대규모 온라인 게임을 위한 XML 기반의 메시지 정의 시스템

박 학 봉[†] · 오 삼 권[††]

요 약

대규모 온라인 롤-플레잉 게임(MMORPG : Massively Multiplayer Role-Playing Game)에서 클라이언트와 서버 간의 상호작용은 대개 네트워크를 통한 메시지 전달을 통해 행해진다. 이런 MMORPG들에서는 게임의 내용이나 구성요소들이 지속적으로 진화해 가므로, 그에 따른 메시지들의 추가, 삭제, 또는 변경이 필요하다. 그러므로 게임 프로그래머들은 반복적으로 새로운 프로그램을 작성하거나 현재 있는 프로그램들을 수정해야 한다. 본 논문에서 소개하는 XML 기반의 메시지 정의 시스템은 게임의 상호작용을 위한 메시지들의 추가 및 변경을 용이하게 해준다. 이 시스템은 프로그래머들이 기존 메시지들을 변경하거나 새로운 메시지들을 추가하는데 드는 시간을 상당히 줄여 줄 뿐만 아니라 프로그래밍 버그 발생의 부담을 상당히 줄여준다. 또한 이 시스템은 일반 클라이언트-서버 응용 프로그램들에서도 사용이 가능하다.

An XML-based Message Definition System for Massively Multiplayer Online Games

XueFeng Piao[†] · SamKweon Oh[††]

ABSTRACT

Client-server interaction in a massive multi-player online role-playing game(MMORPG) is usually made via network-message passing. Since the game contents and elements in such MMORPGs are consistently evolved, messages need to be added, deleted, or modified accordingly. Therefore, game programmers are repeatedly required to write new programs or at least modify existing programs. The XML-based message definition system introduced in this paper facilitates the addition and modification of messages for game interaction ; it not only allows programmers to save a considerable amount of time for modifying existing messages or adding new messages but also relieves them of a serious burden of programming bugs. In addition, this system can be used for general client-server applications.

키워드 : XML(eXtensible Markup Language), 메시지 정의 시스템(Message Definition System), MMORPG(Massively Multiplayer Online Game)

1. 서 론

1980년 보드(board)형 롤-플레잉(role-playing)기능과 채팅(chatting)을 결합한 MUD(Multi-User Dungeon) 게임으로부터 시작된 온라인 게임은 사용자가 네트워크를 통해 서버에 접속하여 진행하는 형태의 게임으로서, 1996년 MUD에 그래픽 출력 기능을 추가한 형태인 MUG(Multi-User Graphics) 게임이 상용화되면서 그 시장이 본격적으로 형성되기 시작하였다[1]. 이러한 MUG 게임은 지리적으로 분산된 수많은 사용자들이 인터넷을 통해 동시에 즐길 수 있는 MMORPG와 같은 대규모 온라인 게임으로 발전하였다[2].

온라인 게임은 오프라인 게임과는 달리 상호작용성과 내용확장성을 가진다. 상호작용성은 클라이언트와 서버 간에 정보가 양방향으로 전송되는 성질을 의미하고, 내용확장성은 게임 내용이 지속적으로 변화하여 새로운 정보가 계속하여 추가되는 성질을 의미한다[1]. 대규모 온라인 게임은 수많은 메시지들을 정의하여 사용함으로써 클라이언트-서버 간의 상호작용이 가능하고, 지속적으로 메시지를 추가하거나 변경함으로써 게임 내용의 확장이 가능하다. 메시지의 송수신으로 이루어지는 클라이언트-서버간의 상호작용은 일반적으로 다음과 같다. 클라이언트의 전송 모듈에서 서버에 요청 메시지를 전송하면, 서버의 유효성 검사 모듈은 수신된 메시지의 아이디(identity), 길이 등의 값이 유효한지를 검사하고, 유효하면 이에 대한 처리를 하여 처리 결과 메시지를 클라이언트에게 전송하고, 그렇지 않으면 오류 메시지를 전송한다. 이와 같이 클라이언트-서버 간의 상호작용이 이루어지기 위해 개발자들은 먼저 게임 응용 프로그램 계층의 프로토콜, 즉 메시지의 형식을 정의하고, 정의한 형식에 맞게 메시지 아이디나 길이 등의 값을 정한다. 그

※ 이 연구는 2003년 호서대학교 특별학술연구비 지원으로 수행되었음.
† 준 회 원 : 서울대학교 대학원 전기·컴퓨터공학부
†† 종신회원 : 호서대학교 컴퓨터공학부 교수
논문접수 : 2003년 9월 18일, 심사완료 : 2004년 6월 3일

