모바일 디바이스 역량 분석 기반 앱 설계 기법

박춘우, 김문권*, 김수동
숭실대학교 컴퓨터학과
e-mail : {cnsdnsla, mkdmkk, sdkim777}@gmail.com

Methods to Design Apps based on Analysis of Mobile Devices Capability

Chun Woo Park, Moon Kwon Kim*, and Soo Dong Kim
Dept. of Computer Science, Soongsil University

요 약

모바일 컴퓨팅은 모바일 디바이스의 사용 증가로 인해 연구가 활발히 진행되고 있는 분야이다. 스마트폰은 대표적인 모바일 디바이스로, 많은 앱들이 오픈 마켓을 통해 활발히 유통되고 있다. 고품질의 앱을 설계하기 위해서는 모바일 디바이스의 특징에 대한 이해와 제한적인 자원에 의해 발생할수 있는 문제들에 대한 효과적인 해결책이 필요하다. 본 논문에서는 모바일 디바이스의 역량을 정량적으로 분석하고 이를 기반으로 앱 설계시 발생할 수 있는 문제들을 효과적으로 대처하기 위한 설계 지침 제시 및 이를 기반으로 한 앱 설계 기법을 제안한다.

1. 서론

모바일 컴퓨팅이란 휴대성이 있고 무선 네트워크 역량을 지닌 모바일 디바이스를 통한 컴퓨팅을 의미한다. 스마트폰은 이러한 모바일 디바이스의 대표적인 예이며 앱은 오픈 마켓의 등장으로 그 개수와 사용량이 급격히 증가하고 있다[1].

모바일 디바이스는 휴대성과 무선 네트워크 역량 등의 데스크톱 컴퓨터와는 다른 특징들이 존재하며 일반적으로 제한적인 컴퓨팅 역량을 지닌다. 이에 의해서 앱 설계는 데스크톱 컴퓨터 기반의 소프트웨어 설계와의 차이을 지닌다. 이러한 차이를 고려한 앱에 대한 여러 연구들이 진행되고 있다.

그러나 기존 연구들은 모바일 디바이스의 특징에 대한 이해를 바탕으로 앱의 성능, 보안 등의 향상을 위한 설계 기법, 데이터 모델링 기법, 도구, 프레임워크 등을 제안하고 있으나 모바일 디바이스 역량의 정량적 분석이 부족하여 앱 설계시 고려해야하는 여러가지 문제에 대한 명확한 지침 및 기법은 다루어지지 않고 있다.

본 논문에서는 모바일 디바이스의 역량을 데스크톱 컴퓨터의 역량과 정량적으로 비교한다. 모바일 디바이스의 특징에 대한 고려와 측정된 정량적 비교 데이터를 기반으로 앱의 특징에 따라 설계시 발생할 수 있는 문제들을 도출하고 이러한 문제들을 효과적으로 해결하기 위한 지침을 제시한다. 이러한 지침들을 바탕으로 고품질의 앱을 설계하기 위한 효과적인 기법을 제안한다.

2. 관련연구

Thompson[2]의 연구에서는 배터리 소모량의 최적화를 위한 Model-Driven Engineering(MDE) 도구를 제시하였다. 제시된 MDE 도구는 앱 아키텍처 모델 제안과 모바일 디바이스의 배터리 소모를 측정하는 에뮬레이션 코드 생성 기능을 제공한다. 이 연구에서는 배터리를 제외한 다른 모바일 디바이스 역량을 고려하지 않는다.

Lin[3]의 연구에서는 배터리 소모량이 많은 GPS 센서 대신에 WI-FI, 삼각측량법을 이용한 a-LOC 기법을 제안하고 a-LOC 의 높은 정확성과 효율적인 배터리 관리 능력을 검증하기 위해 다양한 모바일 디바이스를 이용하여 실험하였다. 이 연구에서는 모바일 디바이스의 배터리 역량만을 고려하고 있으며 위치기반 앱에 한정된 기법을 제시하고 있다.

Zhang[4]의 연구에서는 모바일 디바이스의 제약을 극복하기 위해 클라우드 자원을 효과적으로 사용하기 위한 프로그래밍 모델과 아키텍처를 제안하고 관련된 미들웨어들을 설계하였다. 이 연구에서는 모바일 디바이스의 역량을 정량적으로 분석하고 있지 않으며 클라우드 컴퓨팅을 통한 앱 설계 기법만을 제시하고 있다.

