NaCl 실행환경을 위한 동기식 POSIX 소켓의 비동기식 웹 소켓으로의 인터페이스 변환 기법

*오영섭[0], *성백재, **박일우, **권재욱, *박찬익
*포항공과대학교 컴퓨터공학과, **삼성전자
youngsup@postech.ac.kr, jays@postech.ac.kr, ilwoo.park@samsung.com,
jook.kwon@samsung.com, cipark@postech.ac.kr

Translating Synchronous POSIX socket calls into Asynchronous WebSocket calls in NaCl Runtime Environment

*Youngsup Oh[0], *Baegjae Sung, **Ilwoo Park, **Jaeook Kwon, *Chanik Park
*Department of Computer Science and Engineering, POSTECH
**Samsung Electronics

요 약

NaCl 실행 환경에서 사용하는 비동기식 웹 소켓은 기존 동기식 POSIX 소켓과 실행 흐름에 있어 차이점이 존재한다. 동기식 POSIX 소켓의 경우 메시지 통신 요청 시 해당 요청이 모두 끝날 때까지 프로그램이 대기하였다가 다음 명령어를 수행하는 반면, 비동기식 웹 소켓의 경우 해당 요청이 끝나기 전에 다음 명령어가 수행될 수 있고, 요청이 끝나면 callback 함수를 호출하여 처리하게 된다. 이는 기존에 개발된 수 많은 POSIX 소켓 프로그램들이 웹 소켓을 사용 하기 위해 많은 수정을 필요로 하게 한다. 이 연구에서는 NaCl 실행환경에서 POSIX 소켓 인터페이스를 웹 소켓 인터페이스로 변환하는 기법을 제시한다.

1. 서 론

웹 애플리케이션(이하 앱)은 웹 브라우저를 이용하여 동작하는 응용 소프트웨어로, PC 혹은 모바일 기기에 배포하여 설치하지 않아도 웹 브라우저를 통해 이용할 수 있다는 장점이 있다. 비록 웹 앱이 웹 브라우저에 종속적이지만, 플랫폼에 종속적이지 않다는 점과 기존 단점이었던 느린 실행 속도와 단말 자원에 대한 접근 불가 문제가 해결됨에 따라 그 가치가 더욱 증가하였다.

최근 Microsoft Office Web Apps[1]과 같이 설치하여 사용하던 Word와 Excel, Power Point 등이 웹 앱으로 등장하여, 많은 사용자들이 프로그램 설치 없이 웹 브라우저를 통해 문서 작업 등을 할 수 있게 되었다. 이러한 웹 앱은 HTML, CSS, 자바스크립트, PHP, Ajax 등의 기술을 사용해 개발된다. 따라서 기존에 C나 C++, 자바 등으로 개발된 응용 프로그램들을 웹 앱으로 배포하기 위해서는, 이러한 기술들을 사용해 새롭게 개발 해야 하는 추가비용이 발생한다.

구글의 오픈 소스 프로젝트인 NativeClient(NaCl)는 크롬 브라우저에서 컴파일 된 네이티브 코드를 수행할 수 있는 환경을 제공해준다. NaCl을 이용하면 웹 앱을 개발할 때 자바스크립트가 아니어도 다른 언어를 사용하여 개발할 수 있다. 따라서 기존에 개발된 많은 프로그램들이 큰 수정 없이 웹 브라우저에서 동작할 수 있게 해준다. 현재 NaCl에서는 C와 C++를 지원해주고 있다[2].

하지만 기존의 소켓 통신 프로그램을 NaCl 환경에서 수행하기 위해서는 한가지 주의해야 할 점이 있다. NaCl 실행 환경에서는 기존의 POSIX 소켓 통신이 아닌 웹 브라우저에서 지원하는 웹 소켓을 사용하는데, 동기식으로 동작하는 기존 POSIX 소켓 통신과 달리 웹 소켓의 경우 비동기식으로 동작하게 된다. 두 방식의 가장 큰 차이점은, 소켓 함수가 호출되었을 때 동기식의 경우 해당 요청이 완료된 후 결과값을 반환해주는 반면, 비동기식의 경우 우선적으로 요청에 대한 확인 응답을 해준 뒤 실제 요청이 완료되면 요청에 적합한 callback 함수를 호출하여 결과값을 알려주는 구조이다. 이러한 차이로 인해 receive 함수를 사용하는 부분에서 문제가 발생할 수 있다. POSIX 소켓 통신 프로그램의 경우 receive 함수를 통해 메시지를 받고, 해당 메시지를 사용하여 다음 명령어들이 수행된다. 이 경우 단순히 웹 소켓을 사용하도록 바꾸게 된다면 receive 함수 호출 후 실제 메시지를 받기 전에 메시지를 받았다고 착각하여 다음 명령어들이 수행될 수 있다. 이러한 문제로 인해 웹 소켓을 사용하기 위해서는 프로그램의 코드의 많은 부분이 수정되어야 한다. 이는 기존의 코드를 적은 변환 비용으로 사용할 수 있다는 이점을 감소시킨다.

