타이젠 웨어러블 플랫폼의 특성을 이용한 효율적인 어플리케이션 관리

함동읍*

*삼성전자

The Effective Application Management Using Characteristics of Tizen Wearable Platform

Dong-eup Ham*

*Samsung Electronics

E-mail : dongeup.ham@samsung.com

요 약

대부분의 웨어러블 플랫폼(예, 삼성 기어, 안드로이드 웨어)의 기능은 모바일 플랫폼의 기능을 수정 없이 사용하고 있다. 그러나 웨어러블 디바이스는 웨어러블 형태, 작은 배터리, 스크린 모양, 열악한 네트워크 성능, 상대적으로 짧지만 자주 사용하는 사용자 인터페이스 등의 이유로 독특한 특성을 가지고 있다. 일반적으로 웨어러블 디바이스는 모바일 디바이스와 연결을 위해 어플리케이션 또는 서비스 등을 가지고 있는데, 이는 디바이스 모델 이름, 네트워크 능력(3G, LTE, Wi-Fi 등), 제조사, 지원하는 언어 리스트 등의 정보 교환하는 프로토콜을 사용하고 있다. 즉, 웨어러블 디바이스는 연결된 단말(예, 폰, 태블릿 등)에 깊이 의존하고 있고, 따라서 웨어러블 플랫폼은 이를 고려해야만 한다. 본 논문에서는 사용자 편의성과 웨어러블 플랫폼의 성능 하락(Sluggish) 문제를 해결하기 위해서 웨어러블 플랫폼의 특성을 고려한 효율적인 어플리케이션 관리 방안을 제시한다.

ABSTRACT

Most of wearable platforms(i. e. Samsung Gear, Android Wear) are using most of Tizen mobile platform features without any changes. However wearable devices have unique characteristics due to wearable type, small battery, screen shape, poor network and relative short but frequent user interfaces. In general, a wearable device has a process to be paired with the mobile device, which includes capability exchange that includes information such as device model name, network capability (3G, LTE, Wi-Fi and so on), manufacturer and supported languages. In other word, a wearable device depends heavily on the companion device (i. e. phone, tablet), so wearable platform should consider this. In this paper, we provide the effective application management mechanism using these characteristics of wearable platform to enhance user experience and to reduce sluggish of wearable platform.

키워드

웨어러블(Wearable), 타이젠(Tizen), 플랫폼, 최적화, 성능, 어플리케이션 실행

Ⅰ. 서 론

웨어러블 디바이스는 웨어러블 형태, 작은 배터리, 스크린 모양, 열악한 네트워크 성능, 상대적으로 짧지만 자주 사용하는 사용자 인터페이스 등의 이유로 독특한 특성을 가지고 있다. 하지만 대부분의 웨어러블 플랫폼은 이러한 특성을 이용하지 않고 기존 모바일 플랫폼의 기술을 큰 변경 없이 차용하고 있다.

웨어러블 디바이스는 연결된 단말(예, 폰, 태블릿

등)의 기능에 깊이 의존하고 있고, 이러한 특성은 웨어러블 플랫폼 자체에 반영되어야 한다. 웨어러블 기기의 특성을 고려하여 효율적인 어플리케이션 관리 방안을 제시하는 것이 본 논문의 목적이다.

먼저, 추론 프레임워크(Inference Framework)를 이용하여 여러 환경 하에서 동작 가능한 어플리케이션 실행 방안을 제시한다. 다음으로 연결 초기화 시점에 어플리케이션 초기화를 위해 필요한 데이터를 전송함으로써 성능과 반응성을 높인다.

웨어러블 플랫폼은 자체 기기의 리소스 제약 때문에 일반적으로 모바일 단말(Phone, Tablet)과 연결되어 수행된다. 웨어러블 기기 중에는 BT(블루투스), Wi-Fi 네트워크만 지원하고 3G, LTE 등은 지원하지 못하는 기기가 있다. 이런 기기는 모바일 디바이스의 네트워크를 사용하여 인터넷에 접근한다.

이러한 웨어러블 플랫폼 특성 때문에 모바일 단말과의 정보 교환(Capability Exchange)은 필수적이고, 이 과정에서 많은 데이터를 교환하기 때문에 시간이 상당히 걸린다(약 15초 이상). 그림 1은 웨어러블 단말과 모바일 단말과의 정보 교환을 보여준다.

그림 1. 단말 간 정보 교환 과정

Ⅱ. 본 론

그림 2는 어플리케이션의 상태 및 상황을 파악하고 그에 상응하여 동작하는 추론 프레임워크(Inference Framework)를 보여준다.

