Your title here

A THESIS

Submitted by

Your name surname

In partial fulfilment for the award of the degree of

Master of Technology

IN

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

Your guide's name

DEPARTMENT OF COMPUTER ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY CALICUT

NIT CAMPUS PO, CALICUT

KERALA, INDIA 673601

April 20, 2015

ACKNOWLEDGEMENTS

Your acknowledgements **Y**

name

DECLARATION

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text".

Place:
Date:

Signature :
Name :
Reg.No:

CERTIFICATE

*This is to certify that the thesis entitled: "***Your title here***" submitted by Sri/Smt/Ms* **Your name surname** *to National Institute of Technology Calicut towards partial fulfillment of the requirements for the award of Degree of Master of Technology in Computer Science Engineering is a bonafide record of the work carried out by him/her under my/our supervision and guidance.*

Signed by Thesis Supervisor(s) with name(s) and date

Place:
Date:

Signature of Head of the Department

Office Seal

Contents

Abstract

Your abstract goes here if@

Tables

Figures

Chapter 1

Introduction

Today, the architecture of distributed computer systems is dominated by client/ server platforms relying on synchronous request/reply. This architecture is not well suited to implement information-driven applications like news delivery, stock quoting, air traffic control, and dissemination of auction bids due to the inherent mismatch between the demands of these applications and the characteristics of those platforms. In contrast to that, publish/subscribe directly reflects the intrinsic behavior of information-driven applications because communication here is indirect and initiated by producers of information: Producers publish notifications and these are delivered to subscribed consumers by the help of a notification service that decouples the producers and the consumers. Therefore, publish/subscribe should be the fist choice for implementing such applications.

In client/server systems two roles exist: A component acts as a client if it requests data or functionality from another component; it acts as a server if it responds to a client's request. Moreover, a client is blocked after it has issued a request, until the corresponding reply arrives. One of the main deficiencies is the tight coupling among the involved components, i.e., the clients and the servers: A client needs to explicitly address the server that shall process the request, the server must be ready and able to process the request, and the client is blocked, until it receives the reply. Because of these inherent disadvantages a large range of applications cannot be realized efficiently by using request/reply. These problems are approached by a new communication paradigm called publish/subscribe that recently gained increased publicity in the distributed systems research area. Publish/subscribe is an asynchronous communication paradigm that is also the basis for extensions and supplementary services that have been added to standard middleware recently.

1.0.1 Publish Subscribe Systems

A publish/subscribe system consists of a set of clients that asynchronously exchange notifications, decoupled by a notification service that is interposed between them. Clients can be characterized as producers or consumers. Producers publish notifications such as current stock quotes, and consumers subscribe to notifications by issuing subscriptions, which are essentially stateless message filters. Consumers can have multiple active subscriptions, and after a client has issued a subscription the notification service delivers all future matching notifications that are published by any producer until the client cancels the respective subscription. Publish/subscribe systems have a number of interesting characteristics. Firstly, producers do not need to address consumers and vice versa. Secondly, communication is asynchronous, thereby removing the disadvantages and inflexibility of synchronous communication described above. Thirdly, producers and consumers do not need to be available at the same time. Finally, publish/subscribe directly reects the intrinsic behavior of information-driven applications because communication is initiated by producers of information.

1.1 Previous Works

1.1.1 PASTRY

Pastry [?] is a scalable, distributed object location and routing substrate for wide-area peer-to-peer applications. Pastry performs application-level routing and object location in a potentially very large overlay network of nodes connected via the Internet.

Each node in the Pastry network has a unique identifier (nodeId). When presented with a message and a key, a Pastry node efficiently routes the message to the node with a nodeId that is numerically closest to the key, among all currently live Pastry nodes. Each Pastry node keeps track of its immediate neighbors in the nodeId space, and notifies applications of new node arrivals, node failures and recoveries. Pastry takes into account network locality; it seeks to minimize the distance messages travel, according to a to scalar proximity metric like the number of IP routing hops. Pastry is completely decentralized, scalable, and self-organizing.

