Acme를 이용한 Hyper-torus Architecture 원격의료시스템의 성능평가

최인화° 조민주 방헤미 김명주 이병걸
서울여자대학교 대학원 컴퓨터학과
{urangi°, cmj1228, hemiworld, mjkim, bglee}@swu.or.kr

Performance Evaluation of Hyper-torus Architecture based Telemedicine System Using Acme

Inhwa Choi°, Minjoo Cho, Hemi Pang, Myuhngjoo Kim, Byunggul Lee
Korea Women's University

요 약

현재 개발되어 사용되고 있는 원격의료시스템은 3계층기반의 구조를 이루고 있는데 환자들이 집중적으로 몰리는 지역에서 심각한 bottleneck현상이 발생할 수 있다. 본 논문에서는 3계층기반 원격의료시스템의 성능을 분석하고 bottleneck 현상을 해결하기 위한 방안으로 hyper-torus 구조의 4계층 아키텍처를 제안하고 Architecture Description Language인 Acme를 이용하여 성능을 비교분석 한다.

1. 서 론

환자와 고령자의 수가 급격히 증가하는데 비해 이를 관리할 수 있는 전문인력의 부족현상은 점차 심각해 지고 있다. 이로 인해 사람들은 점차 집에서 전문가 없이 치료할 수 있는 방법을 모색하게 되었다[1][2]. 첨단기술을 이용한 원격의료시스템은 집에서 환자들의 상태를 실시간 모니터링 할 수 있는 방안을 제시한다.

현재 사용되고 있는 이러한 원격의료시스템은 3계층기반구조를 이루고 있는데, 많은 bottleneck 현상이 발생하는 것을 발견할 수 있었다. 본 논문에서는 bottleneck현상을 감소시킬 수 있는 4계층기반 원격의료시스템을 제안하고 Acme를 이용해 성능을 분석한다. Acme는 복잡한 아키텍처의 구조를 정확하고 엄밀하게 설계/분석할 수 있는 방법을 제시하며, 이를 통해 전체구조에 대한 성능평가도 쉽게 할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 3계층기반구조의 원격의료시스템과 Acme를 소개하고 Acme를 사용한 분석과 성능평가의 결과를 제시한다. 3장에서는 2장의 3계층기반구조 원격의료시스템의 bottleneck현상을 감소시킬 수 있는 4계층기반구조 원격의료시스템을 제안하고 Acme를 사용한 분석과 성능평가의 결과를 제시한다. 마지막으로 4장에서 결론과 향후 연구방안을 제시한다.

2. 3계층기반구조 원격의료시스템

2.1 3계층기반 원격의료시스템의 구조

최근까지 진행 된 원격의료시스템은 Internet Area와 Patient Area로 구성된 두 개의 영역과, Sensor Node Layer, Patient Node Layer, Root Node Layer인 3계층기반 구조를 이루고 있다[3][4]. Sensor Node와 Patient Node는 환자 몸에 부착되어 동작하며 Patient Area를 구성한다. 환자에게 부착되는 센서는 환자의 상태와 필요에 따라 달리 구성될 수 있으며, 각 센서는 환자 상태에 대한 정보를 감지하여 Patient Node로 전송한다. Patient Node는 환자의 각 센서를 통해 전달 받은 정보를 조합하여 Root인 의료시스템 서버로 전송하는 역할을 한다. Patient Node에서 Root Node까지는 Internet Area로 구분된다. Internet Area는 새로운 정보가 발생하는 곳이 아니라 Patient Node에서 Root Node까지 정보의 이동이 발생하는 곳으로 bottleneck현상과 같은 일반적인 무선인터넷의 특성이 나타난다.

2.2 Acme: An Architecture Description Interchange Language

시스템의 구조를 표현하기 위해 UML, OCL, IDI, MIL등 많은 언어들을 사용하고 있으나 이 언어들은 복잡하고 큰 시스템을 표현하는데 많은 한계가 있다. 아키텍처를 보다 정확하고 엄밀하게 설계/분석하기 위해 정형명세 언어를 사용하는 ADL(Architecture Description Language)이 개발되어 사용되고 있다.