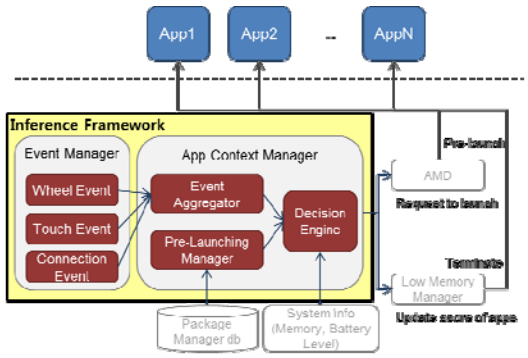

그림 2. 추론 프레임워크

추론 프레임워크는 어플리케이션의 상황을 판단하는 어플리케이션 컨텍스트 매니저(Context Manager), 기기에서의 이벤트를 관리하기 위한 이벤트 매니저(Event Manager)로 이루어져 있다. 또한 어플리케이션 실행을 위한 어플리케이션 관리 데몬, 메모리 관리를 위한 메모리 매니저와 통신을 한다.

● 추론 프레임워크(Inference Framework)

어플리케이션의 상태 정보를 수집, 판단하여 시스템 모듈들에게 동작을 요청한다. 이벤트 매니저(Event Manager)는 터치, 휠 등의 이벤트를 받아서 모으는 역할을 담당한다. 어플리케이션 컨텍스트(App Context Manager)는 이벤트를 전달 받아 판단을 하는 역할을 하며, 어플리케이션의 사전 실행(Pre-Launching)을 위한 모듈을 가지고 있다. 또한 결정 엔진(Decision Engine)을 통해 학습된 사용자 패턴에 의해 어플리케이션을 제어한다.

● 어플리케이션 관리 데몬(Application Manager Daemon, AMD)

어플리케이션을 수행하고, 종료시키는 역할을 담당한다.

● 리소스 관리 데몬(Resourced)

메모리 등 기기에서 사용하는 리소스를 관리하는 역할을 담당한다.

● 웨어러블 연결 관리 서비스(Wearable Manager Service, WMS)

웨어러블의 서비스 어플리케이션으로 연결된 단말의 기어 매니저(Gear Manager)와 통신을 담당하고, 정보 교환(Capability Exchange)을 통해서 기기 간 상태를 동기화 한다.

● 메모리 관리자(Low Memory Manager, LMM)

시스템 메모리 부족 시 동작하는 모듈로서 사용하지 않는 어플리케이션을 정리하여 메모리를 확보한다.

Ⅲ. 어플리케이션의 실행 예측

추론 프레임워크는 다양한 사용자 패턴을 파악하여 추론의 판단 근거로 활용한다. 정확한 추론을 위해서는 학습 패턴을 만들기 위한 데이터가 필요하다.

그림 3에서는 클라우드(Cloud) 서버와 통신하여 일반적인 사용자 패턴을 받아와서 추론 엔진을 학습시킨다. 클라우드에는 어플리케이션 별 실행 빈도에 대한 정보와 일주일 당 어플리케이션 실행 횟수를 이미 가지고 있다. 이 정보를 이용하여 추론 프레임워크의 학습 데이터로 사용하고, 이를 통해 초기 추론 모델(Model)을 생성한다.

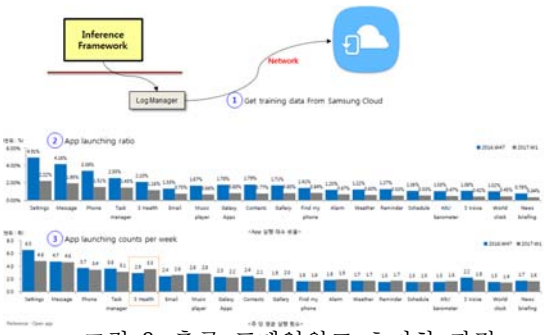

그림 3. 추론 프레임워크 초기화 과정

그림 4는 사용자 패턴이 어떻게 어플리케이션 실행 확률에 영향을 주는지 보여준다. 추론 프레임워크에서는 이러한 이벤트 정보를 이용하여 초기 생성된 모델을 각 개인에게 맞도록 개인화 한다.

삼성 기어의 경우 홈 화면에서 사용자가 시계 방향(Clockwise)으로 돌리면 위젯(Widget)이 표시된다. 위젯에 일정 시간 이상 머물러 있는 경우에는 해당 위젯을 눌러 연관된 어플리케이션을 실행할 확률이 높아지기 때문에 이 정보를 결정 엔진(Decision Engine)에게 전달한다. 또한 사용자가 반 시계 방향(Counter Clockwise)으로 돌리면 알림(Notification)이 표시된다. 이때에는 알림과 관련이 있는 어플리케이션이 실행될 확률이 높아지기 때문에 해당 정보를 결정 엔진에게 전달한다. 결정 엔진은 이와 같은 사용자 입력 정보를 이용하여 어플리케이션의 실행 확률을 조정한다.

