자가 적응 모듈의 오류 탐지와 재사용

이준훈°, 이희원, 박정민, 정진수, 이은석
성균관대학교 정보통신공학부
{trsprs°, lhw1105, jmpark, seba702, eslee}@ece.skku.ac.kr

Fault Detection and Reuse of Self-Adaptive Module

Joonhoon Lee°, Heewon Lee, Jeongmin Park, Jinsu Jung, Eunseok Lee
Department of Information and Communication, Sunkyunkwan University

요 약

오늘날 컴퓨팅 환경은 점차 복잡해지고 있으며, 복잡한 환경을 관리하는 일이 점차 중요해 지고 있다. 이러한 관리를 위해 어플리케이션의 내부 구조를 드러내지 않은 상태에서 환경에 적응하는 자가치유에 관한 연구가 중요한 이슈가 되고 있다. 우리의 이전 연구에서는 자가 적응 모듈의 성능 향상을 위해 스위치를 사용하여 컴포넌트의 동작 유무를 결정하였다. 그러나 바이러스와 같은 외부 상황에 의해 자가 적응 모듈이 정상적으로 동작하지 않을 수 있으며 다수의 파일을 전송할 때 스위치가 꺼진 컴포넌트들은 메모리와 같은 리소스를 낭비한다. 본 연구에서는 이전 연구인 성능 개선 자가 적응 모듈에서 발생할 수 있는 문제점을 해결하기 위한 방법을 제안한다. 1) 컴포넌트의 동작 여부를 결정하는 스위치를 확인하여 비정상 상태인 컴포넌트를 찾아 치유를 하고, 2) 현재 단계에서 사용하지 않는 컴포넌트를 다른 작업에서 재사용한다. 이러한 제안 방법론을 통해 파일 전송이 많은 상황에서도 전체 컴포넌트의 수를 줄일 수 있으며 자가 적응 제어 모듈을 안정적으로 작동할 수 있도록 한다. 본 논문에서는 평가를 위하여 비디오 회의 시스템 내의 파일 전송 모듈에 제안 방법론을 적용하여, 이전 연구의 모듈과 제안 방법론을 적용한 모듈이 미리 정한 상황들에서 정상적으로 적응할 수 있는지를 비교한다. 또한 파일 전송이 많은 상황에서 제안 방법론을 적용하였을 때 이전 연구 방법론과의 컴포넌트 수를 비교한다. 이를 통해 이전 연구의 자가 적응 모듈의 비정상 상태를 찾아낼 수 있었고, 둘 이상의 파일 전송이 이루어 질 때 컴포넌트의 재사용을 통해 리소스의 사용을 줄일 수 있었다.

1. 서론

오늘날 컴퓨팅 시스템 환경들이 갈수록 복잡해지면서 인간이 시스템의 문제에 대처하는 것은 쉽지 않은 일이다. 특히, 복잡한 시스템에서 발생할 수 있는 문제 상황들을 완벽하게 분석하여 설계하는 것은 대단히 어려우며 시스템이 정상 상태를 유지하지 못하는 경우가 발생할 수 있다 [1]. 따라서, 시스템을 문제 상황에 적응시키는 것이 필요하다. 이러한 적응은 인간이 직접 관리를 할 수 있다. 예를 들어, 시스템 관리자가 수동으로 새로운 애플리케이션을 실행하거나 서버의 구성을 변경하여 서버에 주어지는 부하를 줄일 수 있다.

하지만 이러한 방법은 복잡한 단계를 요구하며 시스템 관리자의 능력에 따라 적응 능력이 달라지게 된다 [2]. 이러한 관점에서 점차 늘어가는 비용을 줄이기 위하여 시스템 스스로가 리소스의 가변성이나 사용자의 요구사항 또는 시스템의 오류를 진단하고 회복하는 자가 치유 시스템이 요구되고 있다 [3, 4].

우리의 이전 연구 [5]의 아이디어는 의사도 아플 때 병원에 가서 치료를 받는 것처럼, 시스템의 문제를 적응시키는 자가 적응 모듈 자체에 문제가 발생하였을 때 대처 방안을 적용하는 좋은 출발점이다. 이 연구에서는 동시에 작동하는 컴포넌트의 수를 줄여 자가 적응 모듈의 성능 개선을 했다. 그러나 이전 연구의 방법론에서는 컴포넌트의 상태 변화를 막는 요인(바이러스 같은)에 의해 자가 적응 모듈이 정상적으로 동작하지 않을 수 있다. 또한, 다수의 파일을 전송할 때 여전히 메모리 사용률이 높다.

