Fault Tolerance Mechanism of Agent-Based Distributed Event System

Ozgur Koray Sahingoz[1] and A. Coskun Sonmez[2]

[1] Air Force Academy, Computer Engineering Department, Yesilyurt, Istanbul, Turkey
sahingoz@hho.edu.tr
[2] Yildiz Technical University, Computer Engineering Department, Yildiz, 34349, Istanbul, Turkey
acsonmez@ce.yildiz.edu.tr

Abstract. Event based system development is increasingly becoming popular for large-scale and heterogeneous distributed platforms because it helps diminishing software dependencies, and enhancing system integration and evolution. The architecture of an event based system should be tolerant to error and network fallout especially in dispatching service. Throughout the entire design of event based systems, fault-tolerance mechanism plays very important role in developing large scale middleware. This is a crucial quality of service where node failures are frequent in wide area networks with many brokers. In this paper, we address fault tolerance mechanism of the agent based distributed event system where events are responsible for determining their own paths, in the case of link and broker failures. This mechanism is achieved by dynamically configuring new paths at run time for making the system more scalable and robust on a global scale.

1 Introduction

Event based systems consist of distributed components which communicate through the exchange of event messages which are defined as simple messages, such as records, tuples, or simple objects. Middleware using an event based communication model is appropriate to address the requirements of distributed applications for large scale and heterogeneous environments which require a less tightly coupled communication relationship between their components.

Event based communication generally implements the publish/subscribe model which is very well suited for connecting loosely coupled large-scale applications in the Internet [1-3]. In this model, receivers of messages (subscribers) express their interest by subscribing to a class of events, and they are asynchronously notified if a sender (publisher) publishes an event which matches the subscription. In this way the model allows a flexible n-to-m communication among the communicating parties. The core component of the middleware, the dispatching system, is responsible for collecting subscriptions and advertisements. After that the dispatching system forwards event messages from publishers to subscribers and in doing so achieves a high degree of decoupling among the communicating parties.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 192 – 199, 2006.

Faults and failures are inevitable in a distributed system built over a wide-area network and they should be tolerated for continuing system transactions. Fault-tolerance mechanisms in distributed event systems can cope with different kinds of failures in system middleware and especially integrated with the routing algorithm. This type of fault tolerance results a scalable and robust system.

In this work, we are concerned with the fault tolerance of node or link error in the Agent Based Distributed Event System-(ABDES) [10] in which events are represented by a mobile agent, called as *agvent*, which is treated as a first class citizen of the system. Autonomy and mobility features are given to agvents to select and travel between system components. We assume that brokers (called as *agvent servers* in our system) in this dispatching service of ABDES can fail by crashing and that the failure of a broker is eventually detected by all its neighbor brokers by using well-know failure detection techniques, like heartbeats or by detecting when there is a necessity. A failed broker can cause a gap in the event dissemination tree. To heal the tree, the broker which detects the failure re-routes the subscriptions, advertisements and events (called as agvents) via an alternative route.

The rest of this paper is organized as follows. In the next section, we present fault tolerance mechanisms of event based systems. Next, dispatching mechanism of ABDES is explained shortly in Section 3. Fault tolerance mechanism of ABDES is presented in Section 4 and finally, we present our conclusion and plans for future work in Section 5.

2 Related Works

Researches on fault tolerance mechanism of event based systems are concentrating on two main topics; *Self Stabilization* and *Reconfiguration*. In [4, 5] authors assume that some link may disappear and others appear elsewhere, because of changes in the underlying. *Reconfiguration* in this case means fixing routing tables entries no longer valid after the topology change. Trigger for such reconfiguration is on disappearance of one or more links between brokers, and possibly the appearance of new ones. *Self-stabilization* is an optimistic way of looking at system fault tolerance and scalable coordination, because it provides a built-in safeguard against transient failures that might corrupt the data in a distributed system. Some event systems [6, 7] use self-stabilization in the broker network by discarding broken and outdated information about neighbors. This methodology is used for synchronizing routing tables because of message loses and is accomplished by the use of leases. Both these models have a side effect as significant increase of the network traffic. Therefore some researches [8] have been done to minimize traffic overhead of the system.

