**Chapter 1**

**Introduction**

In pub/sub system subscribers are the users who give their choice in the form subscriptions and providers of information publish events called publishers. Broker also called server matches subscriptions with published events in filtering process. Respective events are sent to interested subscribers. There are two main forms of filtering. First one is Topic based, messages are published to "topics" or named logical channels. Second is content based, where messages are received to subscriber only if the attributes or content of those messages match constraints given by the subscribers. Scalable Interface Event Notification architecture (SIENA) is content based pub/sub system and focuses on distributed architecture. SIENA is free software, first released in 1998. It is available in both C++ and Java language. We need to set up server to connect to publisher and subscriber. There are two matching algorithms in SIENA system. First one is SFF (Siena Fast Forwarding) and second one is covering algorithm.

SFF is counting algorithm and focuses on predicate matching. Message matches constraints if it contains an attribute with the same name and value, and if the value matches the condition specified by subscriber.

* 1. **Problem Statement**

To analyze SIENA content-based publish/subscribe system and existing matching algorithms and to reduce matching time by parallelizing existing matching algorithm.

* 1. **Objectives**
* To study SIENA pub/sub system.
* To develop existing SFF algorithm by parallelizing it.
* To analyze and compare existing sequential matching algorithm and developed parallelized algorithm.

**1.3 Terms and definitions**

**1.3.1 Events:**

An event, also called message or publication is conjunction of attribute value pairs published by publisher. Ex. M= {a1, a2…, am} where M is message and a1,a2…,am are attributes, number of attributes is denoted m.

**1.3.2 Primitive Constraints:**

Primitive constraint is a condition given by subscriber on an attribute. For Ex. C={less than, greater than, equal to, not equal to, any}.

Subscription: subscription is the expression who gives the interest of subscriber.

**1.3.3 Event Matching:**

Given a set of n subscriptions S= {s1, s2, s3, …, sn} and an event e, event matching is to find all subscriptions from S that match e.

**1.3.4 Matching Time:**

Matching time is defined as time running time needed by matching algorithm to match event against subscriptions.

**Chapter 2**

**System Analysis and Design**

This chapter contains the system requirements of different modules of the system. Along with this, it provides the feasibility check for the system.

* 1. **System Requirements**

**For Localhost:**

* Machine: Intel Core i3-3227U CPU @ 1.90GHz x 4, 4 GB RAM
* Operating System: Ubuntu 15.04
* Language: Java(JDK 1.8) and CPP
* Platform: Siena 2.0.4(For Broker), Siena 0.4.3(For publisher and subscriber)

**For Distributed clusters on 2-4 machine**

* Machine: Intel Core i3-3227U CPU @ 1.90GHz x 4, 4 GB RAM
* Operating System: Ubuntu 15.04
* Language: Java(JDK 1.8) and CPP
* Platform: Siena 2.0.4(For Broker), Siena 0.4.3(For publisher and subscriber)
  1. **Feasibility of the project**
     1. **Operational Feasibility:**

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The proposed system is operational due to the following reason. The developed integration uses SIENA (Scalable Internet Event Notification Architectures) architecture. Subscribers requests for the events. Servers matches that events according to subscriptions specified by subscriber and sends only matched events to interested subscribers.

* + 1. **Technical Feasibility:**

In technical feasibility, we study all technical issues regarding the proposed system.

* + 1. **Economic Feasibility:**

It is evaluation of development cost weight against the ultimate income or benefit derived from the system. Economic justification includes the reusability of composites developed. The Siena distributed system is open source content-based system. The hardware and software requirement is already setup. Hence the access cost incurred is negligible.

* 1. **Architectural Design of the system**

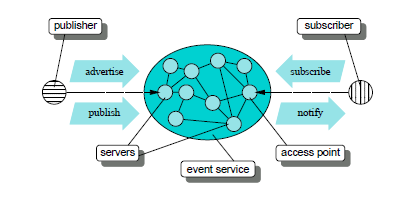


Fig.1 Siena Architecture diagram

Fig. 1 shows Siena architecture. The architecture includes three main composites:

1. subscriber
2. Broker
3. Publisher
   1. **Justification for proposed system**

A content-based pub/sub system consists of subscribers (data consumers), publishers (data providers), and an overlay network of brokers. The publishers input events into the system while the subscribers issue their subscriptions to the system. The core of a content-based pub/sub system is to propagate events from the publishers to the interested subscribers as quickly as possible. For a large-scale content based pub/sub system, there may be millions of clients, including both subscribers and publishers

Event matching is a fundamental component in a content-based pub/sub system. A path for delivering events is composed of consecutive routes that are bridged by intermediate brokers at which event matching is performed. To minimize the queuing time parallelism is achieved using executer services. This will increases the quality of service.

