Nano-Qplus 기반의 USN 응용프로그램 자동생성 기법*

김주일°, 이우진, 이광용*, 정기원
숭실대학교 대학원 컴퓨터학과, 한국전자통신연구원 임베디드S/W연구단 편재형컴퓨팅미들웨어연구팀*
sespop@empal.com°, bluewj@empal.com, kylee@etri.re.kr, chong@ssu.ac.kr

A Technique for the Automatic Generation of USN Applications based on the Nano-Qplus

Juil Kim°, Woojin Lee, Kwangyong Lee, Kiwon Chong
Department of Computing, Graduate School, Soongsil University
Ubiquitous Computing Middleware Team, ETRI*

요 약

본 논문에서는 센서 네트워크를 위한 운영체제인 Nano-Qplus를 기반으로 수행되는 센서 네트워크를 위한 프로그램의 코드를 자동으로 생성하는 기법을 제시한다. 즉, 센서 네트워크를 구성하는 센서, 라우터, 싱크, 엑츄에이터와 같은 노드들이 수행해야 하는 기능에 대한 코드를 자동으로 생성하도록 하는 기법을 제시한다. 센서 네트워크에 대한 모델을 작성하고, 이를 바탕으로 센서 네트워크의 각 노드에 대한 속성을 스크립트를 통하여 설정하면 각 노드를 동작시킬 수 있는 프로그램이 자동으로 생성된다. 이를 위하여 각 노드의 속성을 설정할 수 있는 스크립트와 프로그램을 자동으로 생성하는 알고리즘을 제공한다.

본 논문에서 제시한 기법을 이용하면 센서 네트워크를 구성하는 각 노드에 대한 속성설정만으로 실행코드를 자동으로 생성함으로써 센서 네트워크를 이용하는 어플리케이션을 개발하는데 소요되는 노력을 줄일 수 있으며, 신속한 코드생성을 통해 조기에 테스트를 수행하여, 오류를 찾아내어 수정함으로써 검증된 코드를 생성할 수 있다.

1. 서론

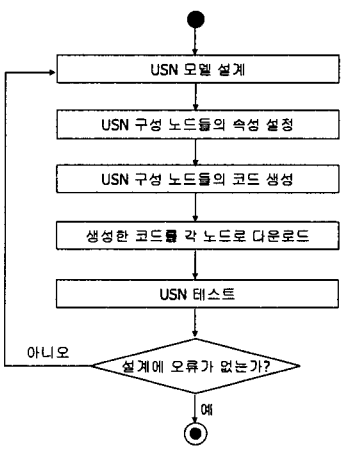
우선통신과 전자기술의 발달로 인해 소형이며, 단거리 통신이 가능한 저비용, 저전력, 다기능의 센서 노드가 개발되어왔다. 이러한 초소형의 센서 노드는 감지, 데이터 처리, 통신 컴포넌트로 구성되어, 노드들 간의 네트워크를 형성할 수 있다. 이러한 개념이 센서 네트워크이다. 센서 네트워크는 전통적인 센서에 비하여 뚜렷한 진보이다 [1]. 이러한 센서 네트워크[2][3]는 홈 네트워크, 건강관리 분야, 로봇 분야 등에서 적용 가능하다. 건강관리 분야에서 예를 들면, 센서 노드는 환자의 상태를 모니터하고, 장애를 가진 환자들을 돕는데 활용될 수 있다.

본 논문에서는 이러한 센서 네트워크를 위한 프로그램의 코드를 자동으로 생성하는 기법을 제시한다. 즉, 센서 네트워크를 구성하는 센서, 라우터, 싱크, 엑츄에이터와 같은 노드들이 수행해야 하는 기능에 대한 코드를 자동으로 생성하도록 하는 기법을 제시한다. 특히, 센서 네트워크를 위한 운영체제인 Nano-Qplus [4][5]를 기반으로 수행되는 센서 네트워크 프로그램에 초점을 둔다.

* 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음

2. USN을 위한 프로그램 생성절차

[그림 1]은 USN (Ubiquitous Sensor Network) 프로그램을 생성하기 위한 절차를 보여준다.



