Autonomous Decentralized System with Event Service for Information Services

Stephen S. Yau, Ning Dong, and Fariaz Karim
Computer Science and Engineering Department
Arizona State University
Tempe, AZ 85287-5406, USA
Email: yau@asu.edu

Abstract

Autonomous Decentralized System (ADS) has the properties of online expansion, online maintenance and fault tolerance, which are among the important features for next generation information services. Since information services require transfer of large amounts of data with different levels of Quality of Services (QoS), it is necessary to have an ADS architecture to satisfy these requirements. This approach uses event service as the communication middleware and ATM network to handle large amounts of data and satisfy various QoS requirements.

Keywords: Autonomous Decentralized System, information services, event service, ATM network, filtering, QoS.

1. Introduction

ADS has been successfully applied in various areas [1,2], primarily related to control systems. In order for ADS to be applied to next generation information services, it needs the following features:

a) The user has access to diverse types of services through one system even though these services have different functionality and various QoS [3].
b) Not only textual information, but also multimedia information are accessible through one system.
c) Service suppliers and consumers can be added and removed easily.
d) Each user has his/her own view of the information services and all the actions of a user are usually confined within the user's own virtual information space.

Accordingly, the ADS system requirements for information services are:

1) Different types of services are combined over a network, but they are autonomous.
2) The amount of data to be transferred is very large.
3) Different QoS requirements need to be satisfied by the system. Even for the same data, different users or the same user in different environments will need different QoS.

The current ADS systems [1-5] use Data Field (DF) as a communication layer and all the information is broadcasted around the DF. An Autonomous Control Processor (ACP) [1,2] receives all the data broadcasted regardless whether it needs the data. DF architecture can satisfy the data communication requirements for those applications that are not information intensive, such as manufacturing control systems, transportation control systems, or those information systems connected to control systems [3]. For information services, where the transfer of large amounts of data is needed, the network traffic is a big problem even when broadband networks are used. Currently, ADS systems are built on Internet layer on top of TCP or UDP layer running on Internet Protocol (IP) [2,4]. The services provided by the Internet layer are not sufficient to satisfy the necessary QoS for information service applications [6,7].

In this paper, we will present an approach for ADS to support the information services using event service and ATM network to satisfy various QoS requirements.

2. Our Approach

In our approach, we use event channels [8] and ATM network to construct the communication backbone of the ADS system. Multiple event channels are connected to each other through ATM network to form a network of channels. User programs communicate with each other through event channels. Figure 1 shows the topology of the communication backbone, where the solid links indicate direct connections and the dotted links represent connections through other event channels. Applications are self-regulated because event service provides de-coupled asynchronous group communication. The communication between user programs is through sending and receiving information. Hence, there are two roles with respect to each application modules: supplier and consumer. Suppliers provide information

and consumers process information. When a supplier needs to send information, it delivers the information to the event channel without telling the recipients of the information. The information is transferred in the form of event, which is attached with event type. The event channels broadcast the information to all the consumers that need the information according to the event type.

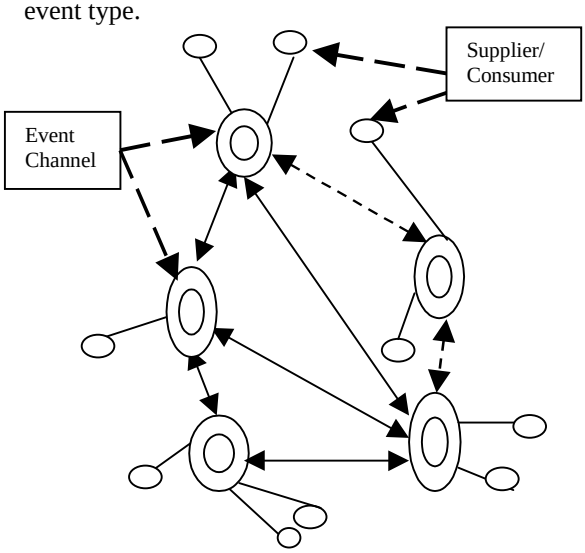

Figure 1: The communication backbone in our approach.

they want by registering the types of events. The information will be broadcasted among event channels. Whenever an event channel receives an event from a supplier, it will transfer the event to the consumers that are connected to it and already registered for this type of event. In addition, the event channel will also deliver the event to other event channels that are connected to it if some consumers of this type of event are connected to these channels.