본 연구에서는 이러한 문제를 해결하기 위해 다음과 같은 제안사항들을 제시한다.

- 컴포넌트의 동작 여부를 결정하는 스위치를 확인하여 비정상 상태인 컴포넌트를 복구한다.
- 현재 단계에서 사용하지 않는 컴포넌트를 다른 파일 전송 작업에서 재사용한다.

제안 방법론을 통해 파일 전송이 많은 상황에서 전체 컴포넌트의 수를 줄여 리소스의 낭비를 막을 수 있고, 비정상 상황을 회복시키는 자가 적응 모듈의 안정적 작동을 가능하게 한다.

본 논문에서는 평가를 위하여 간단한 비디오 회의 시스템을 구현하여 내부 파일 전송 모듈에 제안 방법론을 적용한다. 이를 통해 이전 연구에서 해결하지 못한 상황을 대처할 수 있는지를 비교한다. 또한 제안 방법론을 적용하였을 경우와 그렇지 않은 경우의 시스템에서 작동하는 컴포넌트의 수를 비교하여 [5]에서 제안한 자가 적응 모듈을 보다 안전한 상태로 유지할 수 있음을 확

인한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서는 본 논문에서 제안하는 내용을 소개한다. 4장에서는 본 논문에서 제안하는 내용의 구현과 평가를 한다. 마지막으로 5장에서는 결론을 기술한다.

2. 관련연구

본 장에서는 우리의 이전 연구[5]와 P. Stelling의 Heartbeat Monitoring [6]을 살펴 보고, 그에 대한 특징과 장단점을 요약한다.

2.1. 안전한 파일 전송 시스템

우리의 이전 연구에서는 안전한 파일 전송 시스템을 제안했다 [5]. 이 시스템은 파일을 전송하는 상황을 감시한다. 만약 바이러스와 같은 의심스러운 행동이 감지될 때, 사용자에게 알려 해당 파일을 삭제하거나 주소를 차단하는 결정을 내리고 그 결정을 목표 시스템에 적용을 한다. 이 과정에서 사용되는 많은 컴포넌트들을 기능별로 분류하여 표 1과 같은 단계로 나누었다. 해당 기능이 동작해야 하는 상황일 때만 컴포넌트를 동작시키는 하부 스위치를 두어 하부 스위치가 ON이 된 컴포넌트만 동작하게 된다. 이를 통해 모든 컴포넌트가 동시에 동작하지 않도록 한다. 이 방법론을 통해서 자가 적응 시에 낭비되는 리소스를 방지할 수 있었다.

표 1. 안전한 파일 전송의 네 가지 단계

번호	단계	설명
1	주소 허가 판단	허가된 주소인지 판단한다.
2	예상 시간 계산	파일 전송 시간을 예상한다.
3	파일 전송	파일 전송을 감시한다.
4	자가 적응	의심스러운 파일을 차단한다.

그러나 외부 요인(메모리를 감염시키는 바이러스 등)에 의해 컴포넌트가 동작을 멈출 수 있고, 상태 변경을 하지 못할 수 있기 때문에 비정상적인 상황에 대한 대책을 가지고 있지 않다. 또한 다수의 파일을 전송할 때 메모리와 같은 리소스의 많은 낭비가 존재하며, 파일 전송이 많으면 많을수록 메모리에 실행중인 컴포넌트의 수가 많아지는 단점이 있다.

2.2. 오류탐지 모니터링 방법론

P. Stelling [6]는 컴포넌트의 오류 탐지를 주기적으로 확인하는 모니터링 방법을 제안하였다. 컴포넌트가 "I am alive" 라는 신호를 주기적으로 보내면 그 신호를 모니터에서 받아 컴포넌트가 제대로 동작함을 알게 된다. 만약 신호가 주기적으로 발생하지 않는다면 해당 컴포넌트에 문제가 있음을 의미한다.

이 방법은 직접적으로 컴포넌트의 상태를 확인해 볼 필요가 없다는 장점이 있으나, 주기적인 신호가 지연되거나 주기가 특정 시간보다 짧을 경우 모니터에서 신호 수신 부하가 커지는 단점이 있다.

결론적으로, 본 논문에서는 기존 연구들을 통해서 분석된 문제점들은 다음과 같이 요약할 수 있다.

