Bluetooth 의 OPP 를 적용한 Embedded System 의 구현과 성능 향상

이 찬 향, 이 태 진
성균관대학교 정보통신공학부
e-mail hyang3842@ece.skku.ac.kr, tjlee@ece.skku.ac.kr

s

Performance Improvement and Implementation of Embedded System with Bluetooth OPP

Chan-Hyang Lee, Tae-Jin Lee
School of Information and Communication Engineering
Sungkyunkwan University

요 약

Wireless LAN 외에도 무선 전송 기술은 종류는 다양하지만 비교적 인지도 및 사용성이 낮아 평범한 사용자들에게 친숙하지 않았고 따라서 호환성과 친밀감이 다소 결여된 면이 있다. 그러나 저전력을 소모해야 하는 일반적인 Embedded System 에 적용되는 무선 전송 기술로서 Wireless LAN 보다는 Bluetooth 와 같은 무선 통신 규약이 적용될 필요가 있다. 이에 대하여 Bluetooth 의 OPP(Object Push Profile)는 다른 Bluetooth Profile 에 비하여 가볍고 또한 사용이 어렵지 않으므로, 빠르게 진보하고 있는 무선 전송 기술의 적용 사례 중 하나로서 Bluetooth 의 OPP 를 적용한 Embedded System 을 구현하고, Bluetooth USB Dongle 와 Embedded System 간의 다리 역할을 하는 HCI(Host Controller Interface) layer 에서의 속도의 제약을 가져오는 요인을 개선하는 새로운 알고리즘을 제시하고자 한다.

1. 서론

Embedded System 에는 Wireless LAN 보다 사용이 편리하고 시스템의 전력 소모에 유리한 Bluetooth 와 같은 무선 통신 규약이 필요하다. 현재 Bluetooth 가 적용된 대표적인 장치이며 대중화된 장치로서 그 중 휴대폰을 들 수 있겠다. 또한 전세계적으로 Camera 가 장착된 휴대폰은 전체 휴대폰 중 시장점유율이 2006 년 약 34%에서 2008 년 약 63%를 차지하고 있고 이 비율은 점차 증가할 것이다[1].

그렇다면 시장 점유율이 증가하고 있는 Camera 휴대폰으로 촬영한 Image 를 편리하게 전송하거나 출력하려는 요구 또한 늘어날 것이며 이를 위하여 Bluetooth 가 적용된 Embedded System 의 개발 역시 필요할 것이다. 이에 Bluetooth 의 다양한 Profile 중 Image 를 비롯한 Object 의 전송을 위하여 가장 가볍고 간단한 OPP (Object Push Profile)을 적용하여 Embedded System 을 Bluetooth OPP Server 로 활용하는 방법을 제시하고자 한다[2].

Image 를 촬영할 수 있는 Bluetooth OPP Client 단말인 Camera 휴대폰과 촬영한 Image 를 전송 받을 수 있는 Bluetooth OPP Server 단말인 Embedded System 사이의, Bluetooth 를 이용한 무선 전송 시스템을 구현하는 것이 첫 번째 목적이다. 또한 Embedded System 에 USB interface 로 Bluetooth USB Dongle (RF / Baseband)와 상호 접속된 부분인 HCI (Host Controller Interface) layer 에서 발생할 수 있는 전송 효율의 저하를 방지하기 위해서, Bluetooth USB Dongle 로 전송 받은 Bluetooth Data 자체의 속도를 잃지 않기 위한 HCI layer 의 알고리즘의 실험적 최적화와 HCI layer 의 데이터 수신 알고리즘을 개선하여 전송속도를 개선하는 방법을 제안한다[2].

2. Bluetooth OPP 가 적용된 Embedded System 개요

그림 1 은 Bluetooth 를 이용한 무선 전송 시스템의

Connectivity Features 를 나타낸다.

