안드로이드 디바이스를 위한 에너지 효율적 컨텍스트 모니터링 기법†

(Energy-Efficient Context Monitoring Methods for Android Devices)

김문권‡ (Moon Kwon Kim) 이재유§ (Jae Yoo Lee) 김수동¶ (Soo Dong Kim)

요 약 스마트 디바이스의 보급과 함께 컨텍스트 인지 애플리케이션에 대한 수요가 증가하는 추세이다. 하지만, 센서를 통한 컨텍스트의 수집은 추가적인 배터리 소비가 요구된다. 이것은 제한된 배터리 수명을 가진 모바일 디바이스에서 다수의 컨텍스트 인지 애플리케이션을 구동하는데 큰 제약 사항이 된다. 그러므로 에너지 효율적인 컨텍스트 모니터링 기법이 요구된다. 본 논문에서는 안드로이드 디바이스에서 가능한 네 가지 컨텍스트 모니터링 기법을 제시하고, 각 기법 간의 에너지 소모량을 분석하여 적합한 적용 지침을 제안한다. 본 논문에서 제안된 기법을 컨텍스트 모니터링 애플리케이션에 적용하면 컨텍스트 모니터링에 소비되는 에너지 소모량을 효과적으로 감소시킬 수 있다. 제안된 기법을 검증하기 위해 적용하여 사용자 행동 인지 애플리케이션을 개발하여 각 기법에 대한 에너지 소모량을 정량적으로 비교한다.

키워드 모바일 컴퓨팅, 컨텍스트 모니터링, 에너지 효율, 안드로이드

Abstract Along with increasing supplies of smart devices, a proliferation of context-aware applications is came. However, acquiring contexts through sensors requires considerable energy consumption. It has became big constraints on running many context-aware applications in mobile devices having limited battery capacity. Hence, energy-efficient methods for monitoring contexts are highly required. In this paper, we propose four context monitoring methods, analyse energy consumption in each method, and provide guidelines for applying the methods. It is effective to decrease energy consumption for monitoring contexts with applying the methods. To assess the proposed methods, we implement an application that is aware of a user's motion and show quantitative comparison between each of the methods.

Key words Mobile Computing, Context Monitoring, Energy Efficiency, Android

† 본 연구는 산업통상자원부의 지원을 받는 로봇산업원천기술개발 사업의 연구결과로 수행되었음.
‡ 학생회원 : 숭실대학교 컴퓨터학과
mkdmkk@gmail.com
§ 학생회원 : 숭실대학교 컴퓨터학과
jaeyoo1981@gmail.com (Corresponding Author)
¶ 종신회원 : 숭실대학교 컴퓨터학부 교수
sdkim777@gmail.com

1. 서 론

센싱 능력을 보유한 스마트 디바이스 보급의 증가로 인해 사용자 컨텍스트를 기반으로한 개인화된 애플리케이션에 대한 수요가 증가하는 추세이다[1]. 컨텍스트는 어떠한 상황을 특징지을 수 있는 정보로서 컨텍스트 인지 애플리케이션은 스마트 디바이스의 센서로부터 수집되는 미가공

데이터를 이용하여 특정 상황의 변화에 동적으로 반응할 수 있어 전통적인 애플리케이션과는 차별화된 특징을 제공한다.

스마트 디바이스의 컨텍스트 수집은 디바이스에 장착된 다양한 하드웨어 센서를 활용한다. 따라서, 컨텍스트 인지 애플리케이션의 실행은 센서를 구동하는데 필요한 추가적인 에너지 소모가 요구된다[2][3][4]. <표 1>은 세 가지 안드로이드 디바이스의 배터리 용량, 가용센서의 수, 모든 가용 센서를 통해 컨텍스트를 한번 측청했을 때의 배터리 소모량, 배터리가 방전될 때까지의 소요 시간을 보여준다. 본 연구의 실험을 통해 도출한 이 결과에 따르면 배터리 수명이 제한적인 스마트 디바이스에서 지속적인 컨텍스트 모니터링을 유지하기 힘들다.

<표 1> 스마트 디바이스의 센싱 배터리 소모량

측정치 \ 디바이스	Nexus 7	Galaxy S3	Galaxy Note 1
배터리 용량(mAh)	3950	2100	2500
가용센서 수	12	10	9
배터리 소모량(mA)	1.76×10^{-10}	1.01×10^{-10}	0.96×10^{-10}
배터리 방전 시간	1시간 44분	1시간 36분	1시간 56분

본 논문에서는 안드로이드 디바이스에서 에너지 효율성을 고려한 컨텍스트 모니터링 기법과 각 기법의 적용 지침을 제안한다. 컨텍스트 모니터링을 필요로하는 애플리케이션은 각 기법의 적용 지침을 기반으로 적합한 모니터링 기법을 선정하고 적용할 수 있다.

