Event-Based Communications for Equipment Supervisory Control

Dan Hong Zhang, Jing Bing Zhang, Yi Zhi Zhao, Ming Mao Wong
Singapore Institute of Manufacturing Technology
71 Nanyang Drive, Singapore 638075
dzhang@simtech.a-star.edu.sg

Abstract

Event-based approaches are becoming more and more popular for building large scale distributed systems and applications, especially in the Internet environment. In this paper, the concept and techniques of event-based communications are first discussed. A hybrid approach of event-based communications for real-time manufacturing supervisory control is then proposed together with an "Event-based Communication Agent" (ECA) framework to provide extendable and reliable communications, as well as efficient development method in multi-OS, multiple types of network protocol environment. A case study involving an industrial implementation for a large-scale distributed supervisory control system based on this approach is then discussed.

1. Introduction

In computer science, event-based systems are built in terms of distributed components that communicate through the exchange of events. As event-based communications provide asynchronous, loosely coupled and point to multi-point communications, they are suitable for many different application domains. It is not surprising to find many researchers and software vendors investigating and developing event-based communication middleware for the integration of a wide range of components in a loose-coupled distributed environment [1].

Similarly, in the distributed manufacturing environment, there are a lot of subsystems and applications that are distributed across heterogeneous computing platforms and over different networking environment. The communications between different systems have always been one of the most critical issues in integration and often the most difficult portion to be implemented due to the complexities involved. Most of the existing event-based middleware are mainly used for large-scale distributed systems, especially in the Internet environment. Though they might not be used directly in manufacturing control systems, which typically include multiple types of computer and equipment systems and connected in LAN or with equipment specific protocol, we found that the concepts are suitable for distributed manufacturing control systems, which also require loose-coupled communications between sub-systems.

In this paper, we review the event-based communication techniques, discuss and compare the various approaches, and then propose a hybrid approach for applying event-based communication in the manufacturing supervisory control systems. A case study of actual industry implementation based on this approach is then presented.

2. Event-based Communications

The concept of event-based systems involves the abstraction of the exchanged information between different components into events, and the generalization of data transfer between components into event handling services so that all components work together harmonious in a loosely coupled environment. Here, the "event" is more like a container object to carry data from the producer to the consumer rather than a specific status data. The primary aim of "event-based communications" is to investigate the generic methodology on how to carry the right event to right consumer.

In the past decade, many techniques have been reported and each has demonstrated specific advantage(s) in certain application circumstances.

2.1. Usage Paradigms

Generally, there are two popular event-based system usage paradigms:

Queuing paradigm provides a point-to-point communication model to ensure reliable data transfer in an orderly way. The event producer places the event/message in a queue, and the consumer retrieves the event from the queue. Typical examples include Microsoft Message Queue (MSMQ) Server and IBM WebSphere Messaging Queue. They are also referred to as Messaging-Oriented Middleware (MOM).

Publish-subscribe paradigm provides many-to-many communication model where the events are disseminated across a large-scale distributed system. The

0-7803-9402-X/05/$20.00 © 2005 IEEE

event producer first publishes the event by type and topics. Each consumer specifies its interest by means of "subscription", and later receives the events of interest via event notifications. Typical examples include CORBA Notification Server (CORBA-NS)[2], and Siena (Scalable Internet Event Notification Architectures) [3].

2.2. Information Dissemination Approaches

A fundamental issue in a publish/subscribe system is the dissemination, for example, how to match the interests of subscribers with the available messages from publishers. The information dissemination approaches for publish/subscribe systems could be mainly classified into topic-based, content-based, etc.

In **topic-based** system, events are classified as belonging to one of the fixed set of topics, like newsgroups. The publishers label the events they produced to a particular topic, while the consumers express their interests by joining a topic. When an event related to that topic occurs, it would be broadcasted to all related consumers. In real systems, the topics could also be groups, channels, or subjects. SCRIBE [4] is a scalable application-level multicast infrastructure, supports a large number of groups (topics) and members (consumers).

In **content-based** system, the event producer publishes the event with predefined attributes. The subscribers (consumers) are given the ability to express their interests by specifying the predicated values of the attributes. The subscription could also be a query, or in terms of filter or pattern. The advantage of a content-based system is that the subscribers have the flexibility to choose filtering criteria along multiple dimensions of the information spaces, instead of being limited to pre-defined topics.

