General network information
- CPU / memory / disk information
- Historical data and extracted trend information
- Listener subscription / notification

And uses:

- (mobile) Agent filters and alarm triggers
- Algorithms for decision-support

It integrates several existing monitoring tools and procedures to collect parameters describing computational nodes, applications and network performance. It has built-in SNMP support and network-performance monitoring algorithms that enable it to monitor end-to-end network performance as well as the performance and the state of site facilities within the Grid.

In MonALISA, there is a monitoring module, a Java class, to gather data. Data is translated into a format that MonALISA understands. Monitoring agents, deployed on all the MonALISA services, performs ping like measurements using UDP packets.

Measured metrics include round trip time, jitter and loss rate. Measurements are done every 4 seconds.

The reflectors and all peer connections define a graph. The best routing path for reapplication is defined as a Minimum Spanning Tree(MST). This means that the tree that contains all the reflectors (vertices in the graph G) for which the total connection "cost" is minimized should be found.

$$MST = \min(\sum_{(v,u) \subset G} w((v,u)))$$

The cost between two reflectors(w) is evaluated using the UDP measurements from both sides. If the lost packets are detected or the jitter of the RTT is high, the cost function will increase rapidly. In MonALISA the Baruvkas Algorithm[97] is implemented in MonALISA. Momentum factor is added to link when it becomes a part of the MST. This is done to prevent fluctuations.

MonALISA uses SNMP to monitor the network performance which requires special hardware devices to be placed in the network. This is the main difference between our network monitoring system since we do not require any hardware devices to find network performance in the system.

2.1.7 Ganglia

Ganglia[98] is a distributed monitoring system for high performance computing systems such as clusters and Grids. Some of the metrics that it monitors are; CPU, memory, system clock, operating system, etc. Ganglia does not include support for network monitoring.

2.1.8 Peer-to-peer networks

Peer-to-peer networks are decentralized, dynamic and anonymous logical networks for information exchange over Internet. Unlike traditional client/server model in which centralized unit(server) provides information to requesting clients, peer-to-peer applications allow users to share information among themselves. There are different types of peer-to-peer networks. Some of them uses central unit to coordinate the searches like Napster[105], whereas there are those completely decentralized like Gnutella[106].

Peer-to-peer networks also use dynamic routing for better network performance. Skype[107] is one of the well known peer-to-peer application. Skype network is an overlay network. Therefore each Skype client should build and refresh a table, called the host cache, of reachable nodes. There are super nodes and clients in Skype network, so each peer is not same in terms of availability, bandwidth connectivity and CPU power. Super nodes process, distribute and respond to query traffic. It intelligently routes the calls through most effective path possible.(It is not clear what the criteria is for determining the best path). For that reason it keeps multiple connection paths possible and chooses the one that is best suited at the time.

We do not use super nodes in our system for dynamic routing purposes. Each node can monitor its network traffic and determine the best possible routes for its connections.

2.2 NaradaBrokering

We base our investigations, for providing a performance aggregation framework, on our advanced research prototype NaradaBrokering [20-28]. NaradaBrokering is a distributed event brokering system designed to run on a large network of cooperating broker nodes, the smallest unit of this messaging infrastructure, able to intelligently process and route messages while working with multiple underlying communication protocols. Broker nodes can run on separate clients, whether these clients are associated with users or resources. Broker nodes are organized in a cluster-based architecture which allows the system to support large heterogeneous client configurations that scale to arbitrary size. NaradaBrokering provides support for centralized, distributed and peer-to-peer (P2P) [3] models. Communication within NaradaBrokering is asynchronous and the system has been deployed to support audio/video conferencing systems.

The NaradaBrokering transport framework [45] facilitates easy addition of transport protocols for communications between NaradaBrokering nodes. One of the most important elements in the transport framework is the Link primitive, which encapsulates operations between two communications endpoints and abstracts details pertaining to communications and handshakes. Currently TCP, UDP, RTP, Multicast, SSL and HTTP based implementations of the transport framework exist.

Communication within NaradaBrokering is asynchronous and the system can be used to support different interactions by encapsulating them in specialized events. Clients reconnecting after prolonged disconnects, connect to the local broker instead of the remote broker that it was last attached to. This eliminates bandwidth degradations

caused by heavy concentration of clients from disparate geographic locations accessing a certain known remote broker over and over again.

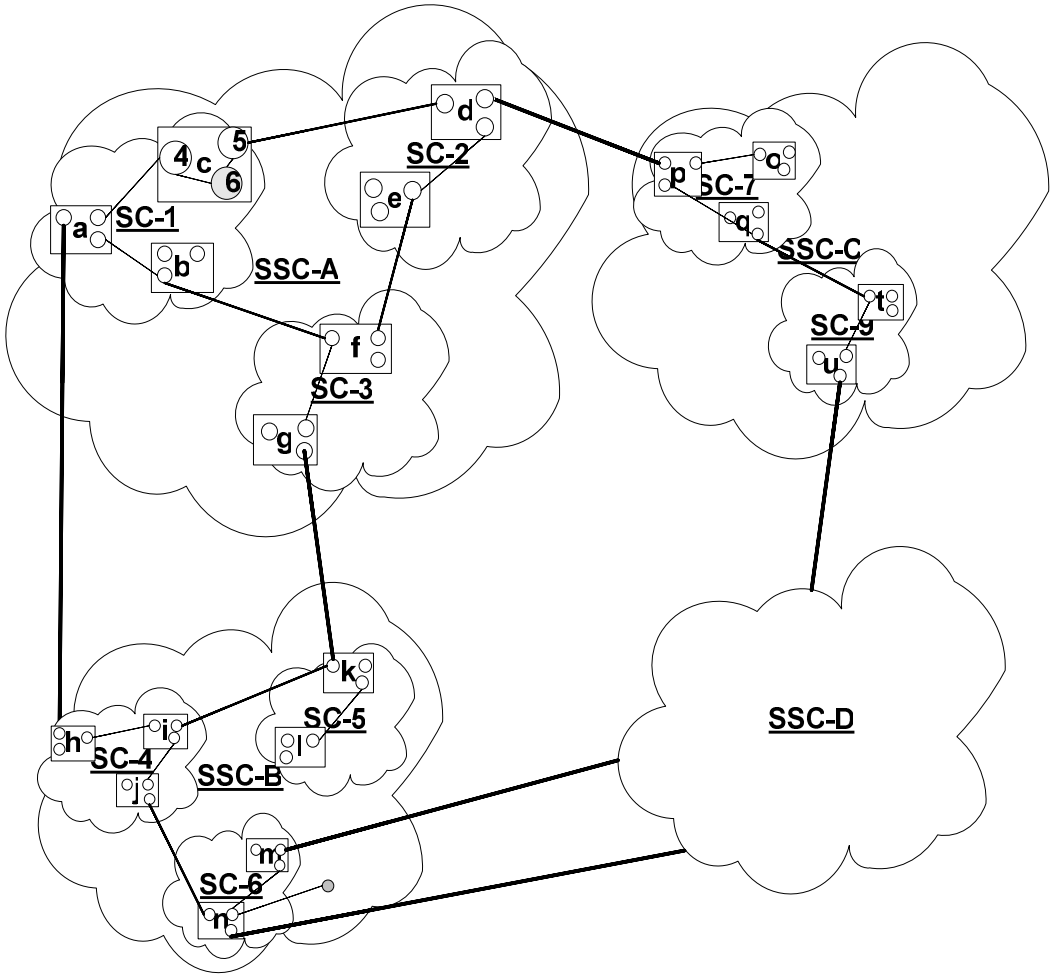

Figure 1: An Example of NaradaBrokering Network

NaradaBrokering goes beyond other operational publish/subscribe systems [34-39] in many ways (support for Java Message Service (JMS) [31], P2P interactions, audio-video conferencing, communication through firewalls among others). Grid Services (including NaradaBrokering) being deployed in the context of Earthquake

Science can be found in [25]. NaradaBrokering supports both JMS and JXTA [32, 33]. In addition to this NaradaBrokering provides support for RTP (A Transport Protocol for Real-Time Applications) [40], which allows it to support audio/video conferencing for RTP clients.

Distributed messaging infrastructures thus must manage communication between external resources, services and clients to achieve the highest possible system performance and reliability. In the distributed NaradaBrokering setting it is expected that when an event traverses an end-to-end *channel* across multiple broker *hops* or *links* the underlying transport protocols deployed for communications would vary. The NaradaBrokering Transport framework aims to abstract the operations that need to be supported for enabling efficient communications between nodes. These include support for:

- Allowing easy addition of transport protocols within the framework.
- Allowing for deployments of specialized links to deal with specific data types.
- Negotiating the best available communication protocol between two nodes
- Allowing for adaptability in communications by responding to changing network conditions.
- Accumulating performance data measured by different underlying protocol implementations.

Operations that need to be supported between two communication endpoints are encapsulated within the “link” primitive in the transport framework. The adaptability in communications is achieved by specifying network constraints and conditions under

which to migrate to another underlying protocol. For e.g. a UDP link may specify that when the loss rates increase substantially communication should revert to TCP. Though there is support for this adaptability in the transport framework, this feature is not yet implemented in the current release. Figure 1 provides an overview of the NaradaBrokering transport framework.

TCP, UDP, Multicast, SSL, HTTP and RTP based implementations of the transport framework are currently available in NaradaBrokering. It is also entirely conceivable that there could be a JXTA link, which will defer communications to the underlying JXTA pipe mechanism. NaradaBrokering can also tunnel through firewalls such as Microsoft's ISA [42] and Checkpoint [43] and proxies such as iPlanet [44]. The user authentication modes supported include Basic, Digest and NTLM. A comprehensive discussion of the NaradaBrokering transport framework can be found in Ref [45].

A "link" is an abstraction that hides details pertaining to communications. A "link" has features, which allow it to specify a change in the underlying communications and the conditions under which to do so. An implementation of the "link" interface can incorporate its own handshaking protocols for setting up communications. The "link" also contains methods, which allow for checking the status of the underlying communication mechanism at specified intervals while reporting communication losses to the relevant error handlers within the transport framework.

For our purposes (registering, transporting and discovering information), we use the term events/messages interchangeably where events are just messages – typically with timestamps. NaradaBrokering provides an interesting test bed for our

investigations since it supports traditional client-server, distributed and peer-to-peer (P2P) [3] interactions. Current research in NaradaBrokering is also focused on deploying the distributed brokering system to federate multiple service realms, which provides Grid based services or traditional Web services. In addition to the reasons enumerated earlier, the performance aggregation infrastructure needs to be in place in NaradaBrokering for reasons itemized below:

1) *State of realms*: NaradaBrokering infrastructure would span multiple realms. Having an aggregation infrastructure in place would allow us to track the state of different realms. The state of the broker network, in a certain realm, provides an excellent indicator of the state of that realm.

2) *Deployment of interactions*: NaradaBrokering supports different types of interactions, some of which are funneled through the broker network and some of which are P2P interactions over the edge of the network. These interactions are fundamentally different, and utilize networks differently. Based on the state of the network reflected in the aggregation framework, decisions can be made regarding the deployment of different types of interactions.

3) *Best available broker*: NaradaBrokering provides users with access transparency, where services hosted on the brokering system can be accessed from any broker. The aggregated information can be used to identify the best available broker node, within a realm, that a client could connect to.

4) *Application specific support*: NaradaBrokering has been deployed to support real time audio/video conferencing and also in distance education based collaboration

systems. Different applications utilize networks and transport protocols in different ways. Knowledge of the state of the network is a precursor to deploying transports efficiently for different applications.

CHAPTER 3

The Problem of Measuring

3.1 Rationale

In order to make necessary performance improvement in the system, performance of the network e.g. bad links, good links should be known. Therefore, performance monitoring is the first step in performance based dynamic routing. The mission of performance monitoring is to monitor network performance and to report the results. For this purpose, we have designed a framework that enables every broker node in the system to conduct network measurements for its links.

Existing monitoring systems mostly works with hardware e.g. routers to get monitored network metrics. Our system ignores the hardware structure of the network. This is useful since there is no need for hardware knowledge to work with our system.

There are some design principles for our performance monitoring system. these principles are as follows;

- It should be hardware independent so that administrators do not have to deal with an underlying hardware structure to accomplish performance monitoring in the network system.
- The performance monitoring framework should be able to work with the existing network structure without making any changes in the system,
- It should be able to measure the performance of the links originating from a node. New links should be included in the monitoring process if desired and the dead links should be removed from it.
- Another principle is that the links that are performing the measurements should not be constrained to work with any specific transport protocol. Our framework is designed in a way that new transport protocols can be included in the performance monitoring without making any changes to the protocol itself.
- Measurements, conducted by the performance monitoring system should be done in a way that it should not cloud the metrics being measured in the first place. For this purpose it should be able to set threshold values to prevent unnecessary monitoring of the links.
- It should also be able to adjust frequency which could cause serious performance problems if it is not set to a right value.
- A node should be able to expose all the performance metrics accumulated over all the communication links originating from that node.

- Another desirable feature of performance monitoring system is being able to use other performance monitoring system's results for dynamic routing purposes.

Considering the principles listed above, we have created a performance monitoring framework that is integrated within every broker node in the system. The framework is hardware independent so that administrators does not have to know about the underlying hardware structure.

It is also designed in a way that it is transport protocol independent New transport protocols can be included in the performance monitoring system easily without making any changes in the transport protocol code. This feature provides scalability to our system by allowing easy addition of new transport protocols.

Performance monitoring framework can also allow changes in the frequency of the measurements by administrators. Frequency of the measurements is important since high frequency measurements could result in performance degradation within the system. More information about the tests regarding this issue is available in section 3.5.5.

The performance monitoring service computes network metrics for a given link after performance measurement is done. These metrics are then wrapped into an XML format for future use purposes. Using an XML format enables compatibility with other systems since other systems can easily understand the structure of these measured metrics and then use them appropriately. Furthermore, their measurements could be converted into our XML format and then used in our system.

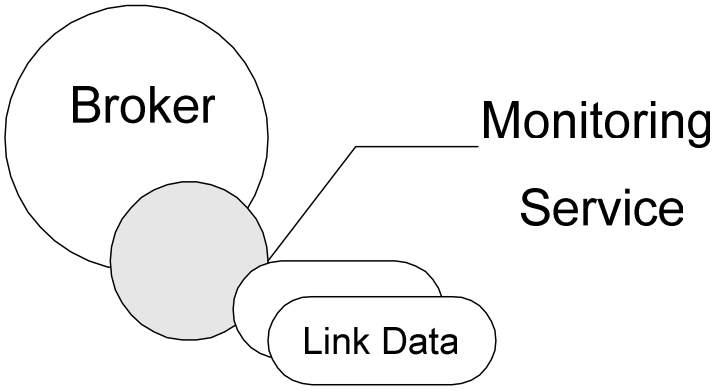

Figure 2: Performance Monitoring Service on Each Node

3.2 Characteristics of the metrics to be measured

Performance measurement system is designed to perform measurements on the links to find the network related metrics. The measured metrics could be used for several purposes.

The type of the metrics that we are measuring is important, since they will help us to solve the problems that we are dealing with. The measured metrics should reveal the state of the network in a way that the application can benefit from it. For example, multimedia applications require the jitter to be low to have a good performance. Therefore, the metrics that the performance monitoring system measures should allow us to calculate the jitter value of a connection.

Administrators can make routing decisions based on the state of the network which could be revealed by the network monitoring system. They can find the links that are bottleneck to the system or they can find the high performance links to be used for demanding applications. Monitoring the network and finding the performance metrics should not be complex and should not introduce high overheads to the system.

3.2.1 Gathering performance metrics over a Link

Measurement of performance factors over a link requires cooperation from the remote end-point of the communication link; this particular detail should be handled within the link implementation itself. How the link implementation computes round trip delays, jitter factors, bandwidth, loss rates etc. should be within the domain of the implementer. The link also has methods which enable/disable the measurement of these performance factors. Links expose the performance related information in the LinkPerformanceData construct using which it is possible to retrieve information (*type*, *value*, *description*) pertaining to the performance factors being measured.

Metrics computed and reported over individual links, originating from a given broker node, include *jitter, latency* (*transit delays)* and *loss rate*. To measure performance metrics over a link, cooperation is needed at both ends of the link. A lot of metrics rely on measurements that require echo behavior from the sink end of the link. For example, the easiest way to measure transit delay over a link is to send a message with a timestamp, and to have this messages echoed back from the sink end of the link. This obviates the need for clock synchronizations and also the need to account for clock drifts. Each node between which a link is established has the option to reset the

intervals at which performance measurements are made. Furthermore, each node also has the option to turn off measurement of metrics over the links.

Factors are measured in a non-intrusive way in order to ensure that the measurements do not further degrade the metrics being measured in the first place. Once a link is deemed to be at the extreme ends of the performance spectrum (either very good or very bad), the measurement of certain factors are turned off while other metrics are measured at a far lower frequency. Each link can measure and report a different set of performance metrics. For e.g. loss rates are an important metric for UDP communications but an insignificant one for TCP. Similarly, depending on the type of application data being routed over a link, the link may be called upon to enable or disable the measurement of certain metrics. The jitter metric, for example, is an important metric for audio/video conferencing applications but is not very relevant in the context of text messaging. The measured factors in our performance monitoring system are explained in the sub-section below.

3.3 The measured metrics and what they capture

Network performance measurement service compute the following metrics; latency, jitter and loss rate.

3.3.1 Latency

Latency is the time that it takes for a packet to travel from source to destination. Round-trip time is the time that it takes for a packet to go to the destination and come back to the sender.

Latency is one of the network metrics that shows the performance of a network link. Having a high latency on the network is a sign of network problems. There are many factors that affect the latency on the networks. Routers are the devices that can create the most latency of any device on the end-to-end path. Packet queuing due to link congestion is one of the main reasons for high latencies on the routers. Satellite links are one of the network technologies that could cause high amount of latencies due to time it takes for a packet to travel across the link. Since the latencies are cumulative, it will increase if the number of the hops increases between the end-points.

In order to find the latency, first step is to find the time that the packet is transmitted. Second step is finding the time that a packet reaches the destination. The last step is subtracting the transmitted time from the second one.

There are some difficulties that exist with the calculation of the latency using this method. One of the main difficulties is the synchronization of the time. As we know, there could be time differences between the devices on the network. Since the transmitted time and the time that a packet reaches the destination on different machines, there will be a time synchronization problem with this method. One of the ways to solve this problem is to have a device which can sync up with a GPS clock to put timestamp in a packet that is sent to measure the latency. The receiving end should have the same device which can compare the time that it receives the packet with the time stamped in that packet by the sender end.

This method is very expensive. Fortunately there is a cost effective method to find the latency with an acceptable accuracy. If the path from sender to receiver and

receiver to sender is same, the round trip latency can be calculated and assumed that the end-to-end latency is half of it.

$$Latency = Round\_Trip \div 2$$

Measuring the round trip latency uses the same device, thus it removes the need of synchronization of time between the sender and the receiver. We use this method to find the latency in our system.

Measuring the latency of one packet is not enough to have an accurate result since the latency can change any time. Therefore, this process is repeated for several packets and the average value is calculated.

Latency changes as the load on the traffic changes. Heavy loads usually cause high latencies. Therefore the measurement of the latency should be done in certain frequencies to have the most up-to-date status of the network.

3.3.2 Jitter

Jitter is the variation in time between the packets arriving. It can also be defined as change in latency of the packets being arrived in time. Jitter can be caused by bad network performance (e.g. network congestion), timing drift or routing changes. Jitter is a very important Quality of Service measurement for audio and video applications.

Audio signals need good network performance for clarity. Especially jitter is very important. High jitter cause audio signals to produce noise and breaks in the signal which prevent audio from being understood clearly. In the case of video, jitter will produce flickering. People can tolerate flickering in the video but can not tolerate noise in the audio. In order to overcome the undesired effects of jitter, real-time

audio/video clients use buffers as one of the solutions. They buffer certain amount of data before sending it.

The benefit of performance based routing can be seen easily in the audio video conferencing systems. The links that has high jitter can easily be found by network performance monitoring. Then the thanks to dynamic routing, these links will be eliminated and routing will be done on low jittered links. Dynamic routing system has a link cost formula. This link cost formula can be updated to favor specific metrics. For example in the case of a audio and video conferencing, the coefficient of jitter can be increased so that high jittered links will have a high cost and will be eliminated during the routing.

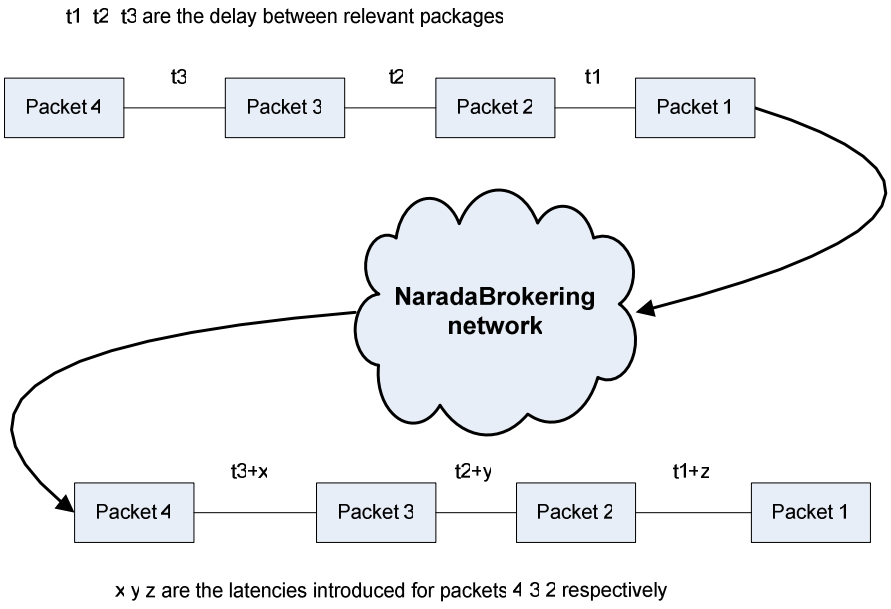

Figure 3 : How Jitter Occurs.

3.3.3 Loss Rate

Loss rate is the ratio of packets that are lost in the network. In other terms, it is the percentage of packets that were supposed to reach to the destination but did not reach there due to various reasons. Some of these reasons could be;

- Packet might get damaged in transit due to poor network conditions.
- The packet might be dropped by the router due to congestion on the network.

