유비쿼터스 컴퓨팅을 위한 위치 감지 시스템의 자가 최적화 기법

최 호 영[†] · 최 창 열[††] · 김 성 수[†††]

요 약

유비쿼터스 환경에서 사용자 혹은 장비들의 위치 이동은 QoS와 밀접한 관계를 가진다. 그러므로 이동 객체의 위치를 정확히 감지하고, 높은 품질의 서비스를 제공할 수 있도록 돕는 위치 감지 시스템은 매우 중요한 역할을 한다. 하지만, 위치 감지 시스템은 적용되는 감지 범위에 따라 프로세스 전략이 다르고 사용이 제한적이기 때문에 사용자에게 신뢰성있는 QoS를 보장하기 힘든 실정이다. 본 연구에서는 지능적인 모니터링 기법을 사용하여 시스템 상황을 분석하고 자동으로 최적의 감지 프로세스 전략을 선택하는 자동화 구조를 설계하였고, 시뮬레이션을 통해 제안한 자동화 구조의 성능을 평가하였다. 실험을 통해 자가 최적화 기법을 이용한 본 구조는 동적인 네트워크 상황에서도 사용자에게 변함 없이 높은 QoS를 보장하는 것을 확인하였다.

키워드 : 유비쿼터스 컴퓨팅, 자율 컴퓨팅, 위치 감지 시스템, 자가 최적화, QoS

A Self-optimizing Mechanism of Location Aware Systems for Ubiquitous Computing

Hoyoung Choi[†] · Changyeol Choi[††] · Sungsoo Kim[†††]

ABSTRACT

The mobility of highly interconnected and communicating devices and users has implications for the QoS in a ubiquitous computing environment. Therefore, it is important for location aware systems to detect location of mobile object correctly and provide high quality services in ubiquitous environment. However, it is not easy that location aware systems offer highly reliable QoS to users because process strategies of location aware systems are limited by the capability according to the applied detection target objects. In this paper, we design an autonomic architecture which analyzes the location aware system condition and autonomously chooses the best appropriate process strategy. We also have simulated the proposed architecture in order to verify its performance. The test results show us that the architecture using self-optimizing mechanism provides higher QoS to users in variable bandwidth.

Key Words : Ubiquitous Computing, Autonomic Computing, Location Aware Systems, Self-optimizing, QoS

1. 서 론

컴퓨터와 통신기술의 발전으로 무선 휴대 장치들은 더욱 작아지고 있지만, 처리 능력은 오히려 급격히 향상되고 있다[1]. 유비쿼터스 컴퓨팅(Ubiquitous Computing) 환경에서 이러한 장치들은 서비스를 사용하는데 다양한 편리성을 가져다 주며 특히, 이동 장치들 간의 상호 작용과 다양한 서비스 요구들은 QoS(Quality of Service)와 밀접한 관계를 가진다[2]. 다양한 사용자들과 장치들이 혼재되어 있는 컴퓨팅 환경에서는 객체들의 위치를 정확하게 감지하여 높은 QoS를 제공해야 한다[3, 4]. 그런 이유로, 유비쿼터스 환경에서는 위치 감지 시스템(Location Aware Systems)의 역할이 매우 중요하다. 위치 감지 시스템은 이동하는 객체들의 위치를 추적하여, 객체가 요구하는 서비스를 효율적으로 제공하는 시스템이다. 대표적인 위치 감지 기술로는 액티브 뱃지(Active Badges), GPS(Global Position System), GSM (Global System for Mobile Communications), Radar, Wi-Fi 등이 있다. 이것들은 높은 감지 성능을 보여주고 있으며 또한, 많은 위치 감지 시스템에서 적용되고 있는 기술들이다. 하지만, 이러한 위치 감지 기술들은 서로 다른 프로

※ 이 논문은 2004년도 두뇌한국21사업에 의하여 지원되었음.
※ 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임.
† 정 회 원 : LG전자 단말연구소 연구원
†† 준 회 원 : 아주대학교 정보통신전문대학원 박사과정
††† 종신회원 : 아주대학교 정교수
논문접수 : 2005년 1월 22일, 심사완료 : 2005년 6월 9일

세스 처리 모델들로 설계되었기 때문에 자원 상황과 적용되는 환경에 적합한 다른 프로세스 전략을 사용하지 못한다[5]. 그러므로 어떠한 조건 속에서도 한결 같이 높은 서비스를 제공해야 하는 유비쿼터스 환경에서는 반드시 이러한 취약점이 개선되어야 한다. 본 연구에서는 자율 컴퓨팅 기술[6] 중 하나인 자가 최적화(Self-optimizing) 기법을 이용하여 위치 감지 시스템의 문제점을 해결하는 것을 목표로 한다. 자가 최적화 기법은 시스템 스스로가 현재 시스템 상황을 파악하여 최적의 성능을 낼 수 있도록 도와주는 기술을 의미한다[7, 8, 9].

본 연구에서는 지능적인 모니터링 기법을 사용하여 상황을 분석하고, 자동으로 최적의 감지 프로세스 전략을 선택하는 자동화 구조를 설계한다. 각 단계별 세부적인 목표와 내용은 다음과 같다.

