UPnP 환경에서 협력작업을 이용한 자율적인 복구 방법

최영수*, 노상욱**, 최경희*, 정기현***
*아주대학교 정보통신전문대학원 정보통신공학과
**가톨릭대학교 컴퓨터정보공학부
***아주대학교 전자공학부
e-mail : drabble@cesys.ajou.ac.kr

Autonomous Recovery Scheme Using Teamwork in UPnP Settings

Youngsoo Choi*, Sanguk Noh**, Kyunghee Choi*, Gihyun Jung***
*Graduate School of Information and Communication, Ajou University
**School of Computer Science and Information Engineering,
The Catholic University of Korea
***Division of Electronics Engineering, Ajou University

요 약

네트워크에 연결된 장치나 자원은 심각한 네트워크 단절로 인해 사용이 불가능할 수도 있다. 견고한 네트워크 연결성을 제공하기 위해 본 논문에서는 협력작업(teamwork)을 이용한 UPnP 환경에서의 자동화된 복구 방법을 제안한다. 제안한 복구 방법에서는 복구 가능한 제어 포인트(control point)들과 이와 연관된 복구 장치들이 팀을 이루어 제어 포인트에 장애(failure)가 발생하여도 다른 팀 구성원이 장치들을 자동으로 연결하여 공통의 목표를 달성한다. 제안한 복구 방법이 효과적임을 실험을 통해 검증하였는데, 이 실험에서 복구 가능한 제어 포인트는 팀 내의 다른 제어 포인트에 장애가 발생하였을 경우에도 장치로부터의 이벤트를 성공적으로 처리하였다.

1. 서론

인터넷의 발달로 인해 현대 사회는 네트워크를 통해 많은 의사소통을 하고 있다. 현재는 컴퓨터뿐만이 아니라 각종 가전기기들도 네트워크에 연결되어 서로 간에 정보를 주고 받고 사용자는 인터넷을 통해 이들을 제어할 수 있다. 이러한 홈 네트워크를 위한 대표적인 표준화된 구조 중의 하나가 UPnP [7]이다.

UPnP 네트워크의 구성요소인 제어 포인트(control point)는 사용자가 직접 제어를 할 수도 있지만 이벤트를 받았을 경우 특정 장치의 동작(action)을 실행함으로써 다른 장치를 간접적으로 제어하는 역할을 할 수도 있다. 이 경우 제어 포인트는 장치들 간의 연결성을 유지하여 주는 중요한 역할을 한다. 따라서 하드웨어나 소프트웨어 결함으로 인해 발생할 수 있는 제어 포인트의 장애(failure)에 대해서도 적절한 처리가 가능하여야 한다. Dabrowski, Mills, Elder 는 그들의 논문 [1]에서 통신 장애가 발생하였을 경우 UPnP 나 Jini [2]와 같은 서비스-검색 프로토콜을 이용하는 네트워크 구조에서의 성능을 평가하였다. 하지만 기존의 UPnP 구조에는 몇 가지의 보안 정책은 존재하지만 제어 포인트에 장애가 발생하였을 경우에 대해 적절히 대처하여 UPnP 네트워크의 신뢰성을 높일 수 있는 방법은 존재하지 않는다.

따라서 본 논문에서는 협력작업(teamwork) [3, 5]을 기반으로 하는 자동화된 복구 방법을 제안한다. 이를 위해 복구 장치라는 새로운 장치 종류를 정의하였고 복구 장치를 통해 같은 팀에 속한 제어 포인트는 복구를 위해 필요한 정보를 주고 받을 수 있게 된다. 제안한 복구 방법은 팀에 속한 주 제어 포인트(primary

control point)에 장애가 발생하였을 경우 사용자의 개입 없이 자동으로 팀 내의 다른 제어 포인트가 복구를 하게 된다.

본 논문의 다음 장에서는 UPnP 에 대한 간략한 소개와 제안한 복구 방법에 대한 설명을 할 것이다. 3 장에서는 본 논문에서 제안한 복구 방법의 효율성을 위한 실험과 실험의 결과에 대하여 설명하고 결론에서 실험 결과를 정리한다.

2. 장애가 발생한 제어 포인트의 복구 방법

2.1 UPnP 네트워크의 구성요소

UPnP 네트워크를 이루고 있는 기본적인 구성요소는 장치, 서비스, 제어 포인트이다 [6].