The main observable effect of packet loss is poor data throughput performance. With UDP(Unreliable Datagram Protocol), used by many streaming media applications and online games, it is not possible to use any system tools to find whether the UDP packet is lost or not. Only the applications could have knowledge of their network usage statistics. TCP(transmission control protocol) is different then UDP in terms of reliability since it guarantees end-to-end delivery and self healing. It could detect the packet losses and might involve in re-transmitting the packet.

3.4 The NaradaBrokering Transport Framework

In the distributed NaradaBrokering environment, there could be different types of underlying protocols deployed for communications when an event travels on an end-to-end channel across multiple broker hops or links. The transport framework aims to abstract the operations which will enable efficient communication between nodes. These include support for:

- Easy addition of transport protocols within the framework

- Deployment of specialized links for specific data types
- Negotiating the best available communication protocol between two nodes.
- Responding the changing network conditions which will allow adaptability in communications.
- Accumulating performance data measured by different underlying protocol implementations.

There are important components that make up the transport framework. The TransportHandler provides an interface between the protocol and transport layers at a node. The TransportHandler manages all registered LinkFactorys. The LinkFactory is responsible for enabling communications for a specific type and managing the Links created in the process. The Link implementation can monitor and report performance data in a specialized class called LinkPerformaceData. Figure 4 provides a brief overview of the main components in the transport framework. We will explain these main components in detail now.

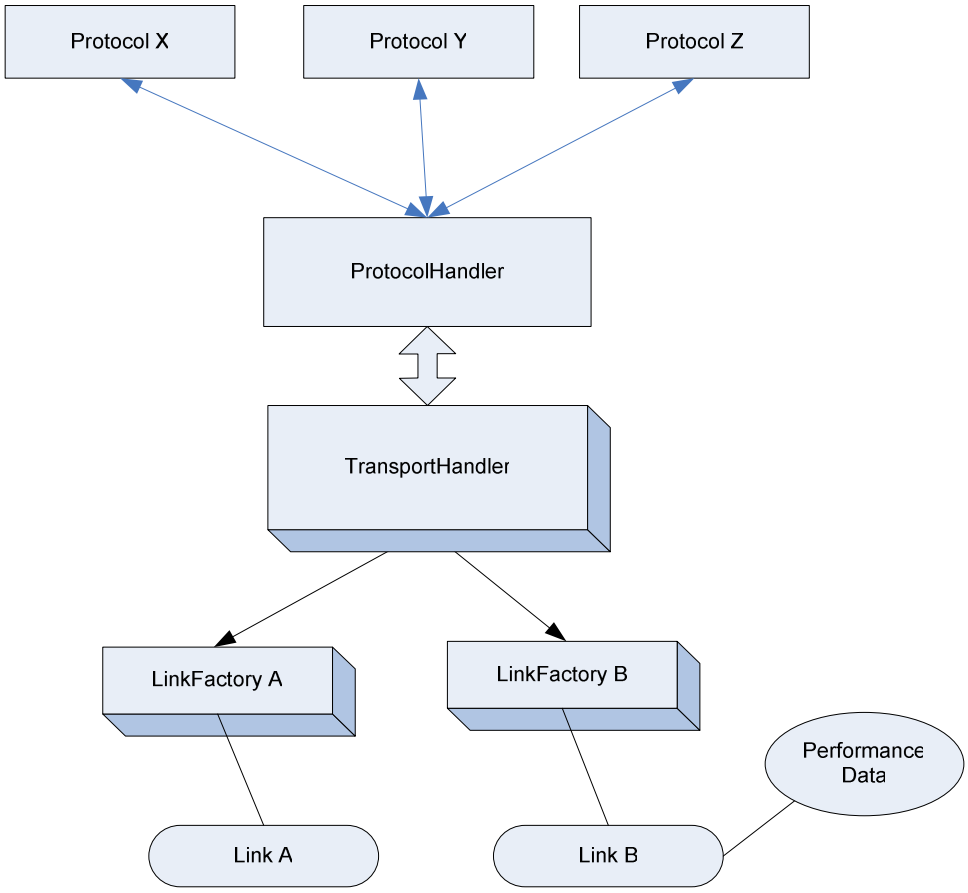

Figure 4 : Transport Framework: Main Components.

3.4.1 Link Structure

Operations that need to be supported between two communication endpoints are encapsulated within the Link primitive in the transport framework. A Link is an abstraction that hides the details regarding communications. An implementation of the Link interface can incorporate its own handshaking protocols for setting up communications. Links can perform the following functions.

- *Security Information:* it can report whether the communication over the link is secure and what type of security/encryption mechanism is used.
- *Failure detection:* It can check the status of the underlying communication mechanism and report the communication losses.
- *Garbage Collection:* It collects the resources associated with the concept of alternate links.
- *Transport protocol migrations:* A link allows the specification of a constraint and the link type that communications migrate to, when the constraint is satisfied.
- *Performance measurements:* Each implementation of the link interface can measure a set of performance factors.

3.4.2 Administrative Link

As explained above, different broker nodes may incorporate support for link implementations with different underlying protocols in distributed NaradaBrokering setting. The problem is that there is a need for information exchange between the nodes before determining the possibility of communications over different protocols for them. This information, which could be used to dynamically load services, can be encapsulated in an URI. This information can be exchanged over an AdministrativeLink. The information that is exchanged over an AdministrativeLink is related to supported protocols such as hosts, ports, multicast group etc.

Communications over AdministrativeLink is generally HTTP based which is the nest way for communications in terms of compatibility.

LinkNegotiators are used by the AdministrativeLink to determine the best available link to create between the nodes. For this purpose, they use the information that is exchanged over the AdministrativeLink.

3.4.3 Link Factory

A LinkFactory is responsible for managing the links that are created in the broker. These links can belong to different communications type. Link factory provides three different capabilities. First, it manages the creation of links (inbound and outbound) from and to other nodes. For example in the case of a NIO TCP communication link, The NioTCPLinkFactory needs to setup a ServerSocket which allows NIO TCP links to be setup based on the socket connections that are enabled by the ServerSocket.accept(). Second, it manages the migration of communications from a different communication protocols. Each implementation of the LinkFactory provides a list of communication types, for which, it can manage the migrations. The last capability is that the LinkFactory can enable or disable failure detection and performance monitoring, and change the measurement intervals associated with these important functions on managed Links.

3.4.4 TransportHandler

The TransportHandler interface is used by protocol layers to invoke methods for communications with other NaradaBrokering nodes. The TransportHandler manages LinkFactorys and Links. A reference to the TransportHandler is passed on to every Link created by LinkFactory.

Each link uses the TransportHandler interface to report data streams, loss of communications and requests to migrate to a different communication protocol. The TransportHandler deals with these notifications within the transport layer and also sends necessary notifications and encapsulated data to the protocol layer.

The TransportHandler also handles the creation of alternate links which is an important feature of NaradaBrokering to enable efficient communications. It is possible for an event routing protocol to specify the creation of alternate communication links while routing events/messages between two NaradaBrokering brokers. For example, an RTP event router, in the protocol layer, could request an RTP based Links for a certain period of time for communication. There is a mechanism that allows you to create a timer for specific links to keep them alive for a certain period of time. After this timer expires the link is destroyed and the necessary garbage collection of all the resources related to link is done.

3.5 Incorporate Support for network Measurements

In order to achieve performance measurements, new framework which will allow performance measurements is needed. There are several ways to achieve performance measurements. Each solution has its pros and cons. Simple approach would be allowing each transport protocol to handle its own measurements. Even though it is simple to design this approach, it is hard to coordinate all the transports if the number of transports increases. The best approach would be designing a framework that will allow easy addition of new transports while coordinating measurements from one center. These approaches are explained in detail below.

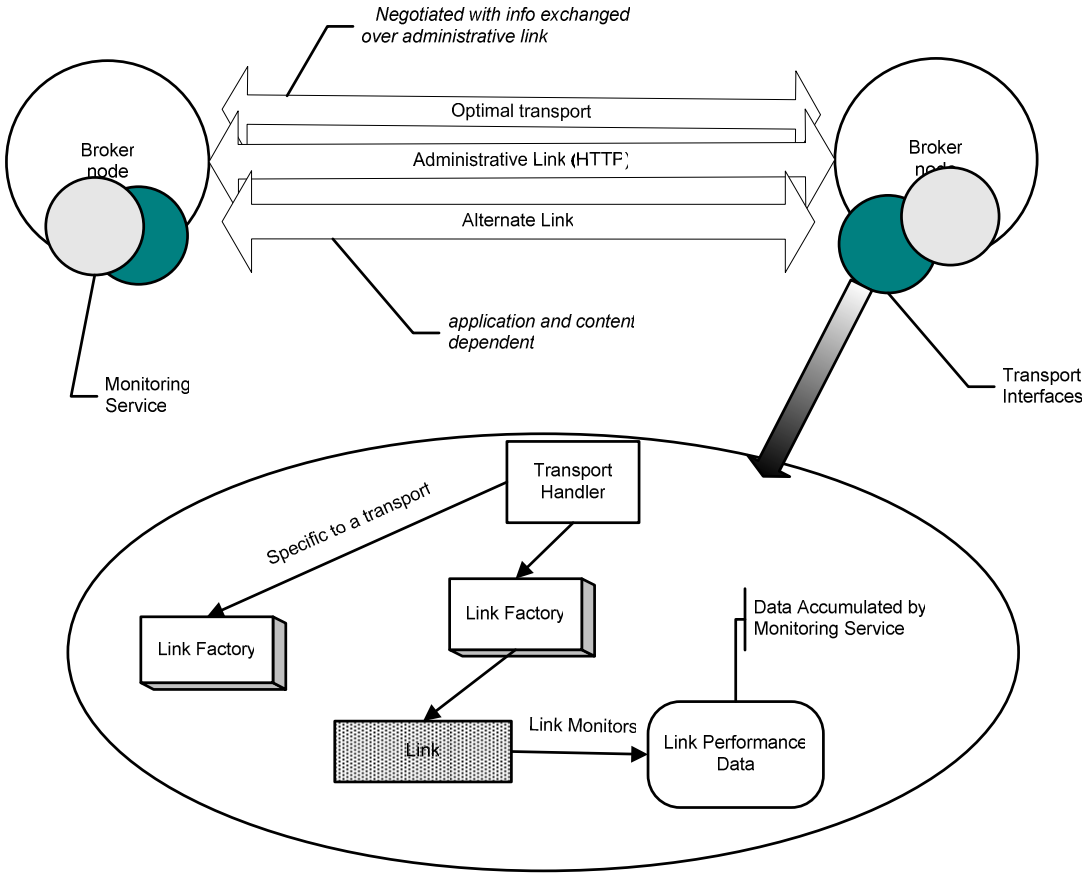

Figure 5 : Transport Framework: The Bigger Picture.

3.5.1 Simple Approach

A network protocol defines a set of rules and conventions for communication between network devices. It includes formatting rules like how data to be sent is packetd into messages. It may also include conventions like data compression or

message acknowledgement to support high performance or reliable network communication.

Working with several transport protocols have advantages. Different types of applications require different protocols. For example, video applications work with UDP protocol. Having a good UDP performance on the link generally indicates that the video applications work well on that link. Dynamic routing could be used to favor UDP performance for video applications. This will ensure that right applications will choose the right link.

The monitoring and aggregation framework is integrated within the distributed NaradaBrokering framework. This integration will allow brokers, individually or collectively, to make decisions on the best possible approach to route packets based on the state of the network fabric and the type of interactions being routed. The performance monitoring scheme within the distributed broker network needs to have two important characteristics. First, it should be able to work with different transport protocols with no straitjacketing of the performance factors being measured. The Link and LinkPerformanceData primitives that abstract transport details and performance data respectively, as outlined in the preceding section, ensure the ability to work with unlimited performance factors over different transport protocols. Different nodes, with different types of links originating from them, can end up measuring a different set of performance factors.

3.5.1.1 Each Transport needs to incorporate its own transport measurements

In today's network environment, there are several transport implementations used together. Many nodes in the distributed systems can support more than one transport protocols. For example a node can have TCP for regular data exchanges and RTP link for video transfers. Therefore, when designing a performance monitoring framework, support for different transports is needed. The simple approach here would be letting each transport to incorporate support for its own transport measurements.

3.5.1.2 Complexity of new transport increases

The problem with this approach is that each transport needs to be worked on to include support for monitoring framework. Whenever there is a new transport, this transport needs to be updated for network monitoring purposes. Detailed information of the transport is needed to make necessary changes. This is not something that is desired in the system.

3.5.1.3 Difficult to coordinate measurements across multiple transports

Another problem with this approach is that since there is no common measurement framework for all of the transports, it is difficult to coordinate the measurements. Whenever a new transport is added into the system, there would be several changes to be done to include this new transport. Its data should also be included in the systems so that it can be used for desired purposes.

3.5.1.4 Design requirements: need to make it easy to perform network measurements

In order to have a good network performance monitoring, there are some requirements. First one is that it should be able to measure all the links that are originating from a node. There could be several links that are originating from a node. The performance monitoring service should be able to measure all of them if necessary. There could be mechanism that lets the user to enable or disable measurements on certain links.

Another requirement is that the performance monitoring should not cloud the network while conducting the measurements. There are several factors affecting the accuracy of the measurements like frequency of the measurements, packet sizes, etc.

Furthermore, performance monitoring should be able to work with different transport protocols to support wide variety of applications.

3.5.2 Our Solution: The Monitored Link

There are several transport protocols that exists today and each has its own characteristics and ways to communicate. Many of these protocols are used in existing network systems. A successful performance monitoring system should be able to monitor most of the transport protocols that exist in the network. Monitoring the network performance requires special operations that are different from regular network operations. For example, you may need special packets to measure different network performance metrics. For this purpose, links have to have some special methods in addition to the methods that it has for its normal operations. Handling these

special performance packets requires operations to be done on both ends of the links. In addition, each transport protocol might require different operations due to their differences. Therefore, all of these issues have to be handled in order to monitor the network performance successfully.

For this purpose we have designed a class that will easily enable links that belong to any transport protocols to monitor network performance. MonitoredLink is the class that we created for the performance monitoring purposes. This class has methods that are necessary to perform network performance monitoring.

One of the design principles of our performance monitoring system is that having an ability to work most of the protocols and having ability to add new transport protocols easily. As we have explained above, this principle is also taken into consideration when designing the MonitoredLink class. Therefore this class is designed in a way that it is transport independent, so that it will be easy to add new transport protocols into the network monitoring system.

MonitoredLink class has methods that will enable/disable performance monitoring, access the measured performance metrics and process the metrics to find desired performance values. It can stop the performance measurement operations on the broker. It also can restart the measurements if desired.

The methods of this class will be used for performance monitoring related operations.

3.5.2.1 Implementations of the link interface

As it is explained in the transport framework of NaradaBrokering, each transport has its own link implementation. There is a Link class which can be

implemented by any transport protocol to have same basic link characteristic. Beside that each transport has its own unique methods to use.

A link that is belong to any transport protocol needs to have methods that are necessary to measure the network performance. Instead of updating its protocol to include these methods, all that needs to be done is to use the methods that the MonitoredLink class has. In order to achieve this, the link implementation of the transport protocol needs to extend MonitoredLink class. This class has methods that will enable the link that belongs to a specific protocol to do network monitoring.

3.5.2.2 Copes with complexity of sending and redirecting the performance related packets

Measuring the network performance requires special operations. There should be a special packet that will allow us to measure necessary elements to calculate the performance metrics. We have created the following performance packet for this purpose.

Our performance packet has the following fields as seen in Figure 6;

- First field is a byte to state that it is a performance packet
- Second field is a byte to state if it is a request or response
- Third field is the timestamp to find the travel time of the packet on the network
- Last field is the sequence number of the packet.

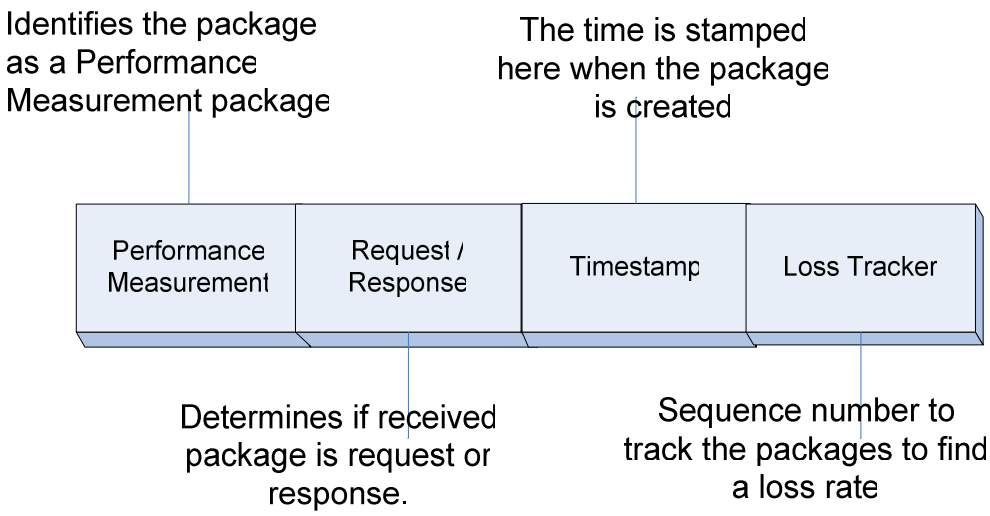

Figure 6 : Performance Measurement Packet

"Performance Measurement" part of the message is a byte that is checked whenever the packets are arrived to destination. According to the result of this check, necessary actions are taken. "Request/Response" byte is checked if it is a performance measurement packet. If it is a request, then response message is sent, otherwise necessary actions are taken to calculate the metrics. "Timestamp" part of the message is used to hold the system time which is recorded at the time of message creation just before sending it. This timestamp is used to calculate the latency when this performance message is returned back to the sender. Return time is calculated when the packet has come back. The half of the difference between the sending time and the returning time is the latency. "Loss Tracker" part is used to find the loss rate in the network. Performance packets are sent to the other end of the link in order to find the required metrics. Some of the other methods are related to accessing to performance data, enabling and disabling performance monitoring either completely or for any of the metrics.

We have designed this framework so that it will be easy to monitor the performance of the network. In addition to that it will allow us to easily add new transport protocols into our performance monitoring system. This is one of the unique features of our system.

3.5.2.3 Methods that are useful for performance measurements

One of the other characteristic that monitoring link will have is the ability to enable and disable performance monitoring. Every broker in the system might have several links.

The following methods are in the monitored link class.

- Enabling or disabling performance monitoring : It enables or disables the performance monitoring on the specified link.
- Enabling or disabling specific factor : This method takes the name of the metrics as a parameter and disables or enables the measurement of it.
- Exposing the specific performance factor or all the factors : this methods takes the name of the factors as a parameter and returns the value of it.
- List the performance factors that are enabled for measurement: It checks all the metrics to see if they are enabled for measurement and lists the enabled ones.
- Process performance factors: it calls the necessary methods based on the values of the metrics.

3.5.3 Measurement initiator

Monitoring the network performance requires special operations. Handling these operations could be very complex if there is no central control unit. Considering these issues, we have created a framework that will control all the network performance monitoring in one center. So the central unit in this framework will handle the following operations;

- Enabling or disabling the performance monitoring.
- Checking the links to find out monitoring links
- Orchestrating and coordinating the measurements of network metrics
- Controls the frequency of measurements.
- Process the measurement results and calculates the link costs.
- Displaying the measured metrics on graphical tables and graphs
- Wrapping the measured metrics into XML and sending them to aggregation points.

In order to manage all the operations that we have mentioned above, we have created a module called “Measurement Initiator”. Measurement initiator starts running when the broker is created. It runs in a sleep mode as a default setting. In this mode, it does nothing and just wait to be enabled. When the performance monitoring is enabled, the first thing it does is to check the links that are created in that broker. The lists of the links are kept in the transportHandler which is a module of NaradaBrokering. Each of these links are then modified to start performance measurements. Each link measures its own metrics. Even though each link measures its own performance metrics, they do it in a controlled manner. What we mean by that is

that they start sending their performance packets at the same time which is controlled by the measurement initiator. This time interval or frequency is set by measurement initiator. This frequency can be updated according to threshold values a explained in section 5.6.4.1.

Measurement initiator also creates a connection to an aggregator node. This connection is then used to send the measured performance metrics to an aggregator node to store it there for future use. The measured performance metrics are wrapped into XML format first before they are being sent to an aggregation node.

When the measurement starts, links that are measuring the performance, start sending performance packets to the other end of the link. When the response is received, it is evaluated in the Measurement initiator module and necessary metrics are calculated from it.

The methods are;

- AddtoConnectionCostTable
- AddtoListofManagedLinks
- CalculateCost
- CheckNewLinks
- DisableMonitoring
- EnableMonitoring
- DisplayMetrics
- GetLinks
- GetNumberWithPrecision
- PublishEvent

- RemovefromManagedLinks
- SetMeasurementIntervals

3.5.3.1 Orchestrates and coordinates the measurement of network metrics at regular intervals

MeasurementInitiator module coordinates the monitoring of the performance on the links which belongs to the broker that it is working on. Considering the number of the links on each broker could be hundreds or even thousands, coordination of the performance monitoring on those links become very important. It will be very costly to deal with each link separately.

Performance monitoring should be done at certain frequencies to have up-to-date status of the links. Performance of a link could change anytime depending on the new packets that two brokers sent to each other. For example, there would not be much traffic if the link is used for text messaging. But it would increase if the users decide to start video conferencing. For this reason measurements should be done in certain intervals. Every link that is measuring the performance, starts sending their performance packets at a frequency controlled by the MeasurementInitiator.

This frequency is set to a specific value by default. It could be changed to an another value by MeasurementInitiator if needed. Adjusting the frequency should be done carefully because it could impact the performance of a network and a broker. If the frequency is too high, performance packets will clog the network. In addition to that, processing the performance packets and updating the costs and routes will cost a lot of processing power.

If the frequency is too low then we will not be able to know the problematic links on time. This will result in unnecessary performance degradation on the network until we change the routes. For these reasons, frequency should be set to a value which will not cause the problems we have mentioned above.

MeasurementInitiator have a thread running on the background. It will activate the processes to measure the performance at links when the corresponding measurement interval has elapsed.

