SDR 시스템을 위한 Ad hoc 방식 분산 소프트웨어 다운로드 제안

이진성° 양형규 이병호
한양대학교 정보통신대학원
jinseong@ihanyang.ac.kr°, sheep@kopo.or.kr, bhrhee@hanyang.ac.kr

A Proposal Of Methods About Downloading Software Based Ad hoc For SDR System

Jin Seong Lee° Hyung Kyu Yang Byoung Ho Rhee
The Graduate School of Information and Communications, Hanyang University

요 약

SDR(Software Defined Radio) 시스템의 발전으로 단말기와 기지국 장비의 교체 없이 서로 다른 표준안을 요구하는 서비스에 대해서 해당 컴포넌트들을 다운로드하고 동적으로 시스템을 재구성하여 해당 서비스에 맞는 시스템을 구성할 수 있다. SDR 시스템의 기본 기능에는 사용하고자 하는 컴포넌트를 여러 가지 방법에 따라 다운로드하는 것이 포함 될 수 있다. 다운로드 할 필요가 있는 컴포넌트는 크게 상용과 공개용으로 나눠 볼 수 있는데 기본적으로 안티-바이러스 패치나 해당 사업자의 중요한 소프트웨어 패치 등의 공개용 컴포넌트를 효율적으로 다운로드 하여 재구성하는 방안이 우선적으로 요구된다. 본 논문에서는 OTA(Over The Air)를 통해 다운로드 하고자 하는 경우를 중심으로, Server/Client 모델의 일 대 일 방식의 다운로드 방식보다 네트워크 자원 및 트래픽의 효율적인 사용을 위해 Ad hoc 모델을 이용하여 다운로드하는 알고리즘을 제안하였다.

1. 서 론

1990년대 초 미군 DARPA의 주관으로 시작된 SDR은 고정된 기능의 통신 시스템 하드웨어 컴포넌트들을 프로그램 제어 가능한 부분으로의 확장을 목적으로 SPEAKeasy 프로젝트가 시작된 이후 최근에는 4세대 이동통신의 기술적 측면, 서비스 측면, 산업적 측면에서 대안으로 가장 주목 받고 있는 개념이다. SDR 시스템은 재구성 가능한 소프트웨어 동작에 의하여 하드웨어 제어 기술의 발전을 확장 시키고, 시스템 내에 증가된 소프트웨어 프로그램 동작 능력을 이용하여 시스템의 유연성을 증대시키는데 목적이 있다. SDR 기반 군용 시스템의 상용화의 접근은 스펙트럼 사용률과 효율성 증대라는 측면에 주요 관심을 가지고 진행 중이다.[1][2] SDR 시스템이 진보하게 되면 단말기나 기지국 장비의 교체 없이 서로 다른 표준안을 요구하는 서비스에 대해서 해당 컴포넌트들을 다운로드 하고 동적으로 시스템을 재구성하여 해당 서비스에 맞는 시스템을 구성할 수 있다.

본 논문에서는 분산 소프트웨어 다운로드에 대해서 알아보고, SDR 시스템의 네트워크 자원 및 트래픽의 효율적인 사용을 위한 Ad hoc 모델 방식의 분산 소프트웨어 다운로드 알고리즘을 제안한다.

2. 분산 소프트웨어 다운로드

SDR 시스템에서 소프트웨어 다운로드는 핵심적인 기술이다. 일반적인 Server/Client 방식의 다운로드는 Server의 역량에 따라 성능이 좌우된다. 한 대의 Server 기능을 여러 Server에서 처리 하게 되면 전체적인 다운로드 부하가 분산되므로 전체적인 성능향상을 이끌어 낼 수 있다.

여러 대의 Server를 사용하지 않고 각 단말기가 Server의 역할도 함으로서 네트워크의 성능 향상을 이룰 수 있는 Ad hoc 방식의 분산 소프트웨어 다운로드에 대해 살펴보도록 하자.

2.1 Ad hoc 방식의 분산 소프트웨어 다운로드

Ad hoc 방식의 분산 소프트웨어 다운로드는 client와 client 간에 소프트웨어를 다운로드 함으로써 네트워크 전체의 부하를 줄일 수 있다. 모든 client들은 자체적으로 SW(software) server 기능을 할 수 있다. 더 빠른 소프트웨어 배포 및 네트워크에서 SW server의 부담이 줄어들게 된다.

Server/Client 모델의 다운로드 시간과 Ad hoc 방식의 다운로드 시간을 비교해 보자.