ADL중의 하나인 Acme(An Architecture Description Interchange Language)는 다수의 ADL을 병용하여 사용할 수 있도록 서로 다른 ADL로 표현된 아키텍처들의 상호변환과 공유를 지원하는 언어이다. 도메인 중립적 구조를 명세하고 세만틱의 인코딩 기능이 있으며 타입과 도구를 확장시킬 수 있는 기능을 갖고 있다.

Acme에서는 구조, 성질, 제약사항, 타입과 스타일의 네 가지를 정의하고 있다. 각각의 정의에 대해 간략히 살펴보면 다음과 같다. 구조는 컴포넌트, 커넥터, 시스템, 포트 역할, 표현, 표현맵을 정의한다. 개방적 부가 정보를 표현하는 성질은 모든 디자인 요소가 가질 수 있는 정의로, 메시지 요청비율이나 소스코드의 위치, 상호작용 프로토콜 등을 정의할 수 있다. 제약사항 또한 모든 디자인요소가 정의할 수 있는 부분으로 미리 정의된 predicate함수를 이용한다 (connected, hasproperty,

[1] 본 연구는 ITRC(Information Technology Research Center)의 지원을 받아 수행되었습니다.

systemname…). 마지막으로 스타일은 디자인 요소 타입의 집합으로 디자인 전문지식의 패키징, 분석 및 코드생성 도구, 아키텍처 표준 적합성 검증 등에 이용된다[5].

2.3 3계층기반 원격의료시스템의 Acme표현

본 논문에서는 3계층기반 원격의료시스템을 컴포넌트-커넥터 기반의 소프트웨어 아키텍처 개념을 채택해 설명한다[6]. 이 개념에서 소프트웨어 아키텍처는 상호작용하는 컴포넌트들을 그래프구조로 표현한 것인데, 커넥터라 칭하는 그래프의 에지는 이들 컴포넌트간의 상호작용을 나타낸다.

이러한 아키텍처를 보다 쉽고 정확하게 표현하기 위해 ADL을 사용하는데, 본 연구는 ADL의 일종인 Acme를 사용하여 시스템 내부의 소프트웨어 컴포넌트간의 구조를 표현한다[7].

그림 1은 AcmeStudio를 통해 가시화된 간단한 3계층기반 원격의료시스템의 구조를 나타낸다. s10, s11, s12, … 컴포넌트들은 환자들에게 직접 부착되어 상태를 감지하는 sensor node이다. 그림의 p1, p2, … 컴포넌트가 한 환자에 하나씩 부여되는patient node이다. Patient node의 데이터는 중앙의 Root node로 전송하게 되고 모든 데이터는 Root node에서 처리된다.

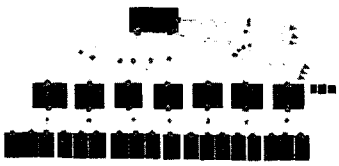

그림 1. Acme를 이용한 3계층기반 원격의료시스템 구조

```
Component Root_Node = {
    Port p = {
        Property bandwidth : int = 50;
        Property extra_bandwidth : int;
        Invariant extra_bandwidth<=bandwidth;
    }
    Port p0 = {
        Property bandwidth : int = 50;
        Property extra_bandwidth : int;
        Invariant extra_bandwidth<=bandwidth;
    }
    Component P1 = {
        Port p = {
        Property bandwidth : int = 10;
        Property extra_bandwidth : int;
        Invariant extra_bandwidth<=bandwidth;
            ....
        }
    Connector conn = {
        Role r1 = { ..... }
        Role r2 = { ..... }
    }
    Attachment P1.p to conn.r1;
    Attachment Root_node.p to conn0.r1
    Attachment P2.p to conn0.r2;
```