3.5.3.2 Links that measurements initiator deals

There is a list of links that are kept in TransportHandler module. Measurement initiator can access the list of these links and check if they are included in the performance monitoring. After that the measuring links are put into another list. This is done because if there is a link it does not mean that it is monitoring its performance.

Whenever a link dies, it is automatically discarded from the list of measuring links since Measurement initiator check the list of links in TransportHandler before notifying them to send performance packets.

Another point with these performance monitoring enabled links is that they all are MonitoredLink. Which means they are updated so that they have the necessary methods to monitor the performance on them.

3.5.3.3 Necessary elements to compute metrics

Performance monitoring system measures three different metrics. These are latency, jitter and loss rates. In order to compute the network performance metrics, we need some special elements. For example in order to compute latency, timestamp is

required to know the transmission time and the return time of the packet, which can only be found if the receiver sends back a response. To compute loss rate, the sequence number of the packets should be known, and response should be gotten by the sender. Furthermore, old latency values should be known to compute jitter on the links.

To be able compute all the metrics we have mentioned above, we need to use a special packet. We have designed a performance packet to use in our system as seen in Figure 6.

Performance Measurement part of the packet is a byte that is checked whenever the packets are arrived to destination. According to the result of this check, necessary actions are taken. Request/ response byte is checked if it is a performance measurement packet. If it is a request, then response message is sent, otherwise necessary actions are taken to calculate the metrics. Timestamp part of the packet is used to hold the system time which is recorded at the time of message creation just before sending it. This timestamp is used to calculate the latency when this performance packet is returned back to the sender. Return time is calculated when the packet has come back. The half of the difference between the sending time and the returning time is the latency. Loss Tracker part is used to find the loss rate in the network.

3.5.3.4 Generating responses

To compute performance metrics, we need a response from the receiver. So we have to make sure that the receiver sends a response message when it gets the performance packets. In order to do that, the brokers in the network should know that the packet they receive is a performance packet. To achieve this, one of the bytes in the packet is set to a value that states it is a performance packet. When the broker get the

message, it checks to see if it is a performance message. If it is a performance message, then it checks the next field in the performance message. This next field states if this message is a "request" or "reply". "request" means that it is a performance message sent by the broker that is at the other end of the link. "reply" means that it is the performance message sent by itself and replied back by the receiver. In other words it means that it is a response.

In the case of this second byte being request, the receiver generates a response and send it back to the sender as it is explained below.

3.5.3.5 Processing responses

When the response, sent by the receiver, reaches the sender, the time of the packet arrival time is recorded. Performance monitoring framework then starts processing to compute the performance metrics. It will extract the timestamp which is the time that packet was sent initially. It will also extract the sequence number from the packet to find the loss rate. We compute the specific metrics as explained below;

3.5.3.6 Computing latency

Latency is the time that it takes for a packet to travel from source to destination. Round-trip time is the time that it takes for a packet to go to the destination and come back to the sender. So to find the latency we calculate the round trip time for a packet and divide that by 2.

$$Latency = Round\_Trip \div 2$$

In order to find the round trip time of a performance packet, first of all, time when the packet is being sent is stamped into the packet. When this packet is returned back to the sender, the time of the arrival is recorded. Subtracting the start time of packet being sent from the arrival of that packet back to the sender will give us the round trip time.

3.5.3.7 Computing loss rates

Some of the packets that are sent to destination are lost due to some reasons. In order to find the loss rate, we need to know if a performance packet, that we have sent, comes back. In order to recognize the performance packet, we have used a sequence number stamped into the packet. This sequence number is checked when the performance packet has returned back to the sender and compared with the previous packet's sequence number. If they are not in the correct sequence then number of lost packets increased by one. This number is divided by the overall number of performance packets sent on that link. This ratio gives the loss rate on that link. It might be very high at the beginning since the number of packets are low which will result in high ratio.

3.5.3.8 Computing jitter

Jitter is the variation in time between the packets arriving. It can also be defined as change in latency of the packets being arrived in time. Jitter can be caused by bad network performance (e.g. network congestion), timing drift or routing changes. Jitter is a very important Quality of Service measurement for audio and video applications.

$$Jitter = Jitter + ABS((NewLantecy - OldLatency) - Jitter)/16$$

The equation above is the formula to calculate the jitter. In order to calculate the jitter, previous value of the latency should be known. The difference between the latencies is subtracted from the old jitter value and divided by sixteen which is the sample number. This value is then added to the old jitter value. The result of this calculation gives the new jitter value.

3.5.4 Other issues in performance measurements

There are some other issues that need to be dealt with. One of these issues is the abnormal values that appears during the measurements. Due to various reasons, sometimes the measurement values are way above the normal values. These abnormal vales cause incorrect performance values which would affect the overall status of the network.

Another issue is the frequency of the measurements. Measurements are done at regular frequencies to have up-to-date status of the links and the network. This frequency should be adjusted in a way that the performance measurements do not clog the network while having the up-to-date values so that routes in the network can be changed in response to network changes.

3.5.4.1 Coping with spikes in performance measurements

When we tested our performance measurement system, we were faced with some abnormal values. These values were way above the usual values. These abnormal values or spikes occur on the net due to various reasons. For example, due to congestion which might occur only for a very short period of time, network metrics that are measured at that time might be few or more times higher than the normal ones.

This abnormalities increases the average value and standard deviation which results in unrealistic link costs. A single outlier is capable of changing the mean considerably. These values are called outliers, anomalous values, in statistics.

3.5.4.2 How to deal with outliers

In order to get rid of these spikes, which are called outliers in statistics, we have implemented an algorithm. This algorithm clears out the outliers. As an example, consider the following results in your experiments for a certain period of time;

{1, 1, 2, 1, 1, 0, 1, 10, 1, 2}

While most of the results are 1 and 2, there is one result that does not fit into this group, 10. It is obvious that there is some kind of glitch in the system that gave the result 10. If we do not discard this outlier, the mean of these results will be 2. Without the outlier the mean is, 1.1. Mean value with the outlier is almost twice the mean value without the outliers.

The method that we used to get rid of outliers is the Z-TEST. In this method, mean and Standard deviation of the data set is found first. And then find the values that falls more than Z_Value(we use 2 for Z_Value) standard deviations away from the mean of the data set. That means, X is an outlier if

$$\frac{abs(X - mean)}{Std\_Deviation} \rangle Z\_Value$$

3.5.5 Frequency

The traffic on the network is not stable at all. It can change any time from normal to very heavy. There could be several clients using the same routes and there could be several of them added later. These clients might not be using the network at the same level at all times. Some times their applications increase the intensity of network usage. In this case the performance of the link will degrade.

The measurements that we do to find the performance of the links is not a one time measurement. It needs to be done at certain frequencies since the traffic on the network can change any time due to reasons we have stated above.

3.5.5.1 Effect of high frequencies

In order to provide non-intrusive measurement in the network, one of the important factors is the frequency of the measurement. Frequency is important because, it will change the traffic that is caused by the monitoring service. It will also affect the performance of the broker since the measurements done at high frequencies will need more processing power to find the network performance values and to make necessary routing changes in the system. We did some experiments to see how frequency affects the performance of our system. As a result of these experiments we see that the frequency of the measurement can make a big difference on the outcome of the monitoring service. There are some limits that are critical for measurement interval. Under these limits, measurement service will not be able to do the measurement. The limits change from link to link but there are some intervals that are safe for every link we have tried.

3.5.5.2 The test regarding to frequency effect

The following test is done to find out how the frequency of the measurement effects the system. We have increased the frequency from 50 ms to 10000 ms and observed the effects on performance monitoring. The table below shows the link costs associated with the frequency of the measurement.

Frequency (ms)	CGL→ Korea (Link Cost)	CGL→ SanDiego (Link Cost)	CGL→ CGL (Link Cost)
50	4000+	900+	300+
100	2000+	93	20
250	900+	24	1
500	94	23	1
1000	93	23	1
2000	93	23	1
5000	93	23	1
10000	93	23	1

Table 1 :Frequency vs. resulting link cost

If we look at the results, we see that performance monitoring will not be able measure the network metrics correctly after a certain threshold. This threshold differs from broker to broker depending on their location. If the delay between two brokers is higher then the threshold is higher. For example the threshold for CGL to CGL link is 100ms whereas this value is 250ms for the link between the brokers located in CGL and Korea.

Performance monitoring was not able to measure the metrics correctly in all cases, if the frequency is higher than 100ms. The measurements are stopped after a

certain time because the link costs were increasing. That's why there is a "+" after some numbers. It shows that performance monitoring system can not measure for those frequencies.

We also see that, frequencies lower than 1000ms does not effect the measurement. We can say that it is safe to use frequencies lower than 1000ms. But there are also other factors that might effect the measurement. For example, if there are more links then the frequency needs to be lowered. Also, if the traffic on the link is too high then higher frequency will bring overhead to the links capacity.

There are some solutions to lower this overhead that is caused by the performance monitoring. For example, if the link cost does not change for a while then the measurement of that link might be stopped for a certain period of time. This will prevent extra overhead caused by the monitoring system.

3.5.6 Displaying performance measurements

The performance values that are measured by the monitoring system are used to update the routes dynamically in the system. This process is done automatically by the system without any input by the administrators or users. However, administrators or users might want to see the current status of the links in the network while the network monitoring is working. For that reasons we have implemented a graphical representation of performance values of the links in the network as seen in Figure 7 and Figure 8.

Measurement initiator takes care of displaying the live results of performance monitoring. Display module is activated by the measurement initiator. Display module

is also updated with the new results by measurement initiator. Frequency of this update is same as the frequency of performance monitoring.

One of the features of our design is to be able to view monitored network metrics at the time of measurement. This view is specific to each broker and can be viewed on the running broker. The name of the links created on that broker and the value of following metrics are displayed; Average latency, standard deviation of latency, jitter and loss rate. The values on this table are updated on each time the measurement done. There is also "Refresh" button the refresh the values.

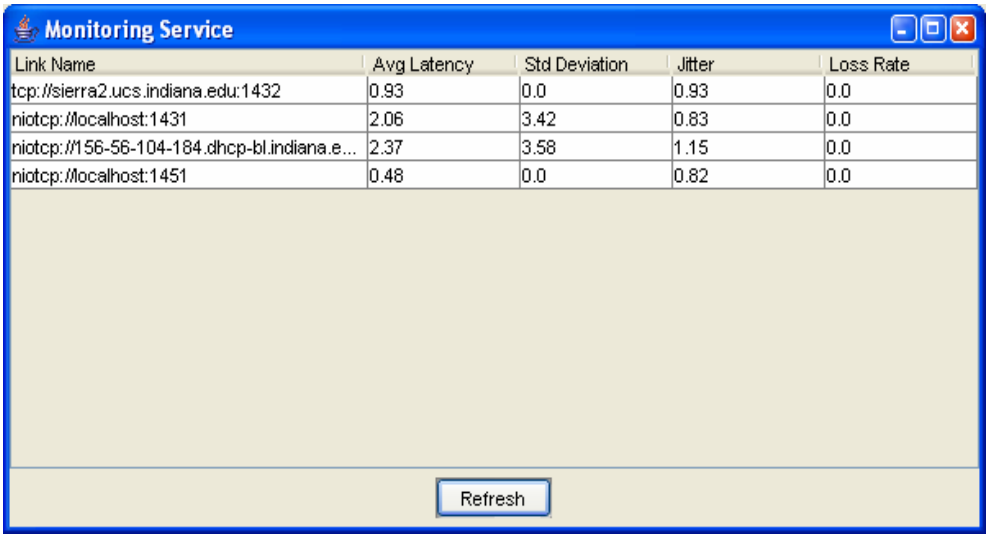

Figure 7: Monitoring Service results on the broker node

There is one another display that views the history of the measured metrics for a certain period of time as a chart graphics. This graphic can be accessed by clicking on any link that is shown on the table. It shows Standard Deviation and mean of the latencies and jitter.

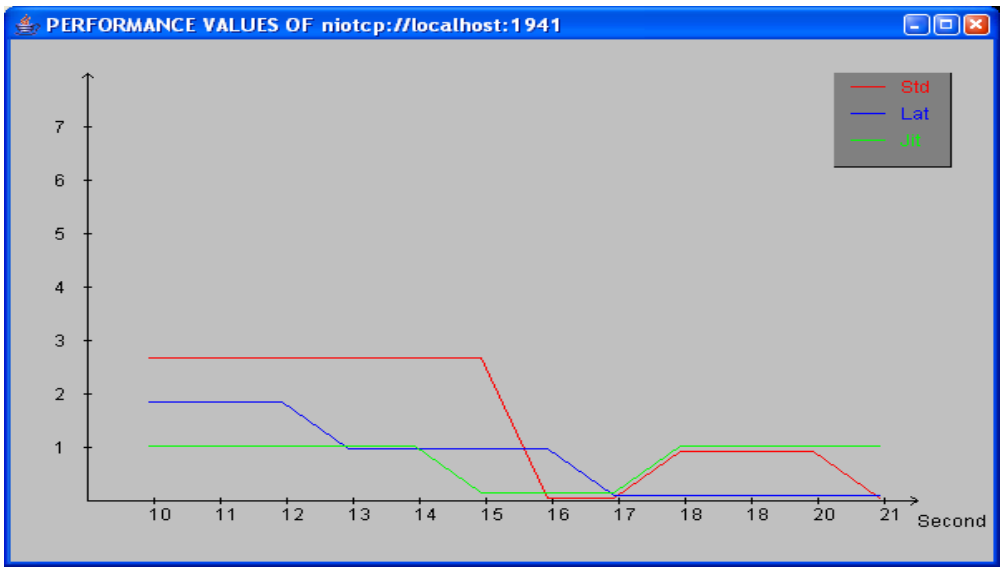

Figure 8 : Graphic display of the link performance metrics

In the performance aggregation framework, every broker in NaradaBrokering incorporates a monitoring service (as shown in Figure 1) that monitors the state of the links originating from the broker node. An implementation of the Link primitive can measure and report performance factors such as jitter, transit delays and loss rates. The Monitoring Service cycles through this list of links at regular intervals to retrieve performance information from each link. The monitoring service then reports these retrieved information to the Aggregation Node. Each Aggregation Node aggregates performance data from monitoring services running at multiple nodes. Each link has a unique ID associated with it which allows us to identify the links being the monitored in the aggregated system.

The monitoring service that runs at every node encapsulates performance data gathered from each link in an XML structure. This performance data is then sent to the Aggregation Node, which stores this data.

3.5.7 Encapsulating performance data

The monitoring service that runs at every node encapsulates performance data gathered from each link in an XML(Extensible Markup Language) structure. XML is a structured document format, in that it represents not only the information to be exchanged, but the metadata encapsulating its meaning, and the structure of the information to be exchanged.

XML is a W3C initiative that allows information an services to be encoded in meaningful structure and semantics that humans and computers can understand. It is great for data exchange.

XML's data encapsulation properties allow us to access relevant fields in the performance data easily. Tags, attributes and element structures provide context information, which can then be used to interpret the meaning of the content which provides intelligent data mining. Also, considering the volume of data that would be aggregated, XML's data description capabilities allow us to mine the data efficiently and effectively.

This encapsulation also enables to deploy sophisticated XPath queries to diagnose network conditions and also to specify thresholds on metrics for administrator notifications. We are also investigating issues pertaining to storing the aggregated performance metrics in a light weight XML database. Knowledge discovery can be achieved by issuing queries to these distributed XML databases.

3.5.8 Federating with other performance monitoring services

Cooperating with other network monitoring services is a plus characteristic for a Network Monitoring Service. What we mean by cooperating is that, a system should be able to use the results of other monitoring services. This usage can be for monitoring purposes. This becomes handy if we do not have an access to some part of the network which is already monitored by other systems. One step ahead of this is using this information for routing purposes.

XML is a widely compatible meta-language. So it is easy to convert information into the format that we use in our system. In order to convert the results of the other system, the only thing that is needed to be done is the have a wrapper that will convert the results into our XML format.

It is easy for other systems to use our monitoring data since the format that the measurement information is encapsulated in XML.

CHAPTER 4

The Problem of Aggregation

4.1 Rationale for performance aggregation

The performance monitoring framework accomplishes the measurement of the network performance metrics. These metrics are used for dynamic routing purposes right after they are computed. Even though real-time usage of this information is sufficient for dynamic routing, it does not give us any idea about the general view of the network system that we are operating in. In order to keep a history of the system or to discover patterns about the performance of the system we need to store measured performance information. For future analysis, we created aggregation nodes. Aggregation nodes are created to hold data from several monitoring services. This data can be used for several other purposes.

This stored information can be mined by network administrators for several purposes like hourly, daily or weekly statistics about the performance of a certain parts of the network system. It is not possible to find a pattern of performance for a certain

link without keeping the previous performance values corresponding to this link for a sufficiently long period of time. In order to do that, we need to have a storage capability in our system. For this purpose we need to use a database to store our performance data. Having the data stored for some period of time will help us to see the status of the network more broadly, while allowing patterns to emerge. There were two types of storage we have considered for our system. One of them is the Flat XML file and the other one is the MySQL database.

The design requirements of the aggregation framework are as follows;

- It should be able to aggregate all the network performance metrics regarding all the links originating from a given node.
- Performance aggregator instances would aggregate performance information from multiple nodes(usually from a certain realm) and thus provide information regarding the state of the fabric in that realm.
- Support for multiple aggregator instances: The entire distributed middleware would possibly span multiple domains and realms controlled by different administrators. It is entirely conceivable that administrators of certain realm would restrict access to performance metrics gathered within its realm. Routing within that realm could be done by the metrics aggregated within that realm
- Assimilate performance data from other services: Ability to integrate information from other network monitoring services is also needed. Different networking monitoring services measure different metrics, which could also be used in arriving at better routing decisions.

- Knowledge discovery: The aggregated performance should be stored so that data mining could be accomplished to discover information that could aid routing algorithms.

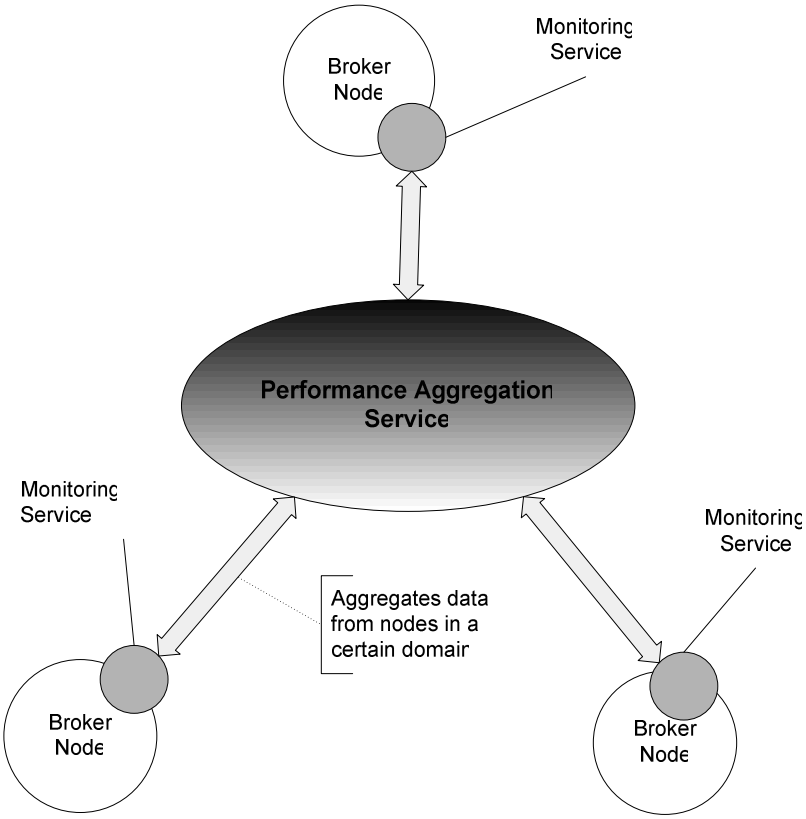

Figure 9: Performance Aggregation Framework

4.1.1 Identifying the bottlenecks using the aggregated data.

One of the reasons that we are storing the network performance data is for analysis of the status of the network. The traffic on the network changes very frequently. These changes could be random or they could have a pattern. For example the traffic on a certain link might be very heavy on certain hours and normal or light at

some other hours. There are several reasons for this changing with patterns traffics. For example for international links, day time and night time traffics are different. Similarly network usage during lunch time is quite different from that in the morning.

In order to find these patterns, or the status of the links that might not have any pattern, we need to have performance data for a certain period of time. This period can vary from hours to days or even weeks depending on the desired statistical results. For example we need at least week-long daily performance results to have some idea about the hourly performance of a certain link. After finding the statistical information about that link, this information can be used to improve the performance of the network. For example if a certain link is very busy during the morning hours, then alternate routing can be found to reduce the traffic on that link. Furthermore, if a link has very bad performance all the time, it can be shut down permanently.

4.1.2 Allows us to have a history of certain parts

One step ahead of the above is to keep a history of the certain links that are most important in the system or that have patterns that are critical for system performance. This history can be used for better tactical changes in the system to improve the performance. For example, if there is a link with a history of bad performance, this link can be destroyed and alternative link can be found or created for the traffic running on that link. Furthermore, the links that have a good performance or low traffic history can be used as an alternative routes in case of a heavy traffic on other links that are connecting the same routes.

4.2 Packaging the computed network metrics

The performance monitoring system calculates the network performance metrics after doing necessary calculations as mentioned in chapter 4. Calculated metrics are used for routing purposes dynamically. After that the MeasurementInitiator module which handles all the performance monitoring related jobs, wrap them into an XML format for future uses. The future uses include;

- Storing the metrics
- Making it available for other performance monitoring systems

We have converted our performance metrics into an XML format for various reasons as explained in the next section. The most important reason for us to use XML is that it is a powerful data description language. The next chapter gives a brief overview about an XML.

4.2.1 Why XML

XML is a markup language for documents containing structured information. XML and HTML both derive from SGML(Standard Generalized Markup Language). SGML is an international standard for document markup. Definition of markup language is that it is a mechanism to identify structures in a document. The XML specification defines a way to add markup to documents. Structured information consists of both content (words, vector graphics, pictures, etc.) and some indication that shows what kind of role this content plays.

XML is different from HTML because in HTML there are fixed tag semantics and tag sets. XML does not specify any tag semantics or tag sets. Instead XML

provides a facility to define tags and the structural relationships between them. Semantics in an XML documents are either be defined by the applications that process them or the style sheets. A DTD(document type definition) is the formal definition of elements, attributes and rules for marking up an XML document. DTDs can be defined either in the XML document or as an external reference. XML is very powerful and very flexible so that it is used for different purposes than just marking up the documents. Some of the areas that XML could be used are described in the sub-section below.