Our approach can satisfy various QoS requirements through ATM network and event service. When consumers register to the event channels, they can also tell the event channel the desired QoS of the information. While the consumer's desired QoS is broadcasted, Virtual Circuits (VCs) between channels are set up. To satisfy multiple QoS, event channels maintain multiple VCs. An individual VC is responsible for providing a single QoS type. This arrangement is necessary to preserve the semantics of each QoS type. Information will be transferred along VCs so that different QoS will be satisfied. Event service performs aging of events by deleting expired events and provides priorities of events by transferring events to other channels according to the priority of events.

To reduce the network traffic, event channels use filters [9] to prevent the propagation of information to unnecessary destinations. In our approach, the consumers tell the event channels what information

Figure 2 shows the ADS architecture of our approach. Application modules send messages to and receive messages from event channels through event service interface. Filters are used to propagate events based on user interests in order to reduce network traffic. QoS requirements are satisfied through QoS management at the event channel level and through VCs at the ATM network level. From the perspective of other ADS system implementations [1-4], our architecture can be viewed as a system where each ACP is connected to multiple DFs simultaneously, and each DF has different QoS characteristics.

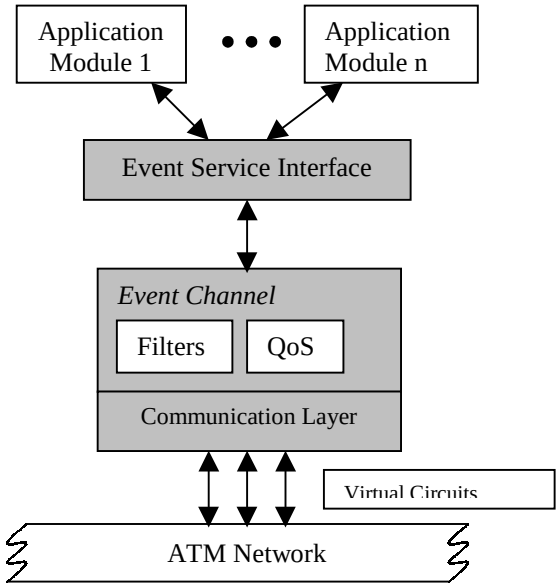

Figure 2: Our ADS architecture using event channels and ATM network.

3. Communication Backbone

Equality, locality and self-containment are the three requirements of ADS components [1]. As one of the asynchronous group communication, event style communication [8] can satisfy the suppliers and consumers with these three requirements. For information services, each supplier or consumer should be self-regulated and hence information services also need the appropriate communication mechanisms to provide suppliers and consumers with equality, locality and self-containment.

In our approach, we use event service to serve as the communication middleware, which is responsible for sending and receiving information between suppliers and consumers. Every supplier and consumer connects to event channels and all the information to be delivered is sent to the channel. The event channel will receive the information and then

send the information to the interested consumers that are connected to it and to other channels that need these information. So the consumers do not need to know the locations of the suppliers, the failure of one supplier or consumer will not affect the others, and new suppliers or consumers can be added easily.

Information is transferred in the form of event. The suppliers and consumers communicate with each other only through events. An event consists of event head and event data. The event data is the passing information. Even though the event data can be of any length and any type, the amount of data in each event should depend on the consideration of QoS. The event type and its QoS requirements are parts of the event head. Instead of specifying the recipient's address, the event type is used to identify the event. The supplier/consumer can dynamically specify QoS parameters while sending the event.

4. Filtering of Events

In information services, the amount of data transferred is very large and the communication governs the system scalability. One of our goals of using event service is to reduce the network traffic by filtering the events [10]. However, current CORBA event service specification [8] does not include the feature to reduce the network traffic. In this section, we will present a filtering technique to reduce data transfer in event service.