2. 관련 연구

Wissen[5]의 연구에서는 에너지 효율성이 높은 컨텍스트 모니터링을 위한 표현 기반(Expression-Based) 프레임워크를 제시한다. 이 프레임워크는 클라이언트 애플리케이션이 개별적으로 컨텍스트 모니터링 하지 않고 특정 표현을 만족하는 상황에서 이벤트를 받는 방식을 제공한다. 이 연구에서는 상황을 판단하여 이벤트를 받는 방식만을 고려하여 다양한 도메인에 적용이 어려울 수 있으며, 에너지 효율성에 대한 정량적 검증이 이루어지지 않았다.

Kramer[6]의 연구에서는 여러 컨텍스트 획득 방법을 제공하는 컨텍스트 엔진을 제시한다. 이 엔진의 주 목적은 컨텍스트 획득 및 제공으로, 여러 클라이언트 애플리케이션에서 이 엔진을 이용하여 여러 종류의 컨텍스트를 포함한 컨텍스트 집합을 획득할 수 있도록 하는 것이다. 이 연구는 클라이언트 애플리케이션이 개별적으로 컨텍스트 모니터링을 수행해야 하는 하는 것을 컨텍스트 엔진에서 컨텍스트 모니터링을 전담하여 효율성을 높일 것으로 기대되지만 이에 대한 정량적인 검증이 이루어지지 않았으며, 각 컨텍스트 획득 방법 적용에 대한 적합한 상황을 제시하고 있지 않다.

Taleb[7]의 연구에서는 에너지 효율성, 센서의 정확도를 고려하여 컨텍스트 모니터링에 사용될 센서를 선택하는 알고리즘을 제시한다. 이 연구에서는 각 센서마다 최소 에너지 소모량, 정확도를 고려하여 센서의 효율성 메트릭을 제시하고 센서 선택 알고리즘에서 이 메트릭을 이용한다. 본 연구에서 제시하는 메트릭은 컨텍스트 획득에 대한 에너지 소모량만을 고려하여 컨텍스트를 처리하는 에너지 소모량, 컨텍스트를 관리하는 에너지 소모량, 컨텍스트 애플리케이션의 개수 등, 컨텍스트 모니터링의 에너지 효율성에 영향을 주는 여러 요소들을 고려하지 않고 있다.

기존 연구에서는 컨텍스트 모니터링 기법에 대한 고려와 각 기법별 정량적인 에너지 효율성 검증이 부족하고, 컨텍스트 모니터링의 다양한 기법에 대한 장단점, 적합한 적용 상황, 적용 지침에 대한 고려가 부족하다. 그러므로 본 연구

에서는 컨텍스트 모니터링 기법의 분류 기준과 그 분류 기준으로부터 컨텍스트 모니터링 기법 분류를 도출하고, 각 분류에서 에너지 효율성을 측정하기 위한 메트릭을 제시한다. 또한 각 기법을 구현하고 제시한 메트릭을 이용하여 각 기법의 에너지 효율성을 비교 및 검증한다.

3. 컨텍스트 모니터링 기법의 분류 기준 및 에너지 효율성 메트릭

본 장에서는 안드로이드의 특성을 고려하여 컨텍스트 모니터링 기법을 제안 및 분류하고 각 기법의 에너지 효율성 측정을 위한 메트릭을 제시한다.

3.1 컨텍스트 모니터링 기법의 분류 기준

본 절에서는 안드로이드 디바이스에서 가능한 컨텍스트 모니터링 기법의 분류 기준을 제시하고, 제시된 기준에 따른 네 가지 컨텍스트 모니터링 기법을 제안한다.

분류 기준 1. 컨텍스트 획득 방식(Acquisition Method): 컨텍스트 모니터링을 위한 메시지 방식은 푸싱(Pushing)과 풀링(Pulling)으로 구분된다. 푸싱은 컨텍스트 이벤트를 감지하여 비동기적으로 컨텍스트를 획득하는 방식이다. 따라서 컨텍스트의 변화에 즉각적인 반응이 가능하지만 클라이언트 애플리케이션의 필요에 따른 능동적인 컨텍스트 획득이 어렵고, 지속적인 컨텍스트 이벤트 감지로 인해 에너지 소모가 크다. 반면에, 풀링 방식은 클라이언트 애플리케이션에서 능동적인 컨텍스트의 획득이 가능하다. 안드로이드 서비스 빌딩 블록은 컨텍스트를 수집 및 관리하고, 다수의 클라이언트 애플리케이션으로 풀링 방식의 컨텍스트 모니터링 기능을 제공하기에 알맞다. 컨텍스트의 수집 및 관리를 위한 공통적인 기능성을 재사용 가능한 서비스를 통해 제공하여 클라이언트 애플리케이션에서 중복 컨텍스트 수집 기능을 제거한다[8][9].