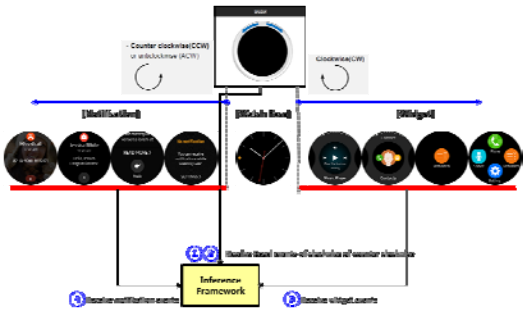

그림 4. 휠 동작에 따른 어플리케이션 실행 확률 변화

그림 5는 결정 엔진(Decision Engine)이 학습된 데이터를 기반으로 예측하여 대상 어플리케이션 프로세스를 미리 실행해 놓는 과정을 보여준다. 이렇게 미리 프로세스가 실행되어 있으면, 실제 사용자에 의해 어플리케이션이 실행되면 상대적으로 빠른 시간에 어플리케이션이 동작하기 때문에 사용자가 반응성이 향상된 것으로 체감할 수 있다.

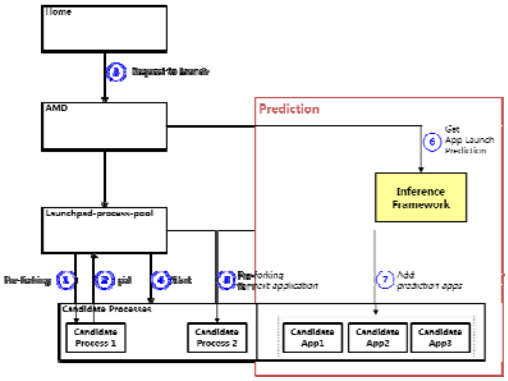

그림 5. 추론 프레임워크에 의한 실행 어플리케이션 예측

표 1은 삼성 기어 S3에서 측정한 어플리케이션 최초 실행 시간과 재실행 시간을 비교한 것이다. 최초 실행의 평균값은 1.28초이고, 재실행의 평균값은 0.77초이다. 실행될 어플리케이션을 예측하여 띄워 놓는 방식은 재실행 로직으로 실행되기 때문에 평균적으로 0.51초(1.28초 - 0.77초)만큼 빨리 실행될 수 있기 때문에 사용자 반응성이 개선될 수 있다. 이 부분을 적절히 사용하여야 사용자 입장에서 어플리케이션이 빠르게 반응한다고 생각할 수 있다.

표1. 어플리케이션 최초 실행 vs 재실행

	First Launch	ReLaunch
Average(s)	1.28	0.77
Call log	1.50	0.76
Clock (Alarm)	1.03	0.61
S Planner	1.17	0.65
Gallery (Albums)	1.49	0.75
Setting	1.22	0.82
Menu	0.65	0.60
Music Player(Google)	1.40	0.72
Message	1.63	1.10
Voice Recorder	1.18	0.64
Contacts entry	1.49	0.75
Recent entry	1.30	1.09

Ⅳ. 빠른 어플리케이션 실행

일반적으로 어플리케이션 실행이라 함은 어플리케이션의 초기 화면이 보일 때까지 측정한다. 하지만 사용자가 실제로 느끼는 시간은 어플리케이션이 동작하고, 필요한 아이콘 및 정보가 모두 표시되는 것까지이다. 웨어러블의 네트워크 어플리케이션이 인터넷으로부터 정보를 가지고 오기 위해서 모바일 네트워크를 일반적으로 사용하는데, 이는 모바일의 Wi-Fi 또는 LTE를 사용하여 데이터를 다운로드 한 다음 이를 BT를 통해 다시 웨어러블로 전송해야 한다. 이 때 시간이 많이 걸리

기 때문에 실제로 데이터 로딩이 완료된 화면을 보기 위해서는 사용자는 로딩 화면을 보면서 기다려야 한다. 표 2는 웨어러블 기기가 모바일 기기의 웹 프락시(Web Proxy) 기능을 사용하였을 때 네트워크 성능을 보여준다. 표에서 볼 수 있듯이 일반 모바일 단말의 네트워크 성능을 고려했을 때 매우 열악한 상황임을 알 수 있다.