- **1단계 : "위치 감지 시스템에서 사용하는 프로세스 전략 모델 파악"**

이 단계에서는 위치 감지 시스템에 포함되는 위치 검색 시스템(Location Searching Systems)과 위치 식별 시스템(Location Sensing Systems)의 프로세스 전략을 파악하는데 중점을 둔다. 위치 검색 시스템에서 주로 사용하는 폴(Poll) 전략의 특징을 살펴보고, 위치 식별 시스템에서는 드롭(Drop) 전략과 연기&드롭(Defer & Drop) 전략에 대해 살펴본다. 그리고 자동화 구조에서 이 전략들을 자율적으로 사용하기 위해, 동일한 환경에서 서로 다른 시스템들을 시뮬레이션 할 수 있는 방법을 알아본다.

- **2단계 : "위치 감지 시스템에서 자가 최적화 기법을 사용한 자동화 구조 제안"**

자동화 구조를 이루기 위해 필요한 두 가지 컨포넌트인 자율 모니터링 관리자와 적응 프로세스 제어기를 제안한다. 자율 모니터링 관리자는 부가적인 작업 없이 데이터들을 분석하여 수집하고, 적응 프로세스 제어기는 높은 QoS를 제공할 수 있도록 감지 프로세스 전략을 선택한다. 이 단계에서는 어떠한 네트워크 상황에서든지 성능을 최적화하기 위해 적용해야 할 기술을 제시한다.

- **3단계 : "제안한 자동화 구조의 QoS 보장 성능 평가"**

시뮬레이션을 통하여 성능을 평가함으로써, 변화하는 네트워크 대역폭에서도 자동화 구조가 최적의 QoS를 제공하는 것을 확인한다. 각 프로세스 전략들의 QoS 정의를 내리고, 적용된 감지 프로세스 전략의 QoS를 측정한다. 그리고 QoS 변화량을 분석하여 지속적으로 사용자에게 높은 QoS를 보장하는 지에 대해서도 살펴본다.

본 연구를 통해, 자가 최적화 기법을 이용한 자동화 구조는 다양한 위치 감지 시스템 상에서 사용자에게 높은 질의 서비스를 한결같이 제공하는 것을 보인다. II 장에서는 위치 감지 시스템과 자율 관리자에 대한 내용을 소개하고, III 장에서는 최적의 위치 감지 프로세스 전략을 선택하는 자동화 구조를 제안한다. 그리고 IV 장에서는 제안된 구조를 시뮬레이션을 통해 성능 분석하며, 마지막으로 V 장에서는 결론 및 향후 연구 계획에 대한 내용을 다루도록 한다.

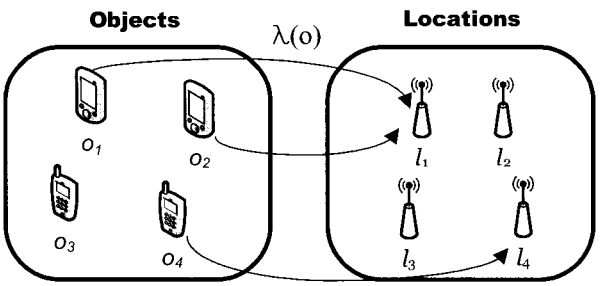
2. 관련연구

2.1 위치 감지 시스템

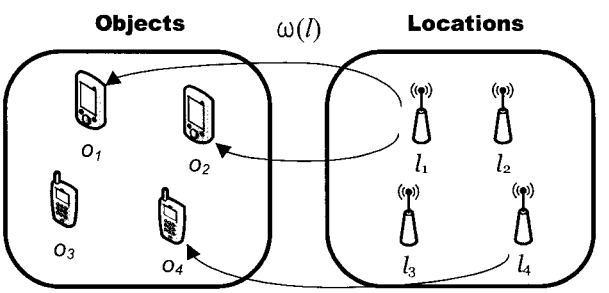
유비쿼터스 환경에서 이동 객체 위치를 감지하는 시스템은 크게 두 가지로 분류할 수 있는데, 그것들은 시한 모델(Timed Model)로 설계된 위치 검색 시스템과 동기 모델(Synchronous Model)로 설계된 위치 식별 시스템이다[10].

위치 검색 시스템은 신원 확인을 통하여 목표 이동 객체의 위치 정보를 검색한다. (그림 1)에서 보는 바와 같이, 이 시스템에서는 객체 o가 어떤 위치 l에 존재하고 있는가를 파악하는 것이 중요하다. 위치 검색 시스템은 λ: O → L 의 함수로 표시할 수 있으며, O는 객체(Object)들의 집합을 뜻하고 L은 위치(Location)들의 집합을 의미한다. GSM과 같은 위치 검색 시스템은 시한 모델을 기반으로 설계된다. 시한 모델에서는 객체들의 위치가 변화되어 이벤트가 발생되면, 일정 시간 동안 객체들을 확인하게 된다. 주로 사용하는 프로세스 전략 방식은 폴 전략이다. 폴 전략은 시간 주입 전략(Time-triggered Strategies)으로서, 폴 간격(Poll Interval) 동안 발생한 이벤트들을 수집한다. 따라서 이러한 전략은 이동하는 객체의 위치 정보를 감지하는데 매우 효율적이다.

