International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 7 4521 - 4525

Optimal Customized Content Dissemination for Rich Content Format in Pub/Sub Framework

Mr. Nikhil Banait
PG Student, Department of Computer Engineering
Alard College of Engineering and Management
Savitribai Phule Pune University
Pune, India
nikhilbanait@gmail.com

Prof. Sonali Patil
Professor, Department of Computer Engineering
Alard College of Engineering and Management
Savitribai Phule Pune University
Pune, India
sonalin69@gmail.com

Abstract:- Publish-Subscribe system is a message passing system which categorized into two types i.e. topic based system and content based system. The publisher is the sender who is responsible for deciding the classes or topics of publish messages to which subscribers can able to subscribe. Subscriber is a receiver who will receive all messages published to the class to which they subscribe. A content based publish-subscribe framework that delivers the content by matching constraints to the subscribers into their required format. Such framework enables the publish-subscribe system to adapt richer content formats including larger text files containing huge number of events to be published with different properties and other content. In Customized Content Dissemination, user's i.e. consumers in addition to specifying their information needs, also specify their profile information which includes the device characteristics used to obtain the content. Our pub sub system as being responsible for matching and distributing the published content, also responsible for converting the content into the desired format for subscribers.

Keywords- *Publish, Subscribe, CCD, Rich Content Formats, Customized Content Dissemination.*

*****

I. INTRODUCTION

Today millions of users are subscribing to content over a network by setting sets of filtering rules. Web applications, such as Twitter, Facebook and RSS feeds, enable users to specify their interests and subscribe to updates provided by information publishers. They usually leverage a Publish/Subscribe infrastructure, which enables distributed components to subscribe to the event notifications (or-events, for simplicity) they are interested to receive, and to publish those they want to spread around. Fig. 1 shows Publish/Subscribe system [2].

A. Models

There are two models of pub/sub systems: topic-based and content-based [3].

Topic-based Model: Events are grouped in topics, i.e. a subscriber declares its interest for a particular topic to receive all events pertaining to that topic. Each topic corresponds to a logical channel ideally connecting each possible publisher to all interested subscribers. The main drawback of the topic-based model is the very limited expressiveness it offers to subscribers. A subscriber interested in a subset of events related to a specific topic receives also all the other events that belong to the same topic.

Content-based Model: Subscribers express their interest by specifying conditions over the content of events they want to receive. In other words, a subscription is a query formed by a set of constraints composed through different operators. Possible constraints depend on the attribute type and on the subscription language. Most subscription languages comprise equality and comparison operators. In content-based Publish/Subscribe, events are not classified according to some predefined criterion (i.e., topic name), but rather according to properties of the events themselves.

B. Requirement of customization in publish/subscribe framework

Publish/Subscribe (pub/sub) systems provide a selective dissemination scheme that delivers published content only to the receivers that have specified interest in it.[3] To provide scalability, pub/sub systems are implemented as a set of broker servers forming an overlay network. Clients connect to one of these brokers and publish or subscribe through that broker. When a broker receives a subscription from one of its clients, it acts on behalf of the client and forwards the subscription to others in the overlay network. Similarly, when a broker receives a produced content from one of its clients, it forwards the content through the overlay network to the brokers that have clients with matching subscriptions. These brokers then deliver the content to the interested clients connected to them.

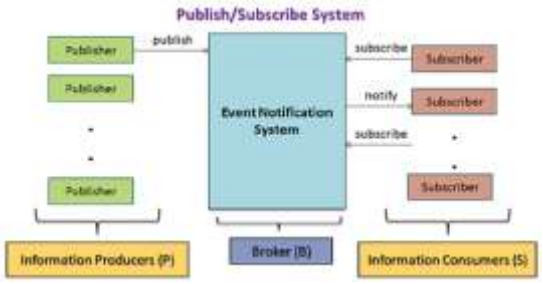

Figure 1. Publish/Subscribe system.