[그림 1] USN을 위한 프로그램 생성절차

USN 프로그램을 작성하기 위해서는 우선, USN에 대한 모델을 설계해야 한다. USN 모델을 설계한다는 것은 USN을 구성하는 센서, 라우터, 싱크, 엑츄에이터와 같은 노드들을 배치하고, 각 노드들의 속성 및 노드간의 통신을 위한 관계를 설정해 주는 것을 말한다. USN 모델 설계가 끝나면, 설

계한 모델을 바탕으로 USN의 각 구성 노드들에 대한 속성을 설정해야 한다. 속성을 설정하고 나면, 설정한 속성값을 바탕으로 각 노드를 위한 실행 코드를 자동으로 생성한다. 이러한 실행코드는 Nano-Qplus 운영체제의 환경에 맞추어, 각 속성값에 따라 미리 정의되어 있는 소스코드를 재사용하여 생성하게 된다. 생성한 실행코드가 설계한대로 동작하는가를 테스트하기 위하여 실제 노드에 실행코드를 다운로드하여 테스트하고, 오류가 없다면 생성한 코드를 사용하게 된다. 이 때, 각 노드의 실행코드를 테스트하기 위해서는 설계한 모델대로 수행되는가를 알아볼 수 있는 최소한의 노드로만 센서 네트워크를 구성하여 테스트한다.

3. USN 프로그램 생성을 위한 스크립트

2장에서 USN 프로그램을 생성하기 위한 절차를 보면, USN 모델을 설계한 후에는 설계한 모델을 바탕으로 각 노드들의 속성값을 설정해야 한다고 설명하였다. 본 논문에서는 노드들의 속성값을 쉽게 설정할 수 있도록 [그림 2]와 같은 스크립트를 제공한다. 이 스크립트는 리눅스에서 환경설정을 위해 사용하는 스크립트[6]를 참고하여 개발한 것이다. 사용자는 각 노드에 대한 속성값을 이러한 스크립트를 통해서 설정해 주면, 스크립트의 내용을 바탕으로 프로그램을 자동으로 생성하게 된다. 이 스크립트의 내용은 Nano-Qplus 운영체제 하에서 센서 네트워크를 수행하기에 적합한 프로그램을 생성할 수 있도록 작성하였다.

```
[ ] Enable EEPROM module (NEW)
[ ] nable Flash memory module (NEW)
[*] nable Timer module
[ ]   nable Digital clock module (NEW)
[*] nable UART module
[*]   nable printf module (NEW)
[ ]   nable scanf module (NEW)
[*] nable actuation(e.g.,Relay,LED,....) modules
[*]   nable LED module (NEW)
[ ] nable ADC or Sensor module (NEW)
[*] cheduler module
[ ]   IFO scheduler (NEW)
[*]   reemption-RR scheduler (NEW)
[ ]     ower management (NEW)
[*] nable Zigbee RF module
[*]   nable Simple Send/Recv module (NEW)
[*]   nable IEEE 802.15.4 MAC module
[*]     nable Star-Mesh route module
(SIMX)  N do type?(SIMX_ROUTER,...) (NEW)
(0)     efault Application Node ID?
(1)     djacent Actuator Node ID?
(TRUE)  s it PAN Cordinator node?(TRUE/FALSE) (NEW)
(NON_BEACON_ENABLE) N N_BEACON_ENABLE or not? (NEW)
(0x1111)  EFAULT_SRC_SHORT_MAC_ADDR?
[ ]     s the DEFAULT_EXTENDED_MAC_ADDR used? (NEW)
(1)     SSOCIATION_PERMIT_NODEID_START?
(99)    TART_MESH_ASSOCIATION_PERMIT_NODEID_END?
[ ]   nable RSSI module (NEW)
[ ] Enable Inter-Thread-Communication (NEW)
```

[그림 2] USN 프로그램 생성을 위한 스크립트

4. USN 프로그램 생성 알고리즘

스크립트를 통하여 선택한 정보를 바탕으로 각 노드의 실행을 위한 소스코드는 다음과 같은 절차를 통하여 생성된다.

Step 1 - 사용자가 선택한 정보를 저장한 Config_Info (.config) 파일을 입력 받는다.