리고 메시지를 전송하는 전송 모듈과 수신된 메시지가 유효한지를 검사하는 유효성 검사 모듈을 개발한다[3]. 그러나 게임 내용의 확장으로 인해 메시지를 추가하거나 변경해야 할 경우, 개발자들은 메시지들의 추가·변경을 위한 문서를 작성하고, 그 문서에 근거하여 전송 모듈과 유효성 검사 모듈을 수정하여 기존 모듈을 수정된 모듈로 대치해야 한다. 이 모든 작업은 개발자들의 수작업으로 진행되므로 오류 발생 확률이 높아지고, 많은 시간 낭비를 초래할 수 있다.

1998년 W3C(World Wide Web Consortium) 권고안으로 발표된 XML(eXtensible Markup Language)은 SGML(Standard Generalized Markup Language)을 기반으로 한 마크업(markup)언어로서 사용자가 임의로 태그를 정의하고 확장할 수 있다[4]. XML의 구성요소인 XML 스키마(schema)는 데이터가 마크업되는 방식으로서 XML 문서 안에서 사용되는 태그들의 구조와 속성, 그리고 어떤 값을 가질 수 있는지에 대한 규정이다[5]. 그러므로 XML은 데이터의 내용뿐만 아니라 구조까지도 포함할 수 있어 데이터를 표현하고 저장하고 교환하는 분야에서 다양하게 활용되고 있다.

본 논문은 먼저, 데이터의 구조와 내용 표현이 동시에 가능한 XML을 이용하여 메시지를 표현하고, 이를 기반으로 대규모 온라인 게임에서 지속적으로 반복되는 메시지의 추가·변경 작업을 용이하게 해주는 XML 기반의 메시지 정의 시스템을 제안한다.

이 시스템은 메시지 정의 문서, 파싱 모듈, 유효성 검사 모듈의 세 부분으로 구성되며 메시지 정의 문서는 메시지 형식을 정의하는 스키마 문서와 이 문서에 정의된 태그들을 사용하여 작성한 메시지 내용 정의 문서로 구성된다. 파싱 모듈은 이 스키마 문서와 내용 문서를 입력으로 받아서 메시지의 구조에 맞는 구조체 파일과 메시지의 헤더(header)와 바디(body) 정보를 담고 있는 정보 파일들을 생성한다. 생성된 파일들은 라이브러리로 존재하는 유효성 검사 모듈내에 포함되며, 게임 응용 프로그램은 이 라이브러리에서 제공하는 함수들을 호출하여 클라이언트-서버간에 송수신되는 메시지들의 유효성을 검사한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 살펴보고, 3장에서는 전체 시스템 모델을 제시한다. 4장에서는 대규모 온라인 게임에서 일반적으로 사용되는 메시지들을 분류하여 다양한 형식의 메시지들을 표현할 수 있는 태그들을 정의하고, 5장에서는 파싱 모듈에 대해 설명한다. 6장에서는 송수신되는 메시지의 유효성 검사에 대해 설명하고, 7장에서는 구현내용을 설명하고, 8장에서 결론을 맺고 향후 연구계획을 제시한다.

2. 관련 연구

정보통신망의 구축과 응용산업 발전의 중요성이 대두되면서 온라인 게임은 중요한 핵심 산업으로 주목받기 시작했다. 따라서 게임 그래픽, 게임 엔진 및 게임 서버 기술과 같은 관련 기술에 대한 연구는 세계적으로 광범위하게 진행되고 있으나, 대규모 온라인 게임에서 지속적인 게임 내용의 확장으로 인한 메시지 추가·변경의 반복적인 작업 문제를 해결하거나 완화하기 위한 연구는 찾아보기 어렵다[6, 7].