1개의 Cell 당 약 300~400개의 터미널이 사용 가능 하므로 50개의 Cell를 기준으로 20000개의 터미널에서 소프트웨어를 다운로드 하는 경우를 가정하고 계산 하였다.

• Server/Client 모델 : 한 터미널이 한 개의 소프트웨어를 다운로드하는데 1분이 걸리고, 10개의 SW server가 작동한다고 가정하자. 그리고 1만개의 터미널에서 다운로드하여야 한다고 가정하자. (20000 terminals * 1 min)/10 servers = 2000min ≅ 33.3 min. 약 33.3분 정도의 시간이 걸린다.

• Ad hoc 모델 : Server/Client 모델과 마찬가지로 한 개 소프트웨어를 다운로드 하는데 1분이 걸리고, 다운로드 하는 터미널이 정확히 2배수로 늘어난다고 가정한다.

2^{14}=16384 이다. 그러므로 Ad hoc 모델로 배포할 경우에는 15~16분 정도 걸린다.

<표 1> 사이클 당 다운로드 된 터미널 수

	1	2	3	4	5	6	...	14	...
	1	2	4	8	16	32	...	16384	...

위 두 모델의 경우를 비교해 본 결과 이상적인 가정 아래의 비교일지라도 50개의 Cell의 터미널에 배포하는 경우에 약 두 배의 시간 차이가 있음을 알 수가 있다. 또한 다운로드를 위해 할당된 네트워크 자원은 Server/Client 모델의 네트워크 자원과

달리 Ad hoc 모델의 그것은 부하가 분산되어 효율성이 좋아졌음을 보여준다.[2]

3. 제안된 다운로드 알고리즘

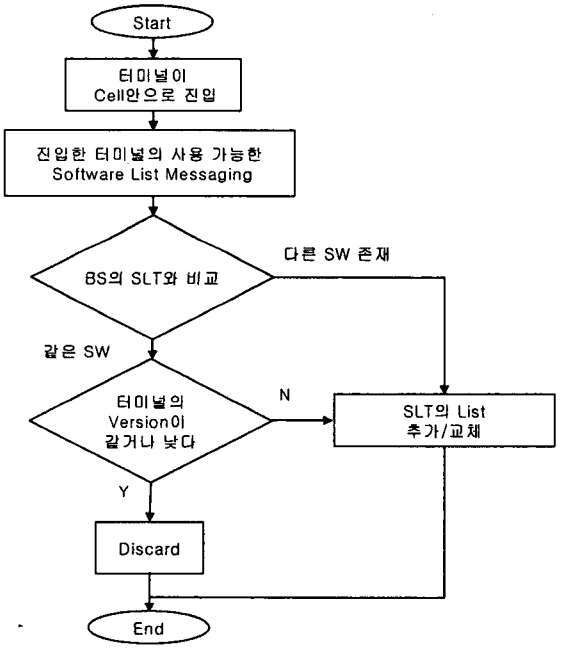
소프트웨어나 컴포넌트는 크게 상용 소프트웨어와 범용적인 목적의 공개용 소프트웨어로 나누어 볼 수 있다. 상용 소프트웨어의 경우에는 보안, 다운로드한 소프트웨어 당 금액 책정 및 여러 가지 요건들이 맞물려 있다. 그러나 공개용 소프트웨어의 경우에는 과금 문제나 배포 문제 등 여러 가지 측면에서 제한이 덜 심하기 때문에 좀 더 자유스럽게 접근이 가능하다. 분산 소프트웨어 다운로드와 관련된 보안 문제는 새로운 방안이 제시 되어있다.[2] 본 논문에서 제안한 알고리즘은 안티-바이러스 패치나 공개용 게임 등의 공개용 소프트웨어를 대상으로 한다.

이용 및 제공 가능한 Software 정보를 담고 있는 SW List Table(SLT)의 내용은 Base Station(BS)/터미널, 터미널/터미널 간의 메시지 전송을 통해 상호 교환된다.

본 논문에서는 상호 교환되는 메시지를 통해 이루어지는 다운로드 알고리즘을 BS와 터미널 측면에서 접근 해 보았다.