그림 2. Acme를 이용한 3계층기반 원격의료시스템 구조 소스

위와 같이 Acme를 이용해 표현된 시스템의 기본구조와 조건을 기반으로 시스템의 성능을 측정하여 그림 3과 같은 결과를 도출했다. 그림 4는 Acme기반구조를 이용하여 성능을 측정하는 Root node의 코드이다.

```
Port1 : extra bandwidth                 : 3
Port2 : the number of waiting patient   : 1
Port3 : extra bandwidth                 : 4
Port4 : the number of waiting patient   : 3
Port5 : extra bandwidth                 : 4
```

그림 3. 3계층기반 원격의료시스템의 성능평가 결과

```
public void port1_require(){
    for( ; port1_patient_require>0; port1_patient_require--){
        int bottleneck = port1.transmit(5);
        if(bottleneck >= (port1.getBandwidth())){
            waiting("Port1 :", (port1_patient_require-1) );
            break;
        }else if(port1_patient_require==1)
                space("Port1 :",(port1.getBandwidth())-bottleneck);
    }
}

public void port2_require(){
    for( ; port2_patient_require>0; port2_patient_require--){
        int bottleneck = port2.transmit(5);
        if(bottleneck >= (port2.getBandwidth())){
            waiting("Port2 :", (port2_patient_require-1));
            break;
        }else if(port2_patient_require==1)
                space("Port2 :",(port2.getBandwidth())-bottleneck);
    }
}

public void port3_require(){
    for( ; port3_patient_require>0; port3_patient_require--){
        int bottleneck = port3.transmit(5);
        if(bottleneck >= (port3.getBandwidth())){
            waiting("Port3 :", (port3_patient_require-1));
            break;
        }else if(port3_patient_require==1)
                space("Port3 :",(port3.getBandwidth())-bottleneck);
    }
```

그림 4. Acme기반구조를 이용한 3계층 원격의료시스템 성능분석 코드

그림 4에서 각각의 Port는 Root node에서 할당하는 Port로, 특정지역의 patient node들이 데이터를 전송하는 통로이다. 모든 Port들은 할당된 bandwidth가 제한되기 때문에 그림 2의 Acme코드에 나타난 것과 같이 정해진 대역폭 이상의 데이터를 수용할 수 없다. 하지만 환자들이 한 지역에 몰리는 경우가 발생하게 되고, 이러한 경우 그림 4와 같이 데이터는 대기상태에 놓이게 된다. 그 결과는 그림 3에서 볼 수 있다. Port1, Port3, Port5는 데이터를 더 수용할 여유가 있지만 Port2, Port4에서는 대기중인 데이터가 발생한다. 본 논문에서는 이와 같은 bottleneck현상을 감소시키기 위해 4계층 원격의료시스템을 제안한다.

3. 4계층기반구조 원격의료시스템

3.1 4계층기반 원격의료시스템의 구조

2.3절에서 살펴본 바와 같이 3계층기반 원격의료시스템에서는 지역에 따라 심한 bottleneck현상이 발생할 수 있다. 본 논문에서는 bottleneck현상의 감소를 위해 기존 3계층기반 의료정보시스템의 구조인 Root node와 Patient node사이에 Super node라는 하나의 node를 추가하여 Super node에서 기존에 발생했던 bottleneck현상을 감소시키는 방안을 제안한다.

Super node는 특정 지역에 위치하여 그 지역에 모이는 patient node의 데이터를 모아서 Root node로 전송하는 역할을 한다. 몇 개의 Super node는 torus구조로 연결되어 있어 한 지역에 patient node가 집중할 경우 이웃한 node로 데이터를 분산시킨다. torus구조로 연결된 Super node들이 여러 개 존재할 수 있는데, 이러한 super node들 간은 hyper torus구조로 연결하여 집중되는 데이터를 보다 효과적으로 분산시켜 bottleneck현상을 감소시킬 수 있다. 그림 5는 AcmeStudio를 통해 가시화된 간단한 4계층 원격의료시스템의 구조를 나타내고 있다. 3계층기반 원격의료시스템에서와 같이 patient node는 환자에게 부착된 sensor node를 통해 감지된 데이터를 전송받는다. 한 환자에게서 감지된 데이터를 모아 patient node가 super node로 전송하고, super node는 전체적인 데이터의 흐름을 파악해 가장 효과적으로 데이터를 전송하는 방법을 찾아 bottleneck현상 없이 Root node로 전송하는 역할을 감당한다. 그림 6은 이에 대한 소스이다.