UPnP 장치는 서비스와 임베디드 장치를 포함하고 있다. 장치의 주요 기능에 따라 장치가 속하는 카테고리를 나누는데 서로 다른 카테고리에 속하는 UPnP 장치에는 서로 다른 종류의 서비스와 임베디드 장치가 포함된다. 따라서 서로 다른 종류의 카테고리를 담당하는 워킹 그룹(working group)은 자신의 카테고리에 속한 특정 장치가 제공해야 할 서비스들을 표준화 한다. 이 표준화된 정보들은 XML 장치 설명 문서(XML device description document)에 기록되고 각 장치들은 이 장치 문서의 위치를 제어 포인트에게 알려주어야 한다.

서비스는 UPnP 네트워크에서 가장 작은 제어 단위이다. 서비스는 상태 변수를 통해 자신의 상태를 나타내고 외부에서 호출할 수 있는 동작(action)을 제공하여 이를 통해 서비스를 제어하거나 작업을 요청할 수 있다. 서비스는 또한 이벤트 서버를 통해 자신의 상태가 변화되었을 때 이에 관심을 가지고 있는 등록자(subscriber)들에게 이벤트로 이를 알려준다.

제어 포인트는 네트워크상의 다른 장치를 검색하고 제어하는 일을 한다. 제어 포인트는 자신의 관심 장치를 검색하는데, 검색된 장치들에 대해서는 장치 목록을 통해 해당 장치가 포함하고 있는 서비스의 목록을 얻어오고 이를 통해 알아낸 서비스의 동작을 실행할 수 있다. 또한 서비스의 이벤트 서버에 자신을 등록하여 서비스의 상태가 변할 때마다 이벤트를 받아서 사용자에게 알려주거나 적절한 처리를 한다.

2.2 복구 방법

본 논문에서 제안한 복구 방법은 협력작업 [3, 5]을 바탕으로 한다. 즉, 여러 개의 서로 다른 제어 포인트가 하나의 팀으로 동작하고 이들은 장치에 대한 연결성을 유지해야 하는 공통의 목표를 달성하려고 한다. 하나의 팀에 속한 제어 포인트들은 자기 팀의 주 제어 포인트에 장애가 발생했을 경우 자신이 장치에 연결을 시도하여 연결성을 유지하려고 한다. 이러한 시도는 사용자의 개입이 없이 제어 포인트가 판단하여 자동으로 이루어지게 되며 이것은 UPnP 의 특징 중 하나인 무설정(zero configuration)과도 부합하는 것이다.

본 논문에서 제안한 복구 방법은 두 가지의 구성요소로 동작한다. 하나는 복구를 위한 새로운 장치 종류인 복구 장치이고, 다른 하나는 기존의 UPnP 구조의 제어 포인트와 동일하게 동작하지만 복구 장치와 정해진 메시지를 주고 받는 기능이 추가된 복구 가능한 제어 포인트이다.

복구 장치는 팀에 속한 다른 복구 가능한 제어 포인트들에게 이벤트를 보내는데 이는 (1) 자신이 담당하는 제어 포인트의 상태와 (2) 새롭게 추가되거나 삭제된 장치를 알리기 위해서이다. 또한 복구 장치는 자신이 담당하는 제어 포인트에 장애가 발생하여 서비스 불능 상태가 될 경우 이를 팀에 속한 다른 제어 포인트들에게 알리게 되고, 이를 통보 받은 제어 포인트들은 주 제어 포인트가 관리하고 있던 장치 목록을 통해 각 장치들에 접속을 시도한다. 접속에 성공하여 장치들의 연결성을 유지하는 팀의 목표를 달성한 제어 포인트는 다른 제어 포인트들에게 자신이 주 제어 포인트가 되었을 알린다. 복구 가능한 제어 포인트는 주 제어 포인트의 복구 장치가 장애를 알리지 않아도 복구 장치가 일정 시간 이상 상태 이벤트를 보내지 않을 경우, 주 제어 포인트의 시스템을 서비스 불능 상태로 판단하고 위의 복구 작업을 수행한다.

2.3 복구 시나리오

설명의 편의를 위해 여기에서는 복구 가능한 제어 포인트의 개수를 2 개로 가정한다.