* 1. **Design Aspect**
     1. **Sequence diagram**

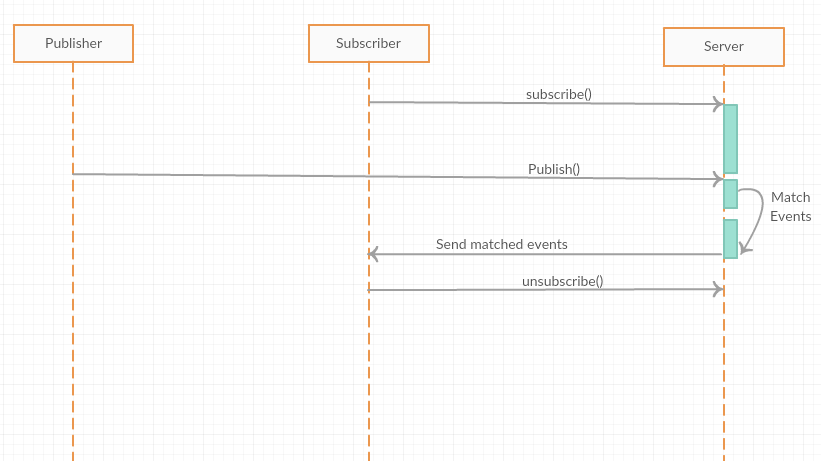


Fig. 2 sequence diagram

**Description:**

Subscribers requests for the events. Servers matches that events according to subscriptions specified by subscriber and sends only matched events to interested subscribers.

* + 1. **Use case diagram**

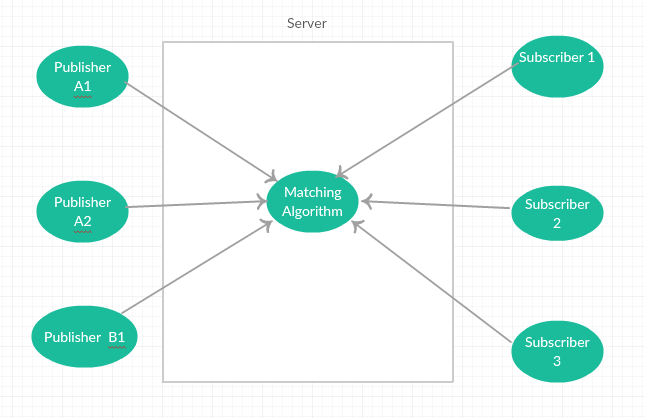
****

Fig. 3 use case diagram

**Description:**

It shows the relationship between the user and the different use cases in which the user is involved. It also shows how different Web Services interact with each other.

* + 1. **Class Diagram**

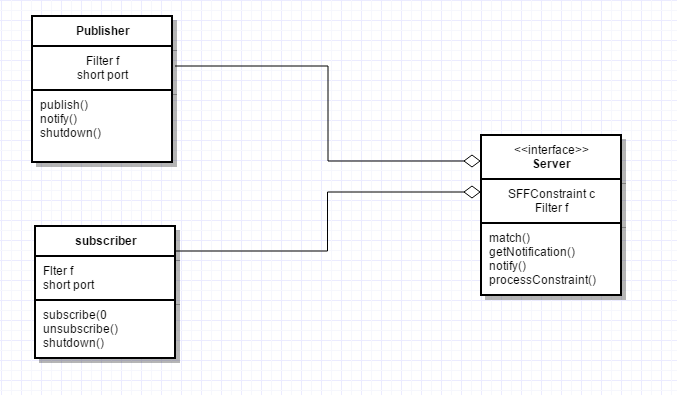
****

Fig. 4 class diagram

**Description:**

Class diagram shown in Fig. 4 describes the static structure of the system.

**Chapter3**

**Methodology**

This chapter explains the overall methodology which explains how different modules of the system need to be implemented. The overview of the system developed is provided in this chapter.

* 1. **g++ compiler**

Installation of g++ compiler is required to compile the pub/sub programs.

To install g++ compiler we have to write commands on terminal.



Firstly go to root and then update the package list with the most recent version of g++ enter your password and press ‘Return’.

Then do

to install the g++ compiler.

* 1. **SIENA distributed system**

*S*iena is a publish/subscribe event notification service. As depicted in Figure 1, in the *S*iena architecture two main entities may be distinguished: clients and servers. The servers are interconnected as a distributed network in order to achieve a scale suitable for large numbers of communicants and high volumes of notifications.

Servers provide clients with *access points* offering an extended publish/subscribe interface. This interconnection of the servers represent the *event-service*. Servers that have no clients other than other servers act as pure routers.