3. 모바일 디바이스 역량 분석

본 장에서는 다음과 같은 실험 환경에서 모바일 디바이스와 데스크톱 컴퓨터의 역량 차이를 정량적으로 비교 분석한다.

- 모바일 디바이스: Galaxy Note I, Android

* Corresponding author

4.0.3, 1.5GHz Dual Core, 4G, WI-FI, 1GB RAM, 16GB SD Card

- 데스크톱 컴퓨터: Windows 7, 3.00GHz Dual Core, 100 Mbps LAN, 4GB RAM, 500GB HDD

본 논문에서는 여러 역량들 중 프로세싱, 네트워크, 저장소에 대한 모바일 디바이스의 역량을 분석한다. 배터리, 센서, 디스플레이 등의 다른 역량들은 향후 연구에서 다룬다.

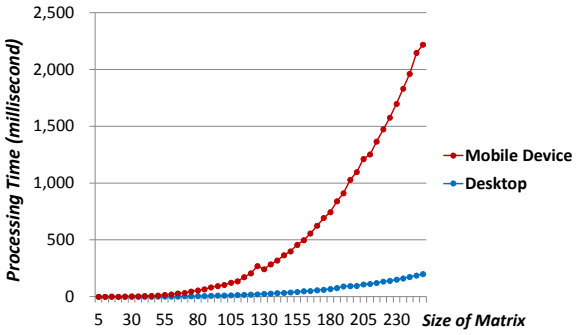
3.1. 프로세싱 역량(Processing Capability)

모바일 디바이스의 프로세싱 역량을 분석하기 위해 모바일 디바이스와 데스크톱 컴퓨터에서 복잡도가 다른 세 가지 알고리즘, 퀵 정렬, 선택 정렬, 행렬 곱셈을 비교 수행하였다.

퀵 정렬의 경우 배열의 크기가 100k 일 때, 모바일 디바이스에서 실행 시간은 53.06ms 가 걸렸고 데스크톱 컴퓨터에서는 5.6ms 가 걸렸다. 배열의 크기가 300k 일 때는 모바일 디바이스에서 208.84ms, 데스크톱 컴퓨터에서는 17.61ms 가 걸렸다.

선택 정렬의 경우 배열의 크기가 1k 일 때 모바일 디바이스에서 실행 시간은 12.28ms, 데스크톱 컴퓨터에서는 1.03ms 로 측정되었다. 배열의 크기가 3k 일 때, 모바일 디바이스에서 117.43ms, 데스크톱 컴퓨터에서는 9.11ms 로 측정되었다.

(그림 1)은 정방행렬 곱셈을 행렬의 크기를 증가시키며 수행하여 도출한 데스크톱 컴퓨터와 모바일 디바이스 사이의 프로세싱 역량을 비교한 결과이다.



(그림 1) 프로세싱 역량 비교 결과

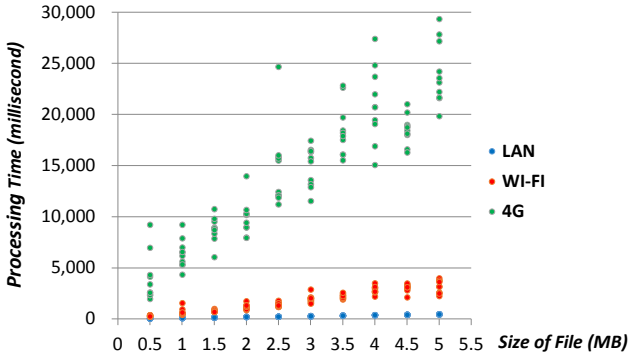
행렬의 크기가 150×150 일 때 모바일 디바이스에서 362.35ms, 데스크톱 컴퓨터에서는 실행 시간이 38.78.ms 로 측정되었다. 행렬의 크기가 250×250 일 때는 모바일 디바이스가 2020.72ms, 데스크톱 컴퓨터가 199.39ms 로 나타났다.

모바일 디바이스와 데스크톱 컴퓨터 사이의 실행 시간 비율은 약 10 에서 12 배로 나타났다.