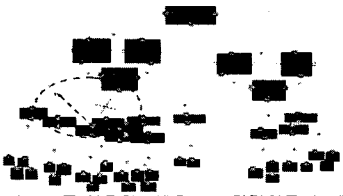

그림 5. Acme를 이용한 4계층기반 원격의료시스템 구조

```
Component  Root_node = {
    Port p = {
        Property bandwidth : int = 50;
        Property extra_bandwidth : int;
        Invariant extra_bandwidth<=bandwidth;
        ....
    }
    Port p0 = {
        Property bandwidth : int = 50;
        Property extra_bandwidth : int;
        Invariant extra_bandwidth<=bandwidth;
        ....
    }
    ......
    Component Super_node10 = {
        Port p = {
        Property bandwidth : int = 50;
        Property extra_bandwidth : int;
        Invariant extra_bandwidth<=bandwidth;
        ....
    }
    Component conn = {
        Role r1 = { .... }
        Role r2 = { .... }
    }
    Attatchment Super_node10.p to conn1.r1;
    Attatchment Super_node10.p to conn1.r2;
    .....
    On (extra_bandwidth == 0) do{
        detach Super_node11.p to conn1.r6;
        Port extra_port = seek_extra_port();
        Attachment extra_port to conn1.r6;
    }
    .....                                          }
```

그림 6. Acme를 이용한 4계층기반 원격의료시스템 구조 소스

위와 같이 Acme를 이용해 표현된 시스템의 기본구조와 조건을 기반으로 성능을 측정하여 그림 7과 같은 결과를 도출했다. 그림 8은 Acme기반구조를 이용하여 성능을 측정하는 Super node의 코드이다.

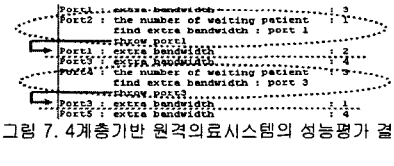

그림 7. 4계층기반 원격의료시스템의 성능평가 결과

```
public void port1_require(int port1_patient_require){
    for( ; port1_patient_require>0; port1_patient_require--){
        int bottleneck = port1.trecmain(5);
        if(port1.extra_bandwidth()){
            Waiting("Port1 :",(port1_patient_require-1) );
            int port = seek_extra_bandwidth(port1_patient_require-1);
            throw require(port, port1_patient_require-1);
            break;
        }else if(port1_patient_require==1)
                extra("Port1 :",port1.extra_bandwidth());
    }
}

public void port2_require(int port2_patient_require){
    for( ; port2_patient_require>0; port2_patient_require--){
        int bottleneck = port2.trecmain(5);
        if(port2.extra_bandwidth()){
            Waiting("Port2 :", (port2_patient_require-1));
            int port = seek_extra_bandwidth(port2_patient_require-1);
            throw require(port, port2_patient_require-1);
            break;
        }else if(port2_patient_require==1)
                extra("Port2 :",port2.extra_bandwidth());
    }
}

public int seek_extra_bandwidth(int bottleneck){

    int extra=0;
    if(port1.extra_bandwidth()>=bottleneck) extra = 1;
    else if(port2.extra_bandwidth()>=bottleneck) extra = 2;
    else if(port3.extra_bandwidth()>=bottleneck) extra = 3;
    else if(port4.extra_bandwidth()>=bottleneck) extra = 4;
    else if(port5.extra_bandwidth()>=bottleneck) extra = 5;
    System.out.println("        find extra bandwidth: port "+extra);

    return extra;
}
```