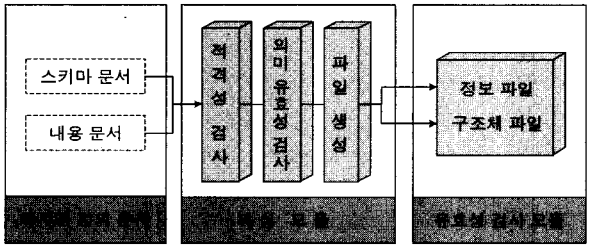
XML은 데이터의 구조와 내용을 동시에 표현할 수 있는 마크업 언어로서, 데이터의 내용을 표현하는 공통 형식으로 그 활용분야가 다양하다. 이를 바탕으로 현재 업종별 비즈니스 표준을 목표로 XML을 이용한 표준화가 진행되고 있으며, 주로 데이터형이나 비즈니스 규칙들을 XML로써 정의하고 있다. 대표적으로 전자상거래를 위한 ebXML, 신문업계의 NewsML 등의 각 비즈니스 영역에서뿐만 아니라 수학식 표현을 위한 MathML, 휴대폰을 위한 WML, 음성 표현을 위한 VoiceXML 등의 많은 업종 및 분야에서 XML을 기반으로 한 표준들이 만들어지고 있다[8, 9].

〈표 1〉 XML 응용 예

XML	설 명
ebXML	기업간의 비즈니스 정보 교환을 위한 언어
NewsML	다양한 수단과 형태의 뉴스들을 미디어에 무관하게 표현하기 위한 언어
MathML	수학 기호나 식 표시를 위한 언어
WML	무선 이동 단말기에서 플랫폼-독립적인 웹-페이지의 디스플레이를 위한 언어
VoiceXML	음성합성을 위한 언어

3. 시스템 모델

본 논문에서 제안하는 XML 기반의 메시지 정의 시스템은 (그림 1)에서 보여주는 것처럼 메시지 정의 문서, 파싱 모듈, 유효성 검사 모듈의 세 부분으로 구성된다.



(그림 1) 시스템 모델

메시지 정의 문서는 대규모 온라인 게임에서 사용되는 메시지들을 정의하는 문서로서, 메시지 형식을 정의하는 스키마 문서와 이 스키마 문서에서 정의한 태그들을 사용하여 작성되는 메시지 내용 정의 문서로 구성된다.

실행 프로그램으로 존재하는 파싱 모듈은 XML로 작성된 메시지 정의 문서를 파싱하여, 문서의 적격성(well-formedness) 검사와 의미 유효성(semantics validation) 검사를 하고, 유효성이 검증된 문서이면 메시지들의 헤더와 바디의 정보를 담고 있는 정보 파일들과 메시지의 구조에 맞는 구조체 파일들을 생성한다. 생성된 파일들은 유효성 검사 모듈 내에 포함되며, 게임 응용 프로그램은 라이브러리로 존재하는 유효성 검사 모듈을 이용하여 클라이언트와 서버간에 송수신되는 메시지들의 아이디나 길이 등의 값이 유효한지를 검사한다.

게임 내용의 확장으로 인해 메시지를 추가하거나 변경해

야 할 경우, 개발자들은 스키마 문서에서 정의한 태그들을 사용하여 추가·변경된 메시지를 포함하는 메시지의 내용을 정의하는 XML 문서를 작성한다. 그리고 이 문서를 파싱하여 메시지 구조체 파일과 정보 파일들을 생성한다. 개발자들은 게임 응용 프로그램에서 기존에 사용하던 구조체 파일과 정보 파일들을 새로 생성된 파일들로 대치함으로써 메시지의 추가·변경을 쉽게 할 수 있다.

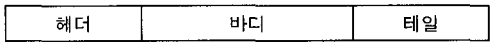
4. 메시지 정의

4.1 메시지 분류 및 구조

클라이언트-서버 기반의 대규모 온라인 게임은 기능별로 여러 대의 서버를 두는 분산구조를 가지는 것이 일반적이다. 분산구조 온라인 게임에서 사용되는 메시지는 네트워크로 전송되는 정보에 따라 다음과 같이 구분할 수 있다.

- 접속 메시지 : 클라이언트의 게임 서버 접속을 위한 메시지들
- 클라이언트 요청 메시지 : 게임진행 중에 발생하는 사용자 요청을 전송하는 메시지들
- 서버 응답 메시지 : 사용자 요청 명령에 따른 처리 결과를 사용자에게 전송하는 메시지들
- 맵(map) 데이터 갱신 메시지 : 사용자가 속한 가상공간(맵) 데이터들의 갱신을 위한 메시지들
- 동기화 메시지 : 분산구조의 대규모 온라인 게임 시스템의 동기화를 위한 메시지들
- 객체 이전 메시지 : 게임의 진행 중 사용자를 관리하는 가상공간 객체의 이전으로 인해 발생하는 사용자 관리 정보 전송 메시지들

이런 메시지들은 일반적으로 (그림 2)와 같이 헤더, 바디, 테일(tail)로 구성된다[3]. 헤더는 메시지에 공통으로 들어가며 일반적으로 메시지를 구분하기 위한 정보와 바디의 길이를 나타내는 정보들로 구성된다. 바디는 데이터의 내용을 담고 있는 필드들을 가지며 필드들의 구조에 따라 나열 구조, 반복 구조, 혼합 구조, 바디가 없는 구조로 구분한다. 테일은 일반적으로 수신 데이터 정보의 오류를 조사하기 위한 체크섬(checksum) 필드와 메시지의 끝을 나타내는 필드들로 구성된다.