4.2.1.1 To describe the metacontent of documents or online resources.

Metacontent is information about the content of documents. This information could be its title, creation date, file size, keywords and so on. Metacontent could be used to search, filter information and manage the document. To see the usefulness of explicit metacontent, consider the following example;

Suppose that you are looking for documents that are written by Geoffrey Fox. If you use the current web searches you will get thousands of hits. Most of these hits will be the ones that mentions Geoffrey Fox in the bodies of documents, not all of which were written by him. Your search would be more effective if you could express the search query as “find documents whose Author element contains the words Geoffrey Fox.

4.2.1.2 To publish and exchange database contents

Many three-tier applications get data from backend database systems. These results are converted into an HTML format and displayed on the screen. If data is

delivered as an XML documents preserving the original information such as data types, column names, then it can be used by the clients for purposes other than displaying. For example it will be possible to do some computation such as calculating averages.

4.2.1.3 To use as a messaging format for communications between applications.

Messaging is the message exchange between the applications. There are several ways to do messaging, the most commonly used one is electronic data interchange (EDI). EDI has greatly contributed to automating business-to-business transactions. The problem with EDI system is the high cost of building and operating it. The simplicity of XML comes to rescue here. A DTD together with a few example messages should be enough to understand the message format and build the prototype implementation.

XML is a character based format which makes it human readable. XML files can easily be edited by the most common text editors. This makes XML messages much easier to understand and analyze.

4.2.2 Our XML Structure

The elements that we have in our XML structure are the following;

- *Node_ID :* Node is a string element which identifies the broker that Performance measurement system is running on.
- *Link_ID :* It is the ID of the link that the measurement is done for. Link ID is unique in the system.

- *User_group :* This specification is added to bring restriction to data access. There are two types of information in the system. One of them can be accessed by all users and the other one can be accessed by only authorized users.
- *Avg_latency:* It is the average value of latency on the link
- *Std_dev:* Standard deviation of latency values on the link
- *Lossrate:* Rate of message loss on the link.
- *Date:* Date of the measurement.
- *Time:* Time of the measurement. Date and the time of the measurements are kept for statistical purposes. They are necessary for hourly, daily and weekly statistics about the links.

The XML structure that we used in our system is shown below;

```
<?xml version="1.0" encoding="UTF-8" ?>
<performancedata>
        <Node_ID>sierra.ucs.indiana.edu</Node_ID>
        <link_ID>niotcp://sierra.ucs.indiana.edu:9000</link_ID>
        <user_group>A</user_group>
        <avg_latency>2.3</avg_latency>
        <Std_dev>-1.2</Std_dev>
        <jitter>3.5</jitter>
        <lossrate>0.0</lossrate>
        <date>03/12/2006</date>
        <time>03:15pm</time>
</performancedata>
```

4.2.3 Encapsulating network performance data using XML

The monitoring service that runs at every node encapsulates performance data gathered from each link in an XML structure. XML is a structured document format, in that it represents not only the information to be exchanged, but the metadata encapsulating its meaning, and the structure of the information to be exchanged. XML's data encapsulation properties allow us to access relevant fields in the performance data easily. Tags, attributes and element structures provide context information, which can then be used to interpret the meaning of the content which provides intelligent data mining. Also, considering the volume of data that would be aggregated, XML's data description capabilities allow us to mine the data efficiently and effectively.

This encapsulation also enables to deploy sophisticated XPATH queries to diagnose network conditions and also to specify thresholds on metrics for administrator notifications. Knowledge discovery can be achieved by issuing queries to these distributed databases.

4.3 Aggregation of network Metrics

The cost of storing the performance data could be very high if we decide to store each broker's data on that broker. Considering the size of the network, which could be very large (hundreds, even thousands of clients), storage of all performance data at one node can result in performance degradation in the system. In order to

prevent that we use one aggregation point for certain parts, close to each other geographically, of the system. Each broker sends its performance values to this aggregation point if the aggregation option on that broker is enabled. When this data arrives to aggregation point it is stored in database. There could be several aggregation points in the system depending on the size of the network.

4.3.1 Accumulating performance metrics for a Node

Every broker in NaradaBrokering incorporates a monitoring service (as shown in Figure 2) that monitors the state of the links originating from the broker node. The TransportHandler within the NaradaBrokering transport framework maintains a list of all links originating from a node. The Monitoring Service cycles through this list of links at regular intervals to retrieve performance information from each link. Since each link is assigned an ID, the performance data can be associated with specific links.

Every NaradaBrokering node has a tiny web server associated with it; the performance information for the node can be viewed in an HTML file served by this web server. Since this information is constantly changing, refreshing the page inside a web server provides the network state of the node being monitored. The monitoring service running at a node can control the frequency at which individual links measure and report performance data. Since the monitoring service deals with Link instances which abstract transport details, the monitoring service is not constrained by the transports that it measures.

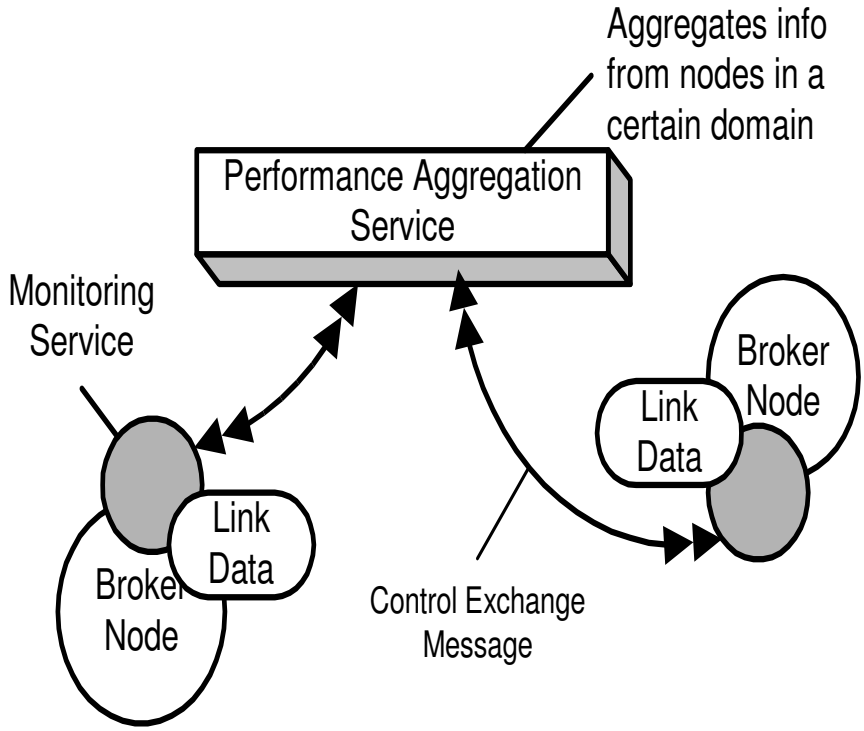

Figure 10: Aggregation Service Overview

The monitoring service at individual broker nodes is akin to sensors in NWS. Each monitoring service instance is configured to report performance data, measured over links originating from a node, to a performance aggregator. The performance aggregator node aggregates information from monitoring services running at multiple nodes. The performance aggregators exchange information with the monitoring services pertaining to the measurement and reporting of performance factors. For example, the aggregator can instruct the monitoring service running at a broker node to stop (or modify the intervals between) the measurement of certain metrics. Similarly, an aggregator may instruct the monitoring service to report only certain performance metrics and that too, only if the factors have varied by the amount (absolute value or a

percentage) specified in its request. The monitoring service at a node can in turn direct links to disable (or vary the intervals for) measurements of certain metrics. Performance aggregators monitor the state of the network fabric at certain realms.

4.3.2 Aggregating performance metrics from multiple nodes

To gather state of the network fabric that spans multiple realms, the aggregators themselves may exchange information with each other. Figure 11 depicts the scenarios where multiple aggregators monitor different realms and also exchange performance information with each other. Since link implementers in NaradaBrokering can measure any metric over their implementations, performance aggregation is not constrained to a specific subset of transport protocols. Furthermore, individual monitoring services can themselves use a variety of transport protocols, to report data to the aggregator. An added feature would be to allow administrators to monitor specific realms or domains.

4.4 Storage

The performance of the links measured by the performance monitoring system is used for dynamic routing purposes. These performance metrics are then stored in the databases for future uses.

4.4.1 The need for storage

The data that is used for dynamic routing purposes is the momentary data. This momentary data does not give any idea about the general performance of the system. What we mean is that we will not have enough data to make findings about the overall

status of the system. For example if we have performance values of a certain part in the system for 24 hours, we can make general comments on the average performance of that part of the system. If we have several days long of performance data of it, we would be able to make comments on the performance at certain times of the day for that part of the system. In order to achieve that, we need to store the performance data. Furthermore, this stored data should be easily accessed any desired time. We have experimented with the storage methods mentioned in the following two sub-sections.

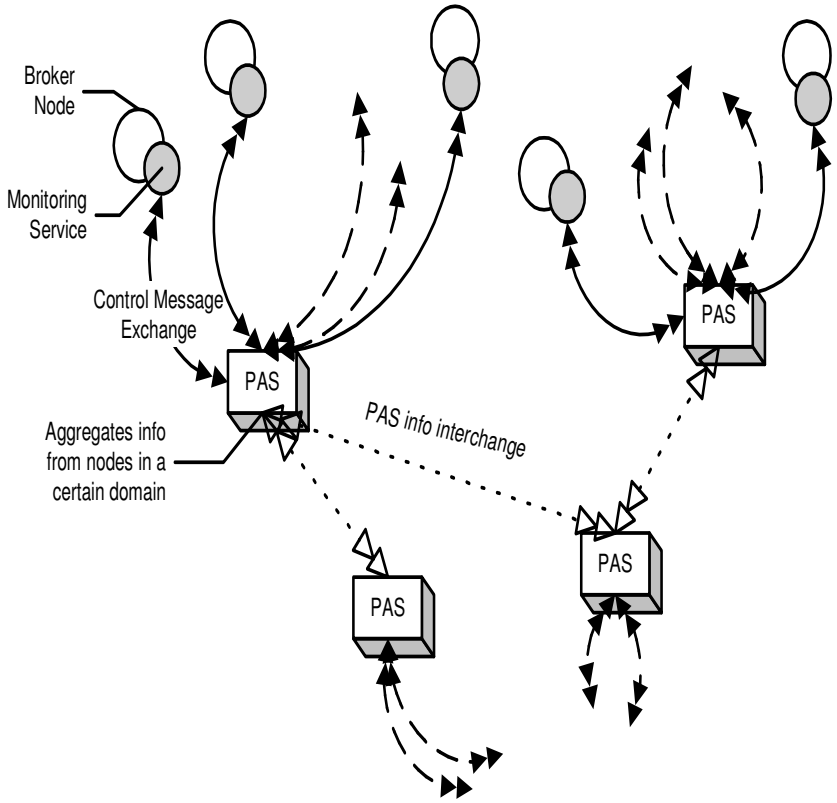

Figure 11: Multiple Performance Aggregators monitoring different domains/realms

4.4.1.1 Flat files

One of them is the Flat file type structure. In this method, data is directly written into the file sequentially. Flat file type of storage does not require any special database software. Data is directly written into the file sequentially and read sequentially from the file. Our performance data is wrapped into an XML format. For that reason, the format of the flat file is XML. The format of the file is listed below;

```
<?xml version="1.0" encoding="UTF-8" ?>
<P_Metrics>
        <performancedata>
           <Node_ID>sierra.ucs.indiana.edu</Node_ID>
            <link_ID>tcp://sierra.ucs.indiana.edu:9000</link_ID>
            <user_group>A</user_group>
            <avg_latency>2.3</avg_latency>
           <Std_dev>-1.2</Std_dev>
            <jitter>3.5</jitter>
            <lossrate>0.0</lossrate>
            <date>03/12/2006</date>
            <time>03:15pm</time>
        </performancedata>
        <performancedata>
                .....
        </performancedata>
        .
        .
</P_Metrics>
```

The access to this file is sequential. Since the file is in an XML format, reading from this file or evaluating the data should be done using XML extracting tools. XPATH[22] is one of the XML querying tool that is effective for finding the desired queries. We use XPATH to search for the necessary information in the XML flat file.

XPATH, a W3C standard, is a language that describes a way to locate specific elements in an XML document. In other words, XPATH is a language for finding information in an XML document. It is born as a result of an effort to provide a common syntax and semantics for functionality shared between XSLT and XPointer. It uses path expressions to select nodes of node-sets in an XML document. For example it lets you write expressions that refer to the XML document's first X element, the fifth child element of the fourth X element, the ID attribute of the third X element whose contents are the string "Syracuse University" and so forth.

It has a library of standard functions which include string manipulation, data and time comparisons, sequence manipulation, numeric value processing, etc. XPATH does not only address the searched information but also matches patterns.

4.4.1.2 Database

Another method is using a database. We chose to use the MySQL database. The reason that we chose it is that it is free and it is fast. Furthermore, it copes well with our system requirements. MySQL is a relational database management system(RDBMS). RDBMS means that it stores data in separate tables rather than putting all data in one big place. Having separate tables brings flexibility and speed.

SQL stands for structured query language, which is the most popular language used in databases. MySQL is a very popular database. It is flexible and scalable. It supports applications that have 1 MB data to terabytes of information.
The storage architecture of MySQL allows different configurations for different applications which results in an increased performance. Thanks to this configurability, it can handle an applications ranging from high speed transactional system to a high volume website that serves billion of queries a day. Queries can be run really fast on MySQL database. There are tables constructed to save our data. This table has the fields that is in our XML document

Flat file database is very easy to use. There is no need for an extra program to run. The disadvantage of flat file storage is that it is very slow when coping with a large amount of data. Maintenance is very easy, however it is not flexible.

4.5 Data Mining

Data mining is the process of searching a large amount of data for patterns. It is also known as knowledge discovery in databases. Data mining can also be defined as the "nontrivial extraction of implicit, previously unknown and potentially useful information from data" [99] and "the science of extracting useful information from large data sets or databases"[100]

4.5.1 Statistical operations on monitored data

There are several statistical operations that could be accomplished on monitored data stored in the database. For example, depending on the time interval of the metrics

that are stored. The average latency of a link for an hour, a day and a week can be found. Another one could be finding the average metrics of a certain transport type that belongs to a specific client. There could be several other statistics that could be extracted from the stored data for several purposes.

Data mining is an important part of the project. Even though it does not do advanced data mining right now, it can be improved to get much more complex statistical information.

4.5.2 Thresholds

There are some performance values that are unacceptable for some systems. Administrators can write scripts that will query the databases at certain frequencies to find threshold values and trigger alerts. These alerts can then be evaluated by the administrators to take necessary actions.

4.5.3 Computing average metrics for a given duration

Computing the average value of a certain metric or all metrics is a relatively easy process in terms of data mining. Depending on the database used, appropriate XPATH queries or MySQL queries can be created to find desired metrics and averages for a certain period of time.

4.6 Viewing the aggregated performance data

There are two different approaches that can be deployed to view aggregated performance data within a portal environment. The first approach would be to use the

XSLT[65] portlet, which comes with the Apache Jetspeed Enterprise Information Portal[46], to access the XML data created by the performance aggregator, convert it into an HTML file using the given XSL[64] style sheet and display them inside the portal. Jetspeed provides for grouping of portlets under a given name. This is useful for providing customized views of the performance metrics. Details pertaining to the viewing of aggregated network performance data inside portals while providing customized user views can be found in Ref [47]. Another approach is to write a customized portlet, which accesses an XML database to retrieve the performance metrics, process the XML data and then display it in a portal.

4.6.1 What is XSL and How is XSL leveraged to display aggregated data

XML tags identify each portion of a document, allowing these fragments to be located, extracted and analyzed. One of the missing part in XML is the representation of data. It does not specify the format to be applied to data in case of representation.

The formatting information needs to be obtained from another source. This information is called a stylesheet. Stylesheets contain a specific set of rules, each one specifying how a specific element should be represented. Some of these specifications include the font, size and style of the text.

Stylesheets must have a syntax specification which is called stylesheet language. XSL is an extensible stylesheet language. Unlike XML, which defines the meaning of each element, XSL defines the appearance of the content of the elements. XSL consists of two parts.

- XSLT A language for transforming XML documents
- XPath – A language for navigating in XML documents

XSLT stands for XSL transformations. It transforms XML documents into XHTML output or into simple text. This transformation is usually done by converting each XML element into an HTML document. XSL is required since XML tags are user defined and therefore browsers do not know how to process each tag.

XSLT can also perform the following operations on XML:

- Add and remove elements
- Rearrange and sort elements
- Add and remove attributes
- Hide or display certain elements
- Find or select specific elements

4.7 Portals

Portals are website or services that offer a different resources and services such as e-mail, search engines, weather information, news, etc. A portal is a web access point. Portals consists of web pages that act as a starting point for using web or web-based services. They are customizable which means each user can select the services that they want in their portal home page. The expected services from a portal platform are the following

- *Easy to use collaboration tools;* communication tools, document management, collaborative spaces, workflow

- *Personalized delivery of desired content;* personal portal, user profiles, subscription to desired contents and alerts.
- *Integration for business applications;* office productivity tools, integration of operation systems, databases and development environments.
- *Search, indexing and categorization of content;* multiple content types, auto categorization and intelligent searches.
- *Centralized management tools;* Flexible deployment options, single sign-on, self management.
- *Security;* Access control list and rights based security.

There are two main portal categories;

- Enterprise information portals
- Content management portals

Enterprise information portals are primarily used to combine a large array of information from different sources onto a single screen. The users of this type of portals are generally consumers which means that they use information published by others, and they do not publish information onto these portals. Consider the following as an example of enterprise information portals;

- Announcements of programs, events etc.
- News, weather information and stock quotes
- Availability of email, meeting schedule tools, calendar and other business tools

- Important reports regarding business that will help to make key decisions

An enterprise information portal can support few people to thousands of people. The presentation of the information stated above is customizable to each user. Yahoo is one of the most well known example of an enterprise information portal.

Content management portals are designed to improve the access and sharing of information. Unlike enterprise information portal, users are allowed to post and share information on content management portals. As an example, consider a development group that consists of several engineers, product managers, quality assurance engineers located at different geographical locations. Each of these people have documents to share and need to access other's information too. So each user need to have the privilege to publish and retrieve documents. Users of a content management portal typically require services like following;

- Security mechanism which will protect the content from unauthorized access
- Version control
- Check-in/check-out capabilities to prevent other users from overwriting content.
- Workflow
- Organizational mechanisms to create a content structure that is easy to browse by portal users

4.7.1 Accumulation and display of data inside portals

Information accumulated within the aggregators could be made accessible to administrators via a portlet residing in a portal such as Apache Jetspeed. A portal is a system that gathers a variety of information sources and services into a single Web page as explained above, while portlets are specialized modules that plug into and run inside a portal.

4.7.2 Advantages of using portals to display information

Portals can display multiple HTML code generated by entities. Portals can also collect content from disparate remote sources such as HTML, XML and images into one page. Besides sophisticated session management, portals also facilitate customized user and group views. The latter feature allows us to restrict access to network performance data by ensuring that different users/groups have access to different features of the aggregated performance metrics. Some entity, either user or group, may thus have access to the entire performance set, while others may have access only to usage patterns or specific metrics or a subset of the aggregated performance data.

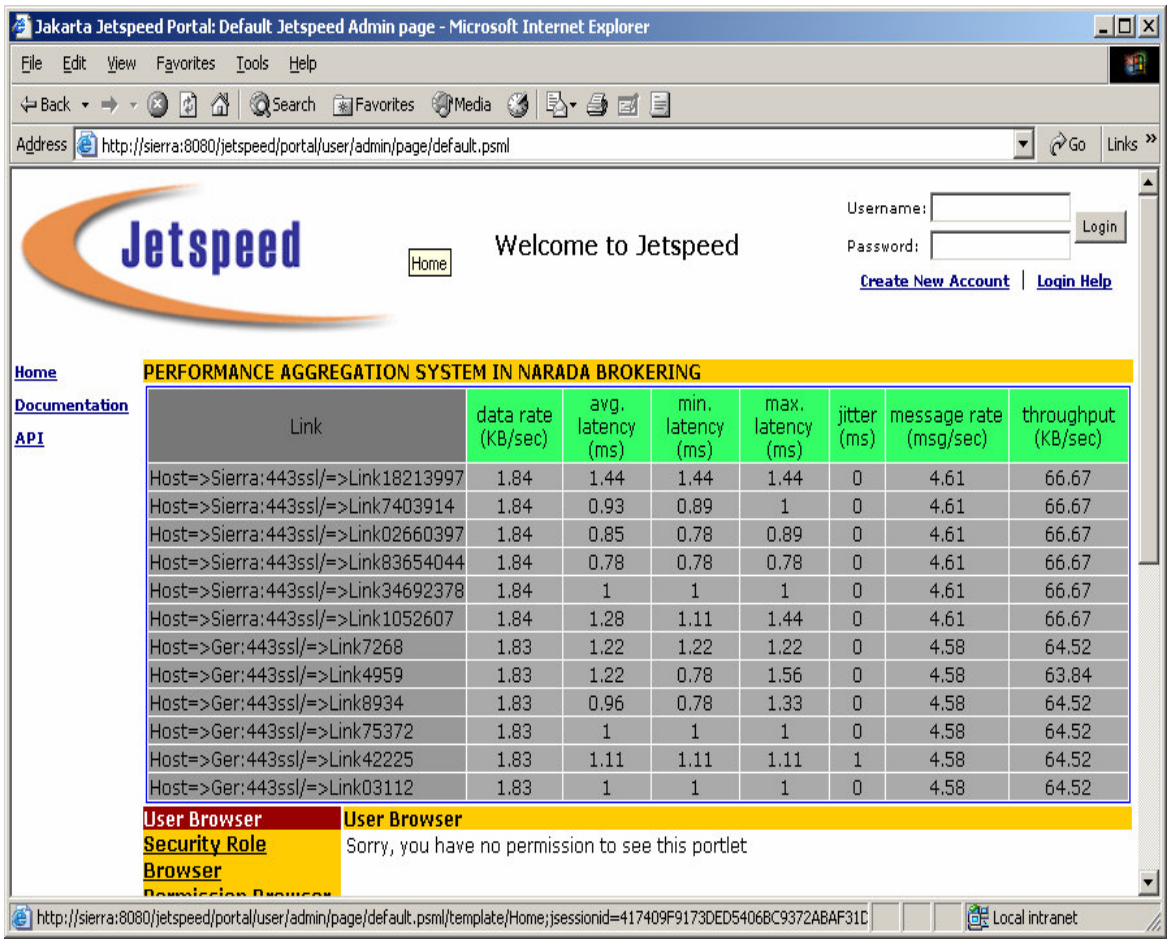

Figure 12 : Screenshot of a portal view

CHAPTER 5

Dynamic Routing

5.1 Rationale

Performance monitoring is integrated into a broker. Each broker measures the performance metrics of its own links. Timely and correct measurements are crucial for correct routing. The link costs in a system should be updated as soon as these measurements are done. Updated costs then should be used to recalculate the shortest routes to reach destinations within the broker network.