그림 8. Acme기반구조를 이용한 4계층 원격의료시스템 성능분석 코드

그림 8에서 각각의 Port는 Super node에서 할당하는 Port이다. 3계층기반 시스템에서는 Root node가 직접 할당했던 부분이다. 그림 3에서와 같이 그림 7의 4계층 시스템도 동일하게 제한된 대역폭이 정해져 있고, 그 이상의 데이터는 수용할 수 없다. 하지만 4계층 시스템의 Super node는 서로 hyper-torus구조로 연결되어 있어 특정지역에 환자가 집중할 경우 효과적으로 분배할 수 있는 기능이 있다. 이에 대해 3계층 시스템의 성능평가와 동일한 조건으로 실험을 한 결과가 그림 7에 나타나 있다. Port 2, Port4에 환자가 집중적으로 몰려있는 현상은 동일하다. 하지만 4계층기반 시스템에서는 그림 8에서와 같이 bottleneck현상이 발생했을 경우 Super node가 서로 연결된 정보를 참고하여 여유 있는 Super node를 찾고, 그 node로 데이터의 처리를 위임한다. 결과적으로 3계층기반 시스템의 Port2, Port4는 환자가 특정지역에서 다른 지역으로 이동할 때까지 bottleneck현상이 지속되지만, 4계층기반 시스템에서는 super node가 Port 2는 Port1으로 데이터를 위임하고, Port 4는 Port3으로 데이터를 위임함으로 모든 Port에 bottleneck현상이 해결됨을 볼 수 있다.

4. 결 론

본 논문은 기존 3계층기반 원격의료시스템의 bottleneck현상을 해결할 수 있는 방안으로 3계층구조에 super node가 추가된 4계층 원격의료시스템을 제안하였다. 두 구조간의 분석과 성능평가를 위해 Architecture Description Language인 Acme를 사용함으로 보다 효과적으로 분석할 수 있었고 hyper-torus구조로의 동적 재구성을 쉽게 할 수 있었다.

앞으로 본 논문을 더욱 확장시켜 patient node에 필터링 기능을 추가하여 데이터의 양을 축소시킴으로 보다 빠르고 안정적인 시스템을 구현하고자 한다. 이를 위해 다른 ADL로의 표현이 연구 되어져야 한다. 또한 다양한 조건들을 기술할 수 있는 보다 견고한 기술방법이 연구 되어져야 하며, hyper-torus구조를 포함한 다른 여러 구조들이 정의되고 분류되어 각 스타일간의 구현이 자동으로 이루어지도록 구현하는 것 또한 중요한 연구과제이다.

참고문헌

1. Alex, P., " Healthwear: Medical Technology Becomes Wearable" , Computer IEEE2004
2. Jeffrey, L./Sanjeev, O., " BIOMEDICAL TELEMEDICINE" , CSCI E-170 January 11,2005
3. Emil, J., Dejan, R., John, P., John, C., Anthony, M., Abhishek, K., " Patient Monitering Using Personal Area Networks of Wireless Intelligent Sensors " , TinyOS Technology Exchange, Berkeley, February 11, 2005.
4. Bauer, P., Sichitiu, M., istepanian, R., Premaratne, K., " The Mobile Patient: Wireless Distributed Sensor Networks for Patient Monitoring and Care" , Proc. Of the First Annual IEEE Conference on Information Technology Applications in Biomedicine, 2000,pp.17-21
5. Garlan, D., Monroe, R., Wile, D.: Acme: an architecture description interchange language. In: Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research, November 1997
6. Garlan, D., Schmerl, B.:Model-based adaptation for self-healing systems. In: Proceedings of the first workshop on Self-healing systems, Charleston, South Carolina, November 18-19, 2002
7. David, G., Robert, M., David, W. Acme: Architectural Description of Component-Based Systems. Foundations of Component-Based Systems, Gary T. Leavens and Murali Sitaraman(eds), Cambridge University Press, 2000, pp.47-68.