- 외부 상황에 의해 자가 적응 컴포넌트에 이상이 있을 경우 대처 방안이 없다.
- 다수의 작업을 처리할 때 낭비되는 리소스가 존재한다.

3. 제안 사항

본 논문에서는 위에서 제시된 문제점을 해결하기 위해, [6]에서 제안하는 heartbeat monitoring을 응용하여 우리의 기존 연구 [5]를 개선하고 새로운 파일 전송 시스템을 제안하여 문제를 해결하고자 한다.

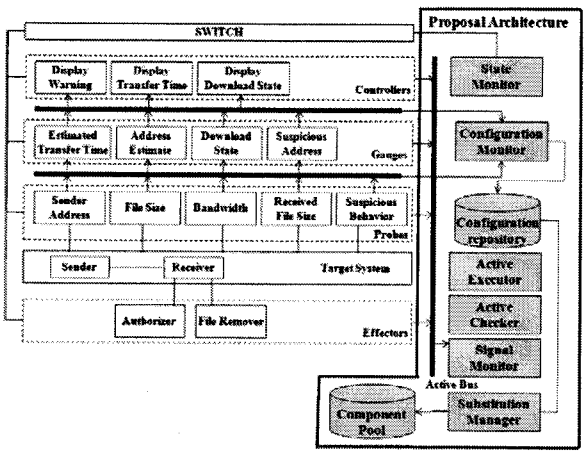

그림 1 제안하는 소프트웨어 구조

3.1. 제안 소프트웨어 구조

제안 소프트웨어 구조는 안전한 파일 전송 시스템 [5]의 구조를 수정하여 그림 1과 같이 구성하였다. 그림 1의 왼쪽 부분은 이전 연구에서 제안하는 구조이다. 본 논문에서 제안하는 확장 구조는 그림 1에서 오른쪽의 굵은 실선 부분이다. 각 컴포넌트의 기능은 표 1에서 간략하게 설명한다. 이것은 다음과 같은 목적을 가진다.

- 자가 적응 컴포넌트들의 비정상 상태를 감지하여 자가 치유를 수행한다.
- 자가 적응 컴포넌트의 기능을 대신하는 대체 컴포넌트를 실행한다.
- 하나의 파일 전송 작업에서 현재 동작을 멈춘 컴포넌트를 다른 파일 전송 작업에서 재사용한다. 이 컴포넌트는 자가적응 컴포넌트이다.

제안 세부 컴포넌트들은 다음과 같은 목적을 수행한다.