There are only a few researches [9] in the literature regarding node and communication link failures. In these researches, when a broker fails, the another broker can take over the role and publish/server model can continue its transactions without any interruption and when the communication link is disconnected data transfer is delayed until the link is reconnected. But, if the link is not reconnected then messages over this link cannot be dispatched.

3 Dispatching Mechanism of ABDES

The historical development of publish/subscribe systems has followed a line which has evolved from channel-based systems, to subject-based systems, next content-based systems and finally object/type based systems. We thought the next step should contain agents in the system and developed an *Agent Based Distributed Event System-ABDES* [10] which defines events (called as agvents-*AGent eVENTs*) as first class members of the system.

ABDES combines the advantages of publish/subscribe communication and mobile agents into a flexible and extensible distributed execution environment. The major novelty of the model is that an event is represented as a mobile agent and given autonomy and mobility features to select and travel between system components. The benefits of the model are; reduced network load, higher adaptability by allowing dynamic changes in system configuration, information hiding, asynchronous communication and flexibility of agent based execution. We think the new model will serve as an effective choice for several information oriented applications, such as e-commerce, information retrieval, publication dispatch systems, distributed software and virus definitions updating.

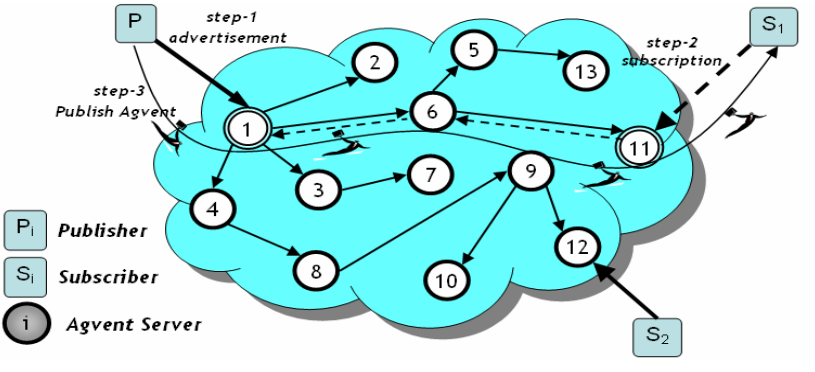

Fig. 1. Dispatching Mechanism of Agent Based Distributed Event System

Dispatching mechanism of ABDES is detailed in [11] and shortly depicted in Fig. 1. In ABDES, Firstly, a publisher advertises its agvent types which he will publish to the system (step-1). This advertisement is dispatched to all agvent servers in the dispatching service via a broadcast message. If a subscriber decides to subscribe on an agvent type, it sends a subscription message to its connected agvent server, and this message is dispatched to all relevant agvent servers by the system (step-2). Whenever a publisher observes an event, it creates and sends an agvent to the agvent server which it is connected (step-3). When the agvent arrives on the agvent server, it starts to execute its pre-specified code to select its targets (neighbor agvent servers and/or registered subscribers) according to the information present in the subscription table.

Faults in ABDES can be in each of these three dispatching operations (advertisement dispatching, subscription dispatching and agvent dispatching).

Therefore we detailed the fault tolerance mechanism of the system on these dispatching mechanisms.

4 Fault Tolerance Mechanism

Dispatching service in most of the event systems is implemented in a distributed fashion to increase the scalability of the system. Dispatching service is constituted as a set of brokers inter-connected in an overlay network. Brokers cooperatively route the messages and events issued by the clients connected to them. Although the topology of brokers based on a graph topology is starting to appear (as we implement), most available event systems are based on a tree topology, because this simplifies routing and provides a high degree of scalability.

The architecture of event based systems should be tolerant to error and network fallout especially in dispatching service. Therefore ABDES has a fault tolerance mechanism that can cope with different kinds of failures in the middleware and it is integrated with the routing algorithm resulting in a scalable and robust system.