이 연구는 프로그램의 코드를 크게 수정하지 않고, POSIX 소켓 통신 프로그램을 웹 소켓을 사용하여 NaCl 실행 환경에서 정상적으로 동작시키기 위한 인터페이스 변환 기법을 제시한다.

2. 시스템 구조

<그림 1>은 전체 시스템 구조와 웹 소켓의 recevice 함수의 실행 과정을 나타낸다. 웹 브라우저의 각 탭은

1331

주 쓰레드로 수행되며, 웹 앱이 포함된 페이지를 방문한 경우 별도의 앱 쓰레드를 생성하여 앱을 실행시키는 역할을 한다. 또한 웹 소켓이나 파일시스템, 그래픽카드 등에 대한 인터페이스를 제공한다. 인터페이스의 각 API들은 실제로 브라우저에 의해 처리된다.

2.1 인터페이스

앱 쓰레드에서 인터페이스를 통해 브라우저에게 어떠한 작업 (사용자로부터 입력을 받거나 웹 소켓에서 메시지를 받는 등)을 요청할 경우, 주 쓰레드는 해당 요청이 완료되었을 때 수행될 callback 함수를 함께 보낸다. 브라우저는 요청을 처리하기 전에 요청을 받았다는 의미의 응답메시지를 반환한다. 이후 요청이 완료되었을 경우 등록된 주 쓰레드의 callback 함수를 실행시킨다. 단순히 POSIX 소켓을 웹 소켓을 사용하도록 변경할 경우 브라우저를 통해 받은 첫 응답메시지가 요청이 완료되었다는 의미로 받아들일 수 있기 때문에 실제로 받지 않은 메시지에 대해 처리하려 하는 문제가 발생할 수 있다.

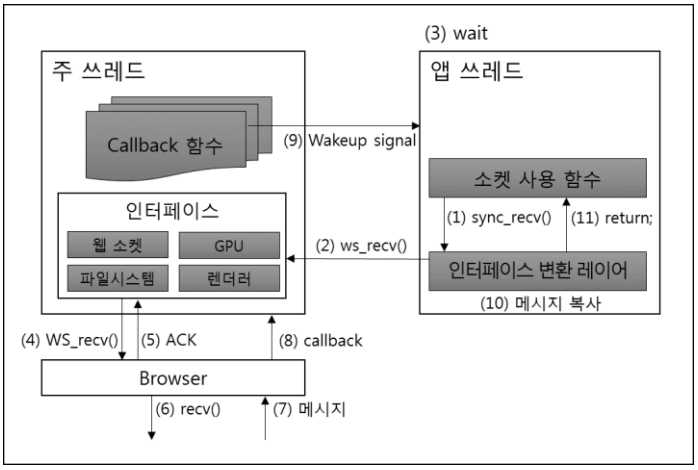
메시지를 받거나 사용자 입력과 같은 이벤트가 발생하면 브라우저에 의해 감지되고, 브라우저는 각 이벤트를 처리하기 위해 미리 등록된 주 쓰레드의 callback 함수를 호출한다. Callback 함수들은 이벤트를 직접 처리하거나 다른 함수를 호출하여 해당 이벤트를 처리한다.

2.2 인터페이스 변환 레이어

이 연구의 목적을 위해 추가된 레이어로, 웹 소켓을 사용하는 함수들로 동기식 POSIX 소켓 통신 프로그램이 정상적으로 수행될 수 있도록 해준다. 인터페이스 변환 레이어는 주 쓰레드의 인터페이스를 통해 브라우저에 요청 후 앱 쓰레드를 대기 모드로 만든다. 이후 요청이 처리되면 브라우저에 의해 실행된 callback 함수에서 대기중인 앱 쓰레드를 다시 시작시킨다. 이를 통해 동기식으로 수행되는 것과 같은 효과를 주어 요청이 완료되기 전에 다음 명령어를 수행하여 잘못된 행동을 하는 것을 막을 수 있다.