- **State Monitor**: [5]에서 파일 전송을 네 단계(주소 허가 판단, 예상 시간 계산, 전송, 자가 적응)에 따라 컴포넌트 동작 스위치의 상태를 결정했다. State Monitor는 파일 전송의 네 단계 중, 현재의 수행 단계가 어떤 단계인지를 모니터링 한다. 모니터링을 통해 현재 작동해야 할 컴포넌트가 어떤 것인지 알 수 있다. 또한 다른 작업에서 컴포넌트를 재사용하기 위해서, 각 컴포넌트의 동작 여부를 나타내는 스위치 상태를 모니터링 하여 작동 중이 아닌 컴포넌트를 파악한다.
- **Configuration Monitor (CM)**: 안전한 파일 전송 시스템 [5]의 아키텍처에는 파일 전송의 자가 적응을 위한 네 가지 계층들(Probes, Gauges, Controller, Effectors)이 있다. Probes 계층에서 측정된 값들은 Probe bus를 통해서 Gauges 계층에 전달되고, Gauges 계층에서 평가한 값들은 Gauge bus를 통해 Controllers 계층으로 전달된다 [2]. Configuration Monitor는 Probe bus와 Gauge bus를 통해 전달되는 값들을 주기적으로 모니터링 한다. 만약 모니터링 한 값들이 이전 주기의 값과 다를 경우 Configuration Repository에 변경된 값을 저장한다. Configuration Repository에 저장된 값은 자가 치유 중인 컴포넌트가 있을 때, 대체 컴포넌트를 작동시키는데 사용된다.
- **Configuration Repository (CR)**: Configuration Monitor에 의해 수집되는 값들을 저장한다. 파일 전송 자가 적응 컴포넌트 중에서 치유 상태인 컴포넌트가 있을 때, 해당 컴포넌트가 사용 중이던 정보를 찾아 해당 컴포넌트의 기능을 대체하는 컴포넌트에게 제공한다.
- **Signal Monitor**: Signal Monitor는 파일 전송 단계별로 작동하는 컴포넌트들이 동작을 시작할 때 보내는 동작 시작 신호를 모니터링 한다. 주기적으로 신호를 수신하는 방법 [6]에서 신호 수신 부하가 클 수 있다. 따라서 동작 시작 신호는 컴포넌트가 동작을 시작하면 Active Bus로 전달한다. 주기적으로 신호를 확인하지 않고, 파일 전송 단계가 변할 때에만 신호를 확인하기 때문에 신호 수신 부하를 줄일 수 있다. 만약 시작 신호가 없는 컴포넌트가 있다면 해당 컴포넌트는 동작 스위치가 ON이지만 현재 동작을 하고 있지 않을 가능성이 있다고 판단한다. 판단된 컴포넌트를 Active Checker에게 알려서 컴포넌트의 상태를 다시 확인하도록 한다.
- **Active Checker (AC)**: 이 컴포넌트는 에러 가능성이 있다고 판단된 컴포넌트들의 동작 상태를 확인한다. 동작 여부를 결정하는 스위치가 켜져 있을 때 컴포넌트는 반드시 작동을 하고 있어야 한다 [5]. 만약 에러 가능성이 있다고 판단된 컴포넌트의 상태가 동작 중이 아님을 알아낸다면, 에러가 발생했다고 인식하고 Active Executor에게 알린다.
- **Active Executor (AE)**: 에러가 발생했다고 판단한 컴포넌트에 자가 치유 전략을 적용시킨다. 본 연구는 에러의 탐지에 초점을 두고 있기 때문에 이 컴포넌트에 대한 자세한 설명은 하지 않는다. 본 논문에서는 에러가 발생한 컴포넌트를 강제로 종료 시키고 새로운 컴포넌트를 시작하는 전략을 사용한다.
- **Substitution Manager**: 파일 전송 단계별로 대체 컴포넌트를 생성하고, 파일 전송 단계에 따라 다른 작업에서 컴포넌트를 재사용을 할 수 있도록 한다. 자가 치유 중인 컴포넌트는 기능적인 부분이 작동하지 않는다 [3]. 자가 치유 단계에 있는 컴포넌트의 기능을 대체하기 위해서 자가 치유 파일 전송 단계에서 다음 단계로 넘어갈 때 다음 단계의 컴포넌트를 미리 생성한 후, Component Pool에 생성한 컴포넌트를 저장한다. 만약 이전의 다른 파일 전송 작업에서 대체 컴포넌트를 만들어 두어 Component Pool에 해당 컴포넌트가 존재하는 경우에는 새로 컴포넌트를 만들지 않는다. 또한, 현재 진행 중인 파일 전송 작업 A 이외의 전송 작업 B가 시작하려 할 때 현재 사용하고 있지 않는 컴포넌트를 전송 작업 B에서 재사용 할 수 있게 한다. 재사용 방법에 관한 내용은 뒤에서 언급한다.
- **Component Pool (CP)**: Substitution Manager에 의해 생성된 대체 컴포넌트를 저장하고, 필요할 때 사용할 수 있도록 한다.

표 2. 제안하는 컴포넌트의 기능 요약

이름	기능
State Monitor	1. 현재 파일 전송 단계를 모니터링 2. 컴포넌트 동작 스위치를 모니터링
Configuration Monitor (CM)	1. Probe Bus, Gauge Bus를 모니터링 2. Bus로 전달되는 값들을 CR에 저장
Configuration Repository (CR)	1. CM에서 모니터링 한 값을 저장
Signal Monitor	1. 자가 적응 파일 전송 컴포넌트가 동작을 시작하는 신호를 모니터링
Active Checker (AC)	1. 의심스러운 컴포넌트의 동작 여부를 확인
Active Executor (AE)	1. 에러가 발생한 컴포넌트에 자가 치유 전략을 적용
Substitution Manager	1. 파일 전송 단계별로 대체 컴포넌트를 생성하고 관리 2. 컴포넌트 재사용 관리
Component Pool (CP)	1. Substitution Manager에 의해 생성된 대체 컴포넌트들을 저장, 관리

3.2. 자가 적응 컴포넌트의 에러 탐지를 위한 프로세스

위에 언급된 소프트웨어 구조에서 자가 적응 컴포넌

트의 에러를 탐지하기 위한 프로세스는 그림 2와 같이 1) 현재 파일 전송 단계를 인식하는 단계, 2) 현재 파일 전송 단계를 대체할 컴포넌트를 생성하는 단계, 3) 컴포넌트들이 작동했음을 알리는 신호를 모니터링 하는 단계, 4) 신호가 오지 않는 컴포넌트의 동작 상태를 확인하는 단계를 진행한다. 마지막으로, 5) 컴포넌트가 동작하지 않는 오류 상황에 대해 자가 치유 전략을 적용하면서 자가 치유에 들어간 컴포넌트의 기능을 대신하는 대체 컴포넌트를 동작 시킨다.

