능동 네트워크를 위한 Java 기반 자료 흐름 언어[1)]

김민영 조은선
충북대학교 전기전자 컴퓨터공학부
tristan88@hanmail.net, eschough@chungbuk.ac.kr

A Data Flow Language for Active Networks based on Java

Min Young Kim Eun-Sun Cho
School of Electrical & Computer Engineering, Chungbuk National University

요 약

능동 네트워크 (active network) 는 각 네트워크 요소들이 단순한 데이터 전달 뿐 아니라 데이터를 다루는 프로그램을 탑재/수행함으로써 네트워크 상에서 부가적인 작업을 가능하게 한다. 본 논문에서는 자료 흐름 모델에 기반 한 능동 네트워크 언어를 제안하고 Java 환경에서 수행을 가능하게 하는 변환기와 보조 API를 제안하였다.

1. 서론

능동 네트워크 (active network) 는 각 네트워크 요소들이 단순한 데이터 전달 뿐 아니라 데이터를 다루는 프로그램을 탑재/수행함으로써 네트워크 상에서 부가적인 작업을 가능하게 함으로써 많은 관심을 받고 있다. 그리고, 최근 들어 인터넷의 다양한 응용과 보안등을 포함한 부가 기능들이 주목을 받음에 따라 이에 적합한 부가 기능들을 능동 네트워크에서 담당해줄 필요성이 생기고 있다 [1].

네트워크 단위 요소의 작업은 하나의 패킷 흐름에 대해 서로 다른 여러 가지의 가공이 단독 혹은 결합 형태로 이루어지게 된다. 또한, 하나의 이벤트가 발생하여 처리되는 도중 또 다시 새로운 이벤트들이 만들어지고 새로운 이벤트의 처리 모듈과 기존 이벤트 처리 모듈이 동시에 수행된다. 이러한 것을 손쉽게 지원할 수 있는 보다 복잡한 계산 모델이 요구되고 있다.

본 논문에서는 자료 흐름 모델에 기반 한 능동 네트워크 언어를 제안하고 Java를 기반으로 구현하였다.

2. 관련 연구

전통적인 능동 네트워크 계산 모델 중에서 일반적인 (procedural) 계산 모델이나 함수 (functional) 모델 기반의 수행 요소들은 각각 고유의 장점은 있으나, 패킷의 흐름 처리라는 사항에 대하여 초점을 맞추지 않기 때문에 패킷 흐름이 단순한 상황에서는 문제가 없지만, 보다 여러 가지 기능을 구현하는 모듈들이 서로 관련을 가지고 능동적으로 결합하는 상황에 적용하기는 한계가 있다. 따라서 이러한 경우 제어 흐름이 불필요하게 복잡해짐으로써 코드 생성 및 수정 비용이 커지며 코드의 오류율도 높아진다는 단점을 가진다.

이에 비해 자료흐름 (dataflow) 모델을 기반으로 하고 있는 능동 네트워크 언어의 경우 여러 가지 가공 단위들 간의 정보 흐름을 쉽게 나타낼 수 있으므로 이벤트의 처리 모듈간의 결합 자체를 비교적 자유롭게 나타낼 수 있다[2].

능동 네트워크에 자료 흐름 모델을 도입한 대표적인 예로 미 Columbia 대학에서 제안된 상위 언어인 NetScript[2]가 있다. Netscript는 각 계산 단위를 '상자(box)' 라고 명명하며, 이를 결합한 상위 단위를 '상자 틀(box template)'이라고 정의한다. 결과적으로 상자 틀에 대해 기술하면 다소 복잡한 능동 네트워크 프로토콜을 구현하게 된다.

MIT의 StreamIt[3]은 Filter를 원소로 사용하여 연결형태를 만들고 이들을 확장하여 자료흐름을 처리 할수 있도록 하는 언어 이다. Filter의 연결형태는 Pipeline, SplitJoin, FeedbackLoop의 세가지 이며, 분기나 결합시

1) 본 연구는 한국 과학재단 목적기초연구 (R04 - 2002 - 000 - 00093 - 0) 지원으로 수행되었음

다양한 방법을 선택할 수 있고 제어를 위한 별도의 통신을 가정한다는 특징이 있다. 하지만 연결형태를 세가지로만 한정시켜 놓음으로써 이벤트 처리 모듈간의 자유로운 결합에 제약이 있다.

본 논문에서 제안되는 언어는 다음과 같은 특징이 있다.

■상자와 상자간의 중첩 결합이 여러 단계로 가능하다. 즉 모든 것을 단일 개념의 상자로 생각함으로써 여러 계층의 상위 상자와 부속 상자들의 자유로운 결합 정의가 가능하다. 따라서 한정된 결합형태를 가지는 StreamIt이나 상자와 상자 틀로 이분화 한 Netscript 구조에 비해서 결합시 높은 자유도를 가지게 된다.

■각 상자의 수행 내용에 대한 기술이 상자의 정의에 포함가능 하도록 하고 자주 사용되는 상자의 수행 연산을 API로 제공하였다.