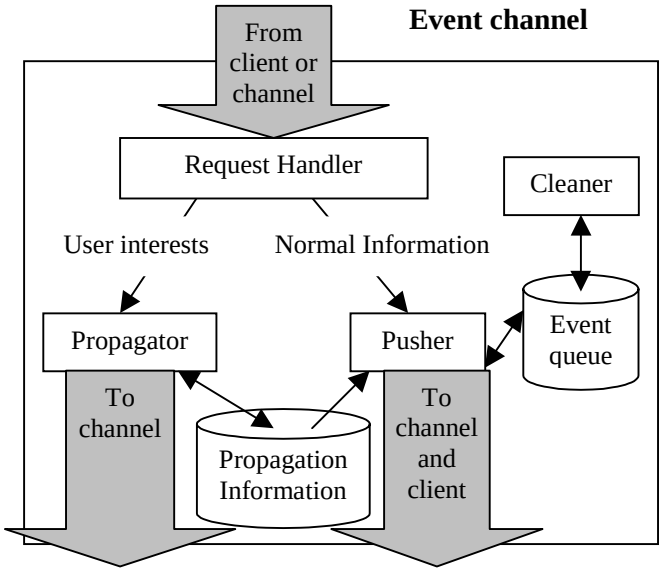

Figure 3: The structure of an event channel.

In our approach, we consider user interests in order to reduce unnecessary data transfer. Consumers express their interests by registering the interests with the channels along with desired QoS. A channel uses this information to propagate events to consumers.

The structure of an event channel is shown in Figure 3. When an event comes from a supplier, the request handler will distinguish the type of the event. If it is registration information about the consumer's interest, the propagator will set up the paths (VCs) from this channel to the consumer and save it as propagation information. Then, the channel will propagate the registration information to its neighbors. If the event is not registration information, the pusher will save it in the proper position in event queue according to its priority and then send it to other channels or interested consumers directly connected to the event channel. The cleaner is responsible for deleting those events that are older than their specified age limits.

For large-scale distributed systems, it is not realistic that each channel knows all other channels. We assume each channel knows only the addresses of its neighbors. The following subsections describe the filtering technique in our approach.

4.1 Setting up Propagation Paths

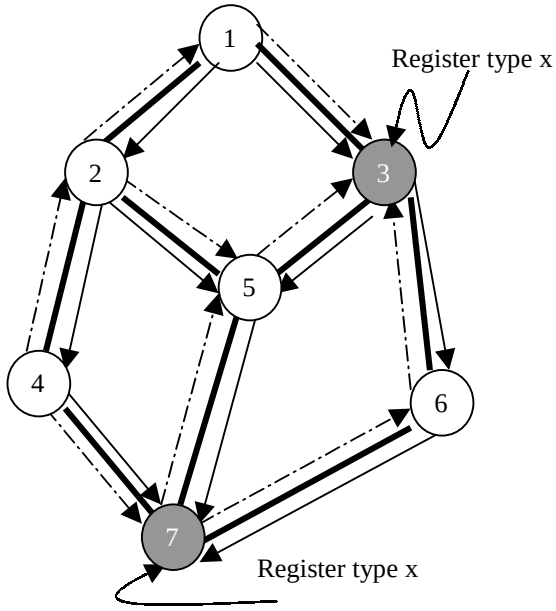

Figure 4: An example for setting up paths.

The propagator in the event channel is responsible for setting up paths between suppliers and consumers. When a consumer registers a type of event to its primary channel, the primary channel propagates the registration information to its neighbors. Each neighbor then notifies its own neighbors until all the channels receive the registration information. In addition, whenever a channel receives the registration information, it identifies all the shortest paths from this channel to the consumers interested in the corresponding event type. The shortest path means

that the number of hops of the path from the channel to the consumer is minimal. For one consumer, there may be several shortest paths from a channel. Consider the example shown in Figure 4. Assume that a consumer of channel 3 and a consumer of channel 7 send the registration information for type x events. In Figure 4, the thick line denotes that the channels on both sides of the line know each other. The solid thin lines represent the paths from other channels to channel 7. The dashed and dotted lines represent the paths from other channels to channel 3. Channel 2 keeps four propagation paths for events of type x: {1, 3}, {5, 3}, {4, 7} and {5, 7}. The first two paths are used for consumers connected to channel 3 and the rest is used for consumers connected to channel 7.

