

Autonomous and Dependable Recovery Scheme in UPnP Network Settings*

Youngsoo Choi¹, Sanguk Noh², Kyunghee Choi¹, Gihyun Jung³

¹Graduate School of Information and Communication,
Ajou University, Suwon, Korea
{drabble, khchoi}@ajou.ac.kr

²School of Computer Science and Information Engineering,
The Catholic University of Korea, Bucheon, Korea
sunoh@catholic.ac.kr

³Division of Electronics Engineering,
Ajou University, Suwon, Korea
khchung@ajou.ac.kr

Abstract. Resources or devices on the network might be unavailable due to serious network partitioning. To provide a robust network connectivity, this paper presents an autonomous and dependable recovery scheme using teamwork in UPnP network settings. For our scheme, we introduce a team of recoverable control points and corresponding recovery device, and, in case of network failures among devices, the recoverable control points autonomously take care of the devices to achieve their mutual goal as a team member. In the experiments, we tested our recovery scheme in terms of the recovery effectiveness. It turned out that the recoverable control point agent in our scheme successfully handled the events from the devices, regardless of one of recoverable control points being killed. We argue that our recovery scheme is fairly consistent, and the control point agents can recover from failures as quickly as possible.

1 Introduction

The Internet, the world's end-to-end communications network, is now pervasive in everyday's life. The users communicate with each other on the Internet, and also control any device through the Internet, if it is accessible. However, the machines or devices may crash due to hardware faults, for example, the breakdown of communication channels, or software faults, for example, the deadlock of protocols. These problems motivate us to develop an autonomous and dependable recovery scheme preventing catastrophic network partitioning.

To provide a robust network connectivity, we suggest a recovery scheme using teamwork [2, 5, 6, 8], particularly in an UPnP [10] network setting, which is a popular network architecture built on the top of TCP/IP. The control point of the UPnP is

* This work has been supported by the Catholic University of Korea, Research Fund, 2005, and by the KISTEP under National Research Laboratory program.

to maintain the network connectivity among devices, hence, it should be recoverable from network failures. In related work, Dabrowski et al. [3] simply presented the performances of UPnP and Jini [4] in case of communication failure. The critical issue is that the service-discovery protocols, such as UPnP and Jini, do not provide any recovery scheme. It is apparent, therefore, that we might need a robust recovery mechanism to improve the dependability of UPnP network.

For our recovery scheme, we introduce new device type of recovery device and recoverable agent of control point. The recovery device keeps track of the status of its control point, and the recoverable agent of control point replaces primary control point, if failed. Our scheme enables the recoverable control point agent in a team to autonomously get back from failures, and provides better dependability as more control points are actively working in the UPnP.

In the following section of this paper, we will describe the UPnP network environment and the details of our framework for autonomously constituting a dependable team to be recoverable in case of the failure of control point. Section 3 describes experimental results to evaluate our recovery scheme in UPnP settings. In conclusions we summarize our work and mention further research issues.

2 Recovery from Control Point Failure

Our online recovery scheme is based on a dynamic teamwork formation [2, 5, 6, 8], and built on UPnP settings. First, we will address the UPnP network architecture.

2.1 Components of UPnP Architecture

The components of UPnP consist of devices, services, and control points [9]. The devices provide services, and further could contain another device called ‘embedded device.’ In a UPnP architecture, the devices can be classified into several categories, according to their functions. The working group of the devices is equipped with standard interfaces to basically define the items of their tasks, which are stored in an XML device description document. The control points search the devices in the UPnP network, get the information of the services from the XML device description document of the devices, and register themselves into the event server of the services. The services use state variables to represent their states, and then they can be invoked through an action committed by a control point. Whenever the states of the services are updated, the event server of the services informs the subscribed control points of their changes as an event.

2.2 Autonomous and Dependable Recovery Scheme

We introduce a new type of recovery device and dependable agents which mean recoverable control points in UPnP network settings. The recovery device keeps track

of the status of its control point, and the dependable agents autonomously recover using teamwork [2, 5, 6, 8] in case that some of them unnoticeably fail.

In our recovery scheme, a group of control points keep monitoring their devices, and replace a failed control point with another control point. Thus, the goal of control point team is to maintain robust connectivity between themselves and their devices. For example, when a primary control point gets killed, any other agent in a team tries to make a connection to take care of the devices. If one of the other agents succeeds in forming a connection, the other control points perceive that their goal has been achieved, and then stop their trials. This recovery scheme, therefore, can autonomously proceed, and control points can robustly work on the UPnP setting in the face of their failure.