Dynamic routing is the part where real-time measured link costs are used to find shortest routes. The design requirements for dynamic routing are the following;

- It should be able to use the result of performance monitoring system to dynamically update routes in the system
- Updating the link costs in the system should be efficient, simple and scalable so that the addition of new links would not require any changes in the system.

- Updating the link costs should be non-intrusive and it should not cloud the network nor should it cause an overhead on the nodes.

5.2 Broker Network Map

Distributed systems consist of several nodes, which could be several thousands or more. NaradaBrokering system that we chose to implement our framework also supports several thousands of brokers. It actually does not bring any size limit so it is a very scalable distributed system. The brokers or nodes in the system might be located in geographically different places. There could be groups of brokers located at the same geographical location which are connected to other brokers that are located at different places by gateways.

In order to address the issues of scaling[10], load balancing and failure resiliency, NaradaBrokering is implemented on a network of cooperating brokers. There is a hierarchical structure on the broker network, in which, a broker is a part of a cluster that is part of a super cluster, which in turn is a part of a super-super cluster and so on. Clusters consist of strongly connected brokers with multiple links to the brokers in other clusters. This ensures alternate communication routes during failures. This organization scheme results in a system where the average communication path lengths between brokers increase logarithmically with geometrics increases in the network size, as opposed to exponential increases in the uncontrolled settings. This architecture allows NaradaBrokering to support large heterogeneous client configurations that scale to arbitrary size. There is at least one unit-controller in every

unit(cluster, super-cluster and so on). Unit controller provides a gateway to nodes on other units. Creation of broker network maps(BNMs) and the detection of network partitions are easily achieved in this technology.

Figure 13 shows an example broker network. SSC-A, SSC-B, SSC-C and SSC-D are super-super clusters, SC-1, SC-2, SC-3, SC-4, SC-5 and SC-6 are super clusters and finally a,b,c,d,e,f,g,l,j,k,l,m, and n are clusters.

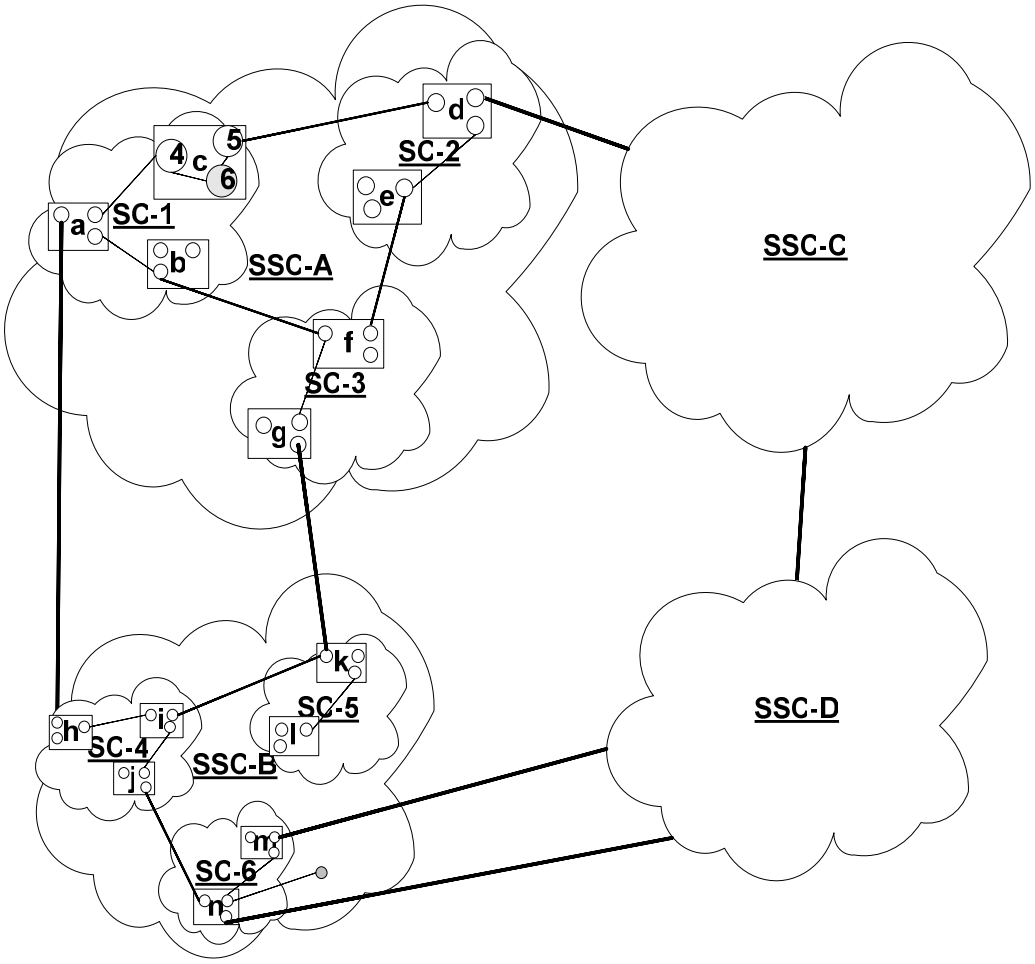

Figure 13 : An Example Broker Network

Brokers in the network system should be able to intelligently route the events. In order to do that, each broker should be aware of the layout and the state of the network. Unfortunately, this will be impractical considering the size of the network could be really large. It will consume a lot of resources and time to update the state of the network for each broker. In order to overcome this bottleneck, an abstract view the broker network is required at each broker.

This view should provide ability to calculation of optimal paths for communication within the system. This information is encapsulated within the Broker Network Map(BNM). BNM provides information regarding the inter-connections between the brokers in the cluster that it is a part of, the inter-connections between the clusters within the super-cluster that it belongs to and so on. Each broker has a different BNM, but they provide a consistent view of the broker network.

In order to provide this locality, changes in the broker network fabric are propagated only to brokers that have their broker network view altered. BNMs at these brokers are updated in response to receipt of information regarding the changes in the network which could be creation or deletion of new connections. (Also updated in case of new links costs that are created by the performance measurement)

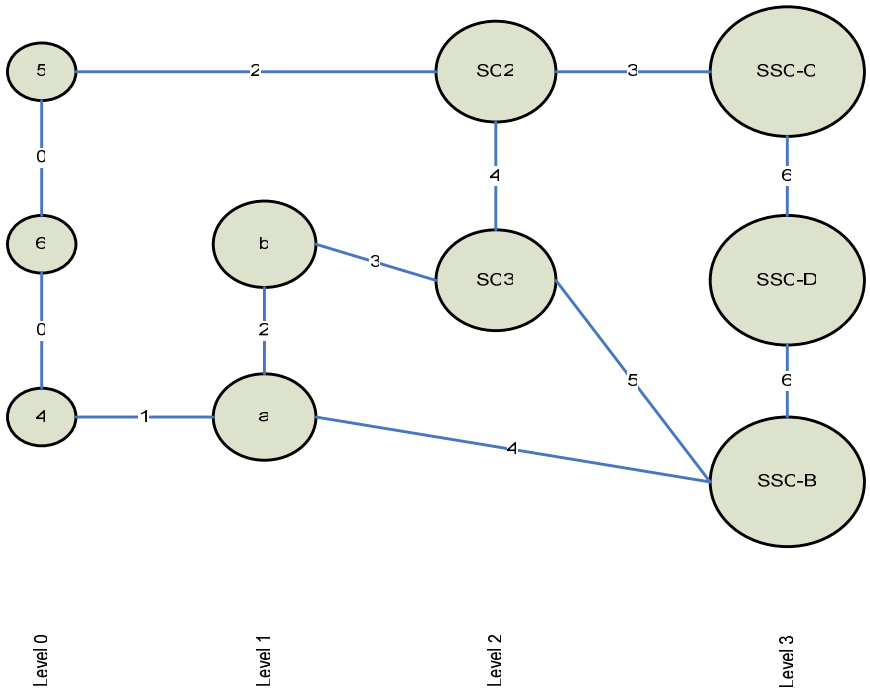

Figure 14 : The Broker Network Map at node 6

Propagation of a connection information is restricted. For example, the connection information between two brokers that are in the same cluster should not be propagated outside of that cluster. Connection information is also modified while it is being propagated through certain sections of the broker network. Considering this rule, the connection between SC-1 and SC-2 in SSC-A is disseminated as one between node 5 and SC-2. When the node 4 receives this information, it is sent over as a connection between the cluster c and SC-2. When the connection between c and SC-2 is sent over the cluster gateway to cluster b, this information is not updated. According to the dissemination rules, the super cluster connection SC-1 and SC-2 information is

propagated only within the super-super cluster SSC-A. Therefore, it is not sent over the super-super cluster gateway available within the cluster g in SC-3 and cluster a and SC-1.

Figure 14 shows the BNM at node 6. As we told before, the BNM hosted at the individual brokers reflect the cost associated with traversal over connections. The cost of the connections are updated based on the costs calculated by the performance monitoring system. The BNM is not only used to compute valid paths but also to compute the shortest path.

5.2.1 Provides snapshot of the broker network

Broker Network Map provides a snapshot of the broker network. It does so by providing information about connections and the costs associated with them in the network. Each BNM is different than other BNMs while providing a consistent view of the system interconnections.

5.2.2 Link Costs

The link cost matrix specifies the cost associated with traversing the link. There are two types of link cost matrix. One of them is the static link cost matrix, which holds the static value for each connection depending on the level of a connection. The other is dynamic link cost matrix which holds the measured costs of connections. It is updated by the dynamic routing system whenever a new value of the cost for a connection arrives.

Level	**0**	**1**	**2**	***Level$_m$***	**Level$_n$**
0	0	1	2	Level$_m$	Level$_n$
1	1	2	3	Level$_m$+1	*Level$_n$+1*
2	2	3	4	Level$_m$+2	*Level$_n$+2*
Level$_m$	Level$_m$	Level$_m$+1	Level$_m$+2	Level$_m$x2	Level$_m$+ Level$_n$
Level$_n$	Level$_n$	Level$_n$+*1*	Level$_n$+*2*	Level$_n$+ Level$_m$	Level$_n$*x2*

Table 2: Link Cost Matrix

The cost of the communication between the units of a cluster is the cheapest if the static link costs are used. Therefore, the cost of communication between the units which are at different levels increases as their levels increase. One of the reasons for implementing this cost scheme is that the dissemination scheme employed by the system is selective about the links employed for finer grain dissemination. In general higher level gateways are busier than the lower level gateways. Table 2 shows the communication costs between the nodes at different levels.

There is a completely different scenario with the dynamic link cost matrix. Levels are not important in this scenario. The link cost matrix is created by the costs that are measured by the performance measurement system.

5.2.3 Destination Paths are computed Based on static values

The smallest unit in the messaging infrastructure should be able to process and route events. This unit is called a broker as it is explained before. The term “server” is not used here to distinguish it clearly from the application servers which would be among the sources/sinks to messages generated within the system.

Efficient organization of brokers is important since it plays important role in routing of events. It is also important in case of failures. Inefficient broker organizations can cause network partitions when the failures occur.

The routing of events to their destinations should be done without flooding the network. Routing decisions and the routes that are created are based on the perceived state of network. Network conditions such as failed and slow links, should be considered while deciding on routes the reach the destinations.

NaradaBrokering is implemented on a network of cooperating brokers to address the issues of scaling, failure resiliency and load balancing. The hierarchical structure of broker network is the following; Each broker is a part of cluster that is a part of super-cluster, which is also a part of super-super-cluster and so on. Clusters consist of strongly connected brokers that are connected to other brokers and different clusters with multiple links. These multiple connections provide alternative routing options in case of a failure. In each unit (cluster, super cluster, super-super cluster and so on) there is at least one unit-controller which provides a gateway to other units.

Event routing is the process of delivering events to relevant clients. This process includes matching the content, computing the destinations and routing the content to its relevant destinations by determining the next broker node that the event must relay to. During the travel of the event in the system via unit controllers, the associated event distribution trace is modified to reflect the event dissemination within the broker network. These routing traces indicate an events dissemination within various parts of the broker network. Routing decisions are made based on this trace

information and the computed destinations. The matching process at a unit-controller computes sub-unit destinations, valid only within that unit.

Figure 15 shows the destination associated with an event in a brokering system which consists of super-super clusters. Individual unit controllers compute the best routes, using the BNMs stored at each node, to reach units contained in the destinations. The sub-unit destinations associated with every event is invalidated when an even arrives at a unit controller. After that it is sent to another unit over the link. This provides broker destinations computed by a cluster controller to be valid only within that cluster and also to be cleared before routing events to another cluster.

Unit controllers analyze the trace information before an event is being sent to another unit to make sure that the event is not routed to the unit where it came from. At every node, best hops to reach destinations are computed. Nodes and links that have not been failure suspected are the only units that can be part of the shortest path. As a result, the best route is taken based on the current state of the network.

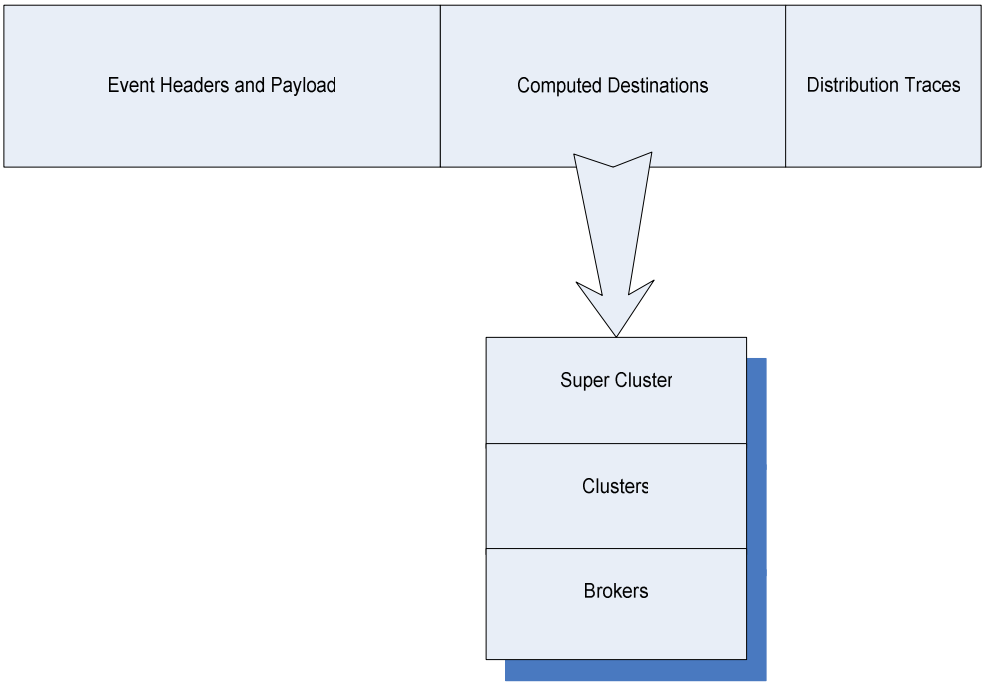

Figure 15 : Event destination and traces

5.3 Making Link Cost Dynamic

As we mentioned above, the cost of the links in the network are statically assigned. They do not reflect the current status of the network. This is an undesired characteristic for a network since there could be two links that have the same static costs but one of them could be way better than the other one in terms of performance. This will result in unwanted routing solutions since both links will have the same priority in terms of routing. The route that chooses the bad link will have a low performance. In order to prevent this, we have to know the current status of the links and choose the routes accordingly. As it was mentioned in chapter 4, we have a framework for network performance measurement. All we need to do is to use this

monitoring service to get the current status of the links. After having the metrics related to links by monitoring service, we need to find a way to associate these results with the links. These values then can be used to find the cost that will be assigned to that link. And after that these costs will be used to find the new routes. Also the condition of the links can change over time, so the status of the network should be checked at regular intervals.

5.4 Computing link costs

In order to have a dynamic routing in the system, the cost of the links need to be updated dynamically. This problem is mentioned above. We know that we can get the performance metrics by using the performance monitoring system. After having the performance metrics by the monitoring service, we need to find a way to calculate the link costs based on these values. There are several factors that can affect the performance of the links in the network. While calculating the cost of the links, all of the performance metrics should be included in this calculation.

LinkCost=*Overall_coeff+Latency_Coeff*Latency+PKT_LOSS_COEF*lossrate+*
*JITTER_COEF*jitter*

Latency → traveling time of a packet from source to destination.

Loss → percent, ranging from 0 to 1 of lost packets sent to the peer

Jitter → sum of variations of latency for a set of samples, divided by the average latency and the number of samples.

There are several applications that work on networks. These applications might have different characteristics. For example there are video applications, text applications, games etc. These applications might require different transport protocols. Beside that these applications might require specific qualifications to be able to run smoothly on the network. For example for sound applications, jitter should be low, but for chat application delay is more important. The formula that we use to find a link cost is a general one so it does not favor any specific performance factor. As a future work, this formula can be updated to favor specific applications. Furthermore, there could be several formulas that could be selected by the user to favor specific applications.

5.5 Publish / Subscribe Systems

Publish/Subscribe systems have evolved from the previous forms of messaging systems. Older distributed systems supported tightly coupled applications and synchronous communication mechanisms such as remote procedure call (RPC), message passing and shared spaces.

RPC is a very powerful technique for constructing distributed, client-server based applications. It extends the notion of local procedure calling so that the called procedure does not need to be in the same address space. They might not even have to be in the same system. The RPC model make distributed programming very easy, thanks to its ability to make the remote interactions appear the same way as local interactions. Participants know each other and have to be active at the time of interaction in this communication method.

Message passing is another way of communication between participants. In this method, participants communicate by sending and receiving messages. It is one of the oldest communication methods. It is rarely used today since physical addressing and data marshalling become visible to the application layer. Both participants have to be active at the same time and the sender knows the recipient.

Another communication method is the use of shared spaces in which synchronization and communication between participants take place through operations on shared data. In this interaction model, participants do not have to be active at the same time and they do not know each other. But the communication is synchronous.

The internet has changed the need of distributed systems considerably. This need has increased the scale of distributed systems to involve thousands of entities, potentially distributed all over the world, whose location and behavior may vary throughout the life time of the system. The applications running on these systems may be dynamic and decoupled. This nature of applications requires more flexible communication models and systems. Old fashioned point-to-point and synchronous communications can lead to inflexible and static applications. In addition, it is very difficult to develop dynamic large scale applications using these communication methods. In order to reduce the complexity of developing applications, the relation between the different entities in large scale settings should be provided by a middleware infrastructure.

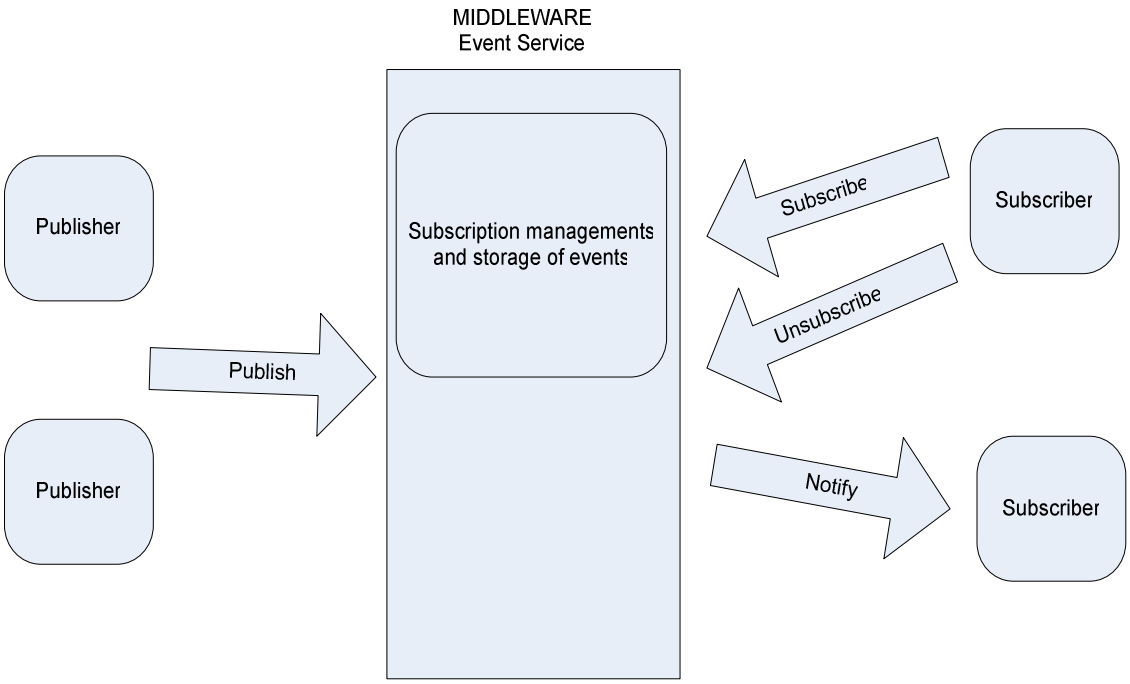

Figure 16: Classic publish/Subscribe system

Publish/subscribe messaging systems provide loosely coupled form of interaction required in such large scale settings. Subscribers can express their interest in an event or a pattern of events. They are notified when a publisher generates an event that matches their interest. There could be more than one subscriber registered their interest in the same event. Events are propagated asynchronously to all subscribers which registered an interest in that published event. In other terms, producers publish information on a software bus (an event manager) and consumers subscribe to the information they want to receive from that software bus. This information is called event, and delivering it is called notification. The bandwidth requirements are not as high as polling, since clients do not return any messages.