위치 검색 시스템과는 대조적으로, 위치 식별 시스템은 일 대 다 관계를 보인다. 위치 식별 시스템은 특정 위치에 존재하는 객체들을 식별한다. (그림 2)와 같이 위치 l에 어떤 객체들이 존재하는지를 파악하는 작업을 한다. 이 시스템은



(그림 1) 이동 객체를 위치로 매핑



(그림 2) 위치를 이동 객체로 매핑

위치 l을 객체들의 집합으로 매핑할 수 있으며, $\omega : L \rightarrow P(O)$로 표시한다. P(O)는 특정 감지 범위에 들어가는 객체 집합을 의미한다. 모든 위치에는 0 개, 혹은 여러 개의 객체들을 포함할 수 있다. Wi-Fi와 같은 위치 검색 시스템은 동기 모델로 설계된다. 동기 모델은 이벤트 주입 전략(Event-triggered Strategies)인 드롭과 연기&드롭 프로세스 전략 방식을 사용하여 이동 객체가 위치를 변화시킬 때마다 이벤트를 발생시켜 그것을 처리하도록 한다. 드롭 전략은 허락하는 네트워크 대역폭 범위 내에서 발생 이벤트들을 실시간으로 수집하여 위치 감지 시스템에 전송하며, 전송량을 초과한 나머지 이벤트들은 탈락(Drop)시킨다. 이와는 달리, 연기&드롭 전략에서는 전송에 실패한 이벤트들에 대해서는 일정 시간(Defer Time)동안 전송을 연기한다. 하지만 연기된 시간 후에도 낮은 대역폭으로 인해 전송되지 못하게 되면 해당 이벤트들은 탈락된다. 이와 같은 이유로, 동기 모델에 속하는 감지 전략들은 동일 위치에 거주하는 객체 집합에 대한 작업을 효율적으로 처리할 수 있다.

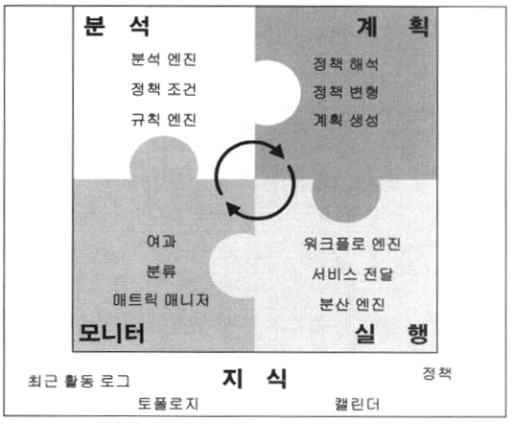
위치 감지 프로세스 전략을 자율적으로 사용하기 위해서 위치 검색 시스템과 위치 식별 시스템 간에 시뮬레이션이 가능할 수 있도록 해야 하며[11], 그러기 위해서는 위치와 이동 객체간의 매핑(Mapping)이 가능해야 한다. 두 요소들의 정보를 매핑함으로써, 서로 다른 모델로 설계된 위치 감지 시스템들에서도 다양한 감지 프로세스 전략들을 적용할 수 있기 때문이다. 먼저, 각 객체들이 포함된 위치들을 알아내기 위해서 주어진 위치 m과 객체들의 위치를 비교한다. 주어진 위치 m에 존재하는 객체들의 집합 P는 $P = \{ o \in O \mid \lambda(o) = m \}$ 로 표기된다. 이러한 경우, ω는 $\omega(l) = P$, $P = \{ o \in O \mid \lambda(o) = l \} \subseteq O$ 로 정의된다. 이 방법은 ω를 λ로 표시함으로써, 위치 식별 시스템을 위치 검색 시스템으로 표현할 때 사용한다. 위치 검색 시스템은 시한 모델을 실행하므로, 이 시스템은 일정한 간격 동안 주기적으로 위치 m에 거주하는 이동 객체들을 감지하는 것을 뜻한다. 특정 장소에 위치한 객체 o를 감지하기 위해서는 각 위치 l에 대한 객체들의 존재 여부를 확인하는 것이 중요하다. 즉, all $l \in L$ 이고, $o \in \omega(l)$ 조건일 경우에 객체들이 감지되는 것을 의미한다. 그래서 λ는 $\lambda(o) = U\{l \in L \mid \omega(l) = o\}$ 로 정의 할 수 있다. 여기서 U는 요구되는 조건을 충족시키는 모든 객체를 의미한다. 이 방법은 위치 검색 시스템을 위치 식별 시스템으로 표현하는데 사용한다. 위치 식별 시스템은 객체가 이동하여 이벤트를 발생시킬 때 동기 모델을 사용하여 감지하고, 생성되는 이벤트들의 총합은 객체들의 이동에 의해 결정된다. 본 연구에서는 위치 검색 시스템과 위치 식별 시스템 간의 시뮬레이션이 가능한 시스템 환경으로 설계한다.

2.2 자가 관리 기법(Self-managing Mechanism)

컴퓨터 시스템들이 더욱 복잡해질수록 관리자들에게 설치를 비롯한 구성, 작동, 조율 그리고 유지 등의 다양하고 세밀한 기술들을 처리하도록 요구되고 있다. 하지만, 이런 모든 업무들을 관리자가 일괄적으로 처리한다는 것은 매우 어려운 문제이다. 그래서 IBM에서 해결책으로 제시한 것이 바로 자율 컴퓨팅 개념이다[12].