- **파일 전송 상태 확인 단계 (Step1)**: [5]에서 제안한 파일 전송 방법론은 네 단계(주소 허가 판단, 예상 시간 계산, 전송, 자가 적응)로 나뉜다. State Monitor를 이용하여 현재의 파일 전송 단계를 모니터링 한다. 만약 현재의 파일 전송 단계가 더 진행되면(예를 들어, 주소 허가 판단 상태에서 예상 시간 계산 상태로 넘어가면) 다음 단계인 대체 컴포넌트 생성 단계로 넘어 간다.
- **대체 컴포넌트 생성 단계(Step2)**: 현재 파일 전송 상태에 따라 작동하는 컴포넌트가 달라진다 [5]. 현재 작동을 시작하려는 컴포넌트들과 같은 종류의 컴포넌트들을 생성하여 Component pool에서 저장하는 단계이다. 만약 Component Pool에 이전에 생성해 둔 컴포넌트가 있다면 해당 컴포넌트는 생성하지 않는다.

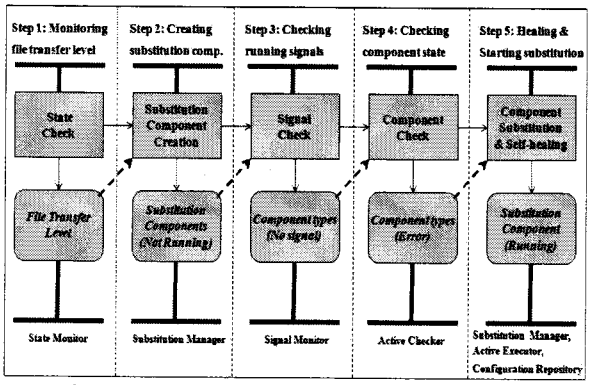

그림 2 자가 적응 컴포넌트의 에러 탐지 프로세스

- **컴포넌트 작동 신호 모니터링 단계(Step3)**: 자가 적응 파일 전송과 관련된 컴포넌트들은 작동을 시작하면 작동 시작 신호를 Active Bus에 보낸다. Signal Monitor는 파일 전송 단계가 변할 때마다 Active Bus를 확인하고 신호가 없는 컴포넌트들이 무엇인지를 Active Checker에게 알린다.
- **의심스러운 컴포넌트 확인 단계(Step4)**: 컴포넌트 작동 신호 모니터링 단계에서 알려 주는 컴포넌트들의 동작 상태를 확인하는 단계이다. Active Checker는 해당 컴포넌트들의 상태를 확인한다. 해당 전송 단계에서 컴포넌트의 동작 스위치는 반드시 ON이므로, 만약 컴포넌트가 동작 중이 아니라면 그것을 오류라고 판단하고 Active Executor에게 알린다.
- **컴포넌트 오류를 자가 치유하는 단계(Step5)**: 오류가 발생한 컴포넌트를 자가 치유하는 단계이다. 자가 치유 과정에서는 컴포넌트의 기능적인 부분이 작동을 멈추게 된다 [1]. Substitution Manager는 오류가 발생한 컴포넌트의 기능을 대체하기 위한 컴포넌트를 Component Pool에서 가져 온다. 이 컴포넌트는 현재 상태에 대한 정보를 필요로 한다. Configuration Monitor는 파일 전송을 위한 네 가지 계층들(Probes, Gauges, Controllers, Effectors)간의 정보 전달에 사용되는 통로인 버스(bus)를 모니터링 하면서 이 버스에서 전달되는 값들이 변경될 때마다 Configuration Repository에 저장하고 있다. 대체 컴포넌트는 Configuration Repository에 저장되어 있는 정보 중에서 오류가 발생한 컴포넌트가 가지고 있던 정보들을 읽는다. 이 정보들을 이용하여 대체 컴포넌트는 동작을 시작한다. 오류가 발생한 컴포넌트는 자가 치유 상태에 들어 간다. 본 논문은 자가 적응 모듈의 에러를 탐지하는데 초점을 맞추고 있으므로 컴포넌트의 자가 치유 방법에 대한 자세한 설명은 하지 않겠다.