Here, we present our work on customized content dissemination (CCD) and extend it to address the effect of heterogeneity and concurrent publications in the system. We consider the problem of customized delivery in which clients, in addition to specifying their interest also specify the format in which they wish the data to be delivered. The broker network, in addition to matching and disseminating the data to clients also customizes the data to the formats requested by the clients. As the published content becomes richer in format, considering

content customization within the pub/sub system can significantly reduce resource consumption. Such content customizations have become more attractive due to recent technological advances that have led to significant diversification of how users access information. Emerging mobile and personal devices, for instance, introduce specific requirements on the format in which content is delivered to the user. Consider a distributed video dissemination application over Twitter where users can publish video content that must be delivered to their followers (subscribers). Followers may subscribe to such channel using a variety of devices and prefer the content to be customized according to their needs [3].

Additionally, device characteristics such as screen resolution, available network bandwidth, etc. may also form the basis for required customization. Another example of such customized content dissemination system is dissemination of GIS maps annotated with situational information in responding to natural or manmade disasters. In this case, receivers may require content to be customized according to their location or language. Simply extending the existing pub/sub architectures by forcing the subscribers or publishers to customize content may result in significant inefficiencies and suboptimal use of available resources in the system. Therefore, there is a need for novel approaches for customized dissemination of content through efficient use of available resources in a distributed networked system. The key issue in customized content dissemination using distributed pub/sub framework is where in the broker network should the customization be performed for each published content? An immediate thought is to perform requested customizations at the sender broker prior to delivery. Such approach could result in significant network cost [3].

Consider a simple broker in Network from Fig. 2 where node A publishes a file in 'pdf' format and nodes G, H, and I have subscribers that requested this content in 'txt', 'doc' and 'otf' formats, respectively. By performing customizations in sender broker, A, the same content is transmitted in three different

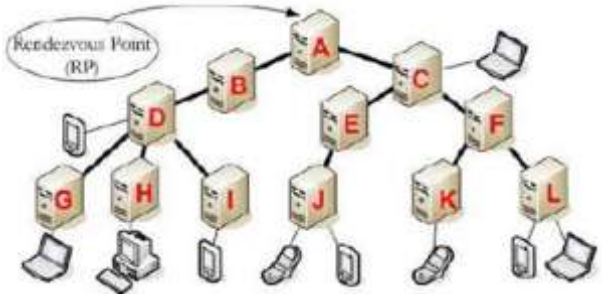

Figure 2. Simple dissemination tree

formats through < A : B > and < B : D > links which results in increased network cost. The alternate might be to defer customizations to the receiver brokers or broker D. Consider another case where J, K and L have subscribers with hand held devices that requested the file in 'txt' format. If the customizations are deferred to receiver brokers, conversion from 'pdf' to 'txt' is done three times, once in each receiving broker which results in higher consumption of computation resource in brokers. This also increases the communication cost by transmitting larger size file in 'pdf' format while it could be transmitted in 'txt' format that has smaller size.

II. LITERATURE SURVEY

A. *Overview of Publish/Subscribe system*

In software architecture, publish-subscribe is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers. Instead, published messages are characterized into classes, without knowledge of what, if any, subscribers there may be. Similarly, subscribers express interest in one or more classes, and only receive messages that are of interest, without knowledge of what, if any, publishers there are. Pub/sub is a sibling of the message queue paradigm, and is typically one part of a larger message oriented middleware system. Most messaging systems support both the pub/sub and message queue models in their API, e.g. Java Message Service. This pattern provides greater network scalability and a more dynamic network topology [7].