Step 2 - Config_Info 파일을 분석하여 선택된 모듈을 찾아내고 해당 모듈의 헤더, 데이터, 함수 코드를 HashTable_Module 클래스의 함수를 통하여 생성하여 Templet에 저장한다.

Step 3 - 선택된 모듈에 따른 main 코드를 Hash Table_Main에서 읽어와 Templet에 저장한다.

이러한 절차에 따라 소스코드를 생성하기 위한 알고리즘은 다음과 같다. 소스코드를 생성하고자 하는 노드의 타입에 따라 HashTable_Module 클래스의 함수를 호출하여 헤더, 데이터, 함수의 코드를 생성한다.

```
Templet Transformation(Config_Info config_info) {
   templet = getTemplet(config_info.NODETYPE);
   templet.setAttribute(config_info.Attribute);

//  HashTable_Module<NODETYPE> 설정
   HashTable_Module.setType(config_info.NODETYPE);
// 각 type에 따라 config_info에 맞도록 코드를 구성
   iterator iterator = Parser.getIterator(conifg_info);
   while( iterator.hasNext() ) {
        templet.addHeader(

        HashTable_Module.getHeader( iterator.ModuleName ) );
        templet.addData(
                HashTable_Module.getData( iterator.ModuleName ) );
           templet.addFunction(

        HashTable_Module.getFunction( iterator.ModuleName ) ) ;
        templet.addMain(
                HashTable_Main.getMain( iterator.ModuleName ) );
        iterator.next();
   }
}
```

[그림 3] USN 프로그램 생성을 위한 알고리즘

소스코드 생성 알고리즘에서 사용하는 HashTable_Module 클래스는 [그림 4]와 같이 구성되어 있다.

```
public class HashTable_Module{
        ...
        public void setType(String nodeType) {
                TempletHashTable templetTable = new TempletHashTable();
                moduleTable = new Hashtable();
                int size = moduleKey.length;
                String templetModule = "";
                this.nodeType = nodeType;
                for(int i = 0; i < size; i++) {
                        templetModule = (String)templetTable.get(moduleKey[i]);
                        templetModule = templetModule.replaceFirst("&1", nodeType);
                        moduleTable.put(moduleKey[i], templetModule);
                }
        }
        public String getHeader(String moduleName) {
                return getReplaceModule(moduleName, HEADER);
        }
        public String getReplaceModule(String moduleName, String value) {
                String templetModule = (String)moduleTable.get(moduleName);
                templetModule = templetModule.replaceFirst("&2", value);
                return templetModule;
        }
        ...
};
```

[그림 4] HashTable_Module 클래스

HashTable_Module은 노드의 타입에 따라 헤더, 데이터, 함수의 코드가 다르기 때문에 동적으로 해쉬테이블을 사

용하여 각 타입에 따른 코드를 생성하여 준다.
정적으로 값이 고정되어 있는 해쉬테이블을 사용하면 각 노드 타입마다 해쉬테이블을 따로 생성해야 하며, 각 해쉬테이블을 제어하기 위한 코드 또한 각각 만들어 줘야 한다. 또한 노드의 타입이 추가될 경우에는 새로운 해쉬테이블을 생성해 주어야 하며, 그에 대한 코드도 새로 추가해 주어야 한다. 그러나 동적으로 해쉬테이블의 값을 바꿀 수 있도록 하면, 새로운 노드의 타입이 추가되더라도 해쉬테이블에 키와 그 값만 넣어주면 되므로, 새로운 코드를 추가하지 않고, 그대로 사용할 수 있다.
HashTable_Module 클래스에서 사용하는 해쉬테이블은 [표 1]과 같은 구조를 갖는다.