In addition to topic-based and content-based systems, the Semantic MOM for publish/subscribe system is also reported [5]. The DARPA Agent Markup Language and Ontology Inference Layer (DAML+OIL) is used as a topic description language and the DAML+OIL reasoner has been implemented for the instance checking and process matching between the subscriptions of subscribers and the advertisements of publishers.

2.3. Distribution Models

The typical event distribution model could be centralized, hierarchical client/server, or peer-to-peer, as summarized in [6].

For event-based communication, the event service could be **centralized** in terms of centralizing the middleware while the producers and consumers are still distributed. All event sources send their events to the single proxy, which then looks up the subscriptions and forward the events to consumers. Obviously, this model applies simple routing logic and more likely to be used for small-scaled distributed systems. The single proxy would be the bottleneck as the system scales up, with potential single point of failures.

In the multi-proxy **hierarchical** architecture [7], all event sources and receivers are connected through hierarchical proxies, where the registration and subscription information are maintained. Each proxy is responsible for the event dissemination and failure handling between its parent proxy and child proxies/nodes. This method could simplify and reduce the complexity of large-scale system routing in event distribution, especially for those systems that are logically organized in hierarchical control architecture.

Peer-to-Peer (P2P) systems and applications are distributed systems without any centralized control or hierarchical organization, where the software running at every node is equivalent in functionality. The connection of nodes is flexible; therefore the whole system is scalable and self-organized [8].

Distributed Hash Table (DHT) technique [9] is commonly used in P2P event-based systems since it provides the infrastructure for efficient, scalable resource lookup in large P2P distributed network. The system maintains a few tables for events, and their attributes, including types, names and constraints on possible value. Each event subscription is represented in a table with an event attribute name and required value range. Each node allows multiple events to be published, and each event allows subscriptions from different nodes.

2.4. Notification and Transmission Models

The newly occurred event could be broadcasted or pushed from the producer or be actively pulled by the consumer in certain case. Which method to adopt depends on the nature of the event as well as the communication protocol and the supporting infrastructure.

Push notification [10] is the ideal model for producers to transmit the information to consumers as when it is ready. The implementation of this model may not always feasible, it relies on the multicast infrastructure. It is most suitable for cases where the events are very critical to the consumers but occurs irregularly and infrequently. The event is actively pushed to the necessary consumers immediately upon the occurrence of the event. This push notification provides quick and effective distribution for the event. A key concern for push notification or unsolicited report is that the connection between producer and consumer has to remain open and available. As such it is seldom used for all events in the system as this would occupy too much system resources, even when it is supported by the infrastructure.

If push notification is not supported by low-level infrastructure, regular producer (server) status or critical events checking could be configured as **pull** operation from the consumers. But the high data accuracy and timeliness can only be achieved at the cost of high

pulling rate, which induces high network traffics. Active pulling is also applied in cases where the consumer needs the event only in certain periods of time, or when the event is required only when requested by the consumer.

A combination of Pull and Push notifications for different situations in the system, such as push for event notification and pull for actual event transmission, may lead to a better solution and higher system performance.

Even though many research works have been published in event-based system and many event-based middleware have been developed [2,3,4,11], to build an application in specific domain is not as simple as purchasing off-the-shelf software and run. Some sophisticated solutions, such as CORBA-NS, provide a large set of features and services, but do not provide the services for specific application domains. In addition, the real application environment might not allow us to use the event communication infrastructure required by these solutions. Nevertheless, the distributed system communication for specific domain could still be developed systematically and more efficiently with the guide and the concept of the event-based systems.

3. Event-based Communication in Equipment Supervisory Control System

Most of the manufacturing equipment control systems consist of a large number and different types of equipment. Typically they are controlled by one or more high-level supervisory controllers. Two-way communication is required between different equipment, as well as between equipment controller and supervisory controllers. Equipment level communication can be Fieldbus, DeviceNet or other specific protocol, depending on the application environment. In contrast to higher level Internet-based inter-application or inter-system communications, equipment level communication is more complex due to multiple protocols and the special requirements for each manufacturing environment, especially when some legacy systems need to be integrated.

On the other hand, flexibility and extendibility are required for the manufacturing equipment control system since the number and type of the equipment may be changed during production extension or re-organization, control features may also be added or enhanced. Hence, the communication control must be re-configurable and extendable.