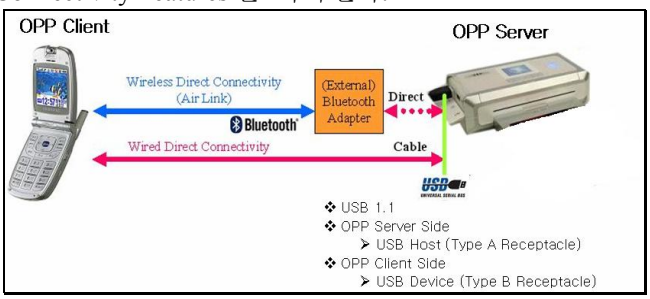

그림 1. Connectivity features

이것은 Bluetooth OPP Client 단말로서 Camera 휴대폰을 예시하고 있고 Object 는 Image 를 직접 촬영하여 얻는 것으로 하였지만, Object Sender 는 Bluetooth OPP Client 기능이 장착된 어떠한 장치이든 상관이 없다.

Bluetooth OPP Server 단말로서 Image 를 전송 받을 수 있는 ARM7 계열의 CPU 및 1M bytes Flash Memory 와 16M Bytes 의 SDRAM 을 가지고 있으며 USB 1.1 Full Speed Host 기능을 지닌 Embedded System 을 예시하였지만, Object Receiver 는 USB Host Interface 기능이 내장되어 있는 장치라면 어떠한 것이든 상관이 없다 [2].

2.1. Embedded System 의 구성

그림 2 는 Bluetooth OPP 를 이용하여 Object 를 전송받기 위한, Embedded System 과 Bluetooth USB Dongle 의 Schematic System for Bluetooth 을 나타낸다[3][4].

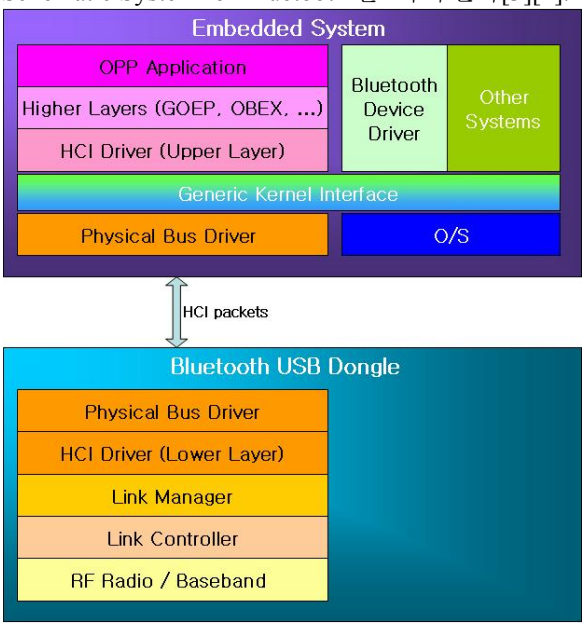

그림 2. Schematic System for Bluetooth

- Bluetooth Device Driver
  - ✓ Bluetooth Device Driver 는 Embedded System 에 장착되는 Bluetooth USB Dongle 의 RF Module 과 Baseband controller 를 구동하고 Bluetooth USB Dongle 과 Embedded System 내에 존재하는 Bluetooth Protocol Stacks 를 초기화하고 제어하는 역할을 한다.
- Mandatory Bluetooth Protocol Stacks
  - ✓ Bluetooth Protocol Stacks 은 다음의 mandatory protocol layers 이 있으며, Bluetooth Core Protocol Specification 과 완전히 호환되는 것이다.
    - ♦ Link Management Protocol (LMP)
    - ♦ Logical Link Control and Adaptation Protocol (L2CAP)
    - ♦ Serial Port Emulation (RFCOMM)
    - ♦ Service Discovery Protocol (SDP)
- Fundamental Bluetooth Protocol Stacks
  - ✓ Bluetooth Protocol Stacks 은 다음의 fundamental protocol layers 이 있으며, OPP 를 이용하기 위하여 필요한 것으로서, Bluetooth Core Protocol Specification 과 완전히 호환되는 것이다.
    - ♦ Generic Access Profile (GAP)
    - ♦ Serial Port Profile (SPP)
    - ♦ Object Exchange (OBEX)
    - ♦ Generic Object Exchange Profile (GOEP)
    - ♦ Object Push Profile (OPP)
- Host Controller Interface (HCI) for USB
  - ✓ Embedded System 와 Bluetooth USB Dongle 상호간의 USB physical bus driver 를 위하여 Embedded System 에는 HCI Upper layer 가 존재한다.
- Bluetooth OPP Server Application
  - ✓ OPP Client 로부터 Object 를 전송받는다.
  - ✓ Embedded System 에 장착된 2x2 LCD 에 전송받은 Image 를 Scaling 하여 Display 한다.