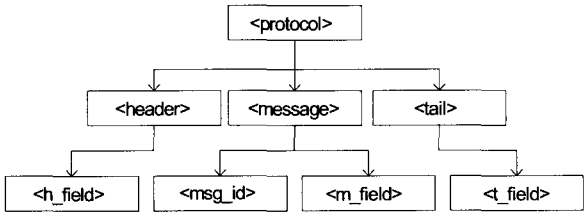


(그림 2) 메시지 일반 형식

- 나열 구조 : 바디가 필드들의 나열인 구조
- 반복 구조 : 여러 필드가 한 그룹을 형성하며, 바디는 이런 그룹들의 반복인 구조
- 혼합 구조 : 나열 구조와 반복 구조가 혼합된 구조
- 바디가 없는 구조 : 메시지에 바디가 없는 구조

4.2 메시지 정의

스키마는 XML 문서가 마크업되는 방식에 대한 정의로서 메시지 형식 정의 스키마 문서는 메시지 내용 정의 XML 문서에서 사용하는 태그들에 대한 정의이다. 정의한 태그들은 메시지의 바디 구조에 따른 다양한 형식을 정의할 수 있어야 한다. 이를 고려하여 태그들을 (그림 3)과 같은 트리 구조로 정의하였다.



(그림 3) 태그 구조

<protocol>태그는 메시지 내용 정의 문서의 최상위 요소이며 문서의 시작과 끝을 나타낸다. 메시지의 헤더 부분을 나타내는 <header>태그와 메시지의 바디 부분을 나타내는 <message>태그, 그리고 테일 부분을 나타내는 <tail>태그를 하위 요소로 가진다. <header>태그는 헤더 부분의 각 필드들을 정의하는 <h_field>를 하위 요소로 가지며, <message>태그는 메시지를 구분하는 식별자를 정의하는 <msg_id>태그와 바디 부분의 각 필드를 정의하는 <m_field>를 하위 요소로 가진다. 그리고 <t_field>태그는 <tail>태그를 하위 요소를 가지며 마찬가지로 테일 부분의 각 필드들을 정의한다. <표 2>는 정의된 태그들의 속성과 데이터형을 보여준다.

메시지의 내용을 정의하는 XML 문서는 <표 2>에서 정의한 태그들을 이용하여 작성된다. 작성된 메시지 내용 문서의 예는 (그림 4)와 같다.

〈표 2〉 태그 속성 및 데이터형

태 그	속 성	데이터형	설 명
protocol	version	string	메시지 정의 문서를 관리하기 위한 버전
	desc	string	메시지 정의 문서에 대한 설명
header	name	string	메시지 헤더의 이름
	desc	string	헤더에 대한 설명
h_field	name	string	헤더 필드의 이름
	type	string	헤더 필드의 데이터 형
	size	unsigned short	필드 데이터가 배열인 경우 배열의 크기, 배열이 아닐 경우 생략
	desc	string	헤더 필드에 대한 설명
message	name	string	메시지 바디의 이름
	desc	string	메시지 바디에 대한 설명
msg_id	name	string	메시지 이름
	value	Hex binary	메시지 아이디 값
	desc	string	메시지에 대한 설명
m_field	name	string	바디 필드의 이름
	type	string	바디 필드의 데이터 형
	size	unsigned short	바디 필드 데이터가 배열인 경우 배열의 크기, 배열이 아닐 경우 생략
	loop	string	반복구조에서 반복을 구분하는 필드이름 지정, 반복을 구분하는 필드가 없을 경우 생략
	desc	string	바디 필드에 대한 설명
tail	name	string	메시지 테일 부분의 이름
	desc	string	메시지 테일 부분에 대한 설명
t_field	name	string	테일 필드의 이름
	type	string	테일 필드의 데이터 형
	size	unsigned short	필드 데이터가 배열인 경우 배열의 크기, 배열이 아닐 경우 생략
	desc	string	테일 필드에 대한 설명