Publish/subscribe systems provides a messaging middleware that decouples producers and consumers based on time, space and synchronization.[108, 66, 67, 68, 69]. Decoupling of producers and consumers in time means that consumers do not have to consume messages when they are produced. The messaging middleware is able to deliver these messages later. For example, subscriber might be disconnected at the time when the publisher publishes an event. The decoupling of producers and consumers in space means that producers and consumers do not have to know each other. Producers produce only one copy of a message and the messaging middleware delivers them to all subscribers. This brings the advantage of scalability. Being a loosely coupled system and multicast capability allows greater scalability. And lastly, the decoupling of producers and consumers on synchronization means that producers and consumers are not blocked when sending or receiving messages. The generation and consumption of events do not happen synchronously. Producers are not blocked while producing events and subscribers can get asynchronously notified of the occurrence of an event while performing other activities.

Decoupling the producers and consumers increases the scalability by removing the dependencies between interacting participants, publishers and subscribers. Removing the dependencies between participants will reduce the coordination thus making the resulting communication infrastructure well adapted to distributed systems.

As we have mentioned before, subscribers are interested in specific events or event patterns. There are different ways of specifying events or patterns which leads to several subscription schemes. The mostly used variations of publish/subscribe systems are;

- Topic based publish/subscribe systems
- Content-based publish/subscribe systems
- Type-based publish/subscribe systems

In a publish/subscribe model, server publishes a message on a certain topic rather than addressing a specific client, then the clients who registered for this topic get this message. Multiple clients can subscribe to a same message and get the published message. Delivery of the message is done by the messaging infrastructure. It is a loosely coupled system since the sender does not know about the recipient. In topic-based systems, users send and receive messages on shared channels. When a producer publishes a message on a topic, all subscribers of that topic receive that message. It is a many-to-many communication medium.

The content-based publish/subscribe systems improves on topics by introducing a subscription scheme based on the actual content of the considered events. In content-based systems, subscribers specify the types of messages that they are interested in and the broker network checks the messages according to user's specification and deliver the message if there is a match. Content-based systems are more flexible than topic based systems since subscriptions are related to specific information content and each combination of information items can be seen as a single dynamic logical channel. This increases the potential logical channels exponentially[69]. Messages can be intelligently routed to destination based on the content of the message.

Topic-based publish/subscribe systems are better suited for our system. Interested brokers will subscribe to a topic and they will get the message whenever a message published on that topic.

5.5.1 The middleware

As we have mentioned above, the transmission of data between publishers and subscribers are handled by the middleware medium. There are different types of middleware, classified by their characteristics. These characteristics could be their architecture or the guarantees that they provide for the data like persistence or reliability.

Publish/subscribe systems allow publisher and subscribers to exchange events between them asynchronously. This is possible in a way that producers publish an event, which is sent to a specific entity that stores them and forwards them to the subscribers. This approach is called a centralized architecture since the central entity takes care of the storing and forwarding the messages. This approach is used by queuing systems like IBM MQSeries[70] and Oracle Advanced Queuing[71]. Both of these are built on top of a centralized database. Applications that use centralized approach have requirements like data consistency, reliability or transactional support. They do not need a high data throughput. Examples of these applications could be banking applications or electronic commerce.

Another approach to provide asynchrony is the distributed architecture in which there is no centralized entity in the system. In order to achieve asynchrony, communication primitives that achieve store-and-forward mechanisms are implemented in both the publisher's and the subscriber's processes. In this approach no process acts as a bottleneck and there is no single unit of failure. This decentralized architecture is good for efficient and fast delivery of data, which is required in applications like multimedia broadcasting.

There is also an intermediate approach in which the event notification service is implemented as a distributed network of servers. In this approach, processes are discharged by using the dedicated servers. There are different topologies for these servers. For example in Jedi[72], event dispatchers are organized in a hierarchical tree structure in which clients can connect to any node. Subscriptions are propagated upward in this tree structure causing heavy load on the root servers. Failure of a server might disconnect the entire sub-tree.

Another approach as in Gryphon[73, 74] is having a graph summarizing the common interests of subscribers. This graph is superimposed with the message broker graph to avoid redundant matches.

Sienna is another system with different approach which uses subscription and advertisement forwarding to set paths for notifications. There are event servers which efficiently match events with subscriptions by keeping track of useful information.

There are two forms of events in publish/subscribe systems: messages and invocations. If it is a message then it is delivered to a subscriber through a single generic operation like notify (). If it is an invocation then it triggers the execution of specific operations of the subscriber. Event notifications, created by the applications, take the form of messages in most systems.

Messages generally consist of header and payload. Header contains message specific information like identifier, issuer, priority and expiration time in a generic format. Payload contains user-specific information.

Invocations have well defined semantics and are directed to a specific type of object. All consumers should have a matching interface for processing the invocations.

5.5.2 Topics: What are they?

In publish/subscribe systems; subscribers are usually interested in specific events. Having several ways of specifying the event types resulted in several types of publish/subscribe schemes. Most widely used ones of these schemes are topic based and content based schemes. There is also another one which is type-based scheme.

Topic based publish/subscribe scheme was the earliest one in which there are topics or subjects. Publishers can publish events which are identified by keywords, topics. Subscriber can also subscribe to individual topics. Topics are similar to the notion of groups. Subscribing to a specific topic X can be viewed as becoming a member of group X and publishing an event on topic X means broadcasting that event among the members of group X.

In a topic based system, participants can exchange information through a set of predefined subjects(topics). Each topic is mapped to a distinct communication channel. The topic abstraction provides platform interoperability by relying only on strings as the keys to divide the event space.

There are additions to the topic based publish/subscribe systems. One of them is to use hierarchies to orchestrate topics. In this scheme, topics can be organized based on containment relationships. A subscription is made to some node in the hierarchy implicitly involves subscriptions to all the subtopics of that node. There are systems that allows the use of wildcards in the topic names so that subscribing and publishing to a several topics at the same time.

5.5.3 The publisher and subscriber

Subscribers do not know the resources when they register their interest in event. They do that by calling the subscribe() operation. Subscription information remains stored in the middleware, event service, and is not forwarded to the publishers. In order to cancel the subscription, the unsubscribe() operation is used.

Publishers generate an event by invoking the publish() operation. Generated event is propagated to all the relevant subscribers by the event service middleware. A subscriber will be notified of all the events relevant to its interest. Publishers can also advertise the nature of their future events by using the advertise() operation. Advertising helps the event service to prepare itself for the expected flows of events. It also allows the subscribers to learn when a new type of information will be available.

5.6 Framework for enabling PBR

To enable dynamic routing in the network environment, we need the following operations to be done;

- First one is to find the current performance metrics in the network. This is done with our Performance monitoring framework.
- Second, use these values to calculate the cost of the links which have unique IDs in the system.
- Third one is to update the link costs in the system so that they reflect the current state of the network.
- The last one is to recalculate the routes based on the newest costs.

5.6.1 Identifying Links

The system that we are working on can have several thousand of links in the network. If there are two links that had the same ID in the network then it is possible for those links to get each others performance values which will result in incorrect broker network maps in the network. The routing changes based on these values will not be accurate so it might cause serious performance bottleneck in the system. In order to prevent this, we need to make sure that each link has a unique ID so that they are associated with their own performance values. In order to do that we use 128 bit universally unique identifiers(UUID)[104].

UUID is an identifier standard used in software construction, standardized by the Open Software Foundation as part of the Distributed computing environment. UUID is an identifier which is unique in both space and time with respect to the space of all UUIDs. The intent of the UUID is to enable distributed systems to uniquely identify information without the need of central coordination. This means that if you create an UUID, it will not be used by anyone else unintentionally.

5.6.2 How a Given Broker Does Receive Performance Data

There are different methods that could be used to update the link costs in the system. For example, it could be broadcast to all of the nodes in the system. These methods bring an extra overhead to the system since every node has to get this updated cost even if they are not interested in it. The overhead on the network could clog the network and effect the performance of its links. There will also be a processing overhead since every node has to check every performance update for each link.

Another method for updating links could be message queuing in which sender directly sends the necessary updates to the specific receivers. This method eliminates the unnecessary overhead in the system. Unfortunately in has some disadvantages. The system that uses this method is not scalable since the sender has to know the receivers. Every time a new receiver comes in , it has to be known by the sender and has to be added to the receivers list.

Considering the overhead problem and scalability issues, we have decided to use topic based publish/subscribe messaging system to update the link costs in the broker system. In this way only the brokers that register their interest receiving the updated link costs get the messages.

5.6.2.1 The topic that interested nodes subscribes

In topic based publish/subscribe systems, senders or publishers label the messages they sent with the name of a topic, rather then addressing it to specific recipients. The messaging middleware then sends the message to all eligible systems that have registered to receive messages on that topic. This form of messaging is far more scalable architecture then point-to-point alternative such as message queuing since the senders or publishers just take care of the task of creating the message and do not worrying about who is going to get it by leaving this task to the messaging middleware. It is a very loosely coupled architecture in which publishers of the messages do not know the registered recipients.

In our system, we want to make sure that only the interested nodes gets the updated performance information. We also want to make sure that it is a loosely

coupled system, which means that the sender or publisher does not know the receivers, so that it is a scalable system.

There is a certain topic that interested nodes can subscribe to get the updated link cost information. The topic that we are using is “PBR/PerformanceData”. If a node wants to get the measured link costs, it has to subscribe to this topic.

5.6.3 What information is published?

The cost of each link is calculated based on the metrics that are measured by the performance monitoring system. The MeasuremetInitiator module which is explained in chapter 3, handles this cost calculation processes for each link. Now, it is time to publish the new cost of the link so that it is get updated by the system. The created message includes the following information;

- The ID of the link
- The new cost
- The topic name

5.6.3.1 Only the nodes that have subscribed to this topic gets this network performance information.

As we explained sections above, topic based publish/subscribe systems delivers the published message only to subscribers. For this reason, only the registered nodes will get the published new performance data. This will prevent the extra overhead that could occur by delivering the new performance values to every node in the system.

5.6.4 When to publish link cost information?

Network performance monitoring service does the monitoring in the network and returns the results to the MeasurementInitiator. When the measurement results are received by the measurement initiator, they need to be processed. There are a few steps that should be taken when processing these results. The first one is to encapsulate them into an XML format and send it to aggregation points. This information is then used for statistical purposes as explained in section 5. Second one is, these performance values are used to update the link costs dynamically.

So, the way we update the cost is by publishing the cost information. Before publishing the cost information, it must be calculated first using the performance metrics. The calculation formula of the cost is explained before in the section 5.4.

After the cost calculation, it is time to publish it to update the cost of the links in the system.

5.6.4.1 Based on thresholds

Threshold values exist for several reasons including safety, unnecessary updates etc. We set some threshold values into our system to eliminate unnecessary updates. These threshold values are checked before publishing the cost information into the network. In order to implement this strategy, we compare the current cost information with the previous cost information. If the difference between them is higher than the 10% of the previous value, this new cost is published to update the cost of the link. If the difference is lower than the threshold value, then the cost of the link remains untouched. The threshold value can be changed depending on system requirements.

5.6.4.2 Eliminates unnecessary updates

Today's network environments are almost limitless in size. That means there could be thousand of links and possibilities for routing purposes. Considering the frequency of the measurements could be as low as couple seconds, there is an enormous amount of calculation and network overhead caused by the monitoring service each time the measurements are done. So, there is a need for a mechanism to prevent monitoring service to perform measurements frequently. Setting threshold values into our system reduces this frequency overhead.

5.6.4.3 Constant updates can cloud metrics

Updating the costs in the network may be costly depending on the size of the network. The reason we use threshold values in the system is that cost of the updating the cost of the links could be very high depending on the number of the links in the system.

5.6.4.4 Minor changes do not effect the routes being calculated

One of the reason that we use threshold values is that small changes do not effect the performance significantly. The cost of updating the links is could be higher than the gain that is provided by the routing changes. There is a test done for this purpose. In this test we see that updating the link cost might take up to 10 ms. So if the gain here is not as good as 10ms then there is no point updating the routes. The conducted tests could be found in chapter 7.

5.6.5 Link Cost information includes the updated value of the link cost

After the link costs are computed by the MeasurementInitiator, a check is made to see if it exceeds the threshold value that we have set before. This threshold value is set to eliminate unnecessary updates in the system. If the new cost exceeds the specified threshold then it is ready to replace the old value in the system. A new message is then created to be published within the system. This new message will include the topic that we have set before which states that this message will only be received by the interested entities who registered to that specific topic. This message will also include the updated cost of the link so that when the registered entities receive this message, they can update the old cost of the link with this new one.

5.6.6 Processing changes in the link cost for a certain link

As it is mentioned above, the new cost of the link is published if it is greater than the threshold value that is set by the measurement initiator. This published information is received by the subscriber. The received message has to be processed to extract information. The link ID and the cost associated with this link is extracted from the received message. Then the system checks if this link exists in the system. If it exists, the cost is updated. As soon as this update occurs, broker network map needs to be updated. Necessary methods are called to achieve this.

5.6.6.1 Update the cost in the BNM

After the new link cost is received, the value of the link cost is updated. This update invokes methods that update the Broker Network Map, when the new link cost is received in the broker network map.

5.6.6.2 Routing cache is updated depending on the destinations that are affected by this change

NaradaBrokering system provides an abstract view of brokers in the network by broker network maps. Broker network maps provide information regarding the inter-connections between brokers in the clusters. Each broker in the network maintains its own broker network map. Changing all the destinations will cause an extra overhead in the system. Beside, there is no need to change all destinations in the routing cache. For example, if a packet traveling in Syracuse, only the routes in Syracuse will be updated. There is no need to update the routes in Chicago since it does not have to do anything with Syracuse traffic.

CHAPTER 6

Some Test Results Regarding the Dynamic Routing

In this chapter we report on the several experiments that we conducted to test the different aspects of our system.

6.1 Routing System Test

This test is done to check the routing algorithm. The link weights are given statically, since it is not easy to construct the network with this complexity in the real world. There are also tests done in the real network environment which you will find in the subsequent sections.

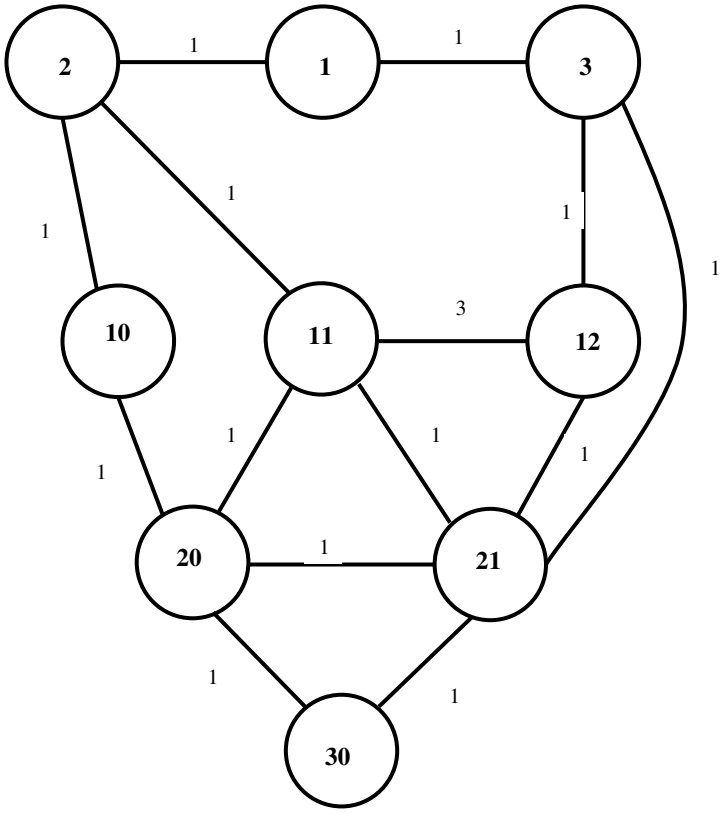

Figure 17 : An Example Broker Setup

In this test there are 9 brokers set up as seen in Figure 17. The cost of the links are given statically and shown in this figure. The broker, numbered 1, is our starting point. We will find the shortest path to other brokers. The shortest path algorithm finds the following paths. It works correctly and finds the exact shortest paths. Shortest path from starting broker 1 to other brokers are shown in table 3 ;

Destination Broker	Followed hop	Cost
2	2	1
3	3	1
10	2	2
11	2	2
12	3	2
20	2	3
21	3	2
30	3	3

Table 3: Destination and Costs to Each Broker from Broker #1

In this example, we wanted to test the dynamic routing system to see if it can dynamically change the routing. For this purpose we have changed the weights of the following links (3,12) and (11,12). The cost of the link (3,12) is increased from 1 to 3 and the cost of the link (11,12) is reduced from 3 to 1. After these changes, the only path that should change here is the one to access broker 12. The dynamic routing algorithm works here and makes that change.

Destination Broker	Followed hop	Cost
2	2	1
3	3	1
10	2	2
11	2	2
12	2	3
20	2	3
21	3	2
30	3	3

Table 4 : Destinations and Costs to Each Broker from Broker #1 with New Changes

There is another slight change here in this setup. The cost of the link (1,2) has changed from 1 to 3. There should be several changes here. The dynamic routing algorithm has responded to this and made the following changes.

Destination Broker	Followed hop	Cost
2	2	3
3	3	1
10	2	4
11	2	4
12	3	4
20	3	3
21	3	2
30	3	3

Table 5: Destinations and Costs to Each Broker from Broker #1 with Additional Changes

These tests have shown us that the routing algorithm works correctly.

6.2 Real World Testing

In this experiment we tried to setup a network that will show that our system is working. For this purpose we setup a network that consists of 4 brokers. One of the brokers is running on a machine which is at San Diego, and the other three are running on different machines which are located at the Community Grid Labs in Bloomington. The specifications of the machines are

Node 1: Windows XP running on Pentium 2.8 GHz

Node 2: Linux running on Pentium 3 800 MHz

Node 3: Linux running on Xeon 2.4 GHz

Node 4: Windows XP running on Pentium 3.4 GHz

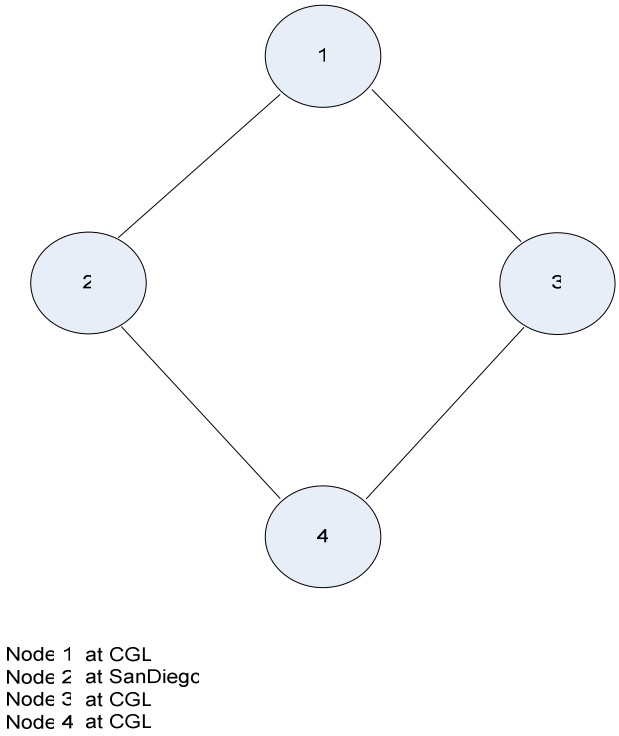

Figure 18: An Example Broker Setup

Without performance based routing, a message sent from node 1 to node 4 could either take the hop, node 2 or node 3. This is because they have the same status in the network and they get the same link cost due to that. It will make a big difference if it takes the node 2 as a hop since this node is in San Diego and it will take much more time to get to the Node 4 in compare to taking node 3 as a hop.

When we run the network with our system, the measurement part will start measuring the performance of the created links. The average link costs after 1 hour of measurements is shown in table 1. As you can see the cost of the link 1 and the link 3 are much higher than the link 2 and link 4. This is as expected since link 2 and link 4 are between the brokers that are in the same geographical location.

Each node has a view of broker network. This information is encapsulated within the broker network map (BNM). BNM provides information regarding the connections between the brokers in the network. Each node has its own BNM. Updating the BNM with new costs will provide us the shortest path to other brokers in the network.

For updating of the links costs, we use publish/subscribe mechanism as explained in dynamic routing section. After each measurement, new costs of the links along with their IDs are published if they are above certain threshold. Subscribers get the updated costs and recalculate the BNMs.

Connection	Cost
CGL1→San Diego	**44.5**
CGL1→CGL2	**1.6**
San Diego→CGL3	**41.2**
CGL2→CGL3	**1.4**

Table 6: Cost of the Links

If we look at the BNM of the node 1 we see the following results,

- Shortest path to node 3 is the direct link from node 1 to node 3.
- Shortest path to node 4 is over node 3.
- Shortest path to node 2 is either direct link to node 2 or taking the hop 3 and 4. Shortest path to this node has switched between these two options depending on the link costs. Sometimes total cost of link2, link3 and link 4 was smaller than link1, so it chose the path consisting of node 3 , and node 4.

If we look at the BNM of the Node 2 we see the following results;

- Shortest path to node 1 is either direct link to Node 1 or Node 4+ Node 3.
- Shortest path to node 3 is either over Node 1 or over node 4.
- Shortest path to Node 4 is either over Node 1 or direct link to Node 4

These results clearly shows us that the dynamic routing is working correctly. There are some factors affecting the link costs in our design. One of the factor is the power of the machine that the broker is running on. As you can see there is a slight difference between the cost of the link 1 and link3. This difference is due to power difference of the machines. CGL1 machine is Pentium 4 2.8 GHz and CGL3 machine is Pentium 4 3.4. The link cost is lower on the faster machines. This is because there are some calculations needs to be done on the computer. This difference gets higher if the number of nodes in the network increases.