■능동 네트워크 요소에 쉽게 사용 될 수 있는 Java를 사용하였다. 수행 내용이나 프로토콜 등을 동적으로 변화시킬 수 있는 능동 네트워크 환경에서는 C와 같은 정적인 언어 보다 Java가 유리하다[4].

3. 제안하는 자료 흐름 기반 언어

```
box template POP3
{
   inport popIn;
   outport popOut;
   import "PopDemuxBox.class" PopDemuxBox;
   import "PopStateBox.class" PopStateBox;
   import "PopUserBox.class" PopUserBox;
   import "PopPassBox.class" PopPassBox;
   import "PopRetrBox.class" PopRetrBox;
   import "OutBox.class" OutBox;
   connect
   {
      popIn->PopDemuxBox.in,
      PopDemuxBox.userCmd->PopUserBox.usecmdIn,
      PopDemuxBox.passCmd->PopPassBox.passcmdIn,
      PopDemuxBox.retrCmd->PopRetrBox.retrcmdIn,
      PopUserBox.setUserState->PopStateBox.setUserState,
      PopStateBox.getUserState->PopUserBox.getUserState,
      PopPassBox.setPassState->PopStateBox.setPassState,
      PopStateBox.getPassState->PopPassBox.getPassState,
      PopRetrBox.setRetrState->PopStateBox.setRetrState,
      PopStateBox.getRetrState->PopRetrBox.getRetrState,
      PopUserBox.userReply->OutBox.userReplyIn,
      PopPassBox.passReply->OutBox.passReplyIn,
      PopRetrBox.retrReply->OutBox.retrReplyIn,
      OutBox.replyOut->popOut
   }
}
```

그림 1. POP3 프로토콜 정의의 예

본 논문에서 제안되는 언어의 구문은 일반적인 객체지향 언어나 인터페이스 기술 언어에서 벗어나지 않으므로 쉽게 이해될 수 있으며, 기호 등은 Netscript에서 상당 부분을 참고하였다. 그림 1은 POP3 프로토콜을 위해 기술된 상자 틀의 예이다. inport와 outport는 외부 입력과 출력을 담당하는 포트의 이름이며, import 구문은 소속되는 부속 상자들의 이름들의 나열이다. connect 문에서는 부속 상자의 포트들 간의 연결 관계를 나타내고 있다.

이 중 connect문의 의미를 전이 시스템으로 나타내보면 다음과 같다.

S : 임시적인 값들을 저장하는 스택
L : port들과 queue들을 산상하는 함수
C : 명령문들을 저장하는 스택

C1. $\langle S, L, p\text{->}p' C\rangle \rightarrow \langle S, L, p\ p' \text{->} C\rangle$ p, p'∈ P, P: set ports
C2. $\langle S, L, p\ p' \text{->} C\rangle \rightarrow \langle pS, L, p' \text{->} C\rangle$
C3. $\langle pS, L, p' \text{->} C\rangle \rightarrow \langle p'\ pS, L, \text{->} C\rangle$
C4. $\langle p'pS, L, \text{->} C\rangle \rightarrow \langle S, (p,q)(p',q)\ L, C\rangle$ new q s.t. q ∈ Q

이러한 언어의 문법은 정형화하여 그림2와 같이 정의된다. 이중 키워드 'ACTION'은 상자의 수행을 위한 코드를 포함하기 위한 구문으로 위의 그림1과 같은 복합 상자에서는 생략될 수 있다. import구문에는 부속 상자들을 정의하는 클래스의 경로를 명시하게 된다.

```
Box_template  ::= box template WORD
                  { Inport_list Outport_list [Import_list]
                    [Connect_list] [ ACTION ] }
Inport_list   ::= Inport { Inport }
Inport        ::= inport WORD ;
Outport_list  ::= Outport { Outport }
Outport       ::= outport WORD ';'
Import_list   ::= Import { Import }
Import        ::= import " Path " WORD ;
Path          ::= WORD [ :/ ] { / WORD} .class
Connect_list  ::= connect { Connect { , Connect } }
Connect       ::= ( Port | WORD ) -> ( Port | WORD)
Port          ::= WORD . WORD
```

그림 3. 문법의 정형화

4. 구현

4.1 전체 구조

제안되는 언어로 정의된 상자들은 변환기를 통해 Java 코드로 변환된다. 이러한 Java 코드들은 javac로 컴파일되며 본 논문에서 제안하는 API들과 결합하여 수행된다. 변환기는 파싱 도구인 JavaCC2.1[5]을 사용하여 구현하였으며, 보조 API는 NetScript 프로그램을 Java 언어로 번역하여 수행하는 데 필요하게 되는 Java 프로그램들이다.