B. *Elements of a Publish/Subscribe system*

Publish/Subscribe systems, shortly named pub/sub systems establish a connection between publishers (producers) and subscribers (consumers) of events, behaving as a mediator between publishers and subscribers. This way, publishers are decoupled from subscribers; they do not need to be aware of each other. Publishers submit events to the Publish/Subscribe system which is responsible for notifying the interested subscribers. Subscribers specify the events they are interested in to the Publish/Subscribe system through a subscription language. The goal of pub/sub matching is to notify a potentially large set of subscribers about matching events quickly (in terms of latency) and efficiently (in terms of throughput) .In such systems, the matching component may easily become the bottleneck of the Publish/ Subscribe infrastructure. On the other hand, in several applications, the performance of the Publish/Subscribe infrastructure may be a key factor. As an example, in financial applications for high-frequency trading, a faster processing of incoming event notifications may produce a significant advantage over competitors. Similarly, in intrusion detection systems, the ability to timely process the huge number of event notifications that results from observing the operation of a large network is fundamental to detect possible attacks, reacting to them before they could compromise the network.

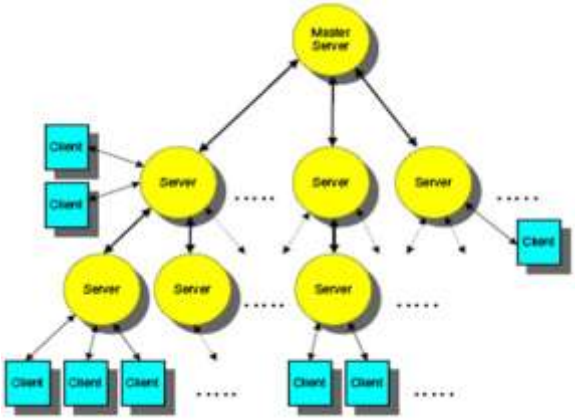

Figure 3. Siena Architecture.

ISSN: 2321-8169
4521 - 4525

C. *Siena Content based publish/subscribe system*

Siena (Scalable Internet Event Notification Architectures) is a research project aimed at designing and constructing a generic scalable publish/subscribe event-notification service. The technical basis of SIENA is an innovative type of network service called content-based networking. Siena is content based publish subscribe system [8].

The SIENA API exists as a prototype for Java (and C++), developed mainly by Antonio Carzaniga and Alexander L. Wolf. A network of servers acts as access-points for clients to publish and subscribe for notifications. The servers will deliver notifications of interest to the clients, and receive and forward publications. Fig. 3 shows architecture of Siena publish/subscribe system. It is hierarchical architecture contains many number of servers called as brokers. These brokers are responsible for filter out events. Client can register to any broker system. It can act as publisher as well as subscriber. Broker receives, filters, and sends events to another broker as well as client system In Siena the pub/sub paradigm states a minimum of functionality. SIENA seeks to optimize and extend this approach and make it more suitable for wide area networks [8].

Existing research works in publish subscribe system mainly focus on many aspects of content-based systems like filtering and routing of events. One of them is to provide a selective dissemination scheme that delivers published content only to the receivers that have specified interest in it. The major hurdle in it is to correct placement of conversion operator. Also publish subscribe system are widely used for transfer of multimedia file. As every portable device does not have same characteristics, there is need of converting multimedia files from one format to another. So we have attempted the customization of content according to device characteristics and operator placement algorithms for content based Publish/Subscribe systems.

III. PROPOSE WORK

A. *Problem Statement*

Design and develop publish/subscribe framework with customization at each node and find the valid customization plan with minimum total cost

B. *System Architecture*

Content based subscriptions systems are an emerging alternative to traditional publish subscribe systems, because they permit more flexible subscriptions along multiple dimensions. In these systems, each subscription is a set of predicates which may test arbitrary attributes within an event. However, the matching problem for content based systems, determining for each event the subset of all subscriptions whose predicates match the event, is still an open problem. But main focus is not on implementation matching algorithms instead delivering such matched contents to user as per their requirements i.e. in suitable format. Delivering the results in required format with minimum possible cost is challenging task. Implement such algorithm which will provide all possible results at correct end with minimum cost. Cost is generally measured in terms of time taken to deliver a matched event. Fig. 4 shows architecture of Propose System [9].