In comparison with the traditional event-based communication for Internet applications, the relationship between supervisory controller and equipment is more "tightly coupled". For example when supervisory controller sends control command to equipment, it may need to get feedback of the acknowledgement or execution result from equipment controller. Different command may request different type of feedback. It would be very time consuming to handle all these communication individually. By separate the communication problem with the control logic, above "tightly coupled" application logic could be implemented in "loosely coupled" manner, whereby both the supervisory controller and equipment can function more independently, reliably and efficiently.

Event-based communication method is beneficial for generalizing the large distributed system communication solution and we believe its concept could be applied to manufacturing equipment supervisory control systems.

3.1. The Model – Event-based Communication Agent (ECA)

From the control point of view, the relationship between supervisory controller and equipment controller is master and slave. But from data communication point of view, supervisory controllers send control commands to equipment whenever necessary, while the equipment report process status to supervisory controllers as and when needed. Both sides initiate (produce) events for others and handle (consume) events from others.

In most cases, the equipment and the controller reside on different types of computer. Their roles and functions as well as processing speeds are also different. Asynchronous communication is necessary, and queuing mechanism is suitable for equipment communication.

To generalize the communications between supervisory controllers and different equipment, we introduce the concept of Event-based Communication Agent (ECA). The ECA is logically located at each controller or equipment. It provides a common queuing mechanism, event subscription control and event distribution capability, as well as error handling feature so that the communication to any other equipment/controller could be handled properly regardless the contents of the information. All node-to-node communication is handled by ECA.

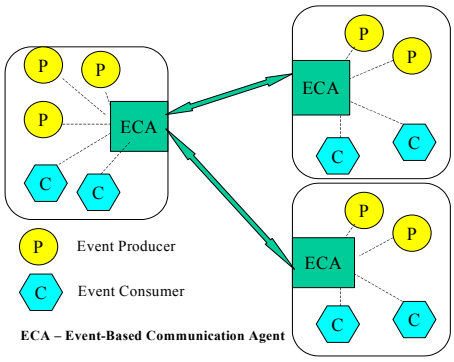


Figure 1. Event-based Communication Agent

Here the *Event* is defined as the manufacturing fact, which is created by the *Producer* module and may need to be transferred to other nodes and handled by the *Consumer* processes. It is an abstracted object and carries useful information, rather than a specific status or data. Events are defined in uniformed way, with a Node ID, an Event ID (or event type) and other attributes as described in the Event Master Table (EMT) shown in Figure 2. The attributes are defined in same data type but with different meaning for different type of event.

Events are generated from producer modules, they are sent from equipment controller to the supervisory controller (or vis-à-vis). They will then be distributed, based on the event condition (or content), to different modules in the controller, for further processing. These event-handling modules are defined as final consumers.

The ECA plays roles of both Event Transmission and Event Reception, as described in the next two sections.

3.2. Event Transmission

Event transmission manages the event publication and distribution process.

Figure 3 shows the event-based communication between node A and node B. One dispatching queue is created and maintained on Node A's ECA for all event transmission from Node A to Node B if it is required. The ID for the channel to transmit this Queuing event is Q_{AB}, it could also be considered as the queue ID. All related information to a queue is maintained in queue master table (QMT) as shown in Figure 2. Some status data, such as "Enable Flag" would be set dynamically based on actual situation (connection status, node mode, etc.).

The event distribution table (EDT, in Figure 2) represents information pertaining to events publication and subscription, which is used for queuing management. The conditions for the event distribution can be defined as logical statements with the operation of event attribute.

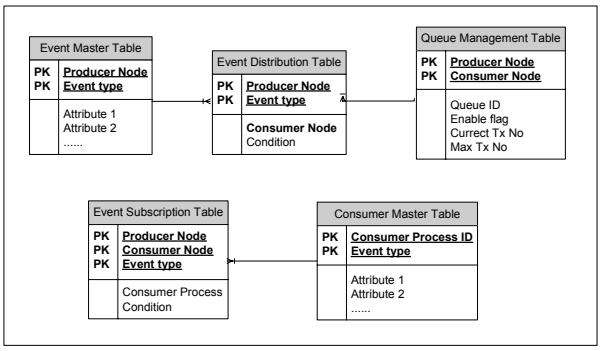


Figure 2. Data Model for ECA

The basic relationship between producers and consumer nodes are described by the EMT, QMT and EDT tables. These are configured in system deployment stage but the conditions and status are allowed to be dynamically changed. When the system is running, the events are generated by event producers. The queuing manager (as shown in Figure 3) looks at the event content and refers to the EDT to decide which consumer nodes it should distribute the event to. It then pushes the event to the related queue if the checking for QMT shows the enable status for the communication channel.