3.3. 자가 적응 컴포넌트의 재사용을 위한 프로세스

자가 적응 모듈에서 파일 전송은 네 가지 단계(주소 허가 판단, 예상 시간 계산, 전송, 자가 적응)에 따라 동작을 하는 컴포넌트가 결정되었다. [5]. 둘 이상의 파일 전송 작업이 있을 때, 각 작업들이 다른 파일 전송 단계에 있다면 단계별로 컴포넌트를 중복해서 생성하는 것을 피할 수 있다. 자가 적응 컴포넌트의 재사용은 다음의 세 단계를 따른다.

- **파일 전송 상태 확인 단계(Step1)**: 새로운 파일 전송이 시작될 때나 파일 전송의 각 단계(주소 허가 판단, 예상 시간 계산, 전송, 자가 적응)를 넘어갈 때마다, State Monitor를 이용하여 다른 파일 전송 작업들의 단계를 확인한다. 만약 현재 작업의 단계와 다른 작업의 단계가 다르다면 현재 작업에서 다른 작업의 단계에 해당하는 컴포넌트들을 사용한다.
- **컴포넌트의 정보 저장 단계(Step2)**: Configuration Monitor는 재사용하려는 컴포넌트들이 가지고 있던 정보들을 Configuration Repository에 저장한다.
- **컴포넌트를 재사용하는 단계(Step3)**: Substitution Manager가 재사용할 컴포넌트에 현재 작업의 정보를 입력하고 동작을 시작하게 한다.

4. 구현 및 평가

화상 회의 시스템의 목적은 성공적인 회의를 하는 것이다. 회의가 진행되는 동안 사용자들은 다양한 종류의 문서들을 교환해야 한다. 이 과정에서 다수의 파일 송수

신이 있다고 가정 한다. 본 논문에서는 제안 방법론을 화상 회의 시스템에 존재하는 파일 전송 모듈에 적용하여 목표를 기준으로 평가하는 것을 초점으로 한다.

4.1 구성 환경

화상 회의 시스템을 통해 사용자들이 회의를 진행하면서 파일 전송을 한다. 테스트를 위하여 화상 회의 시스템을 간단히 구현하여 이 시스템에 제안 사항을 적용하였다. 서버는 Java로 작성하였으며, JDK 1.5를 이용하였다. 서버에서 필요한 모니터링 도구들과 클라이언트는 C#을 이용하였다. 영상 처리를 위하여 클라이언트는 DirectShow를 추가로 이용하였다.

4.2 이전 연구 구현의 수정

본 논문에서는 앞서 언급한 것과 같이 간단한 화상 회의 시스템을 구성하여, 시스템 내에서 작동하는 파일 전송 모듈에 제안 사항을 적용하였다. 이 과정에서 이전 연구 [5]의 일부를 수정하였다.

본 논문은 둘 이상의 파일 전송을 고려하기 때문에 Substitution Manager와 Configuration Repository가 대체 컴포넌트를 동작 시키거나 컴포넌트의 재사용을 할 때 파일 전송 프로세스를 체크하는 것이 필요하다. 그래서 각 파일 전송마다 번호를 부여하여 구별하였다. 번호는 0부터 시작하며 파일 전송 요청이 있을 때마다 1씩 증가하여 번호가 부여 되도록 하였다.

또한 Active Checker에서 컴포넌트의 동작 신호를 모니터링 하기 위해서 자가 적응 파일 전송 모듈 [5]에서 각 컴포넌트가 동작할 때마다 Active Bus에 신호를 보내 주도록 수정 하였다.