분류 기준 2. 획득하고자 하는 컨텍스트의 시점(Timeframe): 컨텍스트의 시점에는 현재, 최근, 과거가 있다. 현재(Current) 컨텍스트는 수집된 가장 최근의 컨텍스트이다. 최근(Recent) 컨텍스트는 일정 기간 이전부터 현재까지 수집된 컨텍스트이다. 과거(Past) 컨텍스트는 일정 기간 이전의 컨텍스트를 의미한다.

제시된 두 가지 분류 기준을 기반으로 모두 여섯 가지의 컨텍스트 모니터링 기법을 도출할 수 있지만 푸싱 방식은 이벤트 발생에 따라 현재 컨텍스트를 갱신하는 방식으로 최신 컨텍스트 또는 과거 컨텍스트에 대하여 적용하는 것은 에너지 효율성의 측면에서 효과적이지 않다.

기법 1은 안드로이드에서의 기본 컨텍스트 수집 방식으로, 시스템으로부터 푸싱 방식으로 현재 미가공 컨텍스트를 수집한다. 기법 2는 서비스를 이용하여 현재 미가공 컨텍스트를 클라이언트에게 풀링 방식으로 제공하며, 기법 3은 최근 컨텍스트를 기법 2와 같은 방식으로 클라이언트에게 제공한다. 기법 4는 과거 미가공 컨텍스트를 콘텐트 프로바이더를 이용하여 클라이언트에게 제공한다.

3.2 에너지 소모량 메트릭

컨텍스트 모니터링을 위한 에너지 소모량(Energy Consumption, ENG)은 컨텍스트 수집(Acquisition, ACQ), 컨텍스트 획득(Retrieval, RET), 획득한 컨텍스트 관리(Managmenet, MAN), 획득한 컨텍스트 처리(Processing, PRO)에 필요한 에너지 소모량으로 이루어진다. 컨텍스트 모니터링을 위한 1분간의 에너지 소모량은 아래와 같이 정의되며 단위는 mAh이다. 아래 수식의 인자는 <표 2>와 같다.

$$ENG = (ACQ + MAN) \times S + \sum_{i=1}^{N} ACQ \times C_i + RET \times R_i + PRO_i \times (C_i + R_i)$$

<표 2> 메트릭 *ENG*의 인자

인자	설명
N	클라이언트 수
S	모니터링 서비스에서의 컨텍스트 수집 수
C	클라이언트에서의 컨텍스트 수집 수
R	서비스로부터 획득한 컨텍스트 집합의 수
ACQ	컨텍스트를 수집 비용
MAN	모니터링 서비스가 수집한 컨텍스트 관리 비용
RET	서비스로부터의 컨텍스트 집합 획득 비용
PRO	클라이언트에서 획득한 컨텍스트/이벤트 처리 비용

각 기법의 에너지 사용량을 측정하기 위한 ENG에서 고려되지 않는 인자를 소거하여 두 가지의 변형 메트릭, ENGa와 ENGb를 제시한다.

기법 1에서는 서비스를 사용하지 않고 각 클라이언트 애플리케이션에서 시스템으로부터 컨텍스트를 수집하기 때문에 S, R, E, MAN, RET를 소거하여 다음과 같은 메트릭 ENGa를 에너지 사용량을 구하는데 사용할 수 있다.

$$ENG^a = \sum_{i=1}^{N} (ACQ + PRO_i) \times C_i$$

기법 2, 기법 3, 기법 4에서는 서비스를 이용하여 시스템으로부터 컨텍스트를 수집하고 관리하고 클라이언트 애플리케이션이 서비스로부터 컨텍스트를 풀링 방식으로 획득하기 때문에 C, E를 소거하여 다음과 같은 메트릭 ENGb를 에너지 사용량을 구하는데 사용할 수 있다.

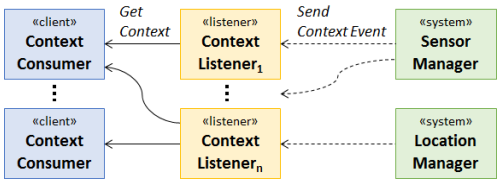
$$ENG^b = (ACQ + MAN) \times \sum_{i=1}^{N} (RET + PRO_i) \times R_i$$

4. 에너지 효율성을 고려한 컨텍스트 모니터링 기법의 설계 및 적용 지침

본 장에서는 각 기법에 대한 특징, 장단점을 소개하고 각 기법을 위한 설계 모델을 제시한다.