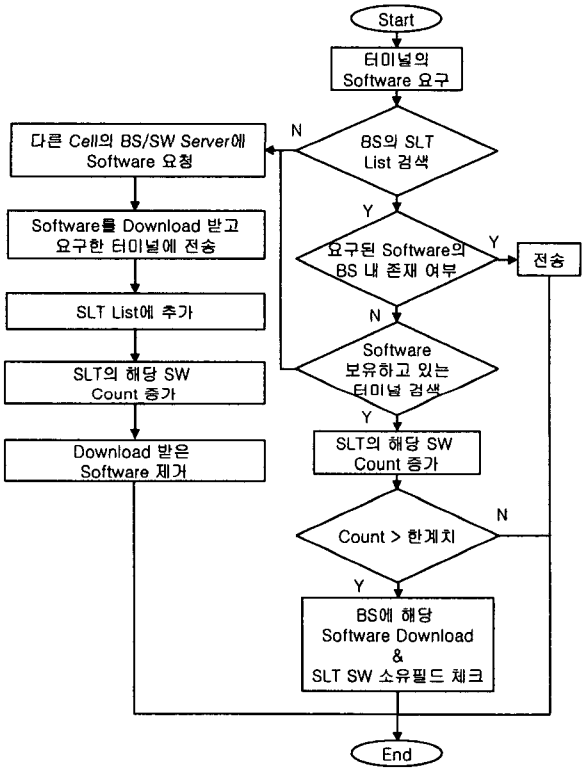
3.1 Base Station 측면에서 다운로드 알고리즘

BS 측면에서의 다운로드 알고리즘은 크게 세 가지 경우로 나눠볼 수 있다. 첫 번째로 Cell안으로 터미널이 들어온 경우, 두 번째로 Cell 안의 터미널이 임의의 소프트웨어를 요구한 경우, 마지막으로 터미널이 Cell을 벗어났거나 내부적인 동작을 위한 타이머에 관한 경우이다.



<그림 1> Cell 안에 터미널이 진입한 경우

<그림 1>은 Cell 안으로 터미널이 들어온 경우에 대한 알고리즘이다. Software List Messaging 이란 것은 터미널이 다른 터미널들에게 제공해 줄 수 있는 소프트웨어 리스트를 브로드캐스팅 하는 것을 말한다. 이 Software List는 각 터미널이나 BS의 SLT라고 하는 테이블에 저장되어 있다. BS에서는 자신의 SLT과 새롭게 등록한 터미널의 SLT를 비교하여 자신의 리스트에 등록된 소프트웨어보다 높은 버전의 리스트나 새로운 소프트웨어가 있으면 리스트를 Update한다. 임의의 터미널이 소프트웨어를 요구하면 SLT를 우선 검색하여 해당 Cell 안에 요구한 소프트웨어가 있는지를 알 수 있다. 요구받은 소프트웨어가 리스트에 존재 한다면 응답을 통해 어떤 단말기가 가지고 있는지를 가르쳐 줌으로서 다운로드 과정을 원활히 할 수 있게 해준다.



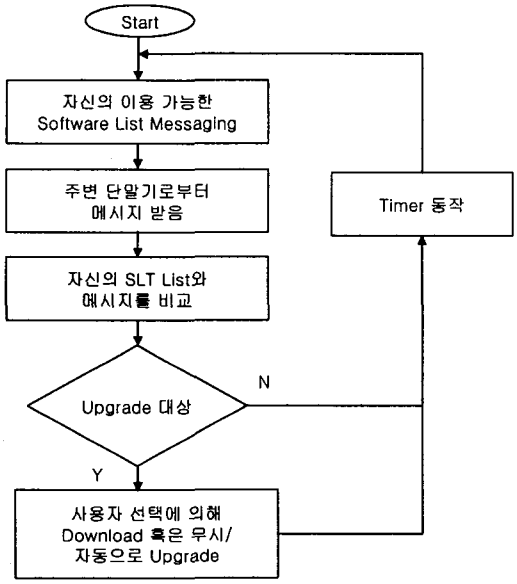
<그림 2> 터미널이 임의의 소프트웨어를 요구한 경우

<그림 2>는 Cell 안의 터미널이 임의의 소프트웨어를 요구했을 경우의 알고리즘이다. Cell 안의 터미널이 필요로 하는 SW를 BS에 요구하면, BS는 일단 자신에게 있는지 검사하고 있으면 전송한다. 없으면 SLT에 요구받은 SW가 있는지 검사 하고 있으면 그 SW 리스트 Count를 증가 시키고 해당 터미널의 위치를 가르쳐준다. Count는 일정 수가 넘을 경우 수요가 많은 것으로 판단하여 BS이 SW를 직접 다운로드 하여 가지고 있다가 요구하는 터미널에 직접 전송해 준다. Cell 안에 그 SW가 없다면 다른 Cell의 BS이나 SW Download Server에 요구하여 다운로드 받고 요구한 터미널에 전송해 준 후 SW를 제거한다.