4.2 Selected Propagation of Events

After the propagation paths are set up, the pushers in event channels are responsible for propagating the events from suppliers to all interested consumers through the propagation paths. Since there may be multiple paths from a channel to consumers, we need to select a set of paths that reduce the network traffic. However, finding an optimal set of paths for reducing traffic in a large network remains a difficult and open research problem. Although several strategies, such as admission control, load shedding, and choke packets [11] address this problem, each of them brings its share of advantages and limitations. Keeping this in mind, we present here a procedure that provides a reasonable event propagation scheme in conjunction with other existing approaches.

1. A design-time analysis should be done on the kind of information that will propagate in the system. Based on the requirements, a set of categories must be determined, such that each category has different priority. These categories are global, and independent of the QoS specified by the consumers during the registration period.
2. Event channels maintain corresponding queues for each category, so that the volume of pending events of each category can be determined easily. In addition, a maximum watermark should be determined for each category.
3. Initially, the supplier's primary channel is responsible for delivering the event to all the interested consumers. The primary channel selects a set of relay channels that appear first on the propagation paths. Each relay channel then becomes responsible for delivering the event to a subset of all the interested consumers in such a way that as a whole the relay channels cover all interested consumers. Each relay channel in turn selects its own relay channels based on its own propagation paths. This procedure continues until all the consumers receive the event. If several alternative relay channels exist, the least loaded one is chosen. If a relay channel is not available, a neighbor channel is chosen as a replacement.
4. Normally, an event channel propagates events of all categories according to their corresponding QoS. If sustained arrivals of higher priority events cause the lower priority events to reach the watermark, the channel simply stops accepting any registration information for the corresponding category. In addition, using any of the congestion control strategies, it requests the neighbor channels to stop the delivery of events of this category. For the highest priority events, such as 911, the watermark can be set very high. Moreover, the channel dynamically upgrades the priority of the lower priority pending events by setting a timer, and start propagating them. The timer stores the length of the period for which the new priority is valid. The timer information is attached with the events, which also store the information related to the old priority.
5. The lower priority events promptly propagate through the network as long as the new priority is active. However, when the time is finished, it resumes its original priority. In this case, the events start following the specified QoS as normal. It also may accumulate in the queues. However, due to the dynamic priority upgrade for a specified period, the events move closer to the destinations even in case of a sustained high network load. The procedure continues until the events are either delivered or deleted due to the aging limit.

Using the above scheme, whenever the channel selects the next channel, it tries to minimize the time to send the event to the consumers according to the configuration and the current network load.

4.3 Addition and Deletion of Channels

Since channels may be added or removed, the configuration of the system may change and affect the propagation path set.

When a new channel is added, it will have its own propagation list through its neighbors. We need to consider whether the number of hops in the shortest paths will decrease when other channels send events passing the new channel. For each neighbor of the new channel, the new propagation paths containing the new channel will be identified and compared with the old paths. The shorter paths are selected and the procedure will continue until no new paths containing the new channel reduces the length of paths. Figure 5 shows the changes of paths after a new channel, channel 8, is added to the example shown in Figure 4. The shortest path from channel 1 to the consumers connected to channel 7 is changed to going through

channel 8. When a channel is removed, only its neighbors are notified and the propagation paths that include the channel to be deleted will be removed. If the channel to be deleted has some consumers, de-registration information will be propagated to all other channels, and those channels will delete the propagation paths containing the deleted channel.

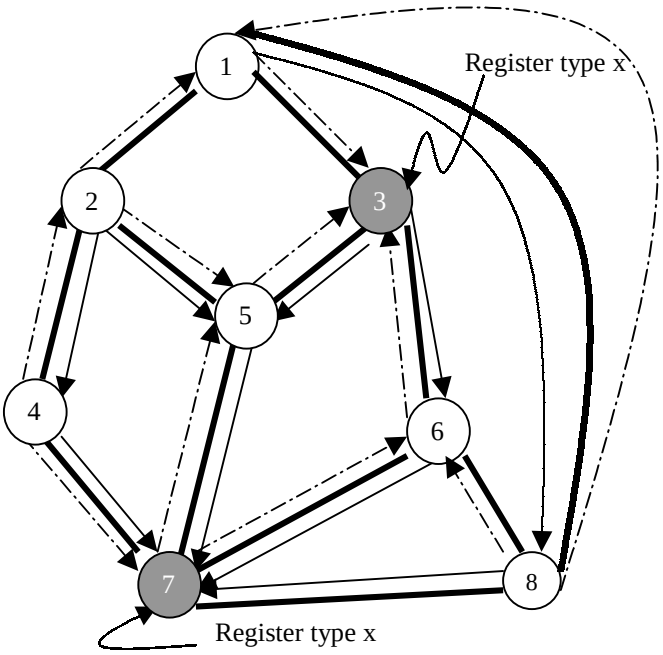

Figure 5: The propagation paths after channel 8 is added for the example in Figure 4.