표2. 삼성 기어 S2의 네트워크 성능

Model	SM-R720
Gear Binary	StmR720XX_20151017.006
iOS GearManager	Host Manager(IOS) #1.0.15101601
Android GearManager	UnifiedHostManager 10/15 #003

Test URL: [illegible]
Size: 90,060 byte

Android Web proxy performance

#1	1.30 sec	67.76 Kbytes/sec
#2	1.60 sec	55.11 Kbytes/sec
#3	1.43 sec	61.72 Kbytes/sec
#4	1.16 sec	75.83 Kbytes/sec
#5	1.61 sec	54.73 Kbytes/sec
AVG	1.42 sec	63.03 Kbytes/sec

iOS Web proxy performance

#1	8.96 sec	9.81 Kbytes/sec
#2	10.31 sec	8.53 Kbytes/sec
#3	7.07 sec	12.44 Kbytes/sec
#4	6.42 sec	13.71 Kbytes/sec
#5	8.00 sec	10.99 Kbytes/sec
AVG	8.15 sec	11.10 Kbytes/sec

그림 1에서의 단말 간 정보 교환 과정 동안 네트워크 어플리케이션에게 Pre-Launching 이벤트를 보낼 수 있다.
그림 6은 해당 로직이 적용된 후 실제 실험 결과를 보여준다. 일반(Normal) 실행의 경우 어플리케이션 실행 시작부터 모든 Contents가 로딩 되는데 시간이 약 30초가 걸리는 것을 보여준다. 이는 어플리케이션 진입 후 모든 Contents가 보여 질 때까지 사용자가 30초를 기다려야 한다는 것을 의미한다.
Pre-Launching의 경우 웨어러블 디바이스와 모바일 디바이스 간 연결 시간 동안 Pre-Launching 이벤트를 해당 어플리케이션에게 전달하여 화면에 보이지 않고 백 그라운드(Background)로 네트워크를 사용하여 Contents를 다운로드 받는 것을 보여준다. 다운로드 받는 시간은 동일하게 30초가 걸리지만 이는 단말 간 연결시간에 포함되기 때문에 사용자는 이 시간을 체감할 수 없다. 따라서 특정 시간이 지난 후(x초) 어플리케이션을 실행하면, 기본 실행 시간인 약 1초만 걸리고 모든 Contents가 로딩 된 화면을 x+1초에 사용자가 볼 수 있음을 의미한다.
추가적으로 네트워크를 사용하는 어플리케이션을 판단하기 위해서는 어플리케이션의 Manifest 파일에 메타데이터(Meta data)로 기술하여 정보를 추출하는 방식이 있다. 하지만 기술하지 않는 어플리케이션이라 하더라도 어플리케이션 실행 시 네트워크를 집중적으로 사용한다고 추론 엔진이 파악하여 Pre-Launching 이벤트를 보내 주는 것도 가능하다.

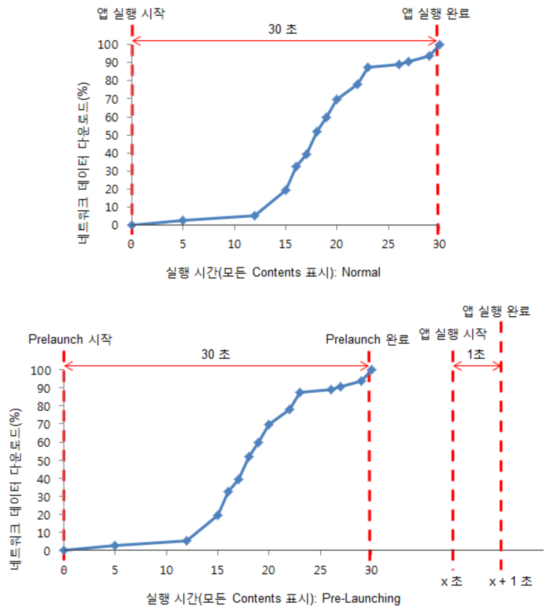

그림 6. 어플리케이션 실행 시간에 대한 결과

V. 결 론

본 논문에서는 웨어러블 플랫폼의 특성에 초점을 두고, 타이젠 웨어러블 플랫폼을 개선하여 사용자에게 최적화된 반응성을 제공하기 위한 방법을 제시한다.

참고문헌

[1] N. D. Lane, S. Bhattacharya, C. Forlivesi, P. Georgiev, L. Jiao, L. Qendro, , and F. Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In IPSN 2016.

[2] Liu, Renju, and Felix Xiaozhu Lin. “Understanding the Characteristics of Android Wear OS.“ Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 2016.

[3] YAN, T., CHU, D., GANESAN, D., KANSAL, A., AND LIU, J. Fast app launching for mobile devices using predictive user context. In Proc. ACM Int. Conf. Mobile Systems, Applications, & Services (MobiSys) (New York, NY, USA, 2012), MobiSys ’12, ACM, pp. 113-126.