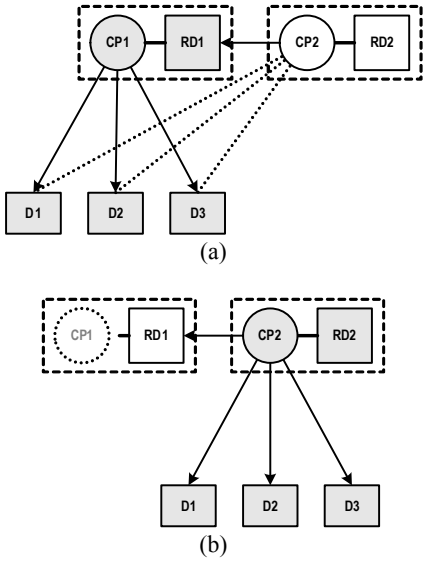

그림 1. 두 쌍의 복구 가능한 제어 포인트(CP1, CP2)와 복구 장치(RD1, RD2)에 대한 복구 시나리오

현재 네트워크에는 하나의 복구 가능한 제어 포인트 CP1 에게 등록된 서로 다른 세 개의 장치 D1, D2, D3 가 존재한다. CP1 과 동일한 시스템에 CP1 의 복구 장치인 RD1 이 존재한다. 이 때 네트워크 상에서 CP1 의 보완 역할을 하는 또 다른 복구 가능한 제어 포인트인 CP2 가 추가된다. 이 CP2 는 현재 네트워크에 존재하는 복구 장치를 검색하므로 RD1 을 발견하게 되고 RD1 에 등록을 하여 CP1 과 팀을 이룬다. RD1 은 주기적으로 CP1 의 상태를 확인하여 CP2 에게 알려준

다. RD1 은 CP1 에 장치가 추가되거나 삭제될 경우 장치 목록도 함께 알려주므로 CP2 는 현재 CP1 이 D1, D2, D3 의 장치들을 관리한다는 것을 알 수 있다. 그림 1.(a)는 이를 나타낸다.

만약 CP1 에 장애가 발생하였을 경우 RD1 은 CP1 이 서비스 불능 상태에 있다는 것을 감지하여 자신에게 등록되어 팀을 이루고 있는 제어 포인트들에게 이 사실을 통보한다. 여기서는 복구 가능한 제어 포인트의 개수를 2 개로 가정하였으므로 CP2 가 이 사실을 알게 되고 CP2 는 RD1 으로부터 받은 장치 목록을 통해 D1, D2, D3 에게 등록을 요청하여 CP1 의 기능을 복구한다.

또는 만약 CP1 과 RD1 이 속한 시스템에 장애가 발생하였을 경우 CP2 는 일정시간 이상 RD1 으로부터 이벤트 메시지를 받지 못하게 되므로, CP1 이 속한 시스템이 서비스 불능 상태가 되어 RD1 이 이벤트 메시지를 더 이상 보내지 못하는 것으로 판단하고 D1, D2, D3 에게 등록을 요청하여 CP1 의 기능을 복구한다. 그림 1.(b)는 CP2 가 복구를 완료한 뒤의 모습이다.

결국 CP2 가 D1, D2, D3 를 관리하게 되므로 각 장치는 CP1 의 장애에도 불구하고 정상적으로 동작하게 되며, 추후에 CP1 이 정상 상태로 복구 되어 네트워크에 추가 되었을 경우 추가적인 설정 없이 바로 CP1 이 CP2 의 보완 역할을 하게 된다.

3. 실험

이 장에서는 본 논문에서 제안한 복구 방법이 효과적임을 증명하기 위한 실험에 대하여 설명한다. 우리는 먼저 실험을 위하여 다음과 같은 환경을 가정하였다.

하나의 제어 포인트는 MD 와 SD 라는 두 종류의 장치에 등록이 되어 있다. SD 는 특정한 상황이 감지되었을 때 이를 제어 포인트에게 이벤트를 통하여 알려주고, 제어 포인트는 이러한 특정 이벤트를 통보 받았을 때에 MD 에게 특정 동작을 요청해야 하는 의무를 갖는다. 제어 포인트는 여러 개의 SD 에 등록될 수 있다. 이와 같은 환경에서 SD 에서 특정 이벤트가 발생하고 MD 에 이 이벤트 발생을 알리는 동작 요청이 도달한 경우, 해당 이벤트가 제어 포인트에 의해 정상적으로 처리된 것으로 간주한다.

우리는 위와 같은 환경을 기반으로 두 가지 실험을 하였다. 첫 번째 실험은 제어 포인트의 장애 발생 주기와 장애 지속 시간을 고정한 채 장치의 이벤트 발생 주기를 변화시켜 이벤트의 발생 빈도에 따라 복구가 얼마나 효과적인지를 측정하였다. 두 번째 실험에서는 첫 번째 실험에서 변화하였던 장치의 이벤트 발생 주기를 고정시키고 대신 제어 포인트의 장애 지속 시간을 변화함으로써 전체 실험 시간 중에 제어 포인트가 서비스 불능인 시간이 길어질 때에 우리의 복구 방법이 어느 정도의 효과가 있는지를 측정하였다.