4.1 기본 컨텍스트 모니터링 기법

안드로이드에서 제공하는 기본 컨텍스트 모니터링 기법을 이용하여 가용한 하드웨어 센서 및 소프트웨어 센서를 파악하고 센서 이벤트 리스너와 로케이션 리스너를 컨텍스트 매니저에 등록하고 컨텍스트 이벤트를 받는다. 본 기법은 컨텍스트 획득을 필요로하는 모든 클라이언트 애플리케이션에서 위의 동일한 절차를 수행해야 한다. [그림 1]은 안드로이드에서 제공하는 기본 컨텍스트 모니터링 기법의 구조를 보여준다.



[그림 1] 기본 컨텍스트 모니터링 구조

기법 1은 메트릭 ENGa의 결과 값이 작게 나오는 상황에 적합하다. 즉, 컨텍스트 수집 에너지 소모량과 컨텍스트 처리 에너지 소모량을 줄여야 한다.

클라이언트 애플리케이션의 수(N): 클라이언트 애플리케이션 별로 컨텍스트 이벤트를 처리해야 한다. 컨텍스트 이벤트는 다량으로 발생되기 때문에, 컨텍스트 애플리케이션의 수가 증가함에 따라 컨텍스트 수집 비용과 컨텍스트 처리 비용이 증가한다. 그러므로 컨텍스트를 획득 및 처리하는 클라이언트 애플리케이션의 수가 적어야 효율적이다.

컨텍스트 수집 주기: 기법 1에서 컨텍스트 수집 에너지 소모량을 줄이는 방법은 컨텍스트 이벤트 주기를 길게 하여서 컨텍스트 수집을 적게 하는

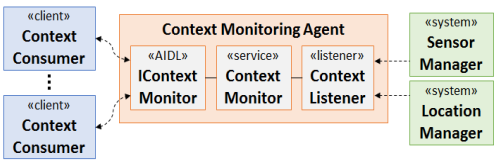
것이다. 센서 이벤트 리스너와 로케이션 리스너를 안드로이드 시스템에 등록할 시점에 이벤트 발생 주기를 정할 수 있다.

컨텍스트 처리 복잡도: 다량의 컨텍스트 이벤트를 처리하기 때문에 컨텍스트 처리의 복잡도가 크면 컨텍스트 처리 비용 크게 증가한다.

4.2 현재 컨텍스트 모니터링 기법

안드로이드에서는 Background 프로세싱을 위해 서비스(Service)와 프로세스 간의 통신을 위한 Android Interface Definition Language(AIDL) [10]을 제공한다. 기법 2는 Background 에서 컨텍스트를 수집 및 관리하기 위한 서비스인 컨텍스트 모니터(ContextMonitor)와 컨텍스트 모니터가 수집한 컨텍스트를 클라이언트에게 제공하기 위한 AIDL 인터페이스인 컨텍스트 모니터 인터페이스(IContextMonitor)를 활용한다.

컨텍스트 모니터가 센서 매니저와 로케이션 매니저에 컨텍스트 리스너를 등록하여 지속적으로 컨텍스트를 획득하고, 최신 컨텍스트를 메모리상에 유지한다. 그러므로 클라이언트 애플리케이션은 컨텍스트 획득을 위해 개별적으로 안드로이드 센서 매니저에 센서 이벤트 리스너를 등록할 필요 없이, 컨텍스트 모니터 인터페이스를 통해 컨텍스트를 필요에 따라 획득한다. [그림 2]는 현재 컨텍스트 모니터링 기법의 구조를 보여준다.



[그림 2] 현재 컨텍스트 모니터링 기법의 구조

기법 2에서는 클라이언트 애플리케이션이 컨텍스트를 풀링 방식으로 획득 및 처리하기 때문에, 기법 1에서 푸싱 방식으로 발생하는 오버헤드를 줄인다. 이 방식은 클라이언트 애플리케이션의 수가 증가함에 따라 더욱 효율적이다. 하지만 기법 2는 최신 컨텍스트만을 유지하기 때문에, 클라이언트 애플리케이션이 축적된 컨텍스트를 요구하는 경우 적합하지 않다. 또한, 서비스로부터 컨텍스트를 획득하는 주기가 짧을 경우, 서비스로부터 컨텍스트를 획득하기 위한 오버헤드로 인해 본 기법을 적용하는 것이 효율적이지 않을 수 있다. 즉, ENGb를 구하기 위한 모든 인자값이 주어졌을 때, 다음과 같은 조건이면 본 기법을 적용하는 것이 효율적이다.

$$\frac{\sum_{i=1}^{N}(RET+PRO_i)\times R_i}{\frac{\sum_{i=1}^{N}PRO_i}{N}\times S}<1$$

이 수식에서 분모 부분은 기법 1을 적용했을 때의 예측 에너지 소모량이다. 본 기법의 컨텍스트 관리 비용(MAN)은 에너지 효율성에 큰 영향을 미치지 않기 때문에 고려하지 않는다.