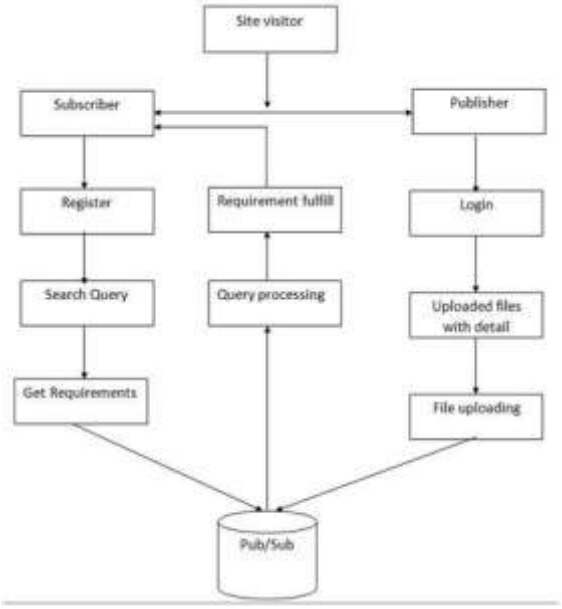

Figure 4. Architecture of Propose System.

1) Publisher: Publisher are used to implement this system are Siena publisher. Siena Publisher can be any system in the network. User can register to any machine available in network as a publisher. For registration, user required to provide necessary details including ip addresses, port numbers. Once user successfully registered, system is ready to accept file to be published. This file can be in any format, publisher loads file to corresponding server or broker. Then broker forwards this data to publish subscribe filter. To register to the system, publisher has to enter

tcp:10.10.7.252:1534

Here Transmission Control Protocol (TCP) used for communication between brokers and clients. Client has to provide brokers IP address and port number to establish a communication channel. Once publisher is connected to broker, now it is ready to upload a file and publish events present in that file. File can contain any number of events. File containing events a structured file, here each word has specific meaning based on their position. Example: Consider a following entry from a input file

Microsoft 1200 open 1400 closed 21/06/14
Amazon 2200 open 2300 closed 22/06/14

Here first word represents company name, next price at which stock for a day opened and next number represents price at which stock closed at that date.

2) Subscriber: Subscriber are used to implement this system are Siena subscriber. Siena subscriber can be any system in the network. User can register to any machine available in network as a subscriber. For registration, user required to provide necessary details including ip addresses, port numbers. Once user successfully registered, system is ready to proved file as a result. This file can be in any format subscriber has to maintain a file format. Then broker retrieves data from publish subscribe filter. To register to the system, subscribe has to enter.

tcp:10.10.7.252:1534 pdf

Here Transmission Control Protocol (TCP) used for communication between brokers and clients. Client have to provide brokers IP address and port number to establish a communication channel. Subscriber also has to provide a parameter indicating a file format. The results of matched events will be provided in this format only. File containing result is a structured file, here each word has specific meaning based on their position. Example: Consider a following entry from a input file

TCS 800 open 850 closed 21/06/14
TCS 850 open 900 closed 22/06/14

Here first word represents company name, next price at which stock for a day opened and next number represents price at which stock closed at that date. This result file contains only matched events.

3) Query Processing: Query processing is a middleware between publisher and subscriber. And it is responsible for filter out unmatched events. Query processing usually works on entire data provided by user. It contains filter and matching algorithm.

4) Filters: Filter is used to finding matching subscriptions from arrived publications. Binary operators are used in Siena filters. A Filter represents a Boolean expression to be evaluated against an Notification. It consists of a set of constraints. Each constraint poses a condition on an attribute of the Notification. A Filter can pose multiple constraints on the same attribute. A Filter is implemented by means of a multimap < string, AttributeConstraint >, therefore a Filter can be manipulated using all the methods of a standard multimap. Two additional utility methods are provided to add constraints. Example of filter constrain applied by a subscriber

f.add_constraint ("stock", SX_eq, "TCS");

Here, f is a object of filter. This constrain will return a publication events having name 'TCS'. SX_eq is a operator checks equality. Other operators are SX_gt (greater than) and SX_lt (less than).