In this study we investigate fault tolerance mechanism in the case of broker and communication link failures. When a link is removed, each of its endpoints is no longer able to route messages and events to the other partition and when a broker is down messages cannot be sent over it. Therefore we developed a fault tolerance mechanism by rerouting of the messages over healthy nodes and links to their clients. Fault tolerance mechanism of ABDES is thought in three dimensions for faults in *advertisement, subscription and agvent dispatching..*

4.1 Fault Tolerance in Advertisement Dispatching

In ABDES, advertisements are used to make agvent types visible to all the participants of the system. Advertisement forwarding, limits the overhead of subscription dispatching by spreading knowledge about agvents throughout the system. When an agvent server receives an advertisement from one of its neighbors, not only it stores the associated agvents behaviors and properties into its advertisement table, but it also forwards it to all the remaining neighbor agvent servers, thereby forming a tree to reach all agvent servers. This process effectively sets up routes for subscriptions through *the reverse path* followed by advertisements.

The technique of flooding is the simplest approach to implement the advertisement propagation in dispatching service. Here, every agvent server forwards an advertisement that is produced by one of its local clients to all of its neighbors, and if an agvent server receives an advertisement from a neighbor, it simply forwards it to all other neighbor agvent servers.

In dispatching of an advertisement, if there is a node or link error and one of agvent servers (or more) is disconnected, then this agvent server cannot get advertisement messages. This fault causes a disadvantage on routing this advertisement message not to all relevant agvent servers. When this agvent server is online or link is repaired, the agvent server synchronizes its advertisement table by controlling the neighbor agvent servers' knowledge bases. If there are some unsynchronized advertisement messages, these are received from them and forwarded to other neighbor agvent servers (if there

is any). This mechanism is also used for adding new agvent as a new neighbor to one already in the system. As is to be expected, improving scalability can be seen as more agvent servers are added to the system. Scalability of brokers in an event system is important today. As the Internet continues to grow in popularity and size, dispatching service without good scalability will result in performance drop or live-lock. When a new agvent server joins to the system, agvent servers gets the advertisement table of the one of its neighbor and uses it.

4.2 Fault Tolerance in Subscription Dispatching

A subscription expresses the subscriber's interest in the occurrence of specific events. It is a logical expression that provides the ability to select a subset of events based on their content or type. The compatibility between subscriptions and advertisements is also important because, in setting up the routing information, the event service takes advertisements into account to see if they are relevant to any subscription. As usage of advertisements to set up routes for subscriptions, subscription messages are used for setting routes for agvents. During the propagation of subscription messages, each agvent server behaves as a subscriber with respect to relevant neighbors. Consequently, each of them records the filter associated with the subscription in its own subscription table and forwards it towards the root agvent servers which send advertisement messages. This process effectively sets up routes for agvents through the reverse path followed by subscriptions.

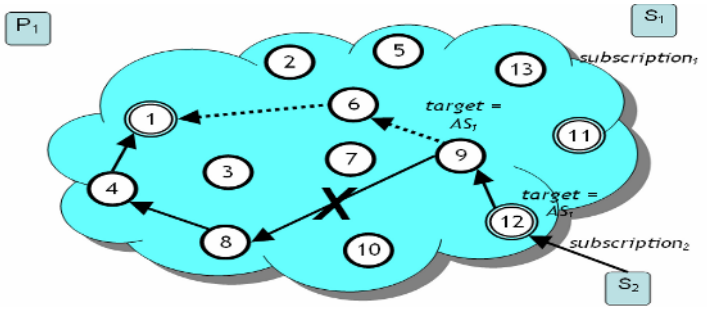

Fig. 2. Fault Tolerance in Subscription Dispatching

In dispatching of subscription, if there is a node or link error in the system (as shown in Fig. 2) then agvent server which detected node/link error should chose an alternative route and forward message on this route. For example if there is a link error between AS9 and AS8 then AS9 cannot forward subscription message over AS8. In this case AS9 can detect the link failure and checks its *Alternative Routing Table (ART)* for an alternative route to AS1. ART is maintained by each agvent server and composed by every incoming message (advertisements, subscription and agvents) which also contains their routes to reach their clients. In this failure case the solution can be in two forms;

- If AS9 contains an alternative route between AS9 and AS1, it forwards this subscription message on this route (see Fig 2). Other agvent servers use these messages for updating their ARTs.
- If there is not any routing information between AS9 and AS1 then AS9 should search an alternative route for it. To achieve this it broadcast a subscription message whose target is set as AS1 to all neighbor agvent servers. This message is not evaluated as a subscription message except AS1.