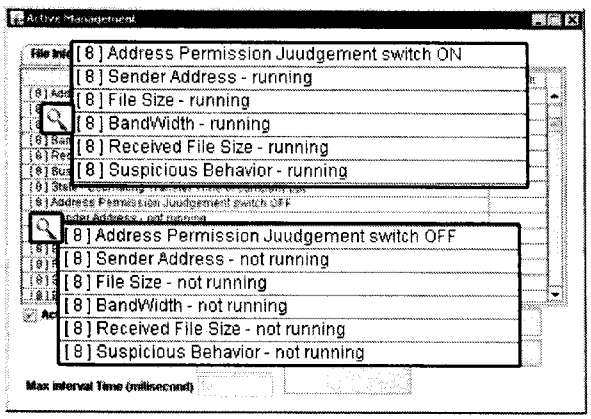

그림 3 정상적인 경우

4.3 정상적인 경우

정상적인 경우, 그림 3과 같이 파일 전송이 순조롭게 이루어 지며 컴포넌트의 동작 스위치가 ON일 때, 해당 컴포넌트가 작동을 시작한다. 또한 컴포넌트의 동작 스위치가 OFF일 때, 해당 컴포넌트는 작동을 멈춘다.

4.4 비정상적인 경우

자가 적응 파일 전송 모듈에 이상이 발생하여 어떤 파일이 정상적으로 동작하지 않는 상황을 임의로 만들어 제안 방법론이 제대로 동작하는지를 확인하였다. 그림 4는 예상 전송 시간을 측정하는 컴포넌트의 동작 스위치가 ON이 되었음에도 불구하고 동작을 시작하는 신호가 없음을 탐지하였다(그림 4의 타원으로 표시된 부분). 그 후 자가 치유 전략을 적용하여 해당 컴포넌트를 복구하는 과정을 보여 주고 있다.

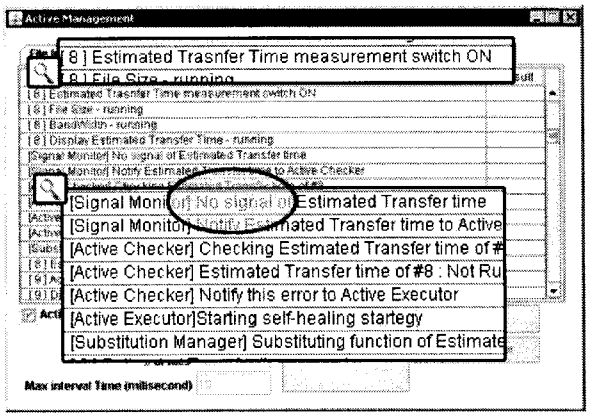

그림 4 동작 스위치와 컴포넌트 상태가 다른 상황

4.5 평가 목적

우리는 제안 방법론을 적용한 화상 회의 시스템의 파일 전송 모듈을 구현하였다. 평가를 위해 먼저 자가 적응 모듈에서 발생할 수 있는 상황들을 제시하고 제안 방법론을 적용하기 전과 적용한 후의 대처 행동에 대해 비교한다. 다음으로 현재 작동하고 있는 컴포넌트의 수이다. 파일 전송이 둘 이상일 때 이전 연구 [5]보다 얼마나 컴포넌트의 수가 줄어드는지를 비교한다.

표 3 제안 방법론 적용 전과 후의 비교

상황	제안 방법론 적용 전	제안 방법론 적용 후
수신 중인 파일이 의심스러운 행동을 함	자가 적응함	자가 적응함
메모리에 바이러스가 있어 컴포넌트의 상태 변경 불가	X	상태 체크
둘 이상의 많은 파일 전송 작업	X	컴포넌트 재사용을 고려함
자가 적응 컴포넌트의 이상 발생	X	체크함
자가 치유 중	X	컴포넌트의 기능 동작 함

(X는 고려하지 않음)

4.6 방법론의 차이 평가

표 3에서 자가 적응 모듈에서 발생할 수 있는 상황들을 제시하고 제안 방법론을 적용하였을 때와 적용하지 않았을 때의 행동에 대해 비교한다. 해당 상황에 대처할 수 있는가를 비교를 위한 것이므로 각 행동에 대한 자세한 언급은 하지 않는다.

제시한 상황에 대해서 제안 방법론을 적용하지 않았을 때에는 자가 적응 모듈의 기능적인 부분만 고려했음을 알 수 있다. 제안 방법론을 적용하여, 제시한 상황을 대처할 수 없었던 것을 대처할 수 있게 만들었다.

4.7 동시에 작동하는 컴포넌트 수 평가

우리는 파일 전송이 둘 이상일 때 제안 방법론을 적용하여 동시에 동작하는 컴포넌트 수가 얼마나 줄었는지를 평가한다. 이를 위해서 화상 회의 시스템에서 100개의 파일이 전송되도록 하였다. 파일 전송이 동시에 진행될 경우, 모두 같은 파일 전송 단계에 있게 되므로 0-1초 사이의 임의의 시간 차이를 두고 각 파일 전송을 시작하였다.