또한 위 그림과 같이 알고리즘의 복잡도가 높은 프로그램에서는 데이터의 크기가 커질수록 데스크톱 컴퓨터와 모바일 디바이스에서의 실행 시간의 차이가 더욱 커짐을 알 수 있다.

3.2. 네트워크 역량(Network Capability)

모바일 디바이스와 데스크톱 컴퓨터간의 네트워크 역량 비교를 위해 같은 크기의 파일 다운로드 시간을 측정한다. 아래 (그림 2)는 모바일 디바이스와 데스크톱 컴퓨터의 네트워크 역량 비교를 나타낸다.



(그림 2) 네트워크 역량 비교 결과

1MB 파일 다운로드 시 LAN, WI-FI, 4G 를 이용했을 때 평균적으로 다운로드 시간이 87.5ms, 663.3ms, 6376ms 가 걸렸다. 2MB 파일 다운로드 시 LAN 은 178.5ms, WI-FI 는 1179.6ms, 4G 는 9849.6ms 가 걸렸다. 3MB 파일을 다운로드 할 때 LAN 은 267.7ms, WI-FI 는 1955ms, 4G 는 14897.2ms 가 걸렸다.

데스크톱 컴퓨터의 유선 랜이 WI-FI 보다 평균적으로 약 7 배 빠르게 측정되었다. WI-FI 는 4G 보다 약 8 배 빠르게 나타났다. 또한, 위 그림에서 볼 수 있듯이 파일 크기가 커져도 다운로드 시간에 대한 차이의 비율이 일정하게 측정되었다.

3.3. 저장소 역량(Storage Capacity)

저장소 역량에는 주 저장소 역량과 보조 저장소 역량이 있다. 주 저장소 역량은 어플리케이션을 실행시키기 위한 주 기억 장치의 크기를 의미하며 보조 저장소 역량은 어플리케이션 및 데이터를 저장하기 위한 보조 기억 장치의 크기를 의미한다.

일반적으로 모바일 디바이스의 주 저장소 역량은 데스크톱 컴퓨터에 비해 매우 한정적이다. 본 논문의 실험 환경의 경우 데스크톱 컴퓨터의 메모리 크기는 4GB 이고 모바일 디바이스의 메모리 크기는 1GB 이다. 데스크톱 컴퓨터의 어플리케이션의 경우 일반적으로 많은 메모리 역량을 요구한다. 일부 데스크톱 컴퓨터 어플리케이션은 1GB 이상의 메모리를 요구하며 이는 모바일 디바이스의 메모리 역량을 초과한다. 모바일 디바이스의 한정적인 메모리 역량에 의해 일반적으로 모바일 어플리케이션의 경우 50MB 미만의 메모리를 사용한다. 모바일 디바이스의 보조 저장소 역량 역시 데스크톱 컴퓨터에 비해 매우 한정적이다. 본 논문의 실험 환경의 경우 데스크톱 컴퓨터의 HDD 크기는 500GB 이고 모바일 디바이스의 SD Card 크기는 16GB 이다. 규모가 큰 데스크톱 컴퓨터 소프트웨어는 일반적으로 5~10GB 의 저장 공간을 요구하며 대부분의 앱의 경우 1GB 이하의 저장 공간을 요구한다.

4. 정량적 분석 기반 앱 설계 기법

본 장에서는 3 장의 분석 결과를 기반으로 앱 설계

시 발생 할 수 있는 문제들을 도출하고 이들을 효과적으로 해결하기 위한 지침을 제시한다.

4.1. 프로세싱 역량을 고려한 앱 설계 지침

모바일 디바이스와 데스크톱 컴퓨터의 프로세싱 역량 차이는 약 10 배에서 12 배로, 모바일 디바이스의 프로세싱 역량은 데스크톱 컴퓨터에 비해 제한적이다. 이러한 역량의 차이는 복잡도가 높은 소프트웨어일수록 체감되는 성능 차이의 정도가 더 커진다. 즉, 복잡도가 낮은 처리를 요구하는 소프트웨어의 경우, 이러한 성능의 차이는 크게 고려될 필요가 없다. 그러나 통계 처리, 이미지 프로세싱 등의 높은 복잡도의 알고리즘으로 많은 양의 데이터를 처리하는 소프트웨어의 경우, 모바일 디바이스와 데스크톱 컴퓨터에서의 처리 시간이 크게 차이가 난다. 즉, 이러한 복잡도가 높은 소프트웨어는 모바일 디바이스에서 처리하기에 적합하지 않다. 그러므로 이러한 문제를 해결하기 위한 다음과 같은 설계 지침을 제시한다.