In the meantime, a recovery device informs the other control point agents in a team of an event in order to let the other control points know (1) the status of a primary control point and (2) the information of newly inserted or deleted devices. When the other control points have been informed that the primary control point is dead, one of them should be new primary control point to achieve the goal of the team. And also, the recoverable control point agent needs to let its recovery device know (1) its status, and (2) a list of devices.

In an example scenario, as depicted in Fig. 1, we assume that there are two pairs of recoverable control points (CP1 and CP2) and recovery devices (RD1 and RD2)¹.

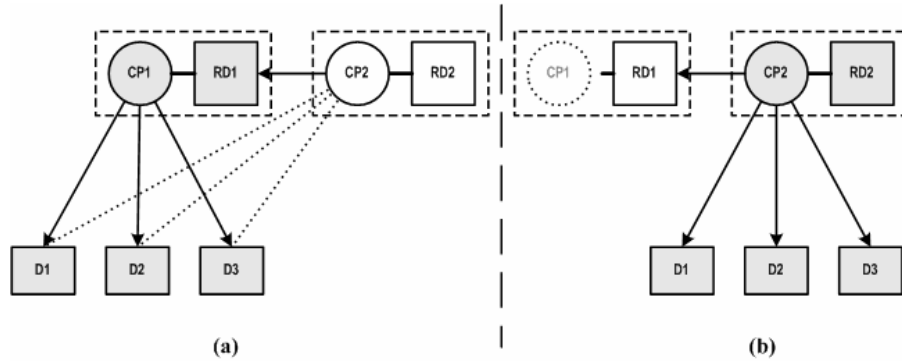


Fig. 1. Two pairs of recoverable control points (CP1 and CP2) and recovery devices (RD1 and RD2). In this example scenario, (a) the control point CP1 checks out the status of three devices D1, D2 and D3, and (b) immediately after the CP1 gets killed, the CP2, in lieu of the CP1, is capable of taking care of D1, D2 and D3.

The primary control point CP1 receives the messages from the devices D1, D2 and D3, and passes its status and a list of the devices into the recovery device RD1. The CP2 in a team is another control point, and similarly, maintains the recovery device RD2. The CP2 searches only the type of recovery devices, and succeeds in finding the RD1, and constitutes a team. The RD1 repeatedly checks out the status of the CP1, and sends an event to notify the CP1's status to the CP2. As a result, the CP2 under-

¹ We made a scaled-up scenario simple for the clarity of presentation here.

stands the current status of the CP1 and, due to the list of devices, knows that three devices are under the control of the CP1. When the RD1 notices that the CP1 gets killed, as depicted in Fig. 1 (b), the RD1 sends an event to a number of recoverable control points registered in a team. In our scenario, the only recoverable control point agent, the CP2, receives the message from the RD1. Even if the machine containing the CP1 and the RD1 fails, further, the CP2 can realize the catastrophic failure of the CP1 since the CP2 cannot receive any messages from the RD1 given a specific interval. To achieve their mutual goal formulated by their joint intention, for both cases, the CP2 autonomously contacts with the disconnected devices D1, D2 and D3, and returns to an original connectivity. Whenever the CP1 wakes up from its failure, the CP1 and the CP2 form a robust team as above.

3 Evaluation

We tested our autonomous and dependable recovery scheme in terms of its recovery effectiveness. Intuitively, the recovery effectiveness presents how autonomously our scheme handles events occurred while control points get killed. We define the recovery effectiveness as a ratio of (the number of events normally processed/the total number of events occurred).

In the experiments, we measured the recovery effectiveness with two parameters: (1) the interval which the events of devices (D1, D2 and D3 in Fig. 1) occurred varied, while the duration which the control points (CP1 and CP2 in Fig. 1) got killed was fixed; and (2) the interval which the events of devices occurred was fixed, while the duration which the control points got killed varied.

3.1 Experimental Setup

During T seconds, the interval which the events of devices occurred, the duration which control points got killed, and the interval which the failures of control points occurred, as depicted in Fig. 2, were determined by the Poisson exponential distribution.

The parameters in Fig. 2 can be computed as follows:

$$F(x) = \frac{1}{\lambda} \ln(1-x), \quad (1)$$

where

- x is a random variable between 0 and 1;
- λ is a constant depending on the generation of each parameter.