제안 방법론을 적용한 결과, 초기에는 대체 컴포넌트를 생성하기 때문에 제안 방법론을 적용하지 않았을 때보다 컴포넌트의 수가 많았다. 하지만 시간이 지남에 따라 파일 전송 작업 수가 늘어 가면서 컴포넌트의 수가 적용 전보다 줄어 들었음을 확인할 수 있었다. 이를 통해서 파일 전송 작업 수가 많을 때 컴포넌트를 더 적게 생성하게 되고, 시스템의 리소스 사용을 줄일 수 있음을 확인할 수 있었다. 본 실험의 결과는 그림 5에 나타나 있다.

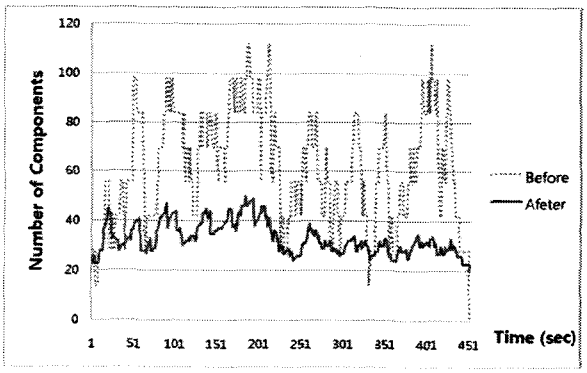

그림 5 제안 방법론 적용 전과 후의 컴포넌트 수

5 결론

본 연구에서는 이전 연구 [5]의 자가 적응 모듈에서 외부 상황에 의해 발생하는 오류를 탐지와 파일 전송 작업이 다수일 때 컴포넌트 재사용을 하는 소프트웨어 구조를 제안하였다. 이것은 마치 현실에서 의사가 아플 때 다른 의사에게 치료를 받는 것과 유사하다. 제안 방법론을 통해 다음과 같은 이점을 얻을 수 있다.

- 이전 연구의 자가 적응 파일 모듈을 예외 상황에서 더 안정적으로 사용할 수 있다.
- 다수의 파일 전송 시 자가 적응 컴포넌트 재사용을 통해 리소스의 사용을 더욱 줄일 수 있다.

하지만, 현실에서 의사 A를 치료하던 의사 B가 아프면 또 다른 의사 C가 있어야 하는 것처럼 자가 적응 모듈을 자가 치유하는 모듈이 있을 때 자가 치유 모듈 자체에 오류가 발생할 수 있다. 이 오류를 치유하기 위한 또 다른 모듈을 추가할 수 있지만, 시스템의 부담을 늘리게 된다. 이러한 경우, 부득이하게 시스템 관리자가 수동으로 관리해야 한다. 이를 위해 현재 상태를 확인할 수 있도록 상태 다이어그램을 생성하고 현재 상태를 표시해 줌으로써 어떤 컴포넌트에 문제가 생겼는지를 관리자가 알 수 있도록 향후 연구에서 다룰 예정이다.

참고문헌

[1] Jiwen Wang, Chenghao Guo and Fengyu Liu, "Self-healing Based Software Architecture Modeling and Analysis Through a Case Study," *Proceedings of Networking, Sensing and Control*, pp.873-877, 2005.

[2] David S. Wile and Alexander Egyed, "An Externalized Infrastructure for Self-Healing Systems," Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture, pp.285-288, Sep., 2004.

[3] Robert Laddaga, "Creating robust software through self-adaptation," Proceedings of Intelligent Systems and Their Applications, Vol. 14, Issue 3, pp.26-29, 1999.

[4] Jeffrey O. Kephart, David M. Chess, "The Vision of Autonomic Computing," IEEE Computer Society, Vol. 36, pp.41-50, Jan., 2004.

[5] 최윤규, 박정민, 유길종, 이은석, "외장형 자가적응 인프라스트럭처를 위한 제어 모듈의 성능개선," 제 26회 한국정보처리학회 추계학술발표대회 논문집 제 13권 제 2호, pp.1149-1152, 2006.

[6] P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. V. Laszewski, "A fault detection service for wide area distributed computations," Proceedings of 7th High Performance Distributed Computing, pp.268-278, Jul., 1998.