The clients are of two kinds: *publishers*, which are the generators of events (notification), and *subscribers*, which are the consumers of notification; of course, a client can act as both a publisher and a subscriber.

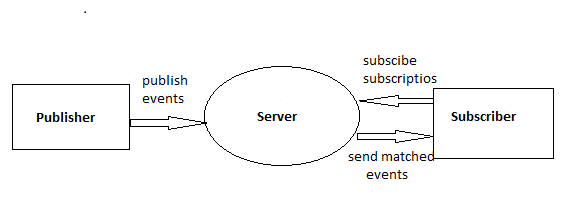


Fig.5 pub/sub system

Publishers use the access points to publish the notifications and subscribers use the access points to subscribe for notifications of interest by supplying a predicate, called filter. A filter is applied to the content of notifications and it allows subscribers to select the events they are interested in. Siena is responsible for selecting the notifications that are of interest to subscribers and then for delivering those notifications to the subscribers via the access points.

**Chapter 4**

**Implementation and results**

This chapter contains the detailed implementation of the system. This chapter contains the respective results of the implemented modules.

**4.1 Matching algorithm:**

SFF: SFF is content based counting matching algorithm.

|  |
| --- |
| Algorithm Counting  //Input=Message m  //Output=Matched Message  Map<filter, int>counters set to null  Set<interface>matched set to null  //use parallelism by using executor service  //for n number of constraints, n executor //services used.  {  Add matching\_constraints(a) to the set <constraints> C  For (each c in C)  {  For (each f in c.filters)  {  If (constraint not matches)  {  If no of constraints not equal to counter  No increment in counter  }  Increment counter  If(size of counter equal to size of filter)  //Message matched  Output Message m  }  } |

Siena Fast Forwarding algorithm is implemented in sequential manner. It matches each individual attribute sequentially with constraints specified by subscriber. Due to this, it takes more matching time with increasing number of subscriptions. To improve the matching time, we focus on parallelization of matching algorithm. For parallelization, we use Executor service packages. Packages of executor service are in-built in java.

**4.2 Results**

If we use Siena Fast Forwarding algorithm for event matching, following results found:

Fig. 6 graph of constraints vs time

In this system for each attribute it matches with every constraint. If attribute matches with constraint it increases the count and when number of constraints is exactly equal to count then message matches with subscription. In sequential Algorithm for n number of constraints loop executes n times. So with increasing no. of constraints matching time increases. As shown in figure matching time for parallel algorithm for same no. of constraints is less than that of sequential algorithm. But in parallel algorithm time remains constant for any no. of constraints because for n no. of constraints n executor services are used and they work in parallel manner. So even if we increase no. of constraints there is slight change in matching time which is negligible.

Fig. 7 graph of subscriptions vs time

Siena pattern follows conjunction of constraints and disjunction of subscriptions. For sequential algorithm time increases with increase in no. of subscriptions. Because if we increase the no. of subscription that may increase in no. of constraints. So matching time increases. In the above graph matching time for parallel algorithm is less than that of Sequential algorithm. Because in parallel algorithm executor services are used for each no. of constraint.

**Chapter 5**

**Conclusion**

**5.1 Conclusion**

In this project, we have presented our experience in analyzing the SIENA distributed system. SIENA is open source. It is free to change behavior of system. Our experience focuses on issues: understanding working of SIENA, studying and analyzing matching algorithms, developing matching algorithm for better performance. We developed existing SFF algorithm by parallelizing it. It gives better performance. So, we provided a formal description of SIENA and existing matching algorithms.

**5.2 Future work**

Future work is on the providing prioritized and deadline aware matching algorithm.

References

[1] ShiyouQian, Jian Cao, Fr\_ed\_eric Le Mouel, Minglu Li, Jie Wang. Towards Prioritized Event Matching in a Content-based Publish/Subscribe System. ACM. 9th ACM International Conference on Distributed Event-Based Systems (DEBS'2015), Jun 2015, Oslo, Norway.

[2] MauroCaporuscio, Paola Inverardi, PatrizioPelliccione . Formal Analysis of Clients Mobility in the SIENA Publish/Subscribe Middleware

[3] AntonioCarzaniga, Alexander L. Wolf. Forwarding in a Content-Based Network

[4] Satyen Kale, EladHazan, Fengyun Cao, Jaswinder Pal Singh. Analysis and Algorithms for Content-based Event Matching.

[5] www.inf.usi.ch/carzaniga/siena/software/java2/index.html

[6] www.inf.usi.ch/carzaniga/siena/software/java2/index.html

[7] www.inf.usi.ch/carzaniga/siena/software/cpp/index.html

[8] www.inf.usi.ch/carzinaga/siena/forwarding/index.html

[9] www.inf.usi.ch/carzinaga/siena/forwarding/doc/index.html