2.2. Embedded System 의 개발 범위

그림 3 은 software 측면에서의 Embedded System 의 Bluetooth 개발 범위를 나타낸다[3][4][5].

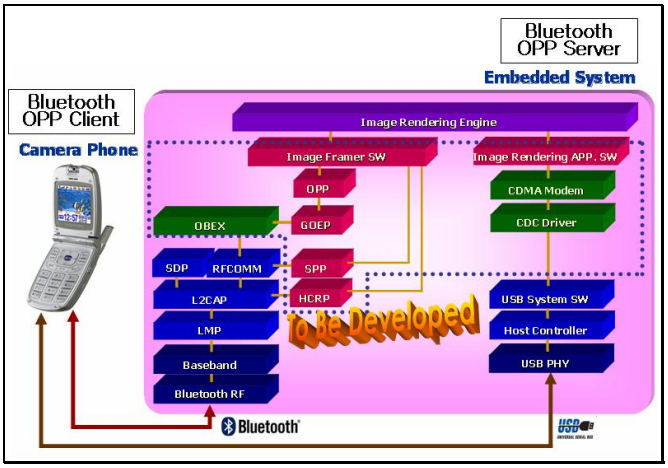

그림 3. Bluetooth 개발 범위

Embedded System 에서 "To Be Developed"라고 명시된 부분이 Bluetooth OPP Client 기능과 Object 를 전송 받은 후 Image 의 경우 Scaling 을 하기 위하여 개발되는 부분이다.

그림 4 는 System Architecture 를 나타낸다[5].

Embedded System 과 USB Interface 로 연결된 Bluetooth USB Dongle 의 RF Module 로부터 전송 받은 데이터는 RF / Baseband 를 시점으로 Link Management

을 거쳐 USB Physical Bus Driver 를 통해 Embedded System 과 연결된 Lower HCI layer for USB 로 데이터를 넘겨준다.

Embedded System 에서는 먼저 USB Host Controller 를 통하여 전송 받은 데이터를 USB Driver 가 넘겨 받고 이것을 다시 Upper HCI layer for USB 로 올려주면, L2CAP 과 Higher Layers 가 데이터를 받아서 OPP Application 까지 전달한다.

OPP Application 은 전달 받은 Object 가 Image 인 경우 LCD 에 출력 가능한 형태로 Image Scaling 을 하거나 일단 메모리에 저장한다[5].

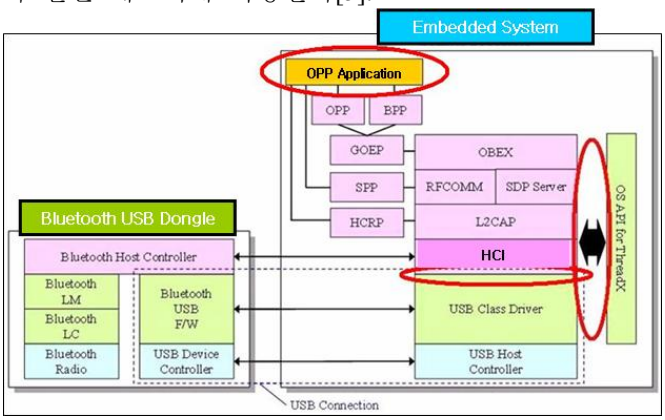

그림 4. System Architecture

3. Embedded System 에서의 Bluetooth 속도 최적화

Bluetooth Protocol Stacks 의 속도에 직접적인 영향을 미치는 부분이 USB Interface 와 dependency 가 있는 알고리즘인 HCI Layer 인데, 이것은 Transfer 부분과 Receiver 부분으로 나뉘어져 있다[3][4][5].