The Queuing Manager of producer node would then notify the consumer node when new event is ready for it. The confirmation response from consumer side is also necessary to ensure the asynchronous queued events to be sent successfully. The Queuing Manager then prepares the next event to be posted to the consumer node.

3.3. Event Reception

Event Reception processes include subscripting events from other nodes, collecting subscripted events when it occurred, and forwarding the events to the related consumer modules.

If node B subscribes events from node A, there will be one QMT record at Node A, and some EDT records are maintained in the Node A EDT table. At node B the reception manager would create one receiving instance to keep looking at the event queue between A and B (which is created and managed by node A). When a new event is notified from A, Node B reception manager will collect the event posted from A and feedback the completion status to the Q_{AB}.

There is an Event Subscription Table (EST) maintained by Event Subscription Manager as shown in Figure 2. When an event is received, the EST is used as a lookup table for the Reception Manager (as shown in Figure 3) to invoke the consumer process after checking the contents of the event.

The overall relationship between two nodes communication is shown in Figure 3.

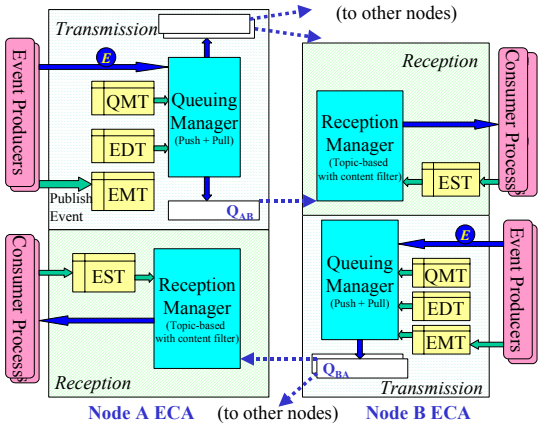


Figure 3. Event Communication between Nodes

3.4. Discussion

In above sections, a hybrid approach that combines the event queuing and publish/subscribe methodologies is described for communication in supervisory control, and ECA model is proposed. The queuing mechanism is adopted for the physical transmission of event information between nodes. The publish/subscribe concept is used in managing the event distribution, based on the lookup tables for describing the relationships between event producer nodes and consumer nodes. The ECA model introduced here offers the flexibility and extendibility to the system. A generic transmission and reception management mechanism handles all output and input events.

The data model described in Figure 2 can be implemented as database tables, data files, or look up tables, depending on the implementation environment. All these can be built during the event publish/subscribe stage or system design and deployment stage. New relationships as well as new producers and consumers can be added by inserting records in the database tables. Dynamic change and on-line re-configuration are also possible. The communication system would be easily extended when new control logic is required or new node is added in the system.

Content-based method is used in the event distribution logic, with filtering feature in queuing management. Similar concept is used in the logic of event forwarding to consumer in the reception manager. The event distribution condition can be defined as the relationships between event attributes. It would be evaluated at run time and used for controlling the event flow.

In supervisory control system, most of the operational events should be handled according to the sequence of occurrence, such as the vehicle picks up a box, moves to another location, then drops the box. The queue therefore can be used as a buffer mechanism for asynchronous communication to ensure the events are distributed and handled orderly. Another advantage of creating dedicated queues (channels) between nodes is to reduce the inter-channel interference. For example, the error or problem in the communication between node A and B, would not affect or delay the data transmission between node A and C.

Error handling is always a critical issue in communication. In the queuing manager, a transaction number is attached with actual event transmission and used as indicator of the progress. Error will be logged and retry will be issued if transmission fails. All these are handled in queuing manager and reception manager, hence transparent to individual event producers and consumers. This method would provide more reliable communication comparing with coding error handling in individual event producer and consumer processes.

The event notification from producer nodes to consumer nodes could be configured in different way. Push method for the event transmission can be adopted if unsolicited report function is supported; otherwise, it is practical that the consumer nodes regularly check the queuing status from producer nodes, and pulls the event from the queue when it is ready to be transmitted. However, in actual situations, not all information between nodes fit into the publish/subscribe paradigm. Event consumer may need to request the event on demand. In this case, ECA model is still applicable in terms of a special queue, with a maximum length of one. The latest data are always ready for the pulling from the consumer.

Due to the constraints of specific industrial environment (operating system, network, available driver, etc.), the implementation of above model may be carried out in different ways. In certain circumstances, the simplified model can be used for solving the actual problem more effectively.