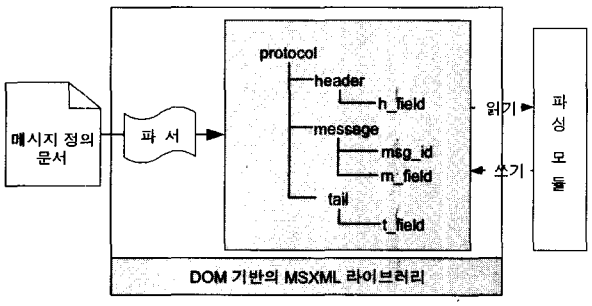
```
<protocol xmlns = "message" version = "1.0" desc = "Message Definition
Document">
  <header name = "HEADER" desc = "Protocol Message Header">
    <h_field type = "WORD" name = "wMsg" desc = "Message ID Field"/>
     <h_field type = "WORD" name = "wLength" desc = "PDU Length
     Field"/>
  </header>
  <message name = "CSP_REQ_LOGIN">
    <msg_id value = "1001" desc = "로그인 요청 메시지"/>
    <m_field type = "char" size = "18" name = "szCharID" desc = "Char ID
    Field"/>
    <m_field type = "char" size = "18" name = "szPwd" desc = "Password
    Field"/>
  </message>
  <message name = "SCP_RESP_LOGIN">
    <msg_id value = "9001" desc = "로그인 응답 메시지"/>
    <m_field type = "BYTE" name = "byResult" desc = "Result Field"/>
    <m_field type = "int" name = "nClientKey" desc = "Client Key"/>
  </message>
</protocol>
```

(그림 4) 메시지 내용 문서의 예

5. 파싱 모듈

5.1 DOM과 SAX

DOM(Document Object Model)과 SAX(Simple API for XML)는 XML 문서의 내용을 액세스(access)하고 조작하기 위한 방식이다. DOM과 SAX는 W3C에서 권고안으로 제안된 것으로써, XML 문서를 다루기 위한 API들이다. DOM 파서는 XML 문서를 파싱한 후, 트리 형태의 자료구조를 만들어 메모리에 문서의 요소들을 적재한다. 사용자는 만들어진 트리 구조의 객체 모델을 통해서 실제 문서의 요소들을 수정, 삭제, 추가 등의 작업을 할 수 있다. SAX는 XML을 위한 이벤트 기반의 API이다. 메모리에 트리 구조를 만드는 DOM과는 달리, SAX는 XML 문서에서 시작 태그, 종료 태그, 속성 등을 만날 때 특정한 메소드를 호출한다. 데이터를 액세스하기 위해 핸들러(handler) 함수를 SAX 파서와 함께 등록한 후, 파서가 XML 요소를 만나면 그 요소에 해당되는 함수를 호출하는 구조를 가진다[9-11].



(그림 5) DOM 기반의 MSXML 파서

DOM은 XML 문서의 데이터를 트리 구조로 메모리에 적재하여 사용하므로 액세스하기 쉬우며, SAX는 이벤트 기반으로 문서를 액세스하므로 성능은 좋으나 응용 프로그램에서의 많은 추가 작업을 필요로 하다. XML 기반의 메시지 정의 시스템은 메시지 정의 문서의 액세스 성능이 문제가 되지 않으므로 마이크로소프트에서 제공하는 DOM기반의 MSXML 파서를 사용하였다. 메시지 정의 XML 문서는 (그림 5)에서 보여주는 것과 같이 MSXML 파서에 의해 파싱되어 메모리에 계층적인 트리 구조로 표현된다. 이 트리 구조에는 메시지들의 헤더와 바디, 그리고 각 필드들의 정보 등이 포함된다. 그리고 파싱 모듈은 DOM API를 사용하여 메모리에 적재되어 있는 이 정보들을 읽어들여 메시지의 구조에 맞는 구조체 파일과 헤더와 바디의 정보들을 담고 있는 정보 파일들을 생성한다.

5.2 파 싱

메시지 정의 문서의 파싱은 MSXML DOM 파서를 사용하여 진행되며 적격성(well-formedness) 검사, 의미 유효성(semantic validation) 검사, 그리고 파일 생성(file generation)의 세 과정을 거친다. 적격성 검사에서는 메시지 정의 문서의 모든 요소들이 XML 1.0 명세서에 맞게 시작 태그와 끝 태그를 가지고 있는지, 중첩 규칙을 위반하지 않았는지 등의 기본적인 사항들을 검사하고, 의미 유효성 검사에서는 메시지의 형식을 표현하는 스키마 문서에서 정의한 태그들의 구조, 속성, 데이터형 등의 제약 조건에 따라 메시지 내용 정의 문서가 작성됐는지를 검사한다.