5) Customization: Customization usually makes results acceptable by the registered subscribers. Customization is applied at middleware. It creates document supported by registered devices and writes results in it.

Algorithm: Customization of documents.
Input: A structured text file, containing large number of events to be published. Subscribers registered with required file format.
Output: Matched events should be delivered to subscriber in file format specified by subscribers while registration and time delay should be minimum.
Step 1: INPUT: matched event set
Step 2: F is Set of formats in which the content is received.
Step 3: CF is Customized File
Step 4: R ← F.getReceivers()
{Subscribers and their requested formats.}
Step 5: if (R > 1) then
CF ← F
Step 6: else
Step 7: F ← F + new F
Step 8: end if
Step 9: for all F do
CF ← F
Step 10: End

In above algorithm, all events that are to be published are in a single file. This file can contain any number of events. This file is kept and forwarded to broker systems. If any subscribers are registered to system with appropriate constrains, then events are filtered out and only matched events are kept in data structures. If number of subscribers is large beyond that broker system asking for same format then required file is created and forwarded to all client systems. Else data structures as it is forwarded to the client system instead of sending result file, and required result file is created at client system.

IV. Mathematical Model

We associate two cost measures with a customization plan: Conversion cost and Transmission cost. Conversion cost of a plan is the sum of costs of carrying out the operators specified for each of its nodes and transmission cost is the sum of costs of transmitting the content in the specified formats over all the links in the dissemination tree. We denote the conversion cost of a plan P by Cp and the transmission cost by Tp.

The total cost of the plan P for content C is denoted by P(C), as a function of its conversion and transmission costs. In general one can use an additive formula such as:

$$P(C) = \alpha Tp + \beta Cp \qquad \text{where } \alpha, \beta \geq 0.$$

The parameters α and β in the above cost function provide flexibility to customize the total cost function based on the system characteristics. For instance, if processing resources in a system are limited and expensive, the total cost function can reflect this by giving more weight to computing cost.

TABLE I. Observations of Objective

No	Event Generated	Matched Events	Delivered Events	Time with Customization (Minimized)
1	150	5	5	0.141
2	500	15	15	0.410
3	2000	62	62	1.213
4	8000	248	248	2.343
5	20000	520	520	5.061
6	50000	1040	1040	9.623
7	100000	2080	2080	16.203
8	150000	3120	3120	34.795

V. Experimental Result

We created a simple text file contains only 150 numbers of events, out of 150 only 5 are matched to registered subscriber's query. For correct working, system should deliver all these 5 matched results to subscribers.

We executed our system number of times (minimum 10), and observed number of events delivered and time taken to deliver these matched results and then computed average time of all the 10 readings. It is 0.141 seconds. Similar procedure

 ISSN: 2321-8169
4521 - 4525

repeated for further observations, with number of events to be published increased vastly (up to 150000). Time taken is observed and mentioned in Table 1.

Fig. 5 X-axis represents matched events. Y- axis represents time taken in seconds to deliver these matched events. Here blue line represents time taken to deliver these results with applying customization, orange line represents time taken to deliver these results without applying customization and gray line represents minimized time taken to deliver matched result in an appropriate file format. As, we can see gray line is very close to orange line, indicates that time is minimized and hence delay is minimized to create and deliver a result file to subscribers.