2.3 인터페이스 변환 레이어의 실행 흐름

앱 쓰레드에서 소켓 통신을 위해 인터페이스 변환 레이어의 receive 함수를 호출할 경우, 해당 함수는 주 쓰레드의 인터페이스를 사용하여 브라우저에 callback 함수와 함께 receive 요청을 보낸다. 요청을 보내고 나면 해당 앱 쓰레드는 대기 상태가 된다. 브라우저가 서버로부터 메시지를 받았을 경우 메시지를 주 쓰레드에 저장한 뒤 등록된 callback 함수를 실행시킨다. Callback 함수에 의해 대기 중이었던 앱 쓰레드가 다시 동작하게 되면 이전에 수행 되고 있던 함수에서 주 쓰레드에 저장된 메시지를 앱 쓰레드에 전달하고 해당 함수는 종료된다.



<그림 1> 인터페이스 변환 레이어가 추가된 전체 시스템 구조

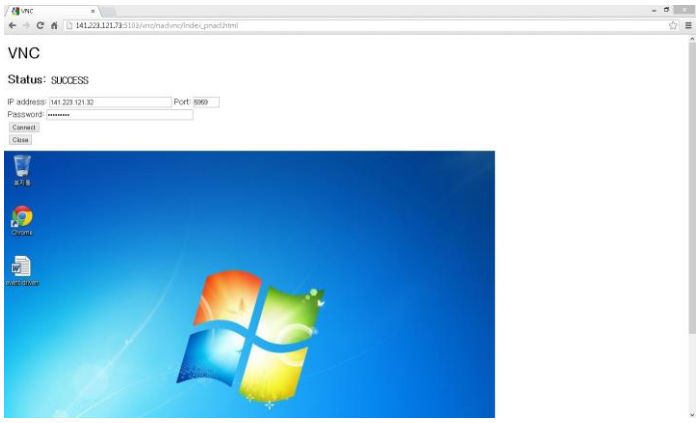
3. 구현

앱 쓰레드의 경우 대기 및 재개되어야 하기 때문에 pthread로 생성 된다. 이후 인터페이스 변환 레이어의 소켓 함수를 사용하게 되면, 각각에 맞는 웹 소켓 함수 호출 후, pthread_cond_wait 함수를 사용하여 웹 소켓 통신이 완료될 때까지 대기 상태가 된다. 웹 소켓 통신이 완료되어 주 쓰레드의 callback 함수가 호출되면, 해당 함수에서 pthread_cond_signal 함수를 사용하여 앱 쓰레드를 깨우게 되고, 앱 쓰레드는 이전에 멈추었던 인터페이스 변환 레이어 함수의 다음 부분을 수행한다. Receive 함수의 경우 주 쓰레드에 저장된 메시지를 앱 쓰레드의 메모리에 복사해주고 종료되게 된다.

이 연구에서는 소켓 통신 프로그램의 예시로 POSIX 소켓을 사용하는 SDL VNC 클라이언트[3]를 인터페이스 변환 레이어를 통해 웹 소켓을 사용하는 웹 앱으로 구현하였다. VNC 클라이언트 앱은 소켓 통신을 통해 서버의 화면을 받아 화면에 보여주고, 사용자의 입력을 서버로 보내는 역할을 한다. VNC 클라이언트의 경우 메시지를 받은 후 바로 해당 메시지를 처리하기 때문에, 비동기식 웹 소켓을 사용할 경우 정상적으로 수행되지 않는다. 따라서 인터페이스 변환 레이어를 사용하여 정상적으로 동작함을 보임으로써, 해당 레이어가 비동기식 웹 소켓을 사용하여 동기식 소켓통신과 같이 동작함을 확인 할 수 있다. SDL VNC 클라이언트를 NaCl 앱으로 변환하기 위해 naclports[4]에서 기존의 SDL 라이브러리를 NaCl에서 사용하도록 변환하여 제공하는 라이브러리(입출력을 브라우저를 통해 제어)를 사용하였으며, 기존 프로그램 코드 중 POSIX 소켓 함수 호출을 인터페이스 변환 레이어의 API 호출로 수정하였다. 또한 웹 페이지로부터 연결하려는 서버의 주소, 포트, 암호를 받아오는 부분을 추가하였다.

1332

4. 성능 평가



<그림 2> VNC 웹 앱 실행 화면

성능 평가를 위해, Intel Xeon 프로세스, 48GB 메모리의 서버와 Intel i5-M520 프로세서, 4GB 메모리의 클라이언트를 사용하였으며, 운영체제는 Ubuntu 12.04 64bit를 사용하였다. 웹 앱을 위해 크롬 브라우저 (v26.0.1410.64)를 사용하였다.