파일 생성은 시스템의 파싱 모듈에서 DOM을 통해 메시지 정의 문서를 액세스하여 얻은 데이터들에 근거하여 메시지의 구조에 맞는 구조체 파일과 헤더 정보와 바디 정보를 담고 있는 정보파일들을 생성하는 것을 뜻한다. 정보파일에는 메시지들의 아이디나 길이 등의 값이 저장되어 있으며, 유효성 검사 모듈은 이런 값들에 근거하여 수신된 메시지들이 정의된 값과 일치하는지를 검사한다.

6. 유효성 검사 모듈

메시지 유효성 검사 모듈은 파싱 모듈에서 생성된 파일들에 근거하여 클라이언트-서버 간에 송수신되는 메시지들이 유효한지를 검사하는 라이브러리의 형태로 존재한다. 이 모듈은 (그림 6)과 같이 선행 처리기 #define 지정어를 사용하여 파싱 모듈에서 생성된 파일들을 포함하며, 메시지의 길이를 검사하는 CheckMsgLength(int_nMsgID, int_nMsgLength) 같은 외부호출 함수들을 제공한다.

```
#define          STRUCT_FILE        PMStruct.h
#define          MSGINFO_FILE       PMMsg.pmi
#define          HDINFO_FILE        PMBody.pmi
... ... ...
int CheckMsgLength(int _nMsgID, int _nMsgLength)
{
      //_nMsgID는 수신된 메시지의 아이디
      //_nMsgLength는 수신된 메시지의 길이
      READ(MSGINFO_FILE)
      READ(HDINFO_FILE)
      if (FIND(_nMsgID) == true) {
         if (MsgLength == _nMsgLength)
            return 1 ;
         else
            return 0 ;
      }
      else
            return 0 ;
}
```

(그림 6) 선행처리 및 제공되는 함수의 예

게임 응용 프로그램은 이 함수들을 호출하여 수신된 메시지의 아이디나 길이 등의 값들을 파라미터(parameter) 형태로 유효성 검사 모듈에게 전달하면 유효성 검사 모듈은 이 값을 정보 파일에 저장된 값과 비교하여 일치하는지를 검사한다. 그리고 그 결과를 응용 프로그램에게 반환한다. 그러므로 메시지의 추가 및 변경은 새로 추가·변경한 메시지들을 포함한 구조체 파일과 정보 파일들의 갱신만으로도 이루어진다.

7. 구 현

7.1 구현 예

본 절에서는 대규모 온라인 게임인 "탄트라(Tantra)"에서 XML 기반의 메시지 정의 시스템의 적용 결과를 제시한다.

(그림 7)은 XML 메시지 정의 문서로 (그림 1)의 파싱 모듈의 적격성 검사, 의미 유효성 검사, 파일 생성을 거쳐 (그림 8)과 같은 3가지의 파일을 생성한다.

```
<?xml version = "1.0" encoding = "utf-8" ?>
 <Protocol xmlns = "MDS.xsd" Version = "1.0" Desc = "XML">
  <Message Type = "HEAD">
    <Body Name = "HEADER" Desc = " ">
     <Field Type = "char" Name = "cMsgID" Desc = "Message ID
     Field"/>
     <Field Type = "short" Name = "sLength" Desc = "Message
     Length Field"/>
    </Body>
  </Message>
  <Message Type = "BODY" Desc = "Server Info Structure">
    <Body Name = "SERVER_INFO" Desc = "Server Info">
     <Field Type = "int" Name = "nSvrID"/>
     <Field Type = "char" Size = "16" Name = "szSvrIP"/>
     <Field Type = "UINT" Name = "uSvrPort"/>
    </Body>
  </Message>
  <Message Type = "MSG" Desc = "Server List Request
  Message">
   <MsgType Name = "REQ_SERVERLIST" Value = "0x10" Desc
   = " "/>
     <Body Name = "SERVER_LIST" Desc = "Server List
     Structure">
      <Field Type = "int" Name = "nServerNum" CountType =
      "SERVER_INFO" Desc = " "/>
      <Field Type = "SERVER_INFO *" Name = "ServerInfo"
      Desc = " "/>
     </Body>
  </Message>
</Protocol>
```

(그림 7) XML 메시지 정의 문서

(그림 9)는 파싱 결과에 의해 생성된 파일을 근거로 (그림 6)과 같은 유효성 검사 모듈의 검사함수들을 사용하여 수신된 메시지의 유효성을 검사하고, 그에 따라 동작하는 것을 보여주고 있다.