5. Fault Tolerance

We incorporate fault tolerance in our ADS architecture at two levels.

At event service level, channel failures are tolerated by dynamically reconfiguring the connections between the consumers/suppliers and event channels. It is accomplished by the interface residing in the consumer/supplier or pushers in event channels as shown in Figure 6. The interface contains the list of event channels that can be physically connected to the application modules with their registration information. The application modules send events to the interface and the interface selects one physical channel that the application modules will be connected to. Whenever a channel breaks down, the interface is responsible for connecting the application modules to another working channel and sending the registration information to that event channel. On the other hand, the event channels keep the information of the addresses of other channels that they can be connected to. Each channel keeps a list of the paths from this channel to the consumers. Whenever an event channel wants to transmit an event and finds that it cannot communicate with one of its neighbor channels, it uses other alternative paths in the list. If all the paths in the list are not available, the channel can send the event to one of its neighbor channels and that channel is responsible for sending the event.

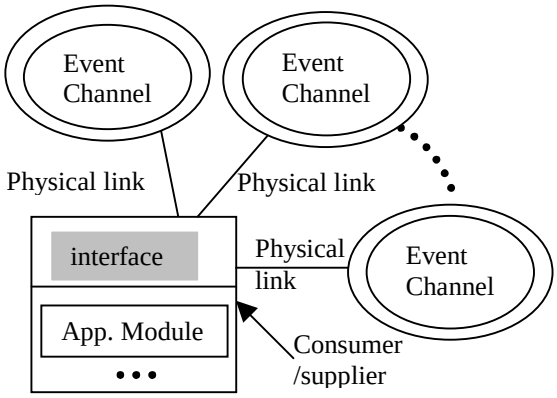

Figure 6: The interface between a consumer/ supplier and event channels.

At the ATM level, channel replication along with ATM group addressing feature [15] is used for providing transparent fault tolerance service. In this scheme, each replicated member of the channel group uses a group address in addition to an individual address. A message bound for a specific group address reaches any member of the associated group. If for any reason a member becomes unavailable, the message simply gets delivered to any of the remaining working members of the same group. The members themselves maintain a consistent state by exchanging messages using their individual addresses, which may be hidden from the information service applications.

6. Bandwidth Management

In order to efficiently use bandwidth and provide priority-based services, we use event service in conjunction with the ATM communication layer. Event channels use aging of events as a criterion to filter out unnecessary events [10]. For a highly loaded system, this approach can save a sufficient amount of bandwidth. Either the supplier or the consumer can specify the aging limit of an event. Since this approach takes the focus on the event channel level, it can be implemented regardless of the nature of the underlying network.

To provide prioritized services, event channels use priority queues. In addition, channels can use the underlying ATM communication layer to reserve and use VCs to provide end-to-end QoS. The QoS type of each VC can be determined as a function of the priority and the size of desired information.

Now, we would like to describe our communication layer, as shown in Figure 7, which has the main distinctions with the introduction of *QoS-aware Connectors* (QC). QCs can be independently configured to satisfy different levels of QoS within a system running various applications. To control the priority and invocation of the QCs, we use *Reactors* that take actions based on send or receive operations of new data or a priority change. Moreover, the communication layer incorporates a *Distributed QoS Manager*, which is used to monitor, specify, and control the types of QoS used across a collection of systems.

The main objective of this layer is to serve as a mediator between the event channels and the underlying ATM-based network in such a way so that both parties agree and abide by a particular QoS contract all the time. Our communication layer has five service components to satisfy this requirement, as shown in Figure 7.