자율 컴퓨팅은 시스템 스스로가 자신을 관리하는 기술로서, 관리자의 최소한 관여 만으로도 복잡한 문제들을 해결하는 것을 목적으로 한다. 기술별로 자가 구성(Self-configuration), 자가 치유(Self-healing), 자가 보호(Self-protecting), 자가 최적화(Self-optimizing)로 분류할 수 있다. 자가 구성 기법은 동작 중 자율적으로 시스템을 구성하는 기술로서, 새로운 시스템 환경에 즉시 적응하여 서비스를 원활하게 제공할 수 있도록 돕는다. 현재, 디스커버리 시스템(Discovery System)에서 각 장비들의 구성요소를 자동으로 설치해주는 기술로 주목 받고 있다. 자가 치유 기법은 시스템의 오류 발생시 내장된 자원들이 활성화됨으로써 스스로 치유 가능하도록 하는 기술이다. 시스템에서 발생할 수 있는 에러나 오류를 미리 예상하거나 감지하고, 이러한 오류를 스스로 치유하여 시스템의 오동작을 최소화한다. 자가 보호 기법은 컴퓨터 시스템 환경에서 인프라 구조와 데이터에 대한 공격을 대비할 수 있도록 자율적으로 적절한 보안 동작을 취하는 기술이다. 최근 들어, 컴퓨터 시스템들을 위협하는 다양한 바이러스들이 급증하고 있어 자가 보호 기술이 시급한 상황이다. 마지막으로, 자가 최적화 기법은 시스템 관리자와 일반 사용자에게 높은 QoS를 제공하기 위해 적용되는 성능 향상 기술로서, 현재 시스템 상황을 관찰하여 자원 상태를 파악하고 그것들을 재배치하거나, 특정 기술을 적용함으로 성능을 최적화 시키는 것을 목표로 한다. 효율적인 시스템 관리 측면에서 매우 중요한 기술로 높은 QoS를 제공해야 하는 유비쿼터스 컴퓨팅 환경에서 더욱 주목 받을 것으로 예상된다.

자율 컴퓨팅 구조에서 자원들은 자율 관리자(Autonomic Manager)에 의해서 취급되고, 자율 관리자는 제어 반복 동작(Control Loop)을 통해 실행된다. (그림 3)에서는 자율 관리자의 구성을 보여준다. 자율 관리자 구성은 책임 역할에 따라 크게 4 개 부분으로 나뉘며 그 내용은 다음과 같다. 모니터 부분에서는 데이터들의 정보를 수집, 여과, 관리, 기록



(그림 3) 자율 관리자의 구성

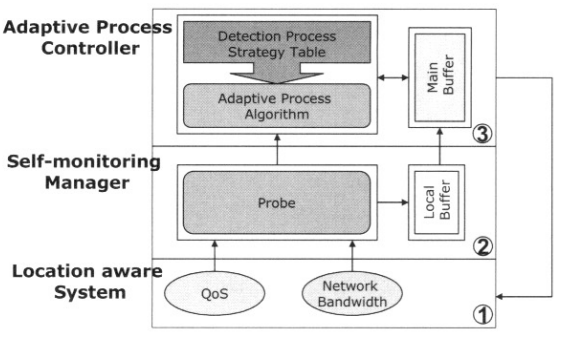
한다. 분석 부분에서는 각 정보들을 가공시키고, 복잡한 시스템 상태를 모델링한다. 자율 관리자는 분석 부분에서 이루어진 결과를 이용하여 상태를 파악하고, 다음 상황을 예측한다. 계획 부분에서는 목표를 성취하기 위해 필요한 동작 기술들을 보고하고 정책(Policy) 정보를 이용하여 해결하기 위한 방법들을 계획한다. 마지막으로 실행 부분에서는 앞서 계획된 기술들을 실행하고 제어하는 작업을 담당한다. 자율 관리자의 4가지 부분에서 사용되는 데이터들은 공유 지식으로 저장되며, 공유 지식은 토폴로지 정보를 비롯한 시스템 로그, 성능 관련 측정값과 정책 등을 포함한다.

3. 자가 최적화 기법을 이용한 구조

3.1 위치 감지 시스템의 자동화 구조

(그림 4)에서는 제안된 자동화 구조를 보여준다. 1) 현재 자원 상태를 파악하기 위해 위치 감지 시스템을 모니터링한다. 2) 그리고, 탐지기(Probe)를 통하여 모니터링된 자원들을 분석한다. 3) 높은 QoS를 제공하기 위해서 시스템은 가장 적절한 프로세스 전략을 선택하고, 그것을 위치 감지 시스템에 적용시킨다.

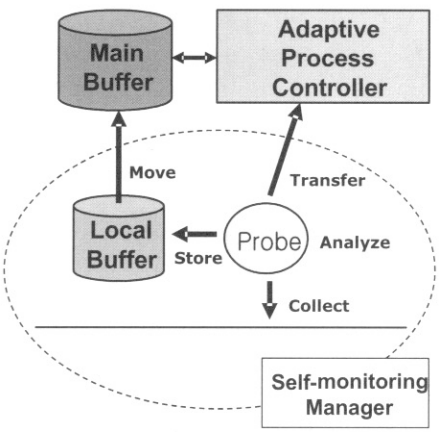
자동화 구조에서 최적의 감지 프로세스 전략을 선택하기 위해 사용되는 컴포넌트들은 자율 모니터링 관리자(Self-monitoring Manager)와 적응 프로세스 제어기(Adaptive Processor Controller)이다. 자율 모니터링 관리자는 위치 감지 시스템 상에서 성능 정보를 수집하는 일을 책임진다. 그것은 수집된 정보들을 분석하고, 성능의 이상 상태를 감지한다. 또한, 분석된 자료들을 적응 프로세스 제어기로 전송하는 작업을 한다. 적응 프로세스 제어기는 위치 감지 시스템의 성능 향상을 목적으로 한다. 현재 QoS와 네트워크 상태를 고려하여 프로세스 전략의 적용 여부를 판단하고, 동작 중에 위치 감지 시스템에 선택된 최적의 프로세스 전략을 적용시킨다. 그리고 마지막으로, 자율 모니터링 관리자를 통해 적용 결과를 다시 관찰한다. 이처럼 피드백을 통하여 주어진 자원 상황에 맞게 최적의 프로세스 전략을 선택할 수 있기 때문에 시스템 QoS를 더욱 향상시킬 수가 있다. 이때, 적응 프로세스 제어기는 적응 프로세스 알고리즘(Adaptive Process Algorithm)을 기반으로 동작한다.