In case of adding new agvent server to dispatching service, advertisement table of this new agvent server is copied (synchronized) from one of neighbor agvent servers. But there is no need to copy the subscription table of the neighbors. Subscription table is composed according to needs of the connected subscribers. In the case of adding an agvent server, two components can join to the system over this agvent server; *publishers* and *subscribers*.

- If a subscriber connects to system over this new agvent server, it sends a subscription message to its connected agvent server and this message is forwarded according to copied advertisement table. After that the subscription table is updated.
- If a publisher connects to the system over this new agvent server, it sends an advertisement message to the agvent server and this message is broadcasting to all agvent servers in dispatching service. Agvent servers which have connected subscribers interested in this type of agvents reply this advertisement message and forward a subscription message to this newly added agent server.

4.3 Fault Tolerance in Agvent Dispatching

An agvent is a collection of code and data that migrates through ABDES. A significant capability of an agvent is its ability to discover target nodes, namely subscribers which demanded to be notified of the occurrence of an event, and to route itself to these subscribers. This is accomplished by enabling the published agvent to search the knowledge base of an agvent server, select the registered subscribers, clone itself and send each agent clone to a subscriber on the selected list. An agvent

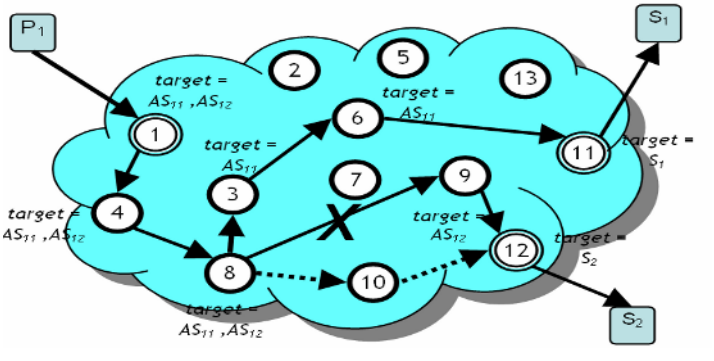

Fig. 3. Dispatching Agvents

determines its own path through the network, utilizing the minimal set of facilities provided by agvent servers.

Route of agvents are decided by subscription messages (reverse path of subscription messages). As shown in Fig. 3, the published agvent should be dispatched to S1 and S2. Agvents forward themselves to these destination nodes. After agvent reaches AS8, a clone of the agvent is created and one is forwarded over AS3, and the other is forwarded over AS9. Because there is a link error between AS8 and AS9, the agvent cannot reach to AS9. Therefore AS8 search an alternative route from its ART. If it finds a route, it forwards this agvent over new route (as depicted in Fig. 3).

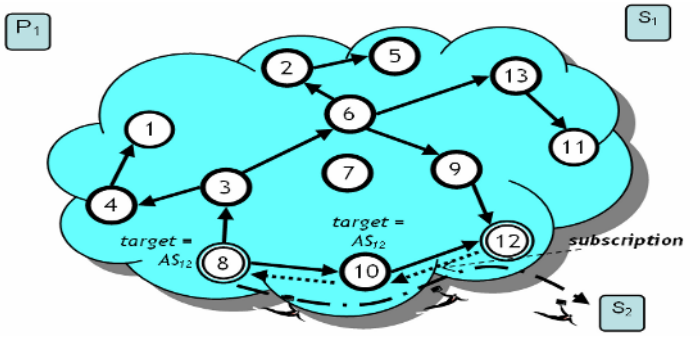

Fig. 4. Fault Tolerance in Agvent Dispatching

Main problem will remain exist if there is not any alternative route. In this case we cannot broadcast the agvent as we do in fault tolerance of subscription dispatching. An agvent is a collection of code and data. Therefore its size could be from 4-6 Kilobytes to 8-10 Megabytes (or more) according to its data block. Consequently, we have to establish a new route to sent message from breaking node to destination node. To achieve this we make a small part of previous messaging processes as follows.