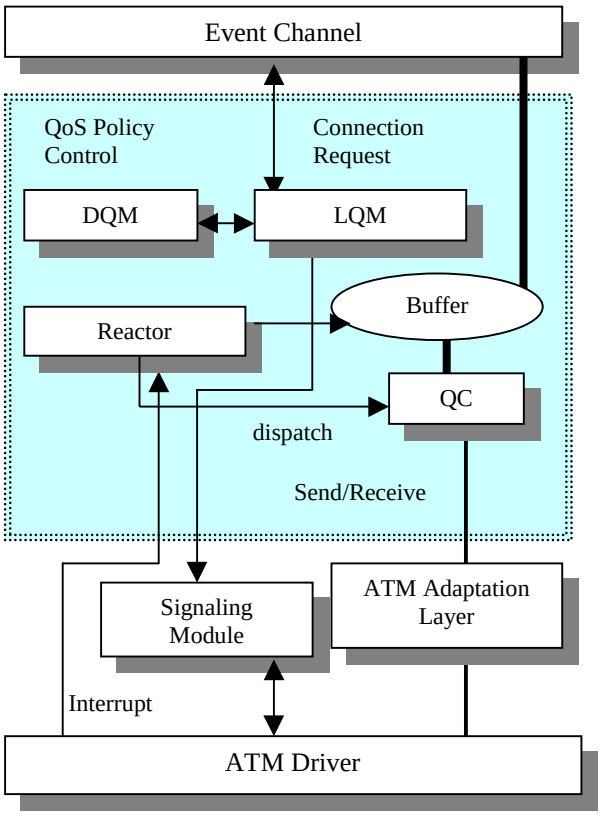

Figure 7: The communication layer between an event channel and ATM network.

Local QoS Manager (LQM): The LQM provides a microscopic view of the local system resources in terms of different QoS. It provides programmable interface to the event channels, which includes request for a new VC or reservation request for a specific amount of bandwidth. It is responsible for admission control mechanisms. By collaborating with the *Distributed QoS Manager*, its resource management policies stay consistent with the global policies, which may be enforced by the higher authorities. Although each request for a new connection goes through the ATM Call Admission Control (CAC) [15], LQM is still required to perform resource management on the end system's side. This enables a system to control a variety of QoS types on per process or thread basis. To be consistent with the CAC scheme, LQM maintains two main resources – bandwidth and buffer.

Distributed QoS Manager (DQM): The DQM provides a macroscopic view of QoS in a collection of systems, which may be governed by a specific group (e.g. the financial branch of a company). Using DQM, it is possible to employ QoS across a collection of channels. For example, it is possible to specify the maximum amount of bandwidth that can be used by all the PCs in an organization. DQM acts as a supervisor to LQMs to enforce the global QoS policies. As a result, if the individual QoS requirements of an application does not meet or exceed the enforced QoS (e.g. security level), LQM will deny or upgrade the request.

QoS-aware Connector (QC): Whenever a request for a connection is approved, a QC is created. Each QC is responsible for almost all aspects of a connection, including VC management, error correction, message retransmission, connection establishment after a failure, and runtime QoS enforcement. A QC communicates directly with the application. Each QC component maintains a QoS profile of a connection, which is used, for example, to provide a constant flow of traffic entering a VC or to maintain the QoS contract. It also provides traffic shaping operation for the applications that exceeds the QoS requirements. Since many VCs can be connected to one system, multiple QCs with different QoS profiles may exist. To provide security, QCs can use the public-key cryptography mechanism. The QCs are scheduled, started, and preempted by the *Reactor* component.

Reactor: A reactor can be considered as a manager for the QCs. It schedules and dispatches QCs whenever data arrives from the network to the channel or applications perform a send or receive operation. Based on the priority policies, it dispatches appropriate QC in order to process the data. In the simplest case, the priority can be based on RMA-like scheduling algorithms, where the priority is based on the frequency of send and/or receive operations.

Signaling Module: The signaling module is responsible for connection establishment and cancellation. In our implementation, we plan to use Q.2931 protocol [15]. A separate signaling module

results in a flexible architecture since it can be changed easily without changing other parts of the communication layer.