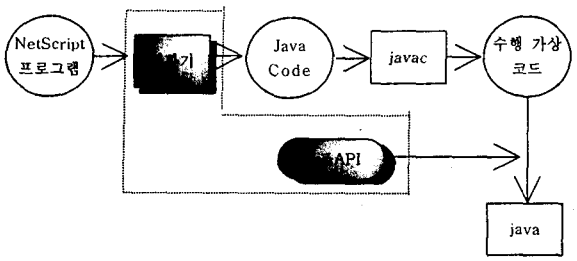

그림 4. 전체 구조

4.2 변환 결과

변환 후의 각 상자는 그림 4 과 같은 계층 구조상의 Java 클래스에 의해 정의된다. 구체적인 상자들은 각각 SimpleBox와 CompositeBox 의 하위 클래스로 정의된다. 이들은 병렬 수행을 위해 Thread의 하위클래스로 정의되게 된다.

인터페이스 Box는 상자들이 가져야 하는 공통적인 특징인 이름과 포트들에 관한 메소드를 명시했다.

AbstactBox는 Java API의 HashMap을 필드로 가지고 있어서 외부 입출력 포트들의 등록에 관한 메소드를 제공한다.

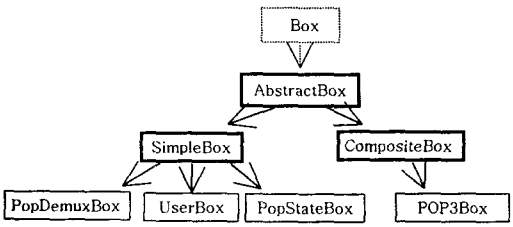

그림 4. 계층 구조

클래스 CompositeBox는 import 되는 상자들의 정보를 저장해 둔다. 클래스 CompositeBox의 run() 메소드는 connect문을 그대로 반영하는 것으로서, import된 상자들의 run() 메소드들을 적절히 결합하여 호출하는 형식으로 구현된다. 이때 각 상자들은 큐 구조로 이루어진 링크로 연결된다.

```
public void run(){
  Iterator itBoxes= ((Set)importedBoxMap.entrySet()).iterator();
  while (itBoxes.hasNext()){
    AbstractBox a =
        (AbstractBox)((Map.Entry) itBoxes.next()).getValue();
    a.start();
  }
}
```

그림 5. CompositeBox에서의 run() 메소드

각 ACTION 구문은 SimpleBox의 하위 클래스의 run() 메소드에 반영된다. 다음은 'move()'를 사용하여 ACTION 내부를 정의하고 변환한 예이다.

ACTION 구문

```
ACTION
  while(true){
    move(in,userCmd);
    move(in,passCmd);
    move(in,retrCmd);
  }
END
```

Java로 변환된 코드

```
public void run(){
  while(true){
    move(in,userCmd);
    move(in,passCmd);
    move(in,retrCmd);
  }}
private void move(InPort in,
OutPort out){
getPort(out.getName()).put(
getPort(in.getName()).get()
;}
```

'move()'의 의미는 전이 시스템을 사용하여 다음과 같은 규칙들로 나타낼 수 있다.

M1. $\langle S, L, \text{Move}(p, p')C \rangle \rightarrow \langle S, L, p, p' \text{Move } C \rangle$

M2. $\langle S, L, p, p' \text{Move } C \rangle \rightarrow \langle\langle p, L(p) \rangle S, L, p' \text{Move} C \rangle$

M3. $\langle\langle p, L(p) \rangle S, L, p' \text{Move} C \rangle \rightarrow \langle\langle p', L(p') \rangle \langle p, L(p) \rangle S, L, \text{Move} C \rangle$

M4. $\langle\langle p', L(p') \rangle \langle p, L(p) \rangle S, L, \text{Move} C \rangle \rightarrow \langle S, L[\text{tail } L(p)/p][(\text{head } L(p))\ L(p')/p'], C \rangle$

5. 결론

본 논문에서는 자료 흐름 모델에 기반 한 능동 네트워크 언어를 제안하고 Java 환경에서 수행을 가능하게 하는 변환기와 보조 API를 제안하였다.

향후에는 제안된 언어를 XML을 지원하고 능동네트워크 수행요소의 주요 기능중 하나인 필터링 기능을 추가 할 계획이다.

6. 참고 문헌

[1] Mahesh V. Tripunitara and Eugene H. Spafford, 'Issues in the Incorporation of Security Services into a Protocol Reference Model', COAST TR-98/03, Purdue University, 1998

[2] Da Silva, Danilo Florissi, Yechiam Yemini, 'Composing Active Services in NetScript', Position paper, DARPA Active Networks Workshop, Tucson, AZ, March 9-10, 1998

[3] W. Thies, M. Karczmarek, and S. Amarasinghe. 'StreamIt: A Language for Streaming Applications', In Proc. of the Int. Conf. on Compiler Construction (CC), 2002.

[4] The Janos Project(Java-oriented Active Network Operating System), http://www.cs.utah.edu/flux/janos, February 2003

[5] Java Compiler Compiler [tm] (JavaCC [tm]) -The Java, http://javacc.dev.java.net