6.2.1 Testing with Different Broker Set-up

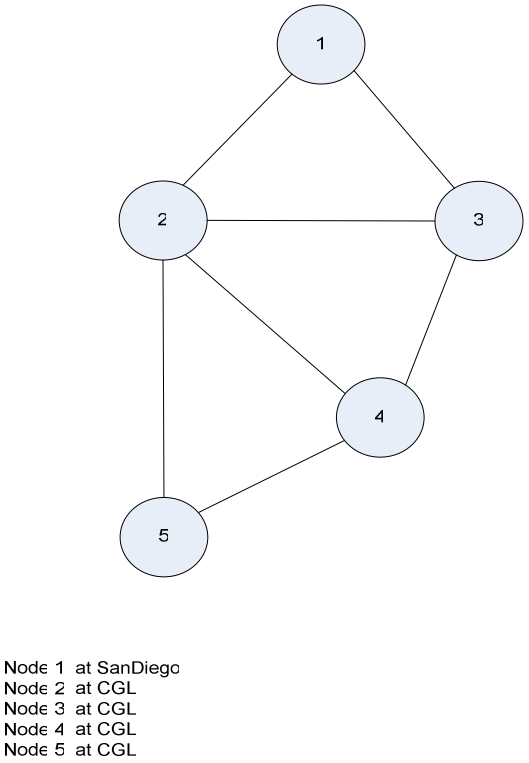

Figure 19: A different Broker setup

In this experiment we tried to expand our setup for more extensive testing. For this purpose we added another broker. And three more connections. In this experiment broker #1 is running in San Diego, other brokers are running in our CGL lab on different machines as stated below;

Node 1: Linux running on Pentium 3 800 MHz

Node 2: Windows XP running on Pentium 2.8 GHz

Node 3: Linux running on Xeon 2.4 GHz

Node 4: Windows XP running on Pentium 3.4 GHz

Node 5: Linux running on Xeon 2.4 GHz

Since there are more links in the new setup, shortest paths change a lot. It is also because four of the nodes are at the same location and the link costs between them are similar. And some spikes on the links enough to change the shortest paths among the brokers. The average cost of the links are shown in table 5.

If we look at the BNM of the node 1 we see the following results,

- Shortest path to node 3 is 1→3 or 1→2→3.
- Shortest path to node 2 is 1→2 or 1→3→2.
- Shortest path to Node 4 and 5 takes hop 2 or 3 depending on the dynamic costs.

Connection	**Cost**
Node 1 → Node 2	39.5
Node 1 → Node 3	43.8
Node 2 → Node 3	2.1
Node 2 → Node 4	1.1
Node 2 → Node 5	2.2
Node 3 → Node 4	1.9
Node 4 → Node 5	1.8

Table 7: Connection Costs of Second Broker Set-up

6.2.2 An Inter-continental Broker Set-up

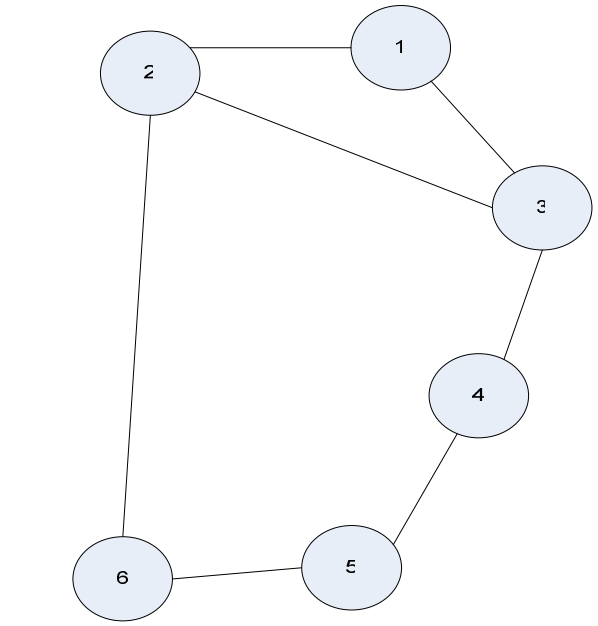

Figure 20: An Inter-Continental Broker Setup

This last experiment includes brokers located in other continents. In this setup we have three brokers in our lab (CGL), one broker in San Diego, one in England and one in Korea. There are total of six brokers and seven connections.

- Node 1: Linux running on Xeon 3.0 GHz
- Node 2: Windows XP running on Pentium 4 3.4 GHz
- Node 3: Windows XP running on Pentium 4 2.8 GHz
- Node 4: Linux running on Pentium 3 800 MHz
- Node 5: Linux running on Xeon 2.4 GHz

- Node 6: Linux running on Pentium 3 1.0 GHz

With the new intercontinental brokers we have links that cost more which will help us to see the benefit of dynamic routing easily. For example without any routing optimization, the packet traveling from node 6(England) to Node 1(Korea) either choose the path of node 2 and node 1 which will cost around 154 or the path node 5,4,3 which will cost around 209. With dynamic routing it chooses the shortest path and save the system from a big delay. As an another example, if we look at the BNM of the node 5(CGL3) we see the following results,

- Shortest path to node 1 is 5→4→3→1.
- Shortest path to node 2 is 5→4 →3→2.
- Shortest path to Node 3 is 5→4 →3. These are the paths chosen dynamically as a result of link costs measured by the performance monitoring.

Connection	**Cost**
CGL1 → Korea	102
CGL1 → CGL3	1.2
CGL1 → England	52
CGL2 → Korea	99
CGL2 → SanDiego	30
CGL3 → SanDiego	29
CGL3 → England	50

Table 8: Connection Costs of Inter-continental Broker Set-up

6.3 System Overhead

The following tests are done to measure the overhead of the dynamic routing in the system. By doing the following test, we want to see when would be good to run the dynamic routing in the system. If the overhead is higher than the traveling time of the packet then there is no need to enable the Performance Based Routing in the system.

I have measured the traveling time of packets between CGL to CGL and CGL to San Diego machine in NB. The following table shows these results.

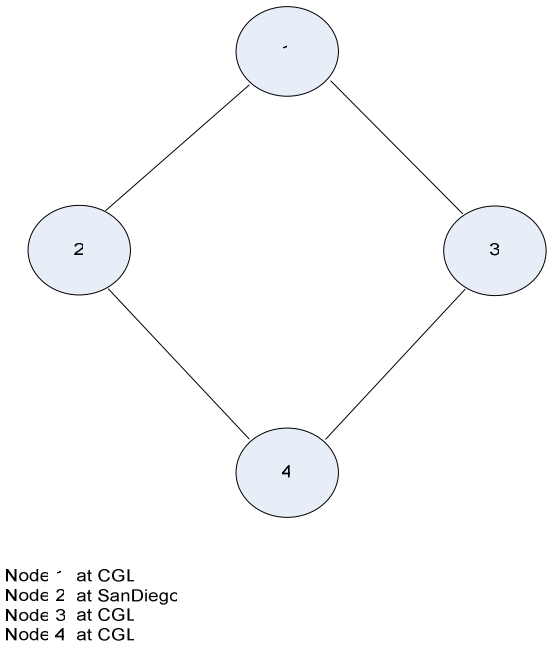

Figure 21: A Broker Network Setup

Nodes	CGL→CGL	CGL→ San Diego
Traveling time of a packet	0.2 - 0.8 ms	14 - 25 ms

Table 9: Delay Associated with Given Connections

The overhead of the dynamic routing system is an average of 8ms. This overhead is directly proportional to the power of the machine it is running on and the number of brokers in the system.

If we look at the setup we have in "test 1", we see that the traveling time of the packet could reach the 26 ms at the worst case. But with dynamic routing it will cost 2-3ms at most. So it is recommended to use dynamic routing in this scenario since the overhead of the Dynamic routing will add 8 ms overhead. Even with this overhead it will cost much less to send a packet on the shortest path.

If all the brokers running in the system was in the same geographical location and the maximum traveling time of the packet was smaller then the overhead, then there would be no reason the run the dynamic routing. There are couple things that could be done at this scenario to reduce the overhead. First one is to disable Performance based routing system. Second one is to reducing the frequency of the measurement, this way it will not add the overhead frequently. Third one is to put some threshold values so that it will not run dynamic routing part unless the threshold value is passed.

If we look at the reasons of this overhead we will see that calculation of the new route creates this. This calculation also includes the time of converting the metrics into

XML format and finding the necessary metrics in this XML. There are advantages and disadvantages of using XML format in our system. It brings extra overhead to the network since the number of bytes you are sending across the network increases with the encapsulation of data in XML format. Accessing the relevant fields in XML format fairly simple and easy but it also brings some calculation overhead. If the number of links and frequency of the measurement are increased then the overhead will be higher since there will be an extra overhead to the network due to more data and extra processing. There are some tests conducted at the end of this chapter which shows the calculation times regarding the XML formatted data.

Even though XML brings extra overhead to the system, it will provide compatibility and ease of use for the system. Integration of other monitoring systems will be easy, since only thing that need to be done is to have a wrapper that will convert that will convert the results of the other monitoring systems into our XML format. Another advantage is ease of display of XML formatted data on web pages. This become handy if the statistics are wanted to be displayed on the web.

6.4 Effect of frequency in performance monitoring

The following test is done to find out how the frequency of the measurement effects the system. We have increased the frequency from 50 ms to 10000 ms and observed the effects on performance monitoring. The table below shows the link costs associated with the frequency of the measurement.

Frequency (ms)	CGL→ Korea (Link Cost)	CGL→ SanDiego (Link Cost)	CGL→ CGL (Link Cost)
50	4000+	900+	300+
100	2000+	93	20
250	900+	24	1
500	94	23	1
1000	93	23	1
2000	93	23	1
5000	93	23	1
10000	93	23	1

Table 10: Link Costs Associated with Different Frequencies

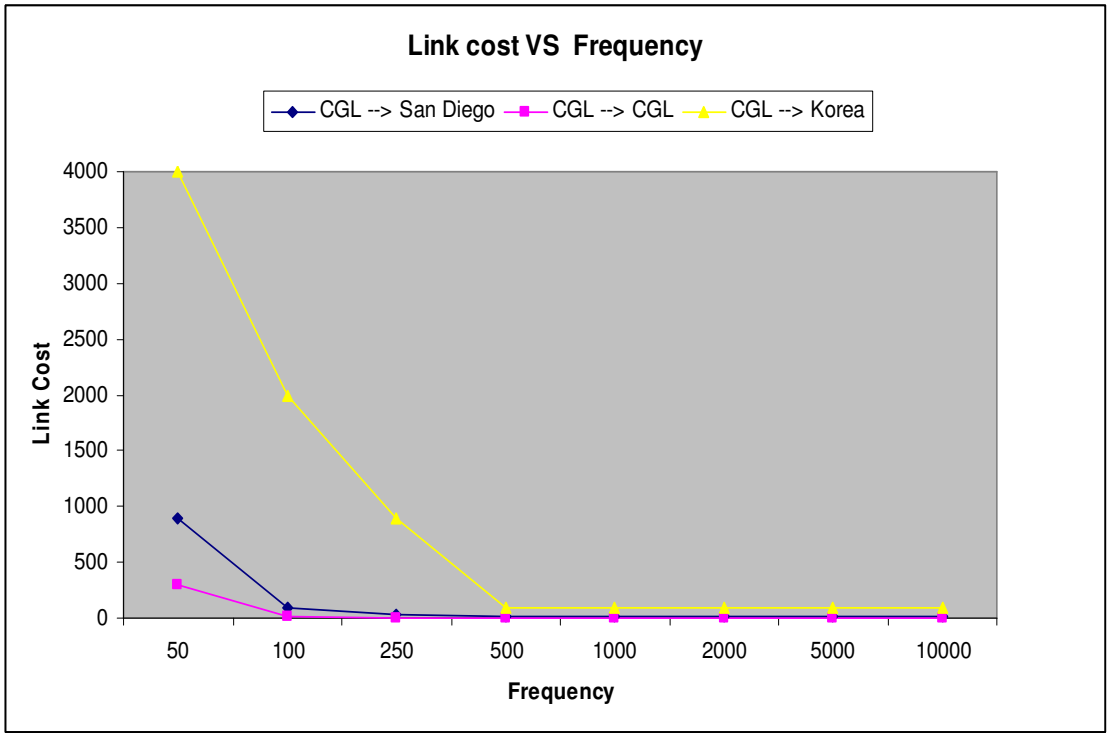

Figure 22 : Effect of Frequency on Link Costs

If we look at the results, we see that performance monitoring will not be able measure the network metrics correctly after a certain threshold. This threshold differs from brokers to brokers depending on their location. If the delay between two brokers is higher then the threshold is higher. For example the threshold for CGL to CGL link is 100ms whereas this value is 250ms for the link between the brokers located in CGL and Korea.

Performance monitoring was not able to measure the metrics correctly in all cases, if the frequency is higher than 100ms. The measurements stopped after a certain time because the link costs were increasing. That's why there is a "+" after some numbers. It shows that performance monitoring system can not measure for those frequencies.

We also see that, frequencies lower than 1000ms does not effect the measurement. We can say that it is safe to use frequencies lower than 1000ms. But there are also other factors that might effect the measurement. For example, if there are more links then the frequency needs to be lowered. Also, if the traffic on the link is too high then higher frequency will bring overhead to the links capacity.

There are some methods to lower this overhead that is caused by the performance monitoring. For example, if the link cost does not change for a while then the measurement of that link might be stopped for a certain period of time. This will prevent extra overhead caused by the monitoring system.

6.5 XML and XSL related tests

We have also conducted some tests regarding to XML conversions and display. The experiments were performed on a Windows 2000 machine (Pentium-4, 1.5 GHz, 512 MB) and all the processes involved ran using the Java 1.4 JRE VM. In our experiments we simulate the presence of multiple links (sometimes reaching several thousands) since it would be infeasible to actually set up the large number of links that we deal with in our experiments. Also, in our experiments the portal is running on the same machine where the database file is on. There is thus no network delay to load the database file.

For our experiments we did 3 different kinds of measurements, First, we measured the time for constructing the W3C Document Object Model from an XML file. The time associated with this construction varies with the number of XML elements in the file, which corresponds to the number of links whose information has been aggregated. Figure 23 depicts the W3C Document creation time as a function of the number of elements in the XML file. The cost varies from 691 milliseconds for an XML file with 100 elements (representing 100 links) to 2.824 seconds for an XML file with 16000 entries.

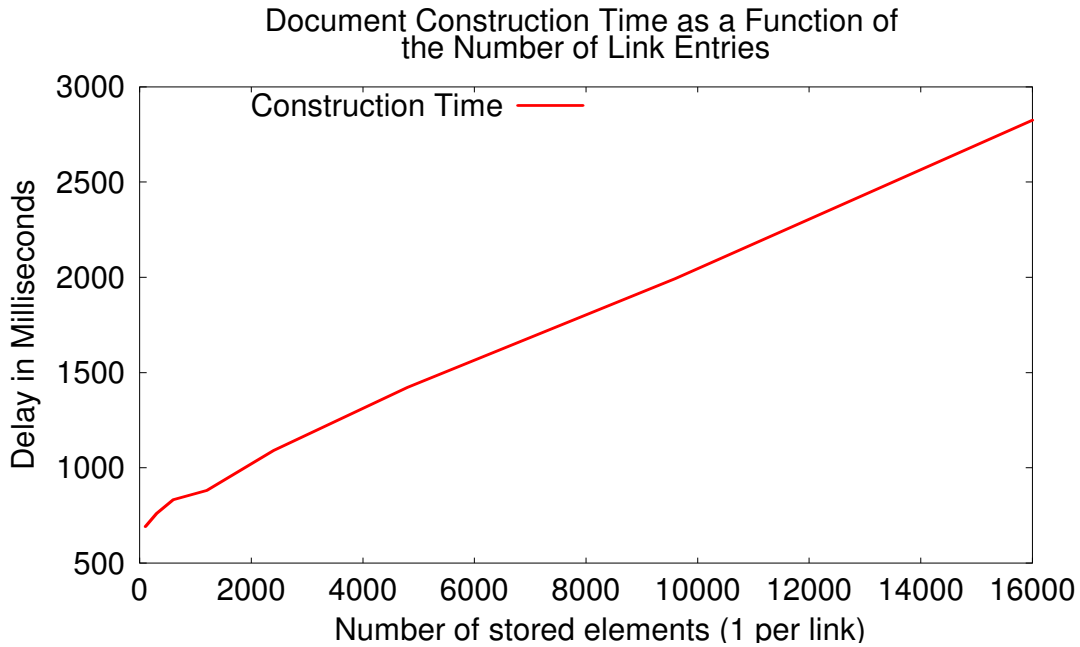

Figure 23: Document Construction Time

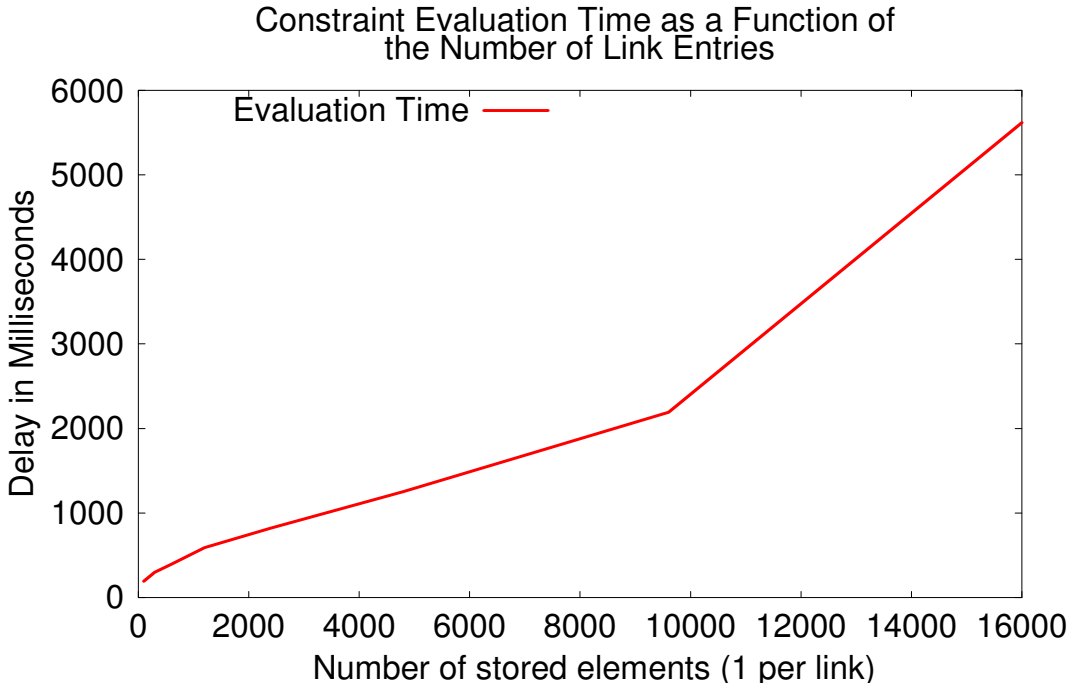

Figure 24 : Query Evaluation Time

Second, we measured the time to evaluate typical queries based on different number of links in the aggregated data. The graph in Figure 24 depicts this increase in evaluation time corresponding to increases in the number of entries in the XML flat-file. The evaluation operation can of course be optimized considerably by evaluating queries only for those links (each with a unique link ID) whose values have changed. In our earlier work we had incorporated a scheme where monitoring nodes would report data about links, only if there is a certain percentage change from its last reported data. This optimization would allow us to reduce query evaluation time very significantly.

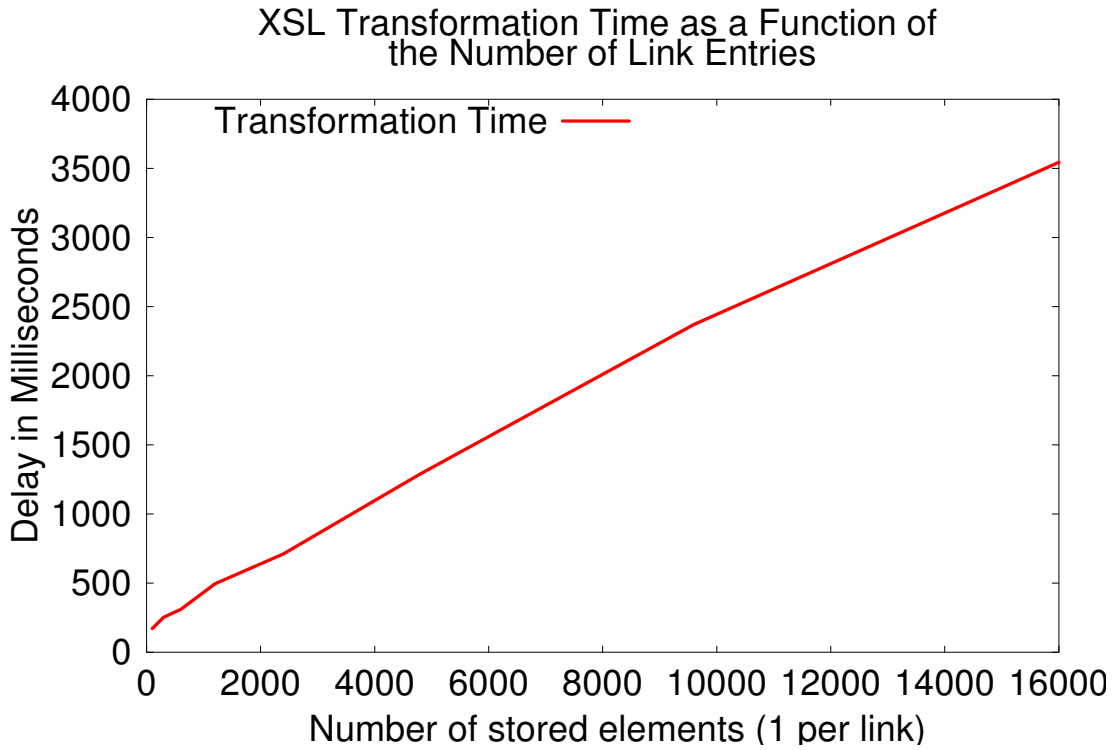

Figure 25 : XSL Transformation Time

Finally, we include timing delays associated with the time it takes the XSLT portlet to transform an XML file into the corresponding HTML file using a given XSL style sheet. Figure 8 depicts the time needed for this transformation.

CHAPTER 7

Conclusion

In this dissertation we have presented a scalable and hardware independent performance based dynamic routing system architecture for distributed systems. This architecture identifies three main tasks to be performed in dynamic routing systems and proposes scalable framework for each component. These tasks are the following; measuring the performance on the network, aggregation of these performance results and dynamic routing based on the measured performance.