- An advertisement is broadcasted to reach the target agvent server (sampled in Fig.4. as AS12). This message is taken into consideration only by the destination server (AS12). Other servers use this message only for updating their ART.
- Destination agvent servers reply this advertisement message and forward the subscription message about that type of agvent. After this subscription message reached to breaking server, an alternative route is set as reverse path of the subscription message.
- Agvent Server re-route the waiting agvent(s) to destination server(s) over this new route.

5 Conclusion

This paper presents the fault tolerance mechanism of the agent based distributed events systems, called as ABDES, which combines the advantages of publish/subscribe communication and mobile agents into a flexible and extensible distributed execution environment.

Faults and failures are inevitable in a distributed system built over a wide area network and they should be tolerated for continuing system transactions. We analyze effectiveness of the system on the failure of agvent servers and disconnection of communication links. In these cases ABDES dynamically reconfigures the connections among agvent servers in order to create paths that increase the performance of message routing. Fault tolerance by reconfiguring connections is integrated with the routing algorithm of ABDES and the method proposed here is highly scalable and robust to partial failures of agvent servers in an agent based distributed event system.

References

1. Carzaniga, A., Rosenblum, O. and Wolf, A.: Design and Evaluation of a Wide Area Notification Service. ACM Transactions on Computer Systems, 3(19), (2001), 332–383.
2. Cugola, G., Nitto, E.D. and Fuggetta, A.: The JEDI Event-Based Infrastructure and its Applications to the Development of the OPSS WFMS. IEEE Transactions on Software Engineering, 27(9), (1998), 827–850.
3. Pietzuch, P. and Bacon, J.: Hermes: A Distributed Event-Based Middleware Architecture, In Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02), (2002), 611-618
4. Picco, G. P., Cugola, G. and Murphy., A. L.: Efficient Content-Based Event Dispatching in the Presence of Topological Reconfiguration, 23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03), (2003), USA, 234–243.
5. Baldoni, R., Beraldi, R., Querzoni, L. and Virgillito, A.: A Self-Organizing Crash-Resilient Topology Management System for Content-Based Publish/Subscribe, International Workshop on Distributed Event-Based Systems (DEBS '04), Edinburgh, Scotland, UK, (2004).
6. Xu, Z. and Srimani, P. K.: Self-Stabilizing Publish/Subscribe Protocol for P2P Networks, Lecture Notes in Computer Science, Springer-Verlag, Volume 3741; DistributedComputing - IWDC 2005: 7th International Workshop, Kharagpore, India, December 27-30, (2005), 129-140
7. Buchmann, A. et al, DREAM: Distributed Reliable Event-based Application Management, Springer, (2004), 319-350
8. Cugola, G., Frey, D., Murphy, A. L. and Picco, G.P.: Minimizing the Reconfiguration Overhead in Content-Based Publish-Subscribe, In Proceedings of the 19th ACM Symposium on Applied Computing (SAC), Cyrprus, (2004), 1134-1140.
9. V.S. Sunderam et al.: Publish/Subscribe Systems on Node and Link Error Prone Mobile Environments, ICCS 2005, Springer-Verlag, LNCS 3515, (2005), 576 – 584.
10. Sahingoz, O. K., and Erdogan N.:Agvent:Agent Based Distributed Event System, Proceedings of 30th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2004), Czech Republic, (2004), 144-153.
11. Sahingoz, O. K., and Erdogan N.: Dispatching Mechanism of an Agent-Based Distributed Event System, LNCS, Springer-Verlag, Vol.3036 ICCS 2004, (2004), 184-191.