(그림 4) 자동화 구조의 개념도

3.2 자율 모니터링 관리자

자율 컴퓨팅에서 모니터링 기술은 시스템 정보를 수집하는 중요한 기술이다. 본 연구에서 제안하는 자율 모니터링 관리자는 별도의 처리부하 없이 네트워크 대역폭과 시스템 QoS를 관찰하고 분석하여 프로세스 전략 적용여부를 결정하고, 해당 데이터들을 적응 프로세스 제어기로 전송하는 기능을 담당한다. 자율 모니터링 관리자는 (그림 5)에서 보는 바와 같이, 탐지기와 지역 저장소(Local Buffer)로 구성된다. 탐지기는 네트워크 대역폭과 시스템 QoS 데이터를 수집하여 실시간으로 분석함으로써 현재 성능을 관찰하고, 지역 저장소는 탐지기에 의해 수집된 데이터들을 임시적으로 저장한다. 만약, QoS와 네트워크 대역폭이 현재보다 높거나 혹은 낮을 때, 탐지기는 관련 데이터들을 적응 프로세스 제어기로 전송시킨다. 그 이유는 변화하는 자원 상태에 더욱 적합한 프로세스 전략을 선택함으로써 사용자에게 현재보다 더 높은 QoS를 제공하기 위해서이다. 만약 이상 상태가 감지되거나 지역 저장소에 더 이상 데이터들을 저장할 수 없다면, 자율 모니터링 관리자의 부가적인 처리량을 줄이기 위해 지역 저장소는 저장된 데이터들을 주 저장소로 이동시킨다.



(그림 5) 자율 모니터링 관리자 구조

다음의 상황이 발생할 경우, 탐지기는 위치 감지 시스템의 상태가 이상이라고 판단하다.

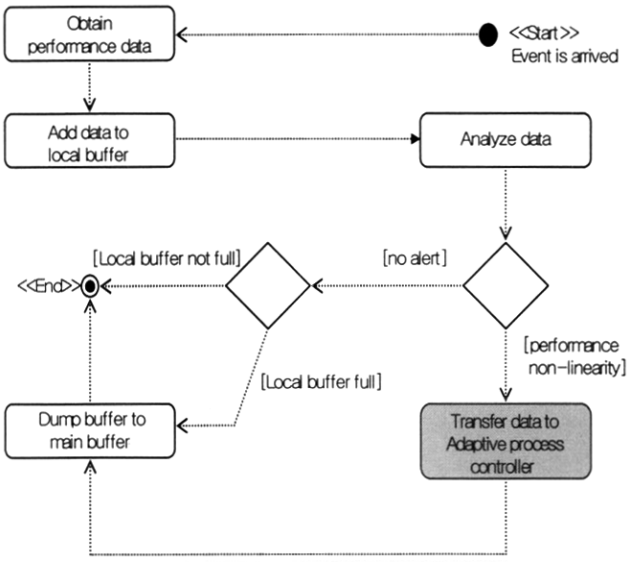
네트워크 대역폭의 이상 상태 (α)

$$\{ \alpha_{before} < \alpha_{measured} \quad or \quad \alpha_{measured} < \alpha_{before} \}$$

QoS의 이상 상태 (β)

$$\{ \beta_{before} < \beta_{measured} \quad or \quad \beta_{measured} < \beta_{before} \}$$

자율 모니터링 관리자는 현재 QoS의 변화량이 없을 때만 자원 상황에 적합한 프로세스 전략이 동작된 것으로 판단하고 적응 프로세스 제어기에 수집한 데이터들을 전송하지 않는다. 이상 상태를 감지하기 위한 탐지기의 절차는 다음과 같다((그림 6) 참조).



(그림 6) UML(Unified Modeling Language) 활동 다이어그램

1) 대상 객체가 이동하여 이벤트를 발생시킬 때 시스템 성능 데이터를 수집.
2) 지역 저장소로 수집된 데이터들을 저장.
3) 저장된 데이터들을 분석.
4) 성능 변화가 감지되지 않았지만 지역 저장소의 용량이 더 이상 남아 있지 않는 경우, 적응 프로세스 제어기에 설치된 주 저장소로 저장된 데이터들을 이동.
5) 성능 변화가 감지된다면, 데이터를 적응 프로세스 제어기로 전송하고 지역 저장소에 저장된 데이터들을 주 저장소로 이동.