지침 1. 서버의 높은 프로세싱 역량 활용: 복잡도가 높은 소프트웨어의 경우, 클라이언트-서버 아키텍처 스타일이 적용 가능하다. 모바일 디바이스에서의 기능 수행 시간을 EXC_TIME_{mobile}, 서버에서의 기능 수행 시간을 EXC_TIME_{server} 이라고 할 때, 다음 식에서 일반적으로 n 은 10 에서 12 의 값을 가진다.

$$EXC\_TIME_{mobile} = n \times EXC\_TIME_{server}$$

그러므로 앱 설계시 복잡도가 높은 기능의 경우 서버에서 처리하고 그 결과를 모바일 디바이스에서 활용하도록 설계하는 것이 좋다. 이 경우, 네트워크 오버헤드가 발생하며, 이로 인해 발생하는 시간이 NET_TIME 이라 할 때, 전체 처리 시간, EXC_TIME_{total} 은 다음과 같다.

$$EXC\_TIME_{total} = EXC\_TIME_{server} + NET\_TIME$$

즉, $NET\_TIME < EXC\_TIME_{mobile} - EXC\_TIME_{server}$ 인 경우 클라이언트-서버 아키텍처가 사용으로 성능을 향상시킬 수 있으며 이는 일반적으로 유효하다.

그러나 NET_TIME 은 서버가 가용하지 않거나 네트워크 상태가 좋지 못할 경우 비정상적으로 증가할 수 있다. 이 경우 앱의 성능이 저하되며, 이 지침이 유효하지 않게 된다.

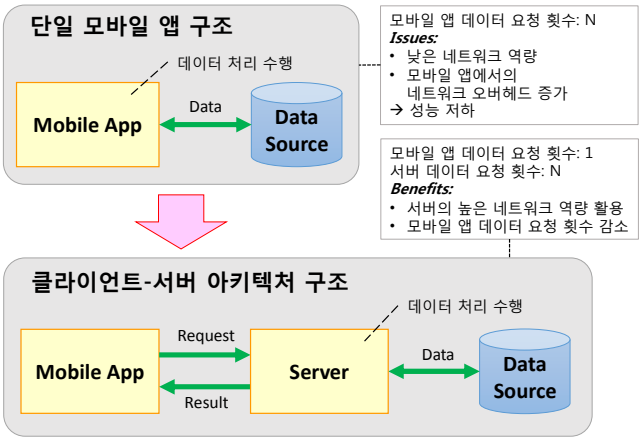
지침 2. 서버의 가용성 및 안정성 최대화: 서버의 가용성 및 안정성을 증가시키기 위해 아마존 EC2 와 같은 클라우드 서버를 사용하는 것이 효과적이다. 클라우드 서버를 사용할 경우 서버 유지 비용을 크게 줄일 수 있으며 서버의 확장 또한 용이하여 더욱 안정적인 서버 운영이 가능하다.

네트워크 상태가 안정적이지 못할 경우 네트워크 통신 횟수를 줄이는 것이 중요하다. 이를 위한 지침은 네트워크 역량을 고려한 지침과 관련이 있으므로 4.2 절에서 자세히 다룬다.

4.2. 네트워크 역량을 고려한 앱 설계 지침

데스크톱 컴퓨터의 유선 랜의 속도는 모바일 디바이스의 WI-FI 속도에 비해 약 7 배 빠르고, 모바일 디바이스의 WI-FI 속도는 4G 에 비해 약 8 배 빠르다. 즉, 모바일 디바이스의 네트워크 역량은 데스크톱 컴퓨터의 네트워크 역량에 비해 한정적이며, 모바일 디바이스의 네트워크 종류에 따라서도 역량의 차이가 존재한다. 따라서 높은 네트워크 역량을 요구하는 소프트웨어를 모바일 디바이스에서 구동하는 것은 효과적이지 않을 수 있다. 본 논문에서는 이러한 높은 네트워크 역량을 요구하는 소프트웨어의 경우 다음과 같은 설계 지침을 제시한다.