우리는 실험을 위해 Intel 의 Linux SDK™ for UPnP™ Device Version 1.2 [4]를 이용하여 프로그래밍하였고, Pentium™ III 600MHz, 256MB RAM 을 가진 커널 버전 2.6.8 의 Debian™ linux O/S 가 설치된 시스템에서 실험을 진행하였다.

3.1 복구 효과성

복구 효과성은 우리가 제안한 복구 방법이 제어 포인트의 장애에 대하여 얼마나 효과적으로 이벤트를 처리하는지를 나타낸다. 우리는 복구 효과성을 이벤트가 발생한 횟수에 대한 이벤트를 정상적으로 처리한 횟수의 비율로 정의하였다.

복구 효과성의 측정을 위해 필요한 각 매개변수들은 다음과 같이 결정하였다. 먼저 실험은 T 시간 동안 이루어진다. 각각의 SD 가 이벤트를 발생시키는 사건과 제어 포인트의 장애가 발생하는 사건은 poisson 분포를 따른다. 제어 포인트에 장애가 발생한 후 다시 복구되기까지의 시간은 지수 분포를 따른다. 이를 그림으로 표현하면 그림 2 와 같다.

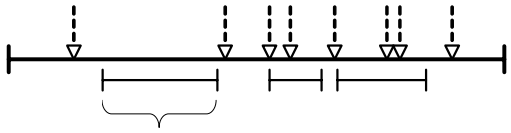

그림 2. 실험을 위한 각 매개변수의 정의

이벤트 발생 주기, 장애 지속 시간, 장애 발생 주기는 모두 식(1)과 같이 지수 분포를 따르는 난수 값을 갖게 된다.

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

- x 는 0 에서 1 사이의 난수이다.
- λ는 매개변수 생성을 위한 변수이다.

장애 발생 주기는 제어 포인트에 한번의 장애가 발생한 후 다음 장애가 발생하기까지 걸리는 시간이다. 장애 지속 시간은 제어 포인트에 장애가 발생한 후 다시 정상으로 동작하기까지의 시간이다. 이벤트 발생 주기는 장치에 하나의 이벤트가 발생한 후 다음 이벤트가 발생 할 때까지의 시간이다.

복구를 하지 않는 일반적인 제어 포인트를 이용한 실험과 제안한 복구 장치를 가지고 있는 제어 포인트를 이용한 실험으로 복구 효과성의 측정치를 비교하였다. 이 때 실험에서는 두 개의 제어 포인트가 필요하므로, 각각의 제어 포인트에 대하여 독립적인 장애 발생 주기와 장애 지속 시간을 구하여 실험하였다.

3.2 실험 결과

실험을 통하여 우리는 다음과 같은 두 개의 그래프를 측정하였다. 두 개의 실험은 모두 1000 초 동안의 결과이다.

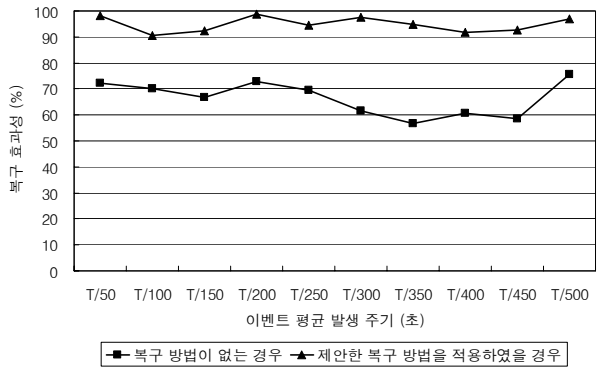

그림 3. 이벤트 발생 주기를 변화하였을 경우의 복구 효과성

첫 번째 실험에서 장애의 평균 발생 주기는 T/10, 장애의 평균 지속시간은 T/40 으로 고정하고 이벤트 발생 주기를 T/50 초에서 T/500 초까지 변화하면서 제안한 복구 방법을 적용한 UPnP 네트워크와 복구 방법이 없는 기존의 UPnP 네트워크 각각에 대하여 실험을 하여 복구 효과성을 측정하였다.