7. Interactive Communication

For highly interactive communication between a consumer and a supplier, it is not efficient if message is broadcasted. On the other hand, the autonomy of the ACPs should not be violated. In our approach, session management service from the underlying communication layer is used to enforce the autonomy and the efficiency.

Initially, the consumer registers for a specific event – a service event. Like normal events, consumers will compose an event registration by including the type of the desired service, which will uniquely identify a specific service category and the semantic information of the service. The consumer will register its interest with its primary channel, and the channel will broadcast the event as usual.

In response, several suppliers may be willing to provide the service. Suppliers can register their service types with their primary channels. The channels will send a *reply-to-service* event to the primary channel of the consumer. The primary channel will choose a supplier based on the order of reply or additional semantic information. Finally, a session will be setup between the primary channels according to an agreed QoS. A session can be realized by setting up a dedicated VC between the primary channels. The session will be torn down after the service is terminated.

8. Transaction-Based Communication

In our approach, a *transaction* is defined as a collection of events that are sent together and the automicity and order of events are preserved.

Atomicity means that either all the events will be received or none of the events will be received by the consumer. A transaction is considered as one big event and given an event number. When a supplier sends an event in a transaction, the event will be given a sequence number in this transaction. The last event in the transaction will also be given an ending flag. The consumer will wait until all the events in the transaction arrive. If any of them is missing, other events in that transaction will be discarded.

For some applications, all the receivers need to receive the events in the same order that they are sent. This can be done using ATM Point-to-Multipoint protocol and ATM cell sequencing techniques. Using the Point-to-Multipoint protocols, a single message can be sent to all the receivers transparently [15]. Since ATM cells in a particular connection are guaranteed to arrive in a given order, a set of events sent to multiple receivers also arrive in the given order.

9. On-line Development and Expansion

Our approach also provides on-line expansion and development, which are indispensable features in ADS. Because event service provides de-coupled asynchronous communication, applications connected to the event channels are autonomous. Consequently, an application can be easily added and deleted without affecting other applications.

On-line development can be achieved using a test flag in the event structure. The applications under development send events with the test flag on while other applications turn the flag off. Events with test flag on are not used for any computation. Hence, these events do not interfere with the regular applications [4].

10. An Illustrative Example

To illustrate our approach, we use a wide area information service system as an example. The requirements of the system are:

1. The user can access diverse types of services without specifying the locations of the service suppliers. The services available are: high priority services: emergencies, such as 911; middle level priority services: high priority business applications; regular priority services: ordinary consumer services such as public transport, news, and library information.
2. The user has the option to specify the quality of the service he/she wants. The desired QoS of emergency applications is that all the updates must be propagated in three seconds. The age limit of high priority business application events is ten seconds.
3. The user is not aware of failure of part of the network or active nodes.
4. Dozens of regions are connected together and the system should have the capability of adding and removing service suppliers and consumers on-line.

In this example, all service suppliers and consumers are connected through their corresponding primary channels. All information is transferred through channels in the form of events. When a supplier sends an event, it reaches the primary channel first. The primary channel then sends the information to all possible consumers. Since the consumers and suppliers communicate only through events, service suppliers are transparent to the consumers. As a result, adding new service suppliers does not affect any existing suppliers and consumers.

Since there are three different levels of priorities, each channel maintains three event-queues, which

correspond to the three priority levels. For instance, 911 events have the highest priority, and thus they are delivered first. The business applications events, such as ordering information, documents, etc. have the second highest priority. The rest, such as library events, have the lowest priority. Before any 911 information is sent, dedicated VCs are set up among the channels for fast propagation. The traffic characteristic of the VCs in this case is CBR, which provides guaranteed bandwidth. The required bandwidth is computed by the LQMs based on the size of every 911 data. On the other hand, when business application and library events arrive on the channels, the business events are processed first. However, in case of sustained arrivals of business events, the library events are propagated in a periodic basis, such that they do not get accumulated in the channels with the possibility of queue overflow and congestion.