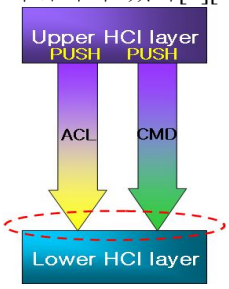

그림 5. 기존의 HCI Transfer 부분

그림 5 는 기존의 HCI Transfer 부분을 나타낸다.

이것은 Upper HCI Layer 에서 데이터를 PUSH 할 때, Lower HCI Layer 에서 USB interface 를 통하여 데이터를 가져가기 전까지, 다음 데이터를 PUSH 할 수 없는 구조이다.

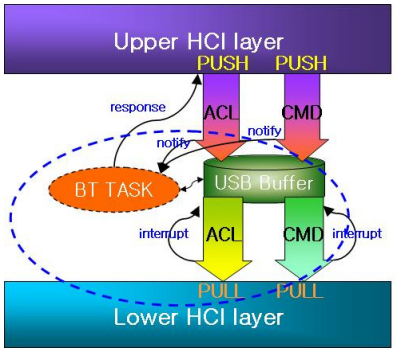

그림 6. 개선한 HCI Transfer 부분

그림 6 은 개선한 HCI Transfer 알고리즘을 나타낸다.

Upper HCI Layer 는 PUSH 한 데이터를 USB Buffer 에 저장한 다음, 이를 Bluetooth Main Task 에 알려준다. Lower HCI Layer 에서는 USB Buffer 에 데이터가 있으면 즉시 USB Interrupt 가 발행되어 데이터를 PULL 할 수 있게끔 되어 있다. Bluetooth Main Task 는 USB Buffer 의 상태를 감지하고 필요 시 Upper HCI Layer 에 알려준다.

다음으로, 기존의 Receiver 부분은 USB interface 를 통하여 데이터가 언제 들어올지 모르므로 계속 Polling 을 하고 있어야 하므로, 별도의 Task 가 필요하다.

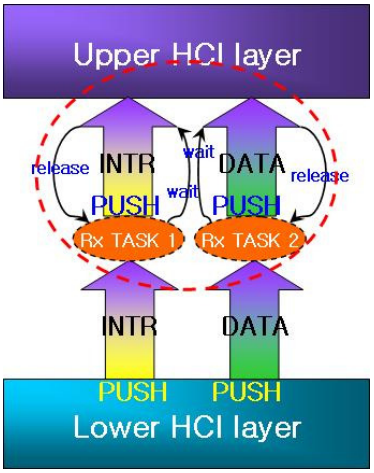

그림 7. 기존의 HCI Receiver 부분

그림 7 은 기존의 HCI Receiver 부분을 나타낸다.

이것은 Lower HCI Layer 에서 USB Interface 를 통하여 데이터를 올려 보낼 때, Polling 하고 있던 Task 는 올라온 데이터를 PUSH 하여 Upper HCI Layer 에 전달한다. 이 Task 는 Upper HCI Layer 로 데이터를 PUSH 하므로 Upper HCI Layer 의 데이터 처리 부분과 연결되어 있어서 데이터 처리가 완료될 때까지 기다려야 한다. 기다리는 동안 Lower HCI Layer 에서 다음 데이터가 들어와도 받을 수가 없는 상태이다.