<그림 2>에서 인터페이스 변환 레이어를 사용한 VNC 클라이언트 앱이 정상적으로 작동 하여, 브라우저에 서버 화면을 보여주는 것을 볼 수 있으며, 인터페이스 변환 레이어가 비동기식 웹 소켓을 동기식 소켓과 같이 동작하도록 지원해줌을 확인할 수 있다.

<표 1>은 화면을 한번 갱신하는데 소요되는 시간으로 1024x768 크기의 화면은 128x64크기의 작은 블록으로 나뉘어 총 96개의 데이터를 받아 부분적으로 갱신한다. Receive는 화면 데이터와 인코딩에 관련된 추가적인 메시지 통신 시간이며, Decompression은 받은 화면 데이터를 압축 방식에 따라 압축 해제하는 시간, SDL 갱신은 실제 보여지는 SDL화면을 갱신하는데 소요되는 시간을 나타낸다. 실험 결과 화면 갱신의 가장 큰 오버헤드는 실제 SDL 화면을 갱신하는 부분으로, 네이티브에 비해 약 202배 정도의 성능 하락을 보여주고 있다. 이에 비해 소켓통신의 경우 네이티브에 비해 약 2.33배 느려졌지만 실제 수행시간이 0.013초로 짧기 때문에 성능에 크게 영향을 미치지는 않는다.

<표 1> 1024x768 Frame 갱신 시간

시간(초)	*NaCl VNC*	*Native VNC*
Receive	0.012580	0.005383
	0.084040	0.023351
SDL 갱신	1.846470	0.009135
Total	1.944410	0.039260

추가적으로 소켓 통신의 성능을 확인하기 위해 서버-클라이언트 간의 메시지 통신 테스트를 수행하였다. 서버 프로그램은 클라이언트와 연결이 되었을 때 8,192 bytes 메시지를 1,000 / 10,000번 보내며, 각 메시지는 클라이언트로부터 확인 메시지 (ACK)를 받은 뒤 다음 메시지를 보내게 된다. 클라이언트 프로그램은 성능 비교를 위해 순수 웹 소켓을 사용하는 앱과, 인터페이스 변환 레이어를 사용하는 앱으로 구성하였다.

실험 결과는 <표 2>에서 보여진다. 시간은 처음 메시지 전송 시점부터 마지막 메시지에 대한 확인 메시지를 받을 때까지의 소요 시간을 나타내며, 웹 소켓에 비해 13%, 6.8%의 오버헤드가 발생했다 (각 1,000개 / 10,000개의 메시지 통신). 해당 오버헤드는 pthread의 대기에 의한 lock 획득 및 스케줄링 문제와, 주 쓰레드로부터 앱 쓰레드로의 데이터 복사에 의한 것이며, 수행시간이 짧기 때문에 앱의 전체적인 성능에 큰 영향을 주지 않을 것이다.

<표 2> 1,000 / 10,000개의 메시지 통신 시 소요 시간

시간(초)	*웹 소켓*	*인터페이스 변환 레이어*
1,000개	2.832	3.207
10,000개	27.507	29.389

5. 결론 및 향후 연구

이 연구에서는 POSIX 소켓 통신 프로그램의 큰 코드 수정 없이 웹 소켓을 사용하여 정상적으로 동작할 수 있게 하기 위해 인터페이스 변환 레이어를 제시하였다.

인터페이스 변환 레이어를 사용하여 기존의 코드를 거의 그대로, 큰 오버헤드 없이 사용할 수 있다.

현재 연구는 선행 연구로, 동기식 소켓과 비동기식 웹 소켓에 대해 진행되었으며, 동기식 인터페이스를 사용하는 많은 프로그램이 적은 코드 수정으로 비동기식 인터페이스를 사용할 수 있도록 인터페이스 변환 레이어를 일반화 시키는 확장작업은 진행 중이다.

Acknowledgement

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2011-0016972)

참 고 문 헌

[1] Finley, Klint (8 June 2010). "Microsoft Rolls Out Office Web Apps". ReadWrite Enterprise. SAY Media. Retrieved 21 January 2013.
[2] "Native Client, Technical Overview", https://developers.google.com/native-client/dev/overview, Last updated 13 April 2013
[3] LibVNCServer, http://sourceforge.net/projects/libvncserver/
[4] naclports, https://code.google.com/p/naclports/

1333