지침 1. 서버의 높은 네트워크 역량 활용: 네트워크 역량을 많이 요구하는 앱의 경우, 서버의 높은 네트워크 역량을 이용하여 앱의 성능을 높일 수 있다. 즉, (그림 3)과 같이 서버에서 데이터 요청 및 처리를 하고 그 처리 결과를 모바일 디바이스에 제공함으로써 모바일 디바이스의 네트워크 통신 횟수를 크게 줄이고 서버의 높은 네트워크 역량을 활용 할 수 있다.



(그림 3) 네트워크 역량 개선을 위한 설계

지침 2. 네트워크 통신 횟수 최소화: 접근 빈도가 높은 데이터를 미리 예측 가능한 경우, 그 데이터를 미리 모바일 디바이스에 저장하고 이를 사용함으로써, 네트워크 통신 횟수를 줄여 앱 품질을 높일 수 있다. 이 방법은 모바일 디바이스의 데이터 저장 공간의 확보를 요구하므로 앱 설계자는 모바일 디바이스의 저장소 역량을 고려하여 모바일 디바이스가 수용할 수 있을 양의 데이터를 미리 저장하도록 LRU 와 같은 기법을 사용하여 설계하는 것이 좋다. 이 방법은 초기의 높은 네트워크 오버헤드를 감수함으로써 전반적인 소프트웨어의 네트워크 오버헤드를 크게 줄인다.

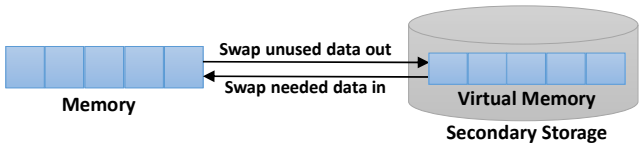
지침 3. 높은 역량의 무선 네트워크 활용: 서버가 가용하지 않은 경우, 모바일 디바이스의 네트워크 중 더 높은 역량을 지니는 WI-FI 를 최대한 활용하는 것이 좋다. 데이터가 즉시에 필요하지 않을 경우, WI-FI 가 가용하게 되었을 때 통신을 수행할 수 있다. 이 방법은 외부 데이터베이스와의 동기화와 같은 통신을 즉시에 수행하지 않아도 되는 기능에 효과적이다.

4.3. 저장소 역량을 고려한 앱 설계 지침

메모리 관점에서 저장소 역량을 분석하였을 때 모

바일 디바이스의 주 저장소 역량은 데스크톱 컴퓨터보다 제한적이다. 또한, 모바일 디바이스의 종류에 따라 한 개의 앱이 사용할 수 있는 메모리 크기는 16~64MB 까지로 제한된다. 즉, 모바일 디바이스의 주 저장소 역량 이상의 메모리 크기를 요구하는 앱은 실행될 수 없다. 그러므로 이러한 문제를 효과적으로 해결하기 위한 다음과 같은 설계 지침을 제안한다.

지침 1. 보조 저장소 역량 활용: 메모리 용량을 많이 요구하는 앱 설계시, 보조 저장소를 최대한 이용하는 것이 좋다. 즉, (그림 4)와 같이 보조 저장소의 일부를 가상 메모리 공간으로 사용하여 당장 사용하지 않는 데이터를 Swap-Out 하고, 필요할 경우에 다시 Swap-In 하는 방식의 데이터 처리가 효과적이다.



(그림 4) 메모리 역량 개선을 위한 지침

그러나 Swap 횟수가 많아지게 되면 Swap 오버헤드가 증가하여 앱 성능을 저하 시킬 수 있으므로 앱 설계자는 Swap 알고리즘을 효율적으로 작성해야 한다.

지침 2. 서버의 높은 메모리 역량 활용: 서버의 메모리 역량을 활용하여 앱의 성능을 높일 수 있다. 서버는 모바일 디바이스에 비해 높은 메모리 역량을 지니고 있으므로, 모바일 디바이스의 메모리 역량을 넘어서는 앱의 기능은 서버에 구동하는 것이 효과적이다. 그러나 데이터 요청이 많은 앱의 경우 서버와의 통신 횟수 증가에 따른 네트워크 오버헤드로 인해 앱의 품질이 저하될 수 있다. 따라서 이 지침은 4.2 절에서 제안한 네트워크 역량을 고려한 지침과 함께 고려하는 것이 좋다.