Routing Performance	logxN where x=2b and N is the total number of nodes. For one it is the order of the diameter of the graph.
No. Of Messages	logxN where x=2b and N is the total number of nodes
Routing Table Size	leaf set=2*2b and neighbor set=2*2b and routing table=2b * logxN where x=2b and N is the total number of nodes.
Overhead on node Addition and deletion	involves passing routing tables among at least logxN nodes (where x=2b and N is the total number of nodes.)
Fault Tolerance	not tolerant
Node Failure	self adjusts
Link Failure	not specified
Traffic adaptability	adaptable to scalar traffic metrics
Requirement for global Knowledge	not required
Scalability	Scalable
Correctness of algorithm	correct

1.1.2 MEDYM

MEDYM: Match Early with DYnamic Multicast[?]. Unlike existing approaches, MEDYM does not build static overlay networks for event delivery. Its event delivery process is as shown in Figure 4. When an event is published, it is first matched against subscriptions from remote servers, to obtain a destination list of servers with matching subscriptions. Then, the event is routed to the destination servers through dynamic multicast: On receiving an event message, based on its destination list, a server dynamically computes the next-hop servers to which to forward the message, as well as the new destination list for each of the next-hop servers. In this way, a transient dynamic multicast tree is constructed on the fly. Along this tree, the event is routed to all the servers with matching subscriptions.

Routing Performance	For one subscriber it is the order of the diameter of the graph
No. Of Messages	For one subscriber it is the order of the diameter of the graph
Routing Table Size	directly proportional to the number of nodes in the network
Overhead on node Addition and deletion	dynamically adjusts to new node arrivals and node deletions
Fault Tolerance	tolerance is not part of MEDYM
Node Failure	The topological changes do not affect the algorithm
Link Failure	The topological changes do not affect the algorithm
Traffic adaptability	The routing can be done to optimize any metric
Requirement for global Knowledge	The complete knowledge of the entire network is required. And the information has to be updated periodically
Scalability	Not Scalable
Correctness of algorithm	correct

1.1.3 SIENA

SIENA(Scalable Internet Event Notification Architecture)[**?**]. SIENA's topology is static. The subscriptions are propagated along the the topology to the publisher. and the subscriptions are stored along the path. The publications are delivered in the reverse path.

Routing Performance	For one subscription it is the diameter of the tree.
No. Of Messages	For one subscriber it is the order of the diameter of the graph
Routing Table Size	proportional to the number of subscriptions in the descendents
Overhead on node Addition and deletion	the node addition and deletion has to ensure that the static overlay topology is maintained.
Fault Tolerance	The system is not adaptable to faults
Node Failure	The node failure has is not handled by the SIENA
Link Failure	Link failure can jeopardize the system as there are no redundant paths.
Traffic adaptability	The optimization depending upon various metrics is not possible due to static overlay
Requirement for global Knowledge	not required
Scalability	Scalable
Correctness of algorithm	correct

1.1.4 Rebeca

Rebeca(Rebeca Event Based Electronic Commerce Architecture) also uses a static architecture. The basic routing is similar to SIENA but the subscriptions are stored using covering and merging techniques.

Routing Performance	For one subscription it is the diameter of the tree.
No. Of Messages	For one subscriber it is the order of the diameter of the graph
Routing Table Size	Subscriptions are stored in a different format. Uses covering, merging techniques to store subscriptions
Overhead on node addition and deletion	should maintain the properties of the static topology overhead depends on the choice of the node. depends on the number of nodes already present in the network
Fault Tolerance	The faults are tolerated by leased subscriptions.
Node Failure	no specific details
Link Failure	cannot adapt to link failures
Traffic adaptability	cannot adapt to traffic due to static topology
Requirement for global Knowledge	not required
Scalability	Scalable
Correctness of algorithm	correct

Chapter 2

Experimental Setup

2.1 Simulation Environment

The Performance evaluation of the algorithms is done on a simulator. The simulator is developed in Java.

The simulator allows the user to create a logical network with nodes and links. The links are of one of the three types(secure ,urgent,normal).The simulation of the network is done by simulation events.(node,link addition and failures,messages).The simulation module contains the classes which processes the simulation events. Each event is processed as a separate thread. the events are processed in the order of the global time stamp which is attached to each event.

The logical network module implements the functionality of the logical node and logical link. This module contains several interfaces which allows different implementations of nodes and links.Routing is also a part of logical network module.The routing Interface provides the flexibility to define different routing algorithms.With each broker node zero or more clients can be associated. EventBroker and EventClient modules define the respective functionalities.The clients can have two kinds of roles either a publishing role or a subscribing role. The roles module defines these roles. The subscriptions are specified as the filter (Expression on the attributes of the Publishing event).The applnEvents contains the classes that process the filters and checks for matches.Finally the evaluation module measures the performance in terms of routing table size, node processing load and data traffic and control traffic.

Appendix A

Appendix