마지막으로 터미널이 Cell을 벗어났거나 내부적인 동작을 위한 타이머에 관한 경우이다. 임의의 시간 간격으로 메시지를 받음으로서 Cell을 떠난 터미널에 해당하는 SW List나 다운로드하여 사용한 후 오랫동안 요구되지 않은 SW를 제거하여 SLT를 효율적으로 관리 할 수 있다.

3.2 Terminal 측면에서 다운로드 알고리즘

터미널의 측면에서 다운로드 역시 여러 가지 경우로 나눠 볼 수 있지만 BS와 겹치는 기능은 다시 언급하지 않고, 주변의 다른 단말기들과의 메시지 통신 및 업그레이드에 관련된 알고리즘을 제안하였다.



<그림 3> 터미널들의 메시지 통신을 통한 Upgrade 경우

<그림 3>은 터미널들 사이에 메시지 통신을 통해 Upgrade 하는 경우의 알고리즘이다. 타이머를 통해 주기적으로 메시지를 Cell 내에 방송한다. 그리고 주기적으로 수신한 다른 단말기들의 SW List와 자신의 것을 비교하여 Upgrade 여부를 결정한다. Upgrade 여부는 사용자가 직접 선택할 수도 있고, 망 관리나 보안 관련 등의 특정 소프트웨어의 경우에는 자동적으로 Upgrade가 되도록 설정해 줄 수도 있다.

3.3 Software List Table (SLT)

SLT는 <표 2>과 같은 형식을 취하고 있다.

<표 2> SLT (Software List Table)

N	Pack-1	3.0	30K	2003. 3.1	0	N
N	LG-Patch	2.2.4	25K	2003.8.6	12	Y
N	Anti-Virus Patch	3.5-1	5K	2003.8.2	20	Y
Y	Game-k	2.0	310K	2003.7.10	2	N

메시지를 교환하여 SW를 Upgrade나 다운로드 하는데 사용하는 SLT는 아래와 같고 SLT 정보 중에서 SW Name, SW Version 등의 정보를 서로 교환함으로써 Upgrade 대상을 선별할 수 있다.

Upgrade 하는 경우를 예를 들어보자. 먼저 상용 SW bit 필드를 확인하여 공개용 소프트웨어인지 상용 소프트웨어인지를 구별한다. 공개용 소프트웨어인 경우에 다운로드에 제한이 없으므로 SW Name을 확인하여 같은 SW가 있으면 SW Version을 확인한다. 자신이 가지고 있는 SW보다 상위 버전이면 다운로드 요청을 하고, 상대방이 해당 소프트웨어의 보유여부 Bit를 확인하여 가지고 있다면 다운로드 받는다. BS 경우 가지고 있지 않을 수 있는데 이 경우에는 다른 가지고 있는 터미널과 연결을 시켜준다.

4. 결 론

서론에서는 SDR의 등장 배경에 대하여 고찰하고, 본론에서는 Ad hoc 기반의 분산 소프트웨어 다운로드와 분산 소프트웨어 다운로드 알고리즘에 대한 아이디어를 살펴보았다.

본 논문에서는 공개용 소프트웨어를 Base Station과 터미널에서 다운로드 할 때 Ad hoc 기반의 분산 소프트웨어 다운로드 방법을 이용하여 접근하였다.

SDR Forum[2]에서는 JTRS(Joint Tactical Radio System)[4]의 SCA(Software Communication Architecture)[5]를 SDR 내장형 시스템을 위한 소프트웨어 구조의 표준으로 인정하였고, 분산 객체 모델의 실질적인 산업 표준인 CORBA(Common Object Request Broker Architecture)를 기반 미들웨어로 채택하여 이기종의 하드웨어와 다양한 언어로 작성된 소프트웨어에 대하여 유연한 통합 환경을 제공하기 때문에 차후에 제안된 알고리즘을 CORBA에 적용하는 연구를 할 것이다.

제안된 방안은 소프트웨어 다운로드 및 배포의 기초 자료로 활용될 수 있을 것으로 사료된다.

참고 문헌

[1] 김지연, 김진업, "SDR 기술의 현재와 발전방향", 한국통신학회지 19권 11호, 2002.11

[2] Software Defined Radio(SDR) Forum, http://www.sdrforum.org

[3] Markus Dillinger, Reinhard Becher, "Decentralized Software Distribution For SDR Terminals", IEEE Wireless Communications, April 2002

[4] Joint Tactical Radio System(JTRS), http://www.jtrs.saalt.army.mil

[5] Software Communications Architecture(SCA) Specification MSRC-5000SCA v2.2, Joint Tactical Radio Systems, November 17, 2001, Available at http://www.jtrs.saalt.army.mil/SCA/SCA.html