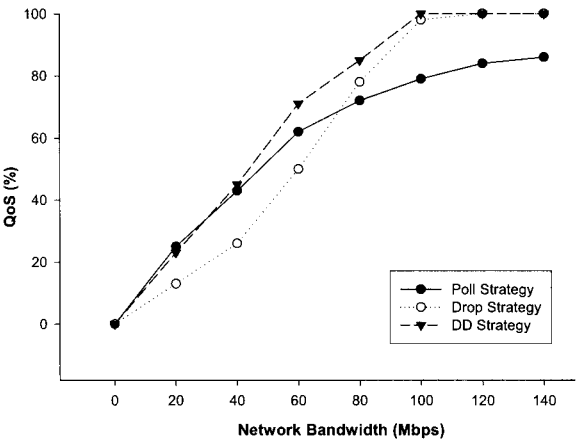
3.3 적응 프로세스 제어기

적응 프로세스 제어기는 위치 감지 시스템의 현재 자원 상태에서 최적의 프로세스 전략을 선택하여 실행시킨다. 적응 프로세스 제어기의 목적은 시스템 동작 중 이동 사용자에게 가장 적합한 프로세스 전략을 자율적으로 선택하여 높은 QoS를 제공하는 것이다. 적응 프로세스 제어기는 자율 모니터링 관리자로부터 전송된 이상 상태에 관한 네트워크 대역폭과 QoS의 값을 입력 받는다. 그리고 수집된 값을 분석하여 최적의 프로세스 전략 적용을 결정한다. 자율 컴퓨팅의 계획 부분에 해당하는 감지 프로세스 전략 테이블은, 적응 프로세스 제어기가 새로운 전략을 선택하고 적용시키도록 돕는다. 감지 프로세스 전략 테이블(Detection Process Strategy Table)은 현재 네트워크 자원 상태에서 높은 QoS를 제공하는 위치 감지 프로세스 전략들의 정책 정보를 가지고 있다(<표 1> 참조). 이 테이블은 네트워크 대역폭 상태에 따른 폴, 드롭, 연기&드롭(DD), 이 세가지 전략 QoS를 비교한 기존의 연구 자료들을 참조·분석하여 재작성한 것이다[13, 14].

(그림 7)에서는 세 가지 프로세스 전략의 시뮬레이션 결과를 보여준다. 실험 결과에 따르면 폴 전략은 대역폭이

〈표 1〉 감지 프로세스 전략 테이블

QoS Boundary	Process Strategies	Bandwidth
Below 43 %	1) Poll 2) DD 3) Drop	Till 32
Between 43 ~ 68 %	1) DD 2) Poll 3) Drop	Till 71
Above 68 %	1) DD 2) Drop 3) Poll	Over 71



(그림 7) 네트워크 대역폭에 따른 전략들의 QoS 시뮬레이션 결과

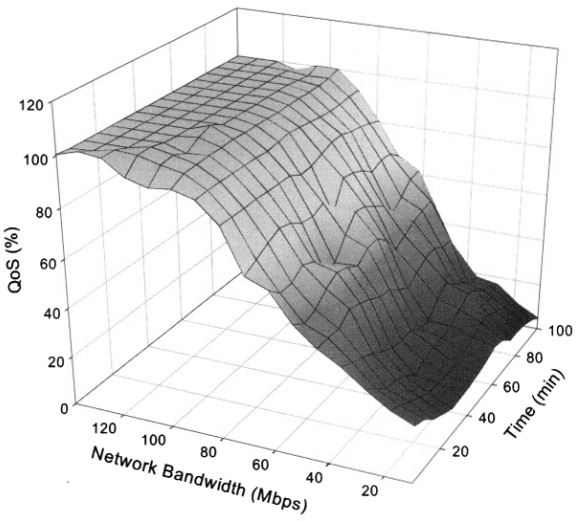
32Mbps 이하일 경우, 다른 전략보다 높은 QoS를 보장한다. 그러나, 32Mbps 이상일 경우에는 연기&드롭 전략이 폴 전략보다 높은 QoS를 제공한다. 또한, 드롭 전략은 71Mbps에서는 폴 전략보다 성능이 더욱 높게 나타난다. 이 그래프

〈표 2〉 적응 프로세스 알고리즘

```
If (QoS Boundary < 43) {
   If (Bandwidth ≤ 32)
      Adapt_Strategy (Poll)
   Else
      Adapt_Strategy (DD)
}
Else {
   If (QoS Boundary > 68) {
      If (Bandwidth > 71)
         Adapt_Strategy (DD)
      Else
         Adapt_Strategy (Drop)
    }
   Else {
      If (32 < Bandwidth)
         Adapt_Strategy (DD)
      Else
         Adapt_Strategy (Poll)
   }
}
```

를 통해 네트워크 대역폭이 증가할수록 연기&드롭과 드롭 전략이 속한 이벤트 주입 전략들이 높은 QoS를 보장하는 한편, 폴 전략이 속한 시간 주입 전략을 사용하는 시스템의 QoS는 더욱 감소되는 것을 볼 수 있다.

위치 감지 전략은 <표 2>에서 보여주는 적응 프로세스 알고리즘을 통하여 실행된다. 이 알고리즘은 감지 프로세스 전략 테이블의 정책을 근거하여 작성되었으며, 최종적으로 적응 프로세스 제어기는 이 적응 프로세스 알고리즘을 통하여 동작하게 된다. 적응 프로세스 제어기는 선택한 감지 프로세스 전략이 올바르게 실행되는지를 확인하기 위해, 동작 중인 위치 감지 시스템의 네트워크와 QoS 정보를 지속적으로 관측하게 된다. 만약, 현재 프로세스 전략이 대역폭 영향이 아닌 시스템 오동작으로 인하여 올바른 성능을 내지 못할 경우에는 사용자에게 지속적이고 신뢰성 있는 서비스를 제공하기 위해서 적응 프로세스 알고리즘에 따라 두 번째 우선순위를 가진 전략이 선택된다.