결과에서 알 수 있듯이 제안한 복구 방법을 적용하였을 때에는 평균적으로 95% 정도의 복구 효과성을 유지하였지만 기존의 UPnP 네트워크에서는 55%에서 76% 사이의 값을 가졌다. 이는 복구 방법이 적용되었을 때에는 제어 포인트에 장애가 T/10 의 주기로 계속 발생하였음에도 불구하고 장애가 발생하였을 때에 또 다른 제어 포인트가 적절한 복구 기능을 수행하였기 때문에 95% 정도의 이벤트가 처리되었고, 단지 모든 제어 포인트가 동시에 서비스 불능 상태가 되었을 경우에만 이벤트에 대한 처리를 하지 못한 것을 나타낸다. 또한 각각의 실험 결과 값들은 poisson 분포와 지수 분포를 따르는 난수 값에 의해서 약간의 차이를 나타내었다.

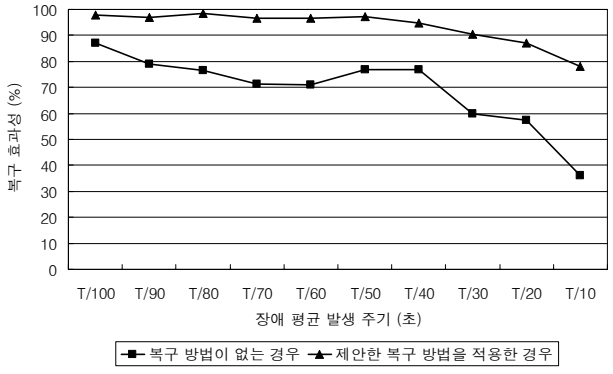

그림 4. 장애 지속 시간을 변화하였을 경우의 복구 효과성

첫 번째 실험에서는 장애 지속 시간을 T/40 초로 고정하였지만 두 번째 실험에서는 T/100 초에서부터 T/10 초까지 장애 지속 시간을 변화하였다. 반면에 이때의 이벤트 발생 주기는 T/200 초로 고정하였다. 그림 4 에서 장애 지속 시간이 길어질수록 복구 방법을 적용한 경우와 그렇지 않은 경우 모두 복구 효과성이 낮아지고 있음을 알 수 있다. 이것은 복구 방법을 적용하더라도 장애 지속 시간이 길어질 경우 두 개의 제어 포인트가 동시에 서비스 불능 상태가 될 확률이 높아지기 때문이다. 하지만 복구 방법을 적용한 경우에는 가장 낮은 값이 80% 근처인 반면에 복구 방법을 적용하지 않은 경우에는 40% 근처로 매우 낮아진다. 따라서 위의 결과는 제안한 복구 방법을 적용하는 경우에는 장애 지속 시간이 길어질 때에도 제어 포인트가 효과적으로 복구되어 이벤트를 처리한다는 것을 보여준다. 반면에 복구 방법을 적용하지 않은 경우에는 장애 지속시간이 길어지면서 복구 효과성이 급격히 떨어지는 것을 알 수 있다.

4. 결론

본 논문에서는 UPnP 네트워크에서 제어 포인트에 장애가 발생하였을 경우 협력작업을 이용하여 해당 제어 포인트가 등록하고 있던 장치들의 연결성을 유지할 수 있는 복구 방법을 제안하였다. 제안한 복구 방법에서는 여러 개의 제어 포인트가 팀을 이루어 장치들의 연결성을 유지하는 공통의 목표를 갖고, 이를 위해 필요한 장치의 정보를 주고 받는다. 복구 작업은 사용자의 개입 없이 팀에 속한 제어 포인트가 자동으로 수행하게 된다.

실험에서는 제안한 복구 방법이 얼마나 효과적으로 동작하는지를 나타내는 복구 효과성을 정의하고 측정하였다. 이벤트 발생 주기와 장애 지속 시간을 각각 변화하여 측정한 두 가지 실험에서, 모두 제안한 복구 방법을 적용하였을 때에 그렇지 않은 경우보다 복구 효과성이 높게 측정되었고 이는 복구 방법을 사용하였을 때에 복구가 효과적으로 이루어진다는 것을 보여준다.

참고문헌

[1] 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).

[2] Ken Arnold et al, The Jini Specification, V1.0, Addison Wesley (1999).

[3] 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.

[4] Linux SDK for UPnP Devices Version 1.2.: http://intel.com/technology/upnp/, Intel (2003).

[5] Tambe, M.: Towards flexible teamwork, Journal of Artificial Intelligence Research, vol. 7, (1997) 83-124.

[6] Understanding UPnP™: A White Paper, Microsoft (2000).

[7] Universal Plug and Play Device Architecture, Version 1.0.: http://www.upnp.org/download/UPnPDA10_20000613.htm, Microsoft (2000).