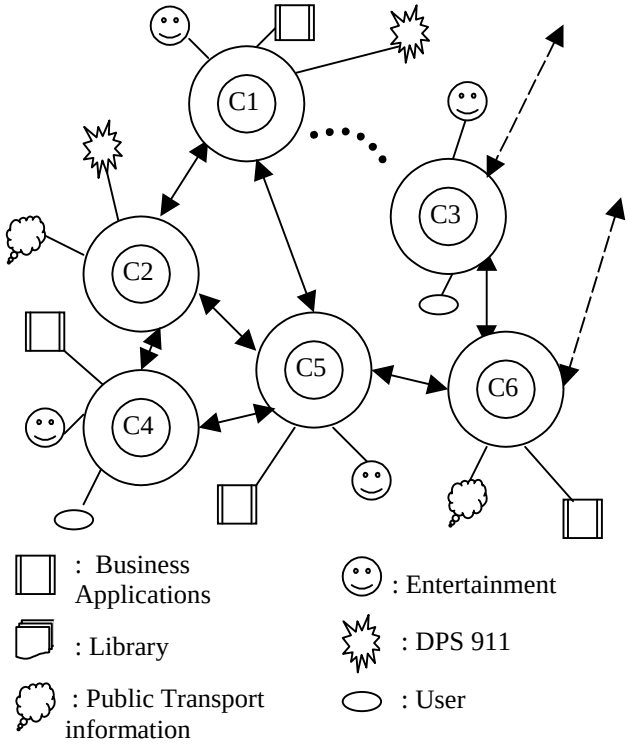

Figure 8: An example of information services system.

Whenever an event channel breaks down or a link between event channels and an event channel or supplier/consumer is broken, the interfaces residing in the consumer/supplier or the event channel will try to find another working event channel or link, which is done transparently to the supplier and consumer.

11. Discussion

In this paper, we have presented an ADS framework using event service and ATM network for the next generation information services. We have showed how to use the filtering technique in event service to reduce traffic and how to satisfy various QoS requirements for various services over a network.

More work needs to be done for the implementation of our approach. The filtering technique may be explored further to have a more efficient and systematic method for reducing network traffic. More research is needed to identify the appropriate watermark values for different system configurations. Additional QoS requirements, such as security, need to be considered in the future. Furthermore, we plan to use metadata as the content indicator to identify different events instead of only using event types since an event type itself is not sufficient for the semantic representation of the event.

Acknowledgment

This work was partially supported under the collaborative agreement between the Arizona State University and Hitachi, Ltd.

References

[1] K. Mori, "Autonomous Decentralized Systems: Concept, Data Field Architecture and Future Trends", *Proc. ISADS 93*, March, 1993, pp. 28-34.

[2] H. Wataya, K. Kawano, H. Keijirou, "The Cooperatin Autonomous Decentralized System Architecture", *Proc. ISADS 95*, April, 1995. pp 40-47.

[3] H. Kuwahara, "Experiences Teach Us the Future of Autonomous Decentralized Systems*", Proc. ISADS 1997*, April, 1997, pp 169-175.

[4] S Sameshima, K. Kawano, J. Kumaama, T. Ito, K. Inoue, S. Fujishiro, "An Autonomous Decentralized System Architecture and Techniques for On-line Development and Maintenance", *Proc. ISADS 97*, April, 1997. pp 121-128.

[5] K. Mori, "Applications in Rapidly Changing Environments", *Computer*, April, 1998. pp 42-44.

[6] R. Ahuja, S. Keshav and H. Saran, "Design, Implementation, and Performance Measurement of a Native-Mode ATM Transport Layer", *IEEE/ACM Transactions on Networking*, Vol. 4, August, 1996, pp. 502-515.

[7] D. Comer, *Internetworking with TCP/IP - Principles, Protocols, and Architecture*, Prentice Hall, 1988.

[8] OMG, CORBA Services, *Common Object Services Specification*, 1997.

[9] S S. Yau, C. Gao, and F. Karim, "Object-Oriented Real-time Event Service for Large-scale Distributed Systems", *Proc. IEEE Workshop on Middleware for Distributed Real-time Systems and Services*, December, 1997. pp 196-203.

[10] C. Gao, "*Object-Oriented Event Service and Management for Large-scale Distributed Systems*", Ph.D dissertation, 1998.

[11] A. Tanenbaum, "Computer Networks", 3rd Edition, Prentice Hall, 1996.

[12] T. Kwok, *ATM - The New Paradigm for Internet, Intranet and Residential Broadband Services and Applications*, Prentice Hall, 1998.