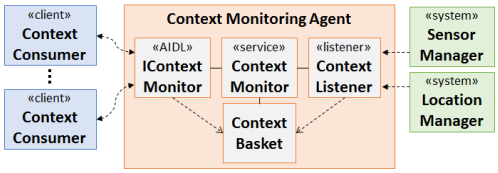
컨텍스트 획득 횟수: 클라이언트 애플리케이션이 필요로 하는 컨텍스트 획득 주기는 안드로이드 시스템이 기본적으로 제공하는 컨텍스트 이벤트 주기보다 길다. 이러한 애플리케이션에서 기법 1을 적용하는 것은 필요치 않게 많은 컨텍스트 처리 작업을 수행해야하므로 효율적이지 않다.

클라이언트 애플리케이션의 수(N): 클라이언트 애플리케이션의 수(N)가 복수개이면 기법 1보다 기법 2가 효율적이다. 기법 2는 컨텍스트 모니터링을 복수개의 클라이언트 애플리케이션에서 개별적으로 수행하지 않고 서비스가 전담하면 중복적인 작업을 제거하여 에너지 효율성 관점에서 좋다.

컨텍스트 처리 복잡도(PRO): 컨텍스트 처리 작업의 복잡도가 높을수록 위의 수식이 0에 더 가까워지기 때문에 본 기법의 적용이 기법 1을 적용하는 것에 비해 효과적이다.

4.3 최근 컨텍스트 모니터링 기법

컨텍스트 바스켓(ContextBasket)은 최근 컨텍스트를 관리하는 클래스로 복수개의 컨텍스트를 메모리 상에서 저장하기 위한 큐(Queue)를 가진다. 기법 3은 컨텍스트 바스켓을 이용하여 컨텍스트 리스너를 통해 수집된 최근 컨텍스트를 유지한다. 클라이언트 애플리케이션은 최신 컨텍스트 뿐만 아니라 축적된 최근의 컨텍스트를 컨텍스트 모니터 인터페이스를 통해 획득할 수 있다. [그림 3]은 최근 컨텍스트 모니터링 기법의 구조를 보여준다.



[그림 3] 최근 컨텍스트 모니터링 기법의 구조

기법 3에서 컨텍스트 모니터는 축적된 최근 컨텍스트 집합을 클라이언트 애플리케이션에게 제공함으로써 클라이언트 애플리케이션이 연속적인 데이터를 가공해야 하는 패턴 인식[11]과 같은 프로세싱을 효율적으로 수행할 수 있도록 한다. 즉, 클라이언트 애플리케이션에서 컨텍스트를 요청하는 횟수를 줄일 수 있는 기회가 많아진다. 하지만 짧은 시간에 다량의 데이터가 발생하는 컨텍스트 모니터링의 특성에 의해 컨텍스트 바스켓 관리 오버헤드가 발생한다. 즉, ENGb를 구하기 위한 모든 인자값이 주어졌을 때, 다음과 같은 조건이면 본 기법을 적용하는 것이 효율적이다.

$$\frac{MAN\times S+\sum_{i=1}^{N}(RET+PRO_i)\times R_i}{\sum_{i=1}^{N}(RET\times\alpha+PRO_i)\times R_i\times\beta}<1$$

이 수식에서 분모 부분은 기법 2를 적용했을 때의 예측 에너지 소모량이다.

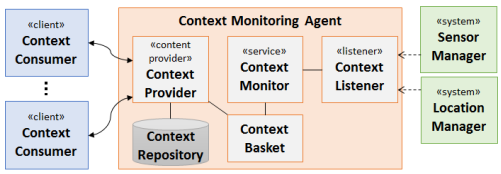
컨텍스트 획득 오버헤드(α): 기법 3에서의 컨텍스트 획득 오버헤드(RET)와 기법 2에서의 컨텍스트 획득 오버헤드(RET$\times\alpha$)는 차이가 크지 않다. 그러므로 α를 구하지 못할 때는 1로 적용해도 효율성 비교에 큰 영향을 주지 않는다.

컨텍스트 획득수 비율(β): 일반적으로, 기법 2에서의 컨텍스트 획득 횟수(R$\times\beta$)에 비해 본 기법의 컨텍스트 획득 횟수(R)가 적다. 왜냐하면 기법 2에서는 최신 컨텍스트만을 제공하고 연속적인 컨텍스트가 필요한 클라이언트 애플리케이션이 짧은 주기로 서비스로부터 컨텍스트를 획득하기 때문이다. 즉, 기법 2와 기법 3의 컨텍스트 획득수 비율인 β가 클수록 본 기법의 효율성은 더욱 높아진다.