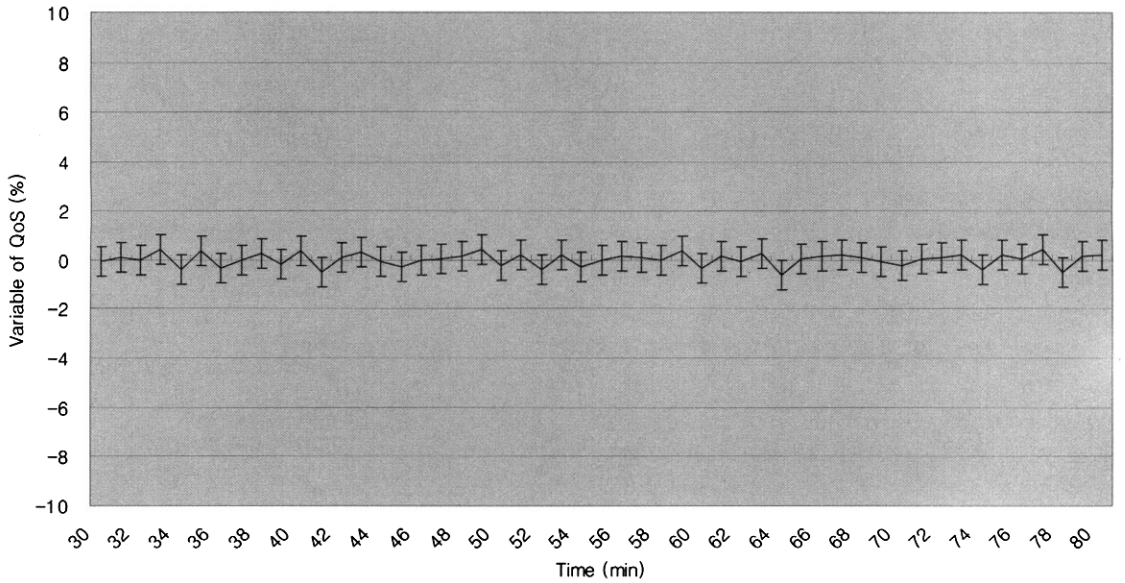


(그림 8) 대역폭에 따른 QoS 변화

4. 성능 평가

이번 장에서는 위치 감지 프로세스 전략을 위한 자동화 구조의 성능을 측정하기 위해 실험된 내용 결과를 보여준다. 제안된 시스템 구조의 성능을 측정하기 위해서 [13]에서 사용한 파라미터를 기준으로 동일하게 시뮬레이션을 수행하였다. [13]에서는 네트워크 대역폭에 따른 각 프로세스 전략별로 보장할 수 있는 QoS를 측정하였다. 사용자 수는 104 로 지정하고, 시스템의 최대 네트워크 대역폭은 140Mbps 로 제한하고 102 사이클(단위: 분) 동안 시뮬레이션을 수행했다.

사용자의 최대 이동 반경을 임의로 결정하여, 정적인 객체와 동적인 객체가 공존하는 유비쿼터스 환경의 특징을 고려하였다. (그림 8)은 시간과 변화하는 네트워크 대역폭에 따른 자동화 구조의 QoS 결과 값을 보여준다. X 축은 각 시간에 측정된 네트워크 대역폭이며, Y 축은 시뮬레이션 시간을 나타낸다. Z 축은 프로세스 전략의 QoS를 표시한 것이다. 각 프로세스의 QoS를 측정하기 위해 해당하는 프로세스마다 QoS를 백분율로 정의하였다. Ntot는 객체들이 위치를 변화할 때마다 생성되는 이벤트들의 수, Ndet는 성공적으로 감지한 이벤트들의 수, 그리고 Ndev는 성공적으로 시스템에 전송된 이벤트들의 수이다. 그러므로, 폴 전략의 QoS는 발생한 모든 위치 변화에 대해 성공적으로 감지된 이벤트 수를 뜻하며, Ndet/Ntot로 정의할 수 있다. 드롭 전략과 연기&드롭 전략의 QoS는 모든 발생된 이벤트에 대해 성공적으로 전송된 이벤트 수이므로, Ndev/Ntot로 정의할 수 있다. 네트워크 대역폭이 낮은 상태에서는 위치 시스템의 QoS도 비교적 낮게 측정되었지만, 대역폭 40Mbps부터



(그림 9) 위치 감지 시스템 자동화 구조의 QoS 변화율

는 자동화 구조에서 원만한 QoS 향상이 이루어지는 것을 확인할 수 있다. 그 이유는 적응 프로세스 제어기가 네트워크 대역폭 상황에서 최적 프로세스 전략을 선택했기 때문이다. (그림 9)에서는 측정시간 30 에서 80 동안 사용자에게 제공된 서비스의 QoS 변화율을 보인다. 자동화 구조를 사용한 위치 감지 시스템은 동적인 네트워크 대역폭 상황에서도 극히 적은 양의 QoS 변화율을 나타낸다. 이것은 시스템 동작 중에 자율 모니터링 관리자가 지속적으로 시스템 QoS 상태를 관찰함으로써 적합한 프로세스 전략으로 신속하게 전환하여 끊김 없는 서비스를 제공하였기 때문이다.