[표 1] 해쉬테이블의 구조

키	키의 값
Zigbee_Simple	"&1_Zig_Simple_&2"
Zigbee_MAC	"&1_Zig_MAC_&2"
Zigbee_MAC_StarMesh	"&1_Zig_StarMesh_&2"
Scheduler_FIFO	"&1_Sche_FIFO_&2"
Scheduler_PreemptionRR	"&1_Sche_PreemptionRR_&2"
Sensor_LIGHT	"&1_Sensor_LIGHT_&2"
Sensor_GAS	"&1_Sensor_GAS_&2"
Sensor_Temperature	"&1_Sensor_Temperature_&2"

해쉬테이블의 구조에서 키의 값은 USN을 위한 프로그램을 자동으로 생성하고자 할 경우에 각 속성값에 따른 실제 코드가 저장되어 있는 파일의 이름이다. 키 값에서 &1, &2 라는 문자열은 노드 및 호출되는 모듈의 타입에 따라서 동적으로 변경되는 부분이다. 각 노드의 타입이 결정되면 &1이 노드 타입으로 대체되고, 소스코드에서 필요한 모듈의 타입에 따라 &2가 변경된다. 모듈의 헤더, 데이터, 함수 코드 중 선택되는 것에 따라 &2는 "H', "D", "F" 로 대체된다. 이것을 예로 들면 다음과 같다.

예) 노드에서 rf 통신을 위한 모듈로 Zigbee_Simple가 선택되었다고 가정하면 해쉬테이블에서 "&1_Zig_Simple_&2" 값을 가져 올 것이다. 그러면 Hash Table_Module 클래스의 각 함수를 호출함에 따라 해쉬테이블의 값이 다음과 같이 변경될 것이다.
setType("SINK"); ➔ "SINK_Zig_Simple_&2"
getHeader("Zig_Simple") ➔ "SINK_Zig_Simple_H"
getFunction("Zig_Simple") ➔ "SINK_Zig_Simple_F"
여기에서 "SINK_Zig_Simple_H"는 싱크 노드에서 ZigBee_Simple 모듈을 사용하고자 할 경우에 필요한 헤더코드를 담고 있는 파일의 이름이며, "SINK_Zig_Simple_F"는 함수의 코드를 담고 있는 파일의 이름이다.

5. 결론 및 향후연구

본 논문에서는 센서 네트워크를 지원하기 위한 운영체제인 Nano-Qplus를 이용하여 수행되는 센서 네트워크 프로그램을 자동으로 생성하는 기법을 제시하였다. 센서 네트워크 프로그램은 모델링, 스크립트를 이용한 속성설정, 스크립트 정보를 기반으로 한 실행코드 자동생성, 다운로드 및 테스트를 통한 코드검증의 단계를 걸쳐 완성된다. 센서 네트워크에 대한 모델을 작성하고, 이를 바탕으로 센서 네트워크의 각 노드에 대한 속성을 스크립트를 통하여 설정하면 각 노드를 동작시킬 수 있는 프로그램이 자동으로 생성된다. 이렇게 생성된 코드를 센서 네트워크를 구성하는 실제 노드에 다운로드 한 후에 테스트하여 오류를 발견하여 수정하고, 오류가 발견되지 않았을 경우에는 생성한 프로그램을 사용한다. 본 논문에서 제시한 기법을 이용하면 센서 네트워크를 구성하는 각 노드에 대한 속성설정만으로 실행코드를 자동으로 생성함으로써 센서 네트워크를 이용하는 어플리케이션을 개발하는데 소요되는 노력을 줄일 수 있으며, 신속한 코드생성을 통해 조기에 테스트를 수행하여, 오류를 찾아내어 수정함으로써 검증된 코드를 생성할 수 있다.
향후에는 현재의 기법을 개선하고, 추가로 센서 네트워크에서 필요한 여러 테스트를 수행할 수 있는 프로그램을 자동으로 생성하는 시스템을 개발하고자 한다.

참고문헌

[1] I.F.Akyildiz, W. Su et al., "A Survey on Sensor Networks," IEEE Communications Magazine, August 2002
[2] 채동현, 한규호, 임경수, 안순신, "센서 네트워크의 개요 및 기술동향," 한국정보과학회 학회지 VOL. 22 NO.12, 2004
[3] 이재용, "유비쿼터스 센서 네트워킹 기술," TTA 저널 제95호, 2004
[4] Kwangyong Lee et al., "A Design of Sensor Network System based on Scalable & Reconfigurable Nano-OS Platform," IT-SoC2004, October 2004
[5] ETRI 임베디드 S/W 연구단, "나노 Qplus," http://qplus.or.kr/
[6] Neil Matthew , Richard Stones , "Beginning Linux Programming 3rd Edition," WROX PRESS, 2003