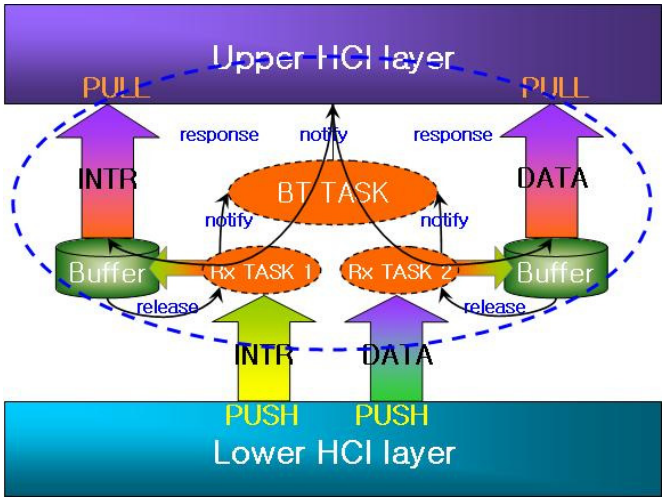

그림 8. 개선한 HCI Receiver 부분

그림 8 은 개선한 HCI Receiver 부분을 나타낸다.

이것은 Lower HCI Layer 에서 USB interface 로 데이터를 PUSH 하면 Polling 하고 있던 Task 는 Buffer 에 데이터를 저장하고 Bluetooth Main Task 에 이것을 알려주고 즉시 다음의 데이터를 받을 준비를 한다. Bluetooth Main Task 는 Buffer 에 데이터가 있음을 알게 되면 Upper HCI Layer 에 데이터가 있음을 알려준다. Upper HCI Layer 는 Buffer 에 쌓여 있는 데이터를 PULL 가져오게 되며, USB Interface 와 무관한 동작을 할 수 있게 된다. Receiver 부분에만 Task 가 존재하는 이유는 USB Interface 는 Interrupt 를 이용하여 데이터를 가져갈 수 있으나, HCI layer 는 Polling 을 하고 있지 않으면 데이터를 받을 수 없는 구조이기 문제가 있기 때문이다.

제안한 알고리즘은 Buffer 로서 USB interface 의 송수신 속도와 Bluetooth Protocol Stacks 에서 처리하는 데이터의 Sync 가 맞지 않는 대부분의 경우에 대하여 실시간으로 데이터를 주고 받을 수 있도록 하여, Hardware 적으로 분리된 Embedded System 과 Bluetooth USB Dongle 과의 최적의 데이터 전송을 구현하는 방법을 제시하고 있다.

4. 개발 System 의 성능평가

그림 9 는 기존 알고리즘과 제안 알고리즘의 전송 소요시간 성능을 분석한 것으로, HCI layer 의 algorithm 개선 효과를 나타내기 위하여, 각각의 Data Size 에 대하여 전송 소요시간을 측정하여 평균값을 취하였다. 여기서 제안한 알고리즘은 Data Size 와 관련 없이 일정한 비율로 전송 소요시간이 단축되었음을 나타내고 있다.

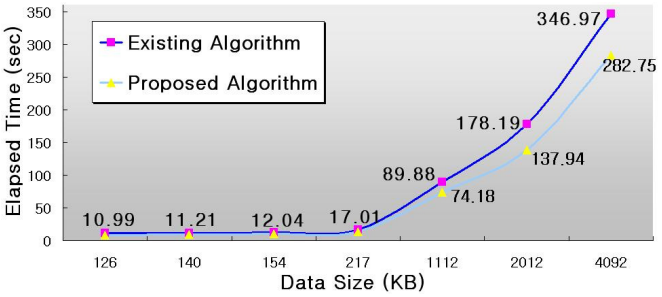

그림 9. 전송 소요시간 비교표

그림 10 에서는 기존 알고리즘과 제안 알고리즘의 전송률 성능을 분석한 것으로, 전송 속도 역시 마찬가지로 기존 대비 평균 약 25%의 속도 개선 효과를 나타내는 것을 확인할 수 있다.