보조 저장소 관점에서 저장소 역량을 분석하였을 때 역시 모바일 디바이스의 역량은 데스크톱 컴퓨터에 비해 매우 한정적이다.

지침 3. 서버의 높은 보조 저장소 역량 활용: 설치 및 운영 시 높은 보조 저장소 역량을 요구하는 앱은 모바일 디바이스에서 실행시키기에 한계가 있다. 모바일 디바이스의 보조 저장소의 크기가 일반적으로 16GB 인 것을 감안하면, 비교적 규모가 큰 데스크톱 컴퓨터 소프트웨어가 요구하는 저장 공간인 5~10GB 은 모바일 디바이스의 보조 저장소 역량으로는 수용하기 어렵다. 이러한 대규모 소프트웨어의 기능을 앱이 필요로 하는 경우, 그 기능을 서버에서 실행 시키고, 그 결과를 모바일 디바이스에서 이용하는 방식이 효과적이다. 앱이 모바일 디바이스의 보조 저장소 공간의 역량을 넘어서는 많은 양의 데이터 관리를 요구하는 경우, 외부 데이터베이스를 활용 하는 것이 효과적이다.

이와 같이 모바일 디바이스의 보조 저장소 역량을 넘어서는 앱 설계에서는 서버의 보조 저장소 역량을 이용해야 하기 때문에, 네트워크 오버헤드가 높아지는 문제가 발생 할 수 있다. 그러므로 이러한 지침들 역시 4.2 절에서 제안한 네트워크 역량을 고려한 지침을 함께 고려하는 것이 좋다.

5. 결론 및 향후 연구

모바일 디바이스는 데스크톱 컴퓨터에 비해 일반적으로 낮은 컴퓨팅 역량을 지니며, 이는 고품질 앱 설계시 여러 문제들을 발생시킨다. 기존 연구의 경우 모바일 디바이스 역량의 정량적 분석과 효과적인 앱 설계 지침 제시가 부족하다. 그러므로 본 논문에서는 프로세싱, 네트워크, 저장소를 관점으로 고품질 앱 설계시 발생할 수 있는 문제를 효과적으로 분석 및 대처하기 위한 설계 지침을 제시하였다. 각 지침이 효과적으로 적용될 수 있는 상황과, 지침을 적용했을 시 얻을 수 있는 장점 및 주의해야 하는 사항에 대해 상세히 기술하였다. 앱 설계자는 설계하고자 하는 앱의 특성에 따라 본 논문에서 제시한 지침들을 적절히 적용하여 앱을 고품질로 설계할 수 있다.

모바일 디바이스의 역량에는 본 논문에서 고려한 프로세싱 역량, 네트워크 역량, 저장소 역량 이외에도 배터리 역량, 센서 역량, 화면 역량 등이 존재하며, 이들은 향후 연구에서 다룬다.

Acknowledgement

본 연구는 중소기업청에서 지원하는 2012 년도 산학연협력 기업부설연구소 지원사업(No. C0016709)의 연구수행으로 인한 결과물임을 밝힙니다.

참고문헌

[1] Qi, Han and Gani, Abdullah, “Research on mobile cloud computing: Review, trend and perspectives,” *In Proceedings of Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, pp. 195-202, 16-18 May 2012.

[2] Thompson C., White J., Dougherty B., and C. Schmidt D., "Optimizing Mobile Application Performance with Model–Driven Engineering," *Software Technologies for Embedded and Ubiquitous Systems Lecture Notes in Computer Science*, Vol. 5860, pp.36-46, November 2009.

[3] Lin K., Kansal A., Lymberopoulos D., and Zhao F., "Energy-accuracy trade-off for continuous mobile device location," *In Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys 2010)*, pp. 285-298, 2010.

[4] Zhang X., Kunjithapatham A., Jeong S., and Gibbs S., "Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing," *Mobile Networks and Applications*, Vol. 16, No. 3, pp. 270-284, June 2011.

[5] Mascolo, Cecilia, “The Power of Mobile Computing in a Social Era,” *Journal of Internet Computing, IEEE*, Vol. 14, No. 6, pp. 76-79, 2010.