클라이언트 애플리케이션의 수(N): 클라이언트 애플리케이션의 수(N)가 많아지면 기법 2을 사용한 예상 컨텍스트 획득수가 가증되므로(R$\times\alpha\times$N) 기법 3의 적용이 더욱 효과적이다.

4.4 과거 컨텍스트 모니터링 기법

기법 3에서 제공하는 최신 컨텍스트 이전에 발생한 과거 컨텍스트를 획득하기 위한 기법이다. 기법 3에서 수집된 최신 컨텍스트는 Context Basket에 저장되고, 지속적으로 갱신된다. Context Basket에서 제외되는 컨텍스트는 과거 컨텍스트로 취급되며, 안드로이드 콘텐트 프로바이더(Content Provider) 형태의 컨텍스트 프로바이더(Context Provider)를 통해 SQLite 데이터베이스의 컨텍스트 저장소(Context Repository)에 저장된다. [그림 4]는 과거 컨텍스트 모니터링 기법의 구조를 보여준다.



[그림 4] 과거 컨텍스트 모니터링 기법의 구조

새로운 컨텍스트 이벤트가 발생하면, 컨텍스트 모니터는 새로운 컨텍스트를 Context Basket에 추가한다. 저장된 컨텍스트의 수가 정해진 Context Basket의 크기를 초과하게 되면, 저장된 순서에 따라 가장 먼저 수집된 컨텍스트가 Context Basket에서 제거되고, 컨텍스트 프로바이더에게 전달된다.

기법 4는 축적되는 콘텐트 프로바이더를 통해 과거 현재까지의 컨텍스트 집합을 클라이언트 애플리케이션에게 제공함으로써 클라이언트 애플리케이션이 컨텍스트의 집합으로부터 사용자의 패턴 분석이나 새로운 가치를 창출하기 위한 컨텍스트 처리과정을 효율적으로 수행할 수 있도록 한다. 따라서, 클라이언트 애플리케이션은 특정 기간의 컨텍스트 집합을 컨텍스트 모니터링 에이전트 서비스로부터 획득하여 각 애플리케이션의 목적에 따라 가공하는 기능만을 수행한다. 짧은 주기로 현재 혹은 최근 컨텍스트를 획득해야 하는 기법 2와 3에 비하여 필요한 시점에 컨텍스트를 획득하는 기법 4의 컨텍스트 요청 횟수가 보다 적어진다. 따라서, ENGb를 구하기 위한 모든 인자 값이 주어졌을 때, 다음과 같은 조건이면 본 기법을 적용하는 것이 효율적이다.

$$\frac{MAN\times S+\sum_{i=1}^{N}(RET+PRO_i)\times R_i}{MAN\times\alpha\times S+\sum_{i=1}^{N}(RET\times\beta+PRO_i)\times R_i\times\gamma}<1$$

이 수식에서 분모 부분은 기법 3를 적용했을 때의 예측 에너지 소모량이다.

컨텍스트 관리 오버헤드(α): 기법 3에 비해 기법 4는 수집한 컨텍스트를 콘텐트 프로바이더를 이용하여 데이터베이스에 저장하기 때문에 컨텍스트 관리 비용이 크다. 즉 α는 1보다 작은 값을 가진다.

컨텍스트 획득 오버헤드(β): 기법 4에서는 클라이언트 애플리케이션이 컨텍스트 획득을 위해 콘텐트 프로바이더를 이용하여 데이터베이스에 연결하기 때문에 컨텍스트 획득 비용이 기법 3에 비해 크다. 즉 β는 1보다 작은 값을 가진다.

컨텍스트 획득수(γ): 앞서 언급한 컨텍스트 관리 오버헤드와 컨텍스트 획득 오버헤드를 감당하면서도 본 기법이 효율적이려면 컨텍스트의 획득 수가 기법 3에 비해 상당히 적어야 한다. 즉 γ이 큰 값을 가져서 기법 4의 컨텍스트 획득 비용과 컨텍스트 처리 비용을 줄여야 한다. 이는 기법 4의 경우 주기적인 컨텍스트 획득에는 큰 효율을 얻을 수 없음을 의미한다.

5. 실험 및 평가

본 장에서는 컨텍스트 모니터링 기법 별로 에너지 소모량 메트릭을 이용하여 에너지 소모량을 측정하고 비교한다. 기법 별 적용 지침을 기반으로 본 실험시나리오에 가장 적합한 컨텍스트 모니터링 기법을 확인한다.

5.1 실험 환경 및 시나리오

본 실험은 Nexus 7에서 수행되었으며, 에너지 소모량을 구하기 위해, 1%의 배터리가 소모되는 동안의 시간을 측정하여 1분동안의 배터리 소모량(ENG)을 유도한다. Nexus 7의 전체 배터리량은 3,950mAh이고, 1%의 배터리량은 39.5mAh이다. 실험 환경을 동일하게 하기 위해 본 실험과 무관한 애플리케이션은 모두 종료하고 화면 밝기를 최소로 유지한다.