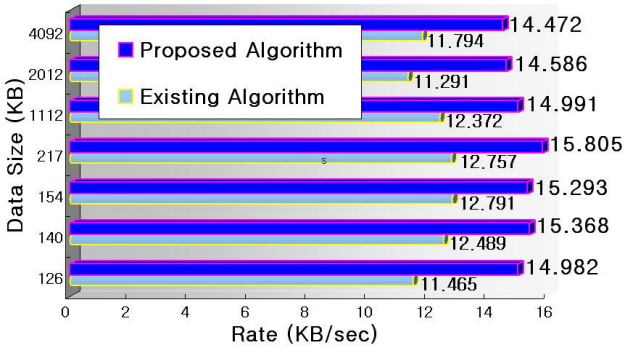

그림 10. 전송 속도 비교표

또한 제안한 알고리즘은 그림 11 에서 USB Packet 으로 이동하는 Encapsulated Bluetooth HCI Packet 에서, 데이터가 전송하는 경우에 구체적으로 아래와 같이 소요시간이 단축됨을 나타내고 있다. 이것은 USB 의 송수신 packet 으로서 HCI Data 가 전송되는 시간을 나타내고 있다.

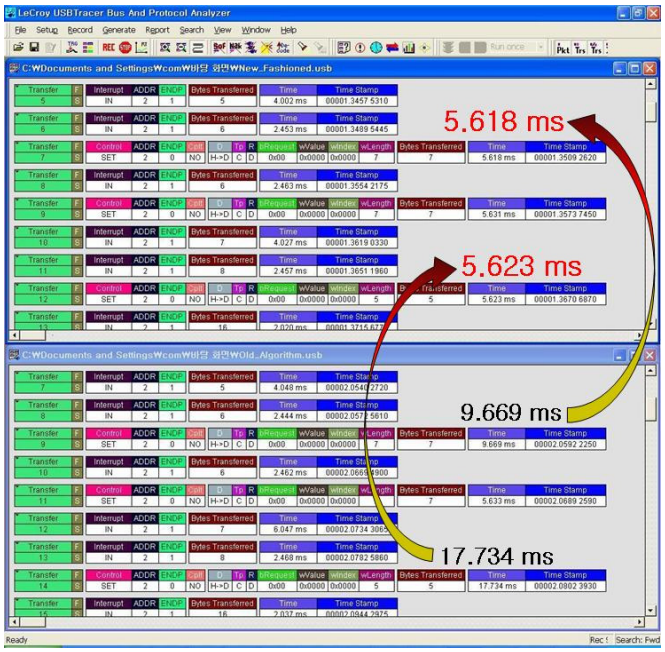

그림 11. USB Packet of Bluetooth HCI layer

5. 결론

본 논문에서는 Embedded System 과 USB Interface 로 접속된 Bluetooth USB Dongle 의 Bluetooth HCI layer 와의 접속을 제시하고 HCI layer 의 속도 향상을 위한 알고리즘을 제안하며, 시뮬레이션을 통하여 제안한 HCI Layer 의 알고리즘의 개선이 속도의 향상에 효과적임을 보였다. 본 논문에서 제안한 방식은 HCI layer 에서 Queuing Buffer 를 두어 송수신 단말간 존재하는 Data Flow 의 병목 현상을 최소화할 수 있도록 논리적으로 분리시킴으로써, 이로 인하여 Lower HCI layer 와 Upper HCI layer 의 상호간 의존성이 크게 줄어, 기존의 방식에 비해 많은 이점을 가짐을 알 수 있다.

참고문헌

[1] Strategy Analytics, Camera Phones Outsell Digital Still Cameras in H1 2006 and Beyond, September 2006.s
[2] Bluetooth Special Interest Group, OPP Specification - http://www.bluetooth.com.
[3] BROADCOM, BCMS1200-BTE Product Brief, June 2004.
[4] Bluetooth Specification Version 2.1 + EDR [vol 4] - http://www.bluetooth.com.
[5] Universal Serial Bus Specification Rev1.1 / USB-IF Administration Headquarter's