5. 결과 및 향후 연구 방향

유비쿼터스 환경에서 위치 감지 시스템은 이동 객체의 위치를 감지하고 요구 사항을 파악하는 중요한 요소이기 때문에 높은 QoS 보장 기법 마련이 시급하다. 특히, 사용자가 어떠한 위치 감지 시스템으로 이동하더라도 동일하게 높은 질의 서비스를 제공받을 수 있도록 해주어야 한다. 본 연구에서는 자율 컴퓨팅의 자가 최적화 기법을 이용해 위치 감지 시스템 성능을 향상시키는데 초점을 맞추었다. 위치 감지 프로세스 전략의 자동화를 위해서 자율 모니터링 관리자와 적응 프로세스 제어기를 가지는 자동화 구조를 제안하였다. 제안된 자동화 구조를 통해 위치 감지 시스템의 네트워크 대역폭과 QoS 상태에 따라 적합한 프로세스 전략을 자율적으로 선택함으로써 시스템의 성능을 최적화 시킬 수 있었다. 특히, 낮은 대역폭에서 높은 성능을 내지 못하는 동기 모델의 취약점을 시한 모델을 기반으로 하는 폴 전략을 동작 중에 적용할 수 있도록 하여 이를 해결하였다. 따라서 제안된 자동화 구조를 이용한 사용자는 어떠한 위치 감지 시스템에 존재하던지 네트워크 상황에 따른 최적의 서비스를 공급받을 수 있다.

본 연구를 통해 설계된 구조는 자율 컴퓨팅을 실현하는데 필요한 모니터링 부분과 분석 부분을 자동화 하였고, 향후 연구에서는 관리자 계획 부분에서도 자동화를 이룰 수 있도록 시스템 상황을 예측하고 자율적으로 프로세스 전략 테이블을 진화시키는 기법을 연구할 계획이다.

참 고 문 헌

[1] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," IEEE Personal Communications, pp.10-17, Aug., 2001.

[2] 김태형, 전광일 외, "유비쿼터스 컴퓨팅을 위한 통합 소프트웨어 구조," 정보과학회지, 제21권, 제5호, pp.51-60, 2003.

[3] A. Diaconescu, "Automatic Performance Management in Component Based Software Systems," Proceedings of the International Conference on Autonomic Computing, pp.214-221, May, 2003.

[4] U. Leonhardt and J. Margee, "Towards a General Location Service for Mobile Environments," Proceedings of the International Workshop on Services in Distributed and Networked Environments, pp.43-50, June, 1996.

[5] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," IEEE Computer, Vol.34, No.8, pp.57-66, Aug., 2001.

[6] J. Kephart and D. Chess, "The Vision of Autonomic Computing," IEEE Computer, Vol.36, No.1, pp.41-50, Jan., 2003.

[7] D. Garlan, B. Schmerl and J. Chang, "Using Gauges for Architecture-based Monitoring and Adaptation," Proceedings of the Working Conference on Complex and Dynamic Systems Architecture, pp.200-205, Dec., 2001.

[8] D. Garlan, S. Cheng and B. Schmerl, "Increasing System Dependability through Architecture-based Self-repair," Architecting Dependable Systems, Lecture Notes in Computer Science, Vol.2677, Springer-Verlag, 2003.

[9] A. Diaconescu, "A Framework for Using Component Redundancy for Self-adapting and Self-optimising Component-based Enterprise Systems," ACM Student Research Competition, OOPSLA, pp.390-391, Oct., 2003.

[10] C. Kirsch, "Principles of Real-Time Programming," Proceedings of the Second International Conference on Embedded Software, pp.61-75, Oct., 2002.

[11] S. Fischmeister and G. Menkhaus, "L2: A Novel Concept for Cell-based Location-Aware Services," Salzburg University, Technical Report C45, 2002.

[12] A. Ganek and T. Corbi, "The Dawning of the Autonomic Computing Era," IBM System Journal, Vol.42, No.1, pp.5-18, Jan., 2003.

[13] S. Fischmeister, G. Menkhaus and A. Stumpf, "Location-Detection Strategies in Pervasive Computing Environments," Proceedings of the IEEE International Conference on Pervasive Computing and Communications, pp.23-26, Mar., 2003.

[14] U. Leonhardt, "Supporting Location-Awareness in Open Distributed Systems," PhD thesis, Imperial College of Science, London University, May, 1998.

최 호 영

e-mail : funkyleo@lge.com

2003년 배재대학교 컴퓨터공학과(학사)

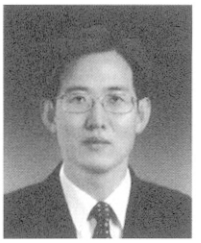
2005년 아주대학교 정보통신전문대학원 (공학석사)

2005년~현재 LG전자 단말연구소 연구원

관심분야 : 유비쿼터스 컴퓨팅, 위치 감지 시스템, 임베디드 시스템 등

최 창 열

e-mail : clchoi@ajou.ac.kr
1999년 아주대학교 정보통신공학과(공학사)
2000년 아주대학교 정보통신전문대학원 (공학석사)
2002년~현재 아주대학교 정보통신전문대학원 박사과정
관심분야 : 유비쿼터스 컴퓨팅, 오토노믹 컴퓨팅, 고가용성 시스템, 미들웨어 등



김 성 수

e-mail : sskim@ajou.ac.kr
1982년 서강대학교 전자공학과(공학사)
1984년 서강대학교 전자공학과(공학석사)
1995년 Texas A&M University 전산학과 (공학박사)
1983년~1996년 삼성전자 수석연구원
2002년~2003년 Texas A&M University 교환교수
1996년~현재 아주대학교 정교수
관심분야 : 유비쿼터스 컴퓨팅 및 네트워크, 오토노믹 컴퓨팅, 결함허용 시스템