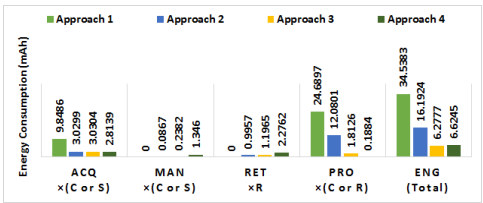
본 실험에서는 획득한 컨텍스트를 이용하여 사용자의 행동을 인지한다. 이를 위해 분류(Classification) 알고리즘인 J48을 사용하고 학습 데이터로 [12]에서 적용한 사용자 행동 데이터를 사용하였다. 기법 1은 클라이언트에서 지속적으로 모니터링한다. 기법 2에서는 0.1초의 주기로 서비스로부터 최신 컨텍스트를 획득하고 기법 3에서는

1초의 주기로 최근 10개의 컨텍스트를 획득하고 기법 4에서는 10초의 주기로 100개의 컨텍스트를 획득한다.

5.2 컨텍스트 모니터링 기법 별 실험 결과

본 실험은 각 기법에서 세 개의 동일한 클라이언트 애플리케이션(N=3)이 컨텍스트를 획득하고, 5회의 실험을 통해 얻은 평균값을 사용한다.

실험 결과를 통해 기법 별로 각 세부 에너지 소모 비용과 전체 에너지 소모 비용을 [그림 5]와 같이 비교한다. 각 에너지 소모량 항목의 막대 그래프는 5개의 막대를 가지며, 각 막대는 순서대로 기법 1, 기법 2, 기법 3, 기법 4를 의미한다.



[그림 5] 시나리오의 기법 별 에너지 소모량 (N=3)

컨텍스트 수집 에너지 소모량(ACQ)은 기법 1의 값이 가장 높으며, 기법 2, 기법 3, 기법 4는 서로 비슷한 값을 가진다. 이는 각각의 클라이언트 애플리케이션에서 컨텍스트를 수집하는 기법 1의 방식과 그 이외 기법의 방식인 서비스에서 컨텍스트 수집을 전담하는 것과의 차이를 보여준다. 컨텍스트 관리 에너지 소모량(MAN)의 경우 기법 2에서는 매우 적은 양의 에너지 소모를 보이며, 기법 3에서도 값이 크지 않지만, 기법 4에서는 비교적 큰 에너지 소모량을 보인다. 컨텍스트 획득 에너지 소모량(RET)은 기법 2에서 가장 적은 값을 가지며 기법 3에서는 기법 2보다는 값이 크지만 큰 차이를 보이진 않는다. 그러나 기법 4에서는 콘텐트 프로바이더를 사용하기 때문에 비교적 큰 에너지 소모량을 보인다. 컨텍스트 처리 에너지 소모량(PRO)은 기법 4에서 가장 적은 값을 가진다. 이는 한번에 대량의 컨텍스트를 획득하여 컨텍스트 처리 횟수를 줄이기 때문이다. 반면에 기법 1에서는 컨텍스트를 클라이언트 애플리케이션에서 수집하여 처리하기 때문에 컨텍스트 처리 횟수가 많다. 기법 2의 컨텍스트 처리 횟수가 기법 3의 컨텍스트 처리 횟수보다 많기 때문에 에너지 소모량 역시 기법 2에서 많다.

본 실험의 시나리오는 최근의 컨텍스트를 기반으로 사용자의 행동을 판단하는 것이다. 행동 판단은 일련의 복수 가속도 컨텍스트를 통해 이루어진다. 그러므로 한번에 복수개의 컨텍스트를 획득 함으로써 컨텍스트 획득수를 줄이는 기법 3이 적합하다. 이 결과 기반으로 4장에서 제시한 기법 3과 기법 2의 비교 메트릭을 적용(α =1, β =10)하면 아래와 같이 결과 값이 1보다 작다.

$$\frac{MAN\times S+\sum_{i=1}^{N}(RET+PRO_i)\times R_i}{\sum_{i=1}^{N}(RET\times\alpha+PRO_i)\times R_i\times\beta}=\frac{0.2382+1.1965+1.8126}{(1.1965+1.8126)\times 10}=0.1044$$

그러므로 본 시나리오에서는 기법 3을 사용하는 것이 기법 2를 사용하는 것보다 에너지 효율성 측면에서 더 적합하다.