Structure File

```
# ifndef __SOURCECODE_H__
# define __SOURCECODE_H__
// Version 1.0
// Structure 문서
#define REQ_SERVERLIST   0x10

typedef struct _S_HEADER
{
   char  cMsgID;
   shrot  sLength;
}S_HEADER;

typedef struct _S_SERVER_INFO
{
   int     nSvrID;
   char   szSvrIP[16];
   UINT  uSvrPort;
}S_SERVER_INFO;

typedef struct _S_SERVER_LIST
{
   int     nServerNum;
   S_SERVER_INFO* ServerInfo;
}S_SERVER_LIST;

#endif
```

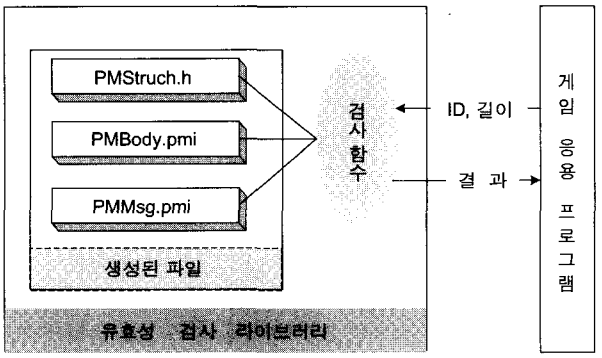
Header Info File

```
Header 정보 문서 1.0

0x10 3 4 24
```

Message Info File

```
Message Info 문서 1.0

HEADER    1 2 1 2
SERVER_INFO 2 4 4 1 1 5
SERVER_LIST  3 2 4 s2
```

(그림 8) 파싱 결과



(그림 9) 유효성 검사

7.2 평가 분석

대규모 온라인 게임에서는 인터넷을 기반으로 서버에서 서버(server to server), 서버에서 클라이언트(server to client), 그리고 클라이언트에서 서버(client to server) 등 다양한 통신이 이루어진다. 따라서 각 통신에서 사용되는 많은 양의 메시지들을 효율적으로 정의하고 관리하기 위해서는 많은 시간과 노력이 필요하다.

또한, 온라인 게임이라는 특수성으로 인해 게임은 계속해서 수정되고, 진보하게 된다. 이러한 기능 추가 및 수정은 곧바로 메시지 포맷과 메시지 검사 라이브러리, 그리고 관련문서의 수정을 동반하게 된다. 실 예로 "A"사의 A게임 개발 당시 메시지 포맷에 관련하여 많은 시간과 노력이 소비되었고, 그 결과 게임개발 시간의 지연이 초래되었다.

본 논문에서 제안한 XML 기반의 메시지 정의 시스템은 실제 "탄트라(Tantra)" 게임 개발 당시 필요여부에 의해 적용한 시스템으로 이전의 게임개발에 비해 다음과 같은 몇 가지 개선효과를 얻을 수 있었다.

- XML 메시지 정의 문서의 수정만으로 시스템에 필요한 3가지 파일이 자동으로 생성되어, 작업의 효율성이 높아지고, 메시지의 추가 및 수정이 용이했다.
- 표준화된 메시징 시스템을 이용하여 게임 메시지 작성 시간 단축 및 노동력이 크게 감소되었다.
- 게임 서버와 클라이언트 개발시 개발자의 실수로 인한 메시지 프로토콜 오류를 미연에 방지할 수 있었다.
- 메시지 분석과 수정시 필요한 문서를 자동으로 생성되어 작업의 효율성을 높였다.
- 런 타임시 메시지 오류 확인 및 유효성 검사를 지원하여 보다 안정적인 통신이 가능했다.

8. 결론 및 향후 연구

대규모 온라인 게임에서 지속적으로 반복되는 메시지 추가 및 변경 작업은 많은 시간을 낭비할 수 있으며, 개발자들의 지속적인 반복 작업으로 인한 게임 응용 프로그램의 오류 발생확률도 높아지게 되므로 결국 안정적인 서비스를 제공하지 못할 수도 있다.