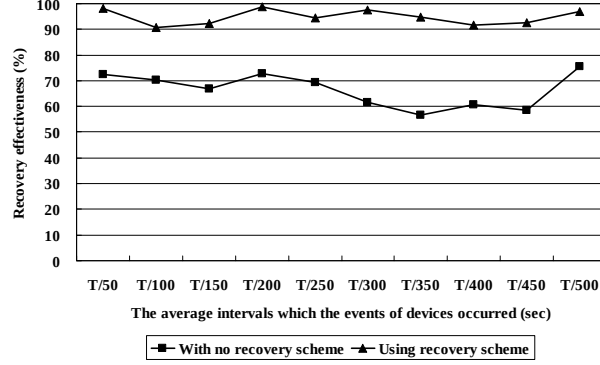


Fig. 3. The recovery effectiveness using our recovery scheme vs. without recovery scheme, when the average intervals which the events of devices occurred, on the x axis, varied.

For the second experiment in Fig. 4, we measured the recovery effectiveness with the intervals which the events of devices occurred was fixed, i.e., $T/200$, while the duration which the control points got killed varied, i.e., ranging from $T/100$ to $T/10$.

In the experiment in Fig. 4, we changed the average duration which the control points got killed, varying from $T/100$ to $T/10$, while the duration in the experiment in Fig. 3 was fixed as $T/40$. As the duration which the control points got killed increased, the recovery effectiveness went down. However, while the performance using our scheme was around 80%, the performance without recovery scheme was below 40%, as depicted in Fig. 4. If more recoverable control point agents were registered in the network, the probability of becoming the agents available should increase, and then the performance of our recovery scheme should be improved.

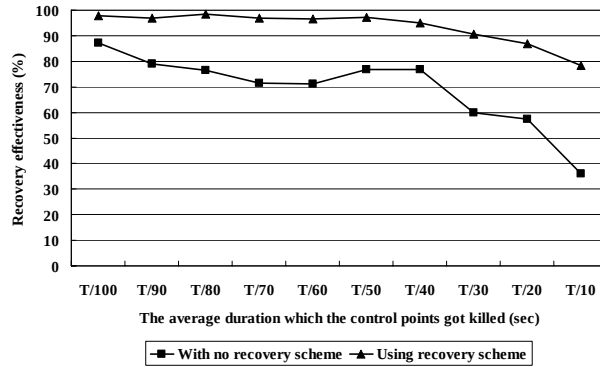


Fig. 4. The recovery effectiveness using our recovery scheme vs. without recovery scheme, when the average duration which the control points got killed, on the x axis, varied.

4 Conclusion

To provide a robust network connectivity, we suggest an autonomous and dependable recovery scheme using teamwork in UPnP network settings. Our recovery scheme enables the recoverable control point agent in a team to autonomously recover from failures, and provides better dependability as more control points are actively working in the UPnP. In the experiments, we tested our recovery scheme in terms of the recovery effectiveness. The recovery effectiveness using our recovery scheme showed, on the average, 95% of the performance in case that all of the events were perfectly processed. The experimental results support that our recovery scheme is fairly consistent, and the recoverable agents have got back from failures as quickly as possible. For future research, we will expand our recovery scheme with much more control points to verify its soundness, and continuously apply our recovery scheme to various network settings.

References

1. Butler, R. W. and Finelli, G. B.: The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software. *IEEE Transactions on Software Engineering* (1993).
2. Cohen, P. R and Levesque, H. J.: Confirmations and Joint Action. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Morgan Kaufman Publishers, Inc., San Mateo, California (1991) 951-957.
3. Dabrowski, C., Mills, K., and Elder, J.: Understanding Consistency Maintenance in Service Discovery Architectures during Communication Failure. In *Proceedings of the 3rd International Workshop on Software and Performance* (2002).
4. Ken, Arnold et al, *The Jini Specification, V1.0*, Addison Wesley (1999).
5. Kumar, S. and Cohen, P. R.: Towards a Fault-Tolerant Multi-Agent System Architecture. In *Proceedings of the 4th International Conference on Autonomous Agents (Agents 2000)*, ACM Press, Barcelona, Spain (2000) 459-466.
6. Kumar, S., Cohen, P. R., and Levesque, H. J.: The Adaptive Agent Architecture: Archieving Fault-Tolerance Using Persistent Broker Teams. In *Proceeding of the 4th International Conference on Multi-Agent Systems (ICMAS 2000)*, Boston MA, USA (2000).
7. Linux SDK for UPnP Devices Version 1.2.: <http://intel.com/technology/upnp/>, Intel (2003).
8. Tambe, M.: Towards flexible teamwork, *Journal of Artificial Intelligence Research*, vol. 7, (1997) 83-124.
9. Understanding UPnP™: A White Paper, Microsoft (2000).
10. Universal Plug and Play Device Architecture, Version 1.0.: http://www.upnp.org/download/UPnPDA10_20000613.htm, Microsoft (2000).