We proposed using NaradaBrokering distributed broker system to implement and demonstrate the ideas and concepts presented in this thesis. However, these ideas can be implemented on other distributed systems with similar features. We made some additions and modifications on NaradaBrokering network to support performance monitoring, aggregation and dynamic routing.

We have conducted several tests regarding different aspects of our system. These tests demonstrated that our framework can find and select the best routes based on the performance of those links. We have also investigated issues related to effect of

the frequency of performance measurements in the system. We have seen that the frequency of the measurements should be set to a right value, otherwise it could cause performance degradation in the network.

The main contribution of this research is designing a complete framework for a scalable, de-centralized and hardware-independent, performance based dynamic routing system which consists of three parts; performance monitoring framework, aggregation framework and dynamic routing framework.

We have used a de-centralized approach when designing the system in which every node can perform its performance monitoring. New additions and removals can be easily made in the system without making any changes or modifications to the existing network structure. Use of publish/subscribe mechanism to update the links costs in the system provides high scalability.

Using XML format to represent network performance metrics measured by the monitoring service provides an easy interaction with other systems. XML is widely known standard which can easily be interpreted by the interested systems so that they can use the information for their purposes. Data mining can easily be achieved on XML data using XPATH queries. Furthermore, XSLT can easily be used to display XML data on portals. We have also found out that XML introduces an extra overhead in the system. Data mining on XML files could cause an extra overhead especially if the file size increases.

More detailed explanations can be found in the section below regarding the architecture we presented in this thesis.

7.1 The Summary of the answers for the research questions

Here we summarize the answers for the research questions presented in this thesis;

1. Is it possible to design a complete performance based dynamic routing?

Yes it is possible. In order to achieve this, performance monitoring, aggregation and dynamic routing frameworks should exist in the system. Performance monitoring system will measure the performance in the network system, dynamic routing system will use the measured performance values to dynamically select the best routes and aggregation system will store the performance values for future analysis of the system.

2. Do you need to update the existing network structure to enable performance based dynamic routing?

Our framework is designed in a way that it does not require any changes on the existing network structure. It works in the application layer so there is no need to make changes to the underlying hardware structure.

3. Is there a de-centralized scalable algorithm?

In order to create a de-centralized system, every node in the network should be able to perform its own performance monitoring. Our system is designed in a way that each node in the system can perform its own network performance monitoring.

Scalability is high in this system since there is no need to make any changes in the system for addition of new nodes and links. Using publish/subscribe messaging system for link cost updates in the system increases the scalability in the system due to reasons explained before.

4. Is performance based dynamic routing hardware independent?

We designed the performance based dynamic routing to be hardware independent. It works on the application level so that it does not interact with any underlying hardware structure like routers, switches, etc. The advantage of this scheme is that administrators do not have to worry about the hardware structure of the network. They do not have to know about the hardware like routers. Another advantage of this scheme is that changes in the hardware devices in the network does not require any changes in our system.

5. Is there an overhead caused by the performance based dynamic routing?

There is an overhead caused by the performance based dynamic routing system. Likewise, network monitoring system also causes an overhead on the links due to performance packets used to find out network performance metrics. Frequency of measurement is set to a right value to make proper measurements. Low frequency values will not be able to update the system in time in case of low performance in the links on the network. High frequency values will cloud the network by generating too much traffic and cause performance degradation.

There is also a processing overhead caused by the system. Processing the metrics found by the monitoring system, updating the links costs in the system and recalculating the routes cause an overhead in the system. High frequency measurements will increase the overhead.

6. What is the advantage of using publish/subscribe system for updating the link costs in the network system dynamically?

Each broker in the network could have several links originating from them to other brokers in the system. After finding the performance metrics of each link in the network, these new values have to be disseminated into the system so that link costs are up-to-date in the system. There could be several brokers and each broker could have several links. Dissemination of the new costs of the links in the system should be timely and it should not cause an overhead in the system. For this purpose we have decided to use publish/subscribe messaging scheme. In this scheme, the new cost of the link is published when it is computed by the monitoring system. Any node which is interested in getting this updated link cost will subscribe to a certain topic. Published costs will only be delivered to interested nodes. This will ensure minimal overhead in the system.

Publish/Subscribe systems could bring an extra traffic if the link costs are changing frequently. This can be prevented by using threshold values which will eliminate unnecessary updates. The threshold value we use is the 10% of the old cost value. If the difference between the new cost, computed by the monitoring service, and

the existing cost is less then this threshold value, then the new cost is not disseminated into the system. This prevents unnecessary information publishing.

Another advantage of this scheme is that the publisher does not have to know the recipients. New brokers can be added into the system thanks to this feature of the publish/subscribe system. This will increase the scalability of the system greatly.

7. Can it interact with other systems?

Our design has features that increases the ability to interact with other performance monitoring systems. We are using XML format to wrap our performance metrics. Using XML has advantages in terms of compatibility with other systems. One can easily understand the structure that we use and convert them into their format to use in their system. We can also convert their performance values into our XML structure and leverage it.

7.2 Future Directions

There are several enhancements and features that can be added to Performance based Dynamic routing framework.

More scalability tests could be conducted to see how the system handles the large number of links and nodes in the system. However, it is not easy to set-up a broker network in larger scales. It is not easy to find computing facilities to perform larger scale tests.

More database performance tests can be conducted to see if there is any better solution then we have now. XML databases can be researched which could be better alternative than the mySQL database.

Applications that can allow easy data mining on the stored performance data can be designed. These applications can also be used to identify, circumvent and prevent system bottlenecks and to aid routing algorithms.

Better user interfaces could be created in which administrators could interact with the system. they could set thresholds in the system to prevent low performance. Warning mechanisms can also be created to alert administrators in case of critical performance problems.

Bibliography

[1] Zhang, L. et al. "ReSource ReserVation Protocol (RSVP) – Functional Specification", Internet Draft, March 1994.

[2] Topolcic, C., "Experimental Internet Stream Protocol: Version 2 (ST-II)", Internet RFC 1190, October 1990.

[3] Oram, A. (eds) 2001. Peer-To-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly, Sebastapol, CA 95472.

[4] IETF Benchmark Working subgroup: http://www.ietf.org/html.charters/ippm-charter.html

[5] CAIDA http://www.caida.org/tools/

[6] R. Wolski, N. Spring, and C. Peterson. Implementing a performance forecasting system for metacomputing: The Network Weather Service. Tech. Rep. TR-cs97-540, University of California, San Diego, May 1997.

[7] R. Wolski. Forecasting network performance to support dynamic scheduling using the network weather service. Proceedings of the 6th IEEE Symp. On High Performance Distributed Computing, August 1997.

[8] R. Carter and M. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical Report TR-96-007, Boston University 1996.

[9] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. Technical Report TR-96-006, Boston University 1996.

[10] B. Lowecamp, N. Miller, D. Sutherland, T. Gross, P. Steenkiste and J. Subhlok. A resource query interface for network-aware applications. In Proc. 7th IEEE Symp. On High Performance Distributed Computing, August 1998.

[11] S. Fritzgerald. I. Foster, C. Kesselman, G. von Laszewski, W. Smith and S. Tuecke. A directory service for configuring high performance distributed computations. In Proc. 6th IEEE Symp. On High Performance Distributed Computing, August 1997.

[12] S. J. Chapin, J. Karpovich and A. Grimshaw. Resource management in legion. Technical report cs-98-09, University of Virginia, Department of Computer Science, May 1998.

[13] K. Obraczka and G.Gheorghiu. The performance of a service for network-aware applications. In proceedings of 2nd SIGMETRICS Conference on Parallel and Distributed Tools, August 1998, to appear.

[14] F.Berman, R. Wolski, S. Figueira, J.Scopf and G. Shao. Application level scheduling on distributed heterogeneous networks. In proceedings of Supercomputing 1996, 1996.

[15] C. Yoshikawa, B. Chun, P. Eastham, A.Vahdat, T.Anderson and D. Culler. Using Smart Clients to build scalable services. In Proceedings of the USENIX 1997 Technical Conference, 1997.

[16] J. Gehrinf and A.Reinfeld. Mars – A framework for minimizing the job execution time in a metacomputing environment. Proceedings of Future General Computer Systems, 1996

[17] R. Jones. Netperf. Available from http://www.cup.hp.com/netperf/NetperfPage.html

[18] M.Mathis and J.Madhavi. Diagnosing internet congestion with a transport layer performance tool. Proceedings of the INET 1996, 1996

[19] V. Jacobson. A tool to infer characteristics of internet paths. Available from ftp://ftp.ee.lbl.gov/pathchar.

[20] The NaradaBrokering System http://www.naradabrokering.org

[21] Geoffrey Fox and Shrideep Pallickara, An Event Service to Support Grid Computational Environments, (To appear) Concurrency and Computation: Practice and Experience, Special Issue on Grid Computing Environments.

[22] Shrideep Pallickara and Geoffrey Fox. A Middleware Framework and Architecture for Peer-to-Peer Grids. (To appear) Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware-2003.

[23] Geoffrey Fox, Shrideep Pallickara and Xi Rao, "A Scaleable Event Infrastructure for Peer to Peer Grids". *Proceedings of ACM Java Grande ISCOPE Conference 2002*. Seattle, Washington. November 2002.

[24] Geoffrey Fox and Shrideep Pallickara, "JMS Compliance in the Narada Event Brokering System". Proceedings of the *International Conference on Internet Computing* (IC-02). June 2002. pp 391-402.

[25] Geoffrey Fox et al, "Grid Services For Earthquake Science" *Concurrency & Computation: Practice and Experience*. 14(6-7): 371-393 (2002).

[26] Hasan Bulut, Geoffrey Fox, Shrideep Pallickara, Ahmet Uyar and Wenjun Wu, "*Integration of NaradaBrokering and Audio/Video Conferencing as a Web Service*". Proceedings of the IASTED International Conference on Communications, Internet, and Information Technology, November, 2002, in St.Thomas, US Virgin Islands.

[27] Geoffrey Fox and Shrideep Pallickara, "An Approach to High Performance Distributed Web Brokering". *ACM Ubiquity* Volume2 Issue 38. November 2001.

[28] Geoffrey Fox, Ozgur Balsoy, Shrideep Pallickara, Ahmet Uyar, Dennis Gannon, and Aleksander Slominski, "Community Grids" invited talk at *The 2002 International Conference on Computational Science*, April 21 -- 24, 2002 Amsterdam, The Netherlands.

[29] D.J. Watts and S.H. Strogatz. "Collective Dynamics of Small-World Networks". *Nature*. 393:440. 1998.

[30] R. Albert, H. Jeong and A. Barabasi. "Diameter of the World Wide Web". *Nature* 401:130. 1999.

[31] Mark Happner, Rich Burridge and Rahul Sharma. Java Message Service Specification". Sun Microsystems. 2000. http://java.sun.com/products/jms.

[32] Sun Microsystems. The JXTA Project and Peer-to-Peer Technology http://www.jxta.org

[33] The JXTA Protocol Specifications. http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html

[34] Gurudutt Banavar, Tushar Chandra, Bodhi Mukherjee, Jay Nagarajarao, Rob Strom, and Daniel Sturman. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems.In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, Austin, Texas, May 1999.

[35] Bill Segall and David Arnold. Elvin has left the building: A publish/subscribe noti.cation service with quenching. In *Proceedings AUUG97*, pages 243–255, Canberra, Australia, September1997.

[36] Fiorano Corporation. A Guide to Understanding the Pluggable, Scalable Connection Management (SCM) Architecture - White Paper. Technical report, http://www.fiorano.com/ products/fmq5 scm wp.htm, 2000.

[37] Talarian Corporation. Smartsockets: Everything you need to know about middleware: Mission critical interprocess communication. Technical report, URL: http://www.talarian.com/products/smartsockets, 2000.

[38] TIBCO Corporation. TIB/Rendezvous White Paper. Technical report, URL: http://www.rv.tibco.com/whitepaper.html, 1999.

[39] The Object Management Group (OMG). OMG's CORBA Event Service. URL: http://www.omg.org/technology/documents/formal/eventservice.htm, June 2000. Version 1.0.

[40] RTP: A Transport Protocol for Real-Time Applications (IETF RFC 1889) http://www.ietf.org/rfc/rfc1889.txt.

[41] XML based messaging and protocol specifications SOAP. http://www.w3.org/2000/xp/.

[42] Microsoft Internet Security and Acceleration (ISA) Server. http://www.microsoft.com/isaserver/

[43] Checkpoint Technologies. http://www.checkpoint.com/

[44] iPlanet. http://www.iplanet.com/

[45] Shrideep Pallickara, Geoffrey Fox, John Yin, Gurhan Gunduz, Hongbin Liu, Ahmet Uyar, Mustafa Varank. A Transport Framework for Distributed Brokering Systems. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications. (PDPTA'03).

[46] Apache Jetspeed. http://jakarta.apache.org/jetspeed/site/index.html

[47] Gurhan Gunduz, Shrideep Pallickara, Geoffrey Fox. A Portal Based Approach to Viewing Aggregated Network Performance Data in Distributed Brokering Systems. http://www.naradabrokering.org

[48] Gurhan Gunduz, Shrideep Pallickara and Geoffrey Fox. A Framework for Aggregating Network Performance in Distributed Brokering Systems. Proceedings of the 9th International Conference on Computer, Communication and Control Technologies.

[49] Geoffrey Fox and Shrideep Pallickara. NaradaBrokering: An Event Based Infrastructure for Building Scaleable Durable Peer-to-Peer Grids. Chapter 22 of "Grid Computing: Making the Global Infrastructure a Reality". John Wiley April'03.

[50] Geoffrey Fox and Shrideep Pallickara. The Narada Event Brokering System: Overview and Extensions. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, June 2002. pp 353-359.

[51] Geoffrey Fox and Shrideep Pallickara. An Event Service to Support Grid Computational Environments. Journal of *Concurrency and Computation: Practice & Experience*. Volume 14(13-15) pp 1097-1129.

[52] *XQuery 1.0: An XML Query Language*, W3C Working Draft (16 August 2002), see http://www.w3.org/TR/xquery.

[53] Robie, J.; Lapp, J.; Schach, D. (1998). *XML Query Language* (XQL). In: Marchiori, M. (ed.): QL'98 --- The Query Languages Workshop. W3C. http://www.w3.org/TandS/QL/QL98/pp/xql.html

[54] S. Ceri, S. Comai, E. Damiani, P. Fraternali, and S. Paraboschi. XML-GL: a graphical language for querying and restructuring XML documents. In Proc. of the Int. World Wide Web Conference, Canada, 1999.

[55] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. *A Query Language for XML*. See http://www.research.att.com/~mff/files/final.html

[56] Don Chamberlin, Jonathan Robie, and Daniela Florescu. *Quilt: an XML Query Language for Heterogeneous Data Sources*. In *Lecture Notes in Computer Science*, Springer-Verlag, Dec. 2000. Also available at http://www.almaden.ibm.com/cs/people/chamberlin/quilt_lncs.pdf. See also http://www.almaden.ibm.com/cs/people/chamberlin/quilt.html.

[57] World Wide Web Consortium. *XML Path Language (XPath) Version 1.0*. W3C Recommendation, Nov. 16, 1999. See http://www.w3.org/TR/xpath.html

[58] International Organization for Standardization (ISO). *Information Technology-Database Language SQL*. Standard No. ISO/IEC 9075:1999. (Available from American National Standards Institute, New York, NY 10036, (212) 642-4900.)

[59] IBM Websphere Portal http://www-3.ibm.com/software/info1/websphere/index.jsp?tab=products/portal

[60] BEA Weblogic Portal http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/portal

[61] Plumtree Portal http://www.plumtree.com/products/midmarket/

[62] Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology. Edited by A. Oram. O'Rielly Press, CA. March 2001.

[63] Java Servlet Technology http://java.sun.com/products/servlet/

[64] World Wide Web Consortium. Extensible Stylesheet Language (XSL). W3C Working Draft. See http://www.w3.org/TR/xsl/

[65] World Wide Web Consortium. XSL Transformations (XSLT). W3C Recommendation. See http://www.w3.org/TR/xslt

[66] EUGSTER, P. AND GUERRAOUI, R. 2001. Contentbased publish/subscribe with structural reflection.In *Proceedings of the 6th Usenix Conferenceon Object-Oriented Technologies and Systems* (COOTS'01).

[67] EUGSTER, P., GUERRAOUI, R., AND DAMM, C. 2001.On objects and events. In *Proceedings of the OOPSLA '01 Conference on Object Oriented Programming Systems Languages and Applications*.ACM Press, New York, NY, 254–269.

[68] EUGSTER, P., GUERRAOUI, R., AND SVENTEK, J. 2000.Distributed Asynchronous Collections: Abstractions for publish/subscribe interaction. In *Proceedingsof the 14th European Conference on Object-Oriented Programming* (ECOOP'2000).

[69] Baldoni, R.; M. Contenti, and A. Virgillito. "The Evolution of Publish/Subscribe Communication Systems." Future Directions of Distributed Computing. Springer Verlag LNCS Vol. 2584, 2003.

[70] LEWIS, R. 1999. Advanced Messaging Applications with MSMQ and MQSeries. QUE.

[71] Oracle 2002. Oracle9i Application Developer's Guide—Advanced Queuing. Oracle, Redwood Shores, CA.

[72] CUGOLA,G.,NITTO, E.D., AND FUGETTA,A. 2001. The Jedi event-based infrastructure and its application to the development of the opss wfms. IEEE Trans. Softw. Eng. 27, 9 (Sept.), 827– 850.

[73] BANAVAR, G., CHANDRA, T., MUKHERJEE, B. NAGARAJARAO, J., STROM, R., AND STURMAN, D. 1999a. An efficient multicast protocol for content-based publish-subscribe systems. In Proceedings of the 19th International Conference on Distributed Computing Systems (ICDCS'99).

[74] BANAVAR, G., CHANDRA, T., STROM, R., AND STURMAN, D. 1999b. A case for message oriented middleware. In Proceedings of the 13th International Symposium on Distributed Computing (DISC 99). 1–18. China).

[75] Internet2 end-to-end performance initiative. http://e2epi.internet2.edu/

[76] EU DataGrid and EU DataTAG work on end-host performance. See for example R. Hughes-Jones et al. Performance Measurements on Gigabit Ethernet NICs and server Quality Motherboards. PFLDnet Workshop, February 2003 http://datatag.web.cern.ch/datatag/pfldnet2003/papers/hughes-jones.pdf

[77] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis. Framework for IP performance metrics. RFC2330, May 1998.

[78] Self-Configuring Network Monitoring Project. http://dsd.lbl.gov/Net-Mon/Self-Config.html

[79] Ping end-to-end reporting(PingER) project. http://www-iepm.slac.stanford.edu/pinger/

[80] Source code and documentation of iperf is available at http://dast.nlanr.net/Projects/Iperf/ .

[81] Source code and documentation of UDPmon is available at www.hep.man.ac.uk/~rich/net .

[82] Traceroute RENO(TRENO) http://www.psc.edu/networking/treno_info.html

[83] Source code and documentation of pchar is available at http://www.employees.org/~bmah/Software/pchar/ .

[84] Source code and documentation of pathrate is available at http://www.cc.gatech.edu/fac/Constantinos.Dovrolis/ .

[85] National Internet Measurement Infrastructure(NIMI) http://www.ncne.org/nimi/

[86] Real-time traffic flow measurement http://www.auckland.ac.nz/net/Internet/rtfm/

[87] NetFlow http://netflow.cesnet.cz/

[88] CoralReef http://www.caida.org/tools/measurement/coralreef/

[89] GridMon http://www.gridmon.dl.ac.uk/gridmon/

[90] GEANT http://www.geant.net/

[91] DOE National Collaboratories projects http://www.doecollaboratory.org/research2/homepage.html

[92] Relational grid monitoring architecture(R-GMA) http://www.r-gma.org/

[93] TIERNEY, B., AYDT, R., GUNTER, D., SMITH, W., TAYLOR, V., WOLSKI, R., AND SWANY, M. A grid monitoring architecture. Tech. Rep. GWD-PERF-16-2, Global Grid Forum, January 2002. http://citeseer.ist.psu.edu/tierney02grid.html

[94] *H.B. Newman, I.C. Legrand, P.Galvez, R. Voicu, C. Cirstoiu* MonALISA: A Distributed Monitoring Service Architecture CHEP 2003, La Jola, California, March 2003

[95] H.B. Newman, I.C. Legrand, J.J. Bunn, "A Distributed Agent-based Architecture for Dynamic Services" CHEP 2001, Beijing, Sept 2001, http://clegrand.home.cern.ch/clegrand/CHEP01/chep01_10-010.pdf

[96] Julian Bunn and Harvey Newman Data Intensive Grids for High Energy Physics Grid Computing: Making the Global Infrastructure a Reality, edited by Fran Berman, Geoffrey Fox and Tony Hey, March 2003 by Wiley

[97] Baruvkas Algorithm http://www.ics.uci.edu/~eppstein/161/960206.html#boruvka

[98] Ganglia http://ganglia.sourceforge.net/

[99] W. Frawley and G. Piatetsky-Shapiro and C. Matheus, Knowledge Discovery in Databases: An Overview. AI Magazine, Fall 1992, pp. 213-228.

[100] D. Hand, H. Mannila, P. Smyth: Principles of Data Mining. MIT Press, Cambridge, MA, 2001. ISBN 0-262-08290-X

[101] JINI http://www.sun.com/software/jini/

[102] Web Services Definition Language http://www.w3.org/TR/wsdl

[103] SOAP http://www.w3.org/TR/soap12-part1/

[104] Universally Unique Identifier http://www.itu.int/ITU-T/asn1/uuid.html

[105] Napster website http://www.napster.com/

[106] Gnutella website http://www.gnutella.com

[107] Skype website www.skype.com

[108] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. Technical Report DSC ID: http://citeseer.ist.psu.edu/649723.html

[109] http://grids.ucs.indiana.edu/ptliupages/publications/disted/erdctraining00.pdf

Vitae

NAME:	Gurhan Gunduz
DATE OF BIRTH:	August 13th , 1974
PLACE OF BIRTH:	TURKEY
EDUCATION:	
SUMMER 1999	M.S. in Computer & Information Science
	Department of Electrical Engineering and Computer
	Science, Syracuse University,
	Syracuse, NY, U.S.A.
JUNE 1996	B.S. in Computer Engineering,
	Ege University,
	Izmir, TURKEY