6. 결 론

스마트 디바이스의 보급과 함께 사용자의 상황을 동적으로 반영할 수 있는 컨텍스트 인지 애플리케이션에 대한 수요가 증가하는 추세이다. 스마트 디바이스의 제한된 배터리 수명은 센서를 이용한 컨텍스트 모니터링의 지속적인 수행 측면에서 주요 제약사항이다. 본 논문에서는 컨텍스트 모니터링을 위한 에너지 소모를 줄이기 위해, 안드로이드의 특성을 고려한 네 가지 컨텍스트 모니터링 기법을 제안하였다. 제안된 네 가지 컨텍스트 모니터링 기법을 적용하여 각 기법 간의 에너지

소모량을 분석하고, 적합한 적용 지침을 제안하였다. 사용자의 행동을 인지하기 위한 애플리케이션을 각 기법 별로 구현하고 실제로 에너지 소모량을 측정해서 제안한 모니터링 기법 및 적용 지침을 검증하였다. 본 논문에서 제안된 기법을 적용 지침을 기반으로 컨텍스트 모니터링 애플리케이션에 적용하면 보다 에너지 효율적인 컨텍스트 모니터링이 가능하다.

참 고 문 헌

[1] MarketsAndMarkets, Context Aware Computing Market-Global Advancements, Emerging Applications, Worldwide Forecasts and Analysis (2013-2018), March 2013.

[2] Kjaergaard, M.B., "Location-based services on mobile phones: minimizing power consumption," Pervasive Computring, IEEE, Vol. 11, No. 1, pp. 67-73, January 2012.

[3] Priyantha, B., Lymberopoulos, D., Liu, J., "LittleRock: Enabling Energy-Efficient Continuous Sensing on Mobile Phones," Pervasive Computing, IEEE, Vol. 10, No. 2, pp. 12-15, April 2011.

[4] S. K. Datta, C. Bonnet, and N. Nikaein, "Android power management: Current and future trends," 2012 The First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT2012), pp. 48–53, 2012.

[5] B. Van Wissen, N. Palmer, and R. Kemp, "Context Droid: an expression-based context framework for Android," In Proceedings of PhoneSense, 2010.

[6] D. Kramer, A. Kocurova, S. Oussena, T. Clark, and P. Komisarczuk, "An extensible, self contained, layered approach to context acquisition," In Proceedings of the Third International Workshop on Middleware for Pervasive Mobile and Embedded Computing, p. 6, Dec. 2011.

[7] S. Taleb, N. Abbas, H. Hajj, and Z. Dawy, "On sensor selection in mobile devices based on energy, application accuracy, and context metrics," In Proceedings of Third International Conference on Communications and Information Technology (ICCIT), 2013, pp. 12–16.

[8] G. Cardone, A. Cirri, A. Corradi, L. Foschini, and D. Maio, "MSF: An Efficient Mobile Phone Sensing Framework," International Journal of Distributed Sensor Networks, vol. 2013, Mar. 2013.

[9] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, "A framework of energy efficient mobile sensing for automatic user state recognition," In Proceedings of the 7th international conference on Mobile systems, applications, and services, pp. 179–192, Jun. 2009.

[10] AIDL, Android Developers, http://developer.android.com/guide/components/aidl.html

[11] Sergios Theodoridis and Konstantinos Koutroumbas, Pattern Recognition, 4thEd.,AcademicPress,2008.

[12] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," ACM SIGKDD Explorations Newsletter, vol. 12, no. 2, pp. 74–82, Mar. 2011.E79-A, No. 9, pp. 1338-1353, Sep., 1996.

저자 소개

김 문 권

2012년 숭실대학교 컴퓨터학과 학사
2013년 숭실대학교 컴퓨터학과 석사
2013년~현재 숭실대학교 컴퓨터학과 박사과정
관심분야는 소프트웨어공학, 모바일 컴퓨팅, 사물인터넷, 컨텍스트 추론

이 재 유

2007년 홍익대학교 컴퓨터학과 학사
2009년 숭실대학교 컴퓨터학과 석사
2009년~현재 숭실대학교 컴퓨터학과 박사과정
관심분야는 소프트웨어공학, 모바일 컴퓨팅, 사물인터넷, 컨텍스트 인지 서비스

김 수 동

1984년 미국 노스웨스트 미주리 주립대학 컴퓨터학과 학사
1988년 미국 아이오와 주립대학 컴퓨터학과 석사
1991년 미국 아이오와 주립대학 컴퓨터학과 박사
1991년~1993년 한국통신 연구개발단 선임연구원
1994년~1995년 현대전자 소프트웨어연구소 책임연구원
1995년~현재 숭실대학교 컴퓨터학과 교수
관심분야는 서비스 지향 아키텍처, 클라우드 컴퓨팅, 모바일 서비스, 객체지향 S/W공학, 컴포넌트 기반 개발, 소프트웨어 아키텍처