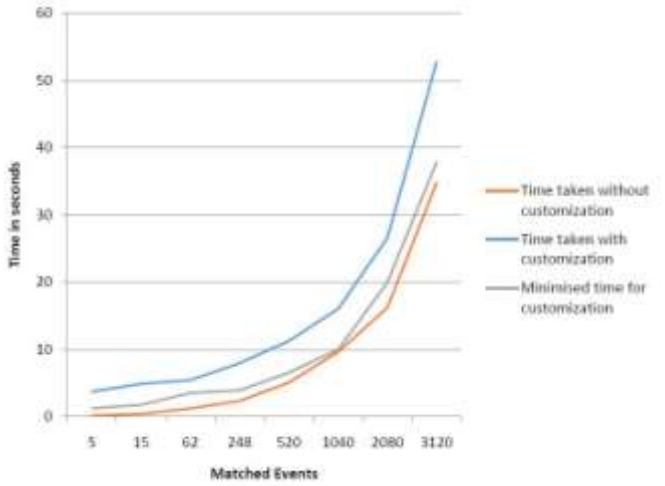

Figure 5. Time taken to delivered result.

VI. Conclusion and Future Work

This paper shows customized content dissemination (CCD) system, where content is only delivered to receivers that have requested it and in their desired format. Operator placement algorithms on top of content based pub/ sub framework to customize content format such that dissemination cost, which we defined transmission (communication) costs, is minimized. Transmission cost can be minimized by transferring data structures to the broker machines instead of transferring entire file.

Siena distributed Pub/Sub System provides a loosely-coupled, asynchronous model which is useful in many fields of network utilization. Several areas are still open for research like Effective routing and filtering algorithms for better performance, Fault tolerance, Security and hence these are the disadvantages carried in this customized content dissemination system.

This system needs to be extended on multiple operating system platforms like Windows, Android, iOS etc. to bring it on large number of mobile devices.

Acknowledgement

I am very thankful to my guide Prof. Sonali Patil for the constant guidance and support. I would like to thank all the faculties who have cleared all the major concepts that were involved in the understanding of techniques behind this paper.

References

[1] S. Castelli, P. Costa, and G.P. Picco, “HyperCBR: Large-Scale Content-Based Routing in a Multidimensional Space,” Proc. IEEE INFOCOM, 2008.

[2] I. Aekaterinidis and P. Triantafillou, “PastryStrings: A Comprehensive Content-Based Publish/Subscribe DHT Network,” Proc. IEEE Int’l Conf. Distributed Computing Systems (ICDCS), 2006.

[3] H. Jafarpour, B. Hore, S. Mehrotra, and N. Venkatasubramanian, “CCD: Efficient Customized Content Dissemination in Distributed Publish/Subscribe,” Proc. ACM/IFIP/USENIX Int’l Conf. Middleware, 2009.

[4] F. Cao and J. Pal Singh, “MEDYM: Match-Early with Dynamic Multicast for Content-Based Publish-Subscribe Networks,” Proc. ACM/Usenix/IFIP Int’l Conf. Middleware, 2005.

[5] G. Li, V. Muthusamy, and H.A. Jacobsen, “Adaptive Content-Based Routing in General Overlay Topologies,” Proc. ACM/Usenix/ IFIP Int’l Conf. Middleware, 2008.

[6] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J. Kubiatowicz, “Tapestry: A Resilient Global-Scale Overlay for Service Deployment,” IEEE J. Selected Areas in Comm., vol. 22,no. 1, pp. 41-53, Jan. 2004.

[7] Hojjat Jafarpour, Bijit Hore, Sharad Mehrotra, and Nalini Venkatasubramanian, “CCD: A Distributed Publish/Subscribe Framework for Rich Content Formats” ieee transactions on parallel and distributed systems, vol. 23, no. 5, may 2012

[8] Ruben Wangberg, Jørgen Frøysadal, Håvard Tegelsrud, “Presentation of an pub/sub implementation”

[9] Marcos Aguilera, Robert Strom, Daniel Sturman, Mark Astley, Tushar Chandra, "Matching Events in a Content-based Subscription System"