4. Industry Case Study

4.1. Application Background

For a large-scale warehouse management, we built an inventory control system with different types of material handling equipment that are controlled by a large number of PLCs. The 3 supervisory controllers (SC) connect to different groups of equipment controllers (EC) and provide supervisory control, track & trace, as well as equipment monitoring and diagnostic features.

A group of legacy PLCs could only work under the proprietary Ethway token ring network environment. In order to integrate them with the supervisory controllers in Ethernet environment, a Gateway Station (GS) is introduced to enable communications in both Ethway as well as Ethernet environment. It helps to forward the events between these two network protocols. The network architecture is shown in Figure 4.

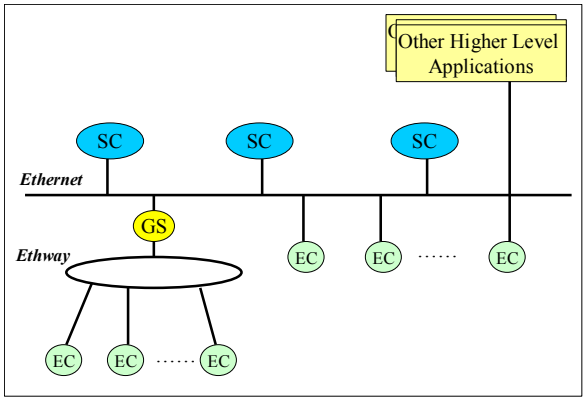


Figure 4. Network Architecture

As typical supervisory control systems, the SCs send control commands, or job instructions to ECs. In return, the ECs report the current equipment operation status,

equipment error status, job progress, or completion status to SCs. SC sends different types of commends to different types of equipment, and different types of equipment may generate different types of status for reporting. On the other hand, for same type of ECs, SC generates similar type of control commands to them and gets similar types of reports from them. On average, one SC needs to communicate to 100 PLCs and produces about 10 types of commands and one EC could communicate to 2 SCs and produces 20 types of status information. In order to handle the large amount of control related information, a generic methodology was applied for reducing the development workload, and the Event-based Communication was adopted for this supervisory control system communication.

4.2. The Solution

As shown in Figure 4, each equipment controller (EC) and supervisory controller (SC) as well as Gateway Station (GS) is considered as one node in a distributed control system. Each type of information communicated between different nodes is considered as one Event. The event-based communication agent (ECA) at each node is responsible for all event distribution and reception related to the node.

Regardless the actual meaning of supervisory control information, this Event-based Communication methodology unified the communication process so that different types of ECs and SCs can communicate in similar ways.

In this system, the primary purpose of Gateway Station is event transfer. So the consumer process for the received events from ECs becomes the queuing manager for the events transmission to the SCs. Conversely, the reverse event flow works similarly.

4.3. Implementation Issues

In this system, the equipment controllers and Gateway Station are PLCs so the EST, EMT, EDT, QMT are implemented in the PLC memory locations. Some of the distribution and filtering features are built as sub-routines due to the constraints of PLC programming. Unified queuing and reception management have demonstrated the advantages of ECA in terms of isolating the communication and control functions in clear software structure and simplifying the development work.

The supervisory controller is implemented on PC in MS Windows environment. Oracle database provides the flexible and re-configurable event management environment. A SCADA software tool is used to build the supervisory control application. The object-oriented methodology is used to develop the template of transmission management, as well as reception management module. In the deployment stage, the configuration of events and consumer/producer relationship is implemented by adding records in database, the creation of event queue and queuing management logic are realized as adding the instance of the objects. This shows significant benefit in reducing the development work.

Due to the unstable network in this manufacturing environment, additional checking and error handling logic are applied. In comparison with Internet-based applications communication, the loss in connection poses more serious problem in real-time supervisory control. If no notification is received for a long time, we need to differentiate whether the two nodes are disconnected or there is no new event. Hence, regular heartbeat checking of the producer-node is introduced in the consumer-node reception manager. On the other hand, a timeout control is also implemented in the queuing manager when waiting for the response from consumer nodes after posting the notification. The checking result is also fed back to the QMT for the event distribution control.

4.4. Results

This implementation shows the advantage of ECA solution for large-scale supervisory control system.

The generic ECA method separates the event generation, event transfer and event handling process. The Event Communication Agent focuses on the reliable transferring of events between controllers, while the event producer and consumer modules in the controllers focus on generating and processing the events without worry about how the events to be transferred.