본 논문은 이런 문제점을 해결하거나 완화하기 위해 메시지의 추가·변경을 용이하게 해주는 XML 기반의 메시지 정의 시스템을 제안하였다. 게임 개발자들은 본 시스템에서 제공하는 태그들을 사용하여 게임에서 사용되는 메시지들을 정의하는 XML 문서를 작성하고, 작성한 문서를 파싱 모듈로써 파싱하여 구조체 파일과 정보 파일들을 생성한다. 게임 응용 프로그램은 파싱 모듈에서 생성된 파일들을 포함한 유효성 검사 모듈 라이브러리를 사용하여 클라이언트와 서버 간에 송수신되는 메시지들의 유효성을 검사한다. 게임 내용의 확장으로 인해 메시지를 추가하거나 변경해야 할 경우, 개발자들을 메시지 정의 문서를 수정하여 파싱 모듈로써 파싱한다. 그리고 생성된 구조체 파일과 정보 파일들로 기존 파일들을 대치하는 것으로 메시지의 추가·변경을 할 수 있다.

본 논문에서 제안하는 XML 기반의 메시지 정의 시스템은 대규모 온라인 게임뿐만 아니라 일반 클라이언트-서버 응용분야에서도 사용될 수 있는 범용 시스템이며, 데이터의 구조와 내용이 분리된 XML의 장점을 이용했으므로 다양한 형식의 메시지 정의도 쉽게 할 수 있다.

향후 연구에서는 본 시스템을 보완하여 메시지의 아이디나 길이 등의 정보뿐만 아니라 메시지들의 순서(sequence) 검사도 가능하고, LAN 환경뿐만 아니라 이동 통신환경에서도 사용 가능한 메시지 정의 시스템에 대해 연구할 계획이다.

참 고 문 헌

[1] 한국게임산업개발원, "게임백서", 한국게임산업개발원, pp. 564-574, 2003.

[2] 최석우, "리니지 개발 사례", 정보처리학회지, 제9권 제3호, pp.85-90, 2002.

[3] 박광우, 강우경, 진용철, "Visual C++.Net Programming Bible", 삼양미디어, pp.80-101, 2002.

[4] 김창수, 정희경, "XML 응용 개발환경", 정보과학회지, 제19권 제1호, pp.15-23, 2001.

[5] 황병연, 김 연, "XML 스키마 발전 동향", 정보처리학회지, 제8권 제3호, pp.3-9, May, 2001.

[6] 이남재, 곽훈성, "완전한 3차원을 지원하는 온라인 RPG를 위한 맵 관리 방법", 정보처리학회논문지B, 제9-B권 제6호, pp. 863-868, 2002.

[7] J. Smed, T. Kaukoranta, and H. Hakonen. "Aspe cts of networking in multiplayer computer gam es," In International Conference on Application and Development of Computer Games in the 21st Century, Hong Kong, November, 2001.

[8] 최영근, 정계동, "XML How To Program", Prentice Hall, 2001.

[9] 이강찬, 손 홍, 박기식, "XML 표준화 동향", 정보과학회지, 제19권 제1호, pp.6-14, 2001.

[10] 이호경, 송순원, "XMLgo.net과 함께 하는 Opening XML", 컴스페이스 구민사, 2002.

[11] W3C, "MSXML 파서", http://www.microsoft.com/korea/msdn.

박 학 봉

e-mail : hbpak76@snu.ac.kr

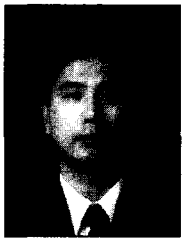
1998년 중국 길림성 연변대학교 법학과 (학사)

2002년 호서대학교 컴퓨터공학부(공학사)

2004년 호서대학교 대학원 컴퓨터공학부 (공학석사)

2004년~현재 서울대학교 전기·컴퓨터공학부 대학원 박사과정

관심분야 : Operating System, Distributed System



오 삼 권

e-mail : ohsk@office.hoseo.ac.kr

1980년 한국항공대학교 항공전자공학과 (학사)

1986년 University of South Florida 컴퓨터과학 및 공학(공학석사)

1994년 Queen's University, 컴퓨터 및 정보과학(공학박사)

1980년~1984년 삼성전자 통신연구소

1994년~1995년 한국전자통신연구원

1995년~현재 호서대학교 컴퓨터공학부 부교수

관심분야 : Distributed System, Computer Game, Communication Protocol, Fault Tolerance