The event relationships between controllers are described as tables (EMT, EDT, QMT, etc.) which are similar to the concept of DHT in P2P publish/subscription event communication. This leads to the re-configurability and extendibility of the event communication in supervisory control system.

The events transfer between controllers through unified queue. The generalized queuing management and error handling shows the advantage in reducing the development efforts in the case of large amount controllers and many types of events.

Compare with traditional tightly coupled, dedicated message communication between controllers, the ECA solution may not be faster. But with the increasing of controller and network speed as well as adopting the unsolicited event push notification in queuing management, the ECA solution may achieve near-real-time performance which should be sufficient for the large-scale supervisory control system. In our implementation, about 99% of events could be transferred within 1 second. The average transferring time is about 50ms. On the other hand, the developing efforts were reduced to about one forth of traditional method and the testing effort had also been reduced significantly due to reliability of the unified event communication agent.

5. Conclusion and Future Work

In this paper, we presented a hybrid approach that combines the event queuing and publish/subscribe methodologies for supervisory control of distribute sub-systems. The Event-based Communication Agent (ECA) is proposed as a generic communication solution for the nodes of the supervisory control system. Data models and function models were discussed in detail, and issues relating to flexibility, extendibility, re-configurability are considered in the model.

The generalized models and solution can help facilitate faster development of reliable and extendable communication system for supervisory control. An industrial case study was discussed which indicated that the proposed method can be effectively used as the communication solution in large-scale distributed supervisory control systems.

The concept of event-based system is becoming more and more important for building large-scale distributed systems and applications. In the manufacturing environment, along with the standardization of equipment communication in supervisory control such as OPC, event-based systems can also be implemented as middleware, and used in many applications.

Another interesting area that we are investigating is to apply ECA based communications to ad-hoc wireless sensor network, which is data-centric in nature and requires self-organizing and self-administrating. Compared with internet based distributed system, wireless sensor network requires more flexible routing, more efficient data transmission and data streaming with the consideration of energy conservation and data storage constraints.

References

[1] David S. Rosenblum and Alexander L. Wolf **"**A design framework for Internet-scale event observation and notification" *Proceedings of Sixth European Software Engineering Conf./ACM SIGSOFT Fifth Symposium on the Foundations of Software Engineering*, Sep. 1997, pp. 344-360.

[2] P. Gore, R. Cytron, D. Schmidt and C. O'Ryan, "Designing and Optimizing a Scalable CORBA Notification Service", *Proceedings of the 2001 ACM SIGPLAN workshop on Optimization of middleware and distributed systems,* pp 196-204, June18, 2001, Snow Bird, Utah, United States

[3] A. Carzaniga, D. S. Rosenblum and A. L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service", *ACM Transactions on Computer Systems,* Vol. 19, No. 3, August 2001.

[4] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron., "SCRIBE: A large-scale and decentralized application-level multicast infrastructure", *Journal on Selected Areas in Communication,* vol. 20, Oct. 2002.

[5] H. Li and G. Jiang, "Semantic Message Oriented Middleware for Publish/Subscribe Network", *SPIE 2004 Defense and Security Symposium*, Florida, April 12-16, 2004.

[6] A. Carzaniga, D. S. Rosenblum and A.L. Wolf, "Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service", *Proceeding of the 19th ACM Symposium on Principles of Distributed Computing (PODC 2000*), pp219-227, July 16-19 2000, Portland, OR.

[7] H. Yu, D. Estrin, R. Govindan, "A Hierarchical Proxy Architecture for Internet Scale Event Services", *Proceedings of the 2nd International Workshop on Internet-Scale Event Notification,* June 1999

[8] P. Triantafillou and I. Aekaterinidis, " Content-based Publish-Subscribe Over Structured P2P Networks", *the Proceedings of the Third International Workshop on Distributed Event-Based Systems (DEBS 2004).*

[9] D. Tam, R. Azimi, H. Jacobsen, "Building Content-Based Publish/Subscribe Systems with Distributed Hash Tables", *Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing*, September 2003.

[10] M. Hauswirth, M. Jazayeri **"**A component and communication model for push systems", *ACM SIGSOFT Software Engineering Notes , Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering.* Pp 20-38, Sept 6-10 1999, Toulouse, France

[11] Silva Filho R. S., De Souza C. R. B., Redmiles D. F. "Design and Experiments with YANCEES, a Versatile Publish-Subscirbe Service", Technical Report UCI-ISR-04-1. University of California, Irvine, April 2004.