Chapter 06

웹 서버의 응답

1. 서버의 개요

클라이언트와 서버의 차이점

- 서버 머신은 용도에 따라 다양한 종류가 있으며 하드웨어나 OS 부분은 클라이언트와 다른 것도 있다.
- 그러나 네트워크에 관한 부분, 즉 LAN 어댑터 프로토콜 스택, Socket 라이브러리 등의 기능은 클라이언트와 조금도 다르지 않다.
- TCP나 IP의 기능은 하드웨어나 OS가 무엇이든지 달라지지 않기 때문에 기능이 통일되었다고 해도 좋다.
- 원래 가지고 있는 기능은 같아도 사용하는 방법까지 같은 것은 아니다.
- 접속 동작을 할 때 클라이언트에서 접속 동작을 수행하고, 서버는 그것을 기다리는 형태가 되므로 Socket 라이브러리의 사용법이 조금 달라진다.
- 그래서 애플리케이션이 호출하는 Socket 라이브러리의 프로그램 부품이 다른 것이다.
- 또한 서버의 애플리케이션은 동시에 다수의 클라이언트 PC와 대화한다는 차이점도 있다.
- 그렇기 때문에 서버 애플리케이션은 클라이언트 애플리케이션과 구조가 다르다.

1. 서버의 개요

서버 애플리케이션의 구조

- 서버는 동시에 복수의 클라이언트와 통신 동작을 실행하지만, 하나의 프로그램으로 여러 클라이언트들의 상대를 처리하기는 어렵다.
- 어느 클라이언트와 어디까지 대화가 진행되고 있는지를 전부 파악해야 하기 때문이다.
- 그래서 클라이언트가 접속할 때마다 새로 서버 프로그램을 작동하여 서버 애플리케이션이 클라이언트와 1 대 1로 대화하는 방법을 선택하는 것이 일반적이다.
- 이 방법은 클라이언트가 접속했을 때 새로 프로그램을 기동하는 부분에서 다소 시간이 걸리고, 응답 시간이 추가로 소요된다는 단점이 있다.
- 그래서 미리 클라이언트와 대화하는 몇 개의 부분을 작동시켜 두고 클라이언트가 접속했을 때 클라이언트의 상대를 처리하지 않는 비어있는 것을 찾아 여기에 접속한 소켓을 건네주어 클라이언트와의 대화를 계속하는 방법도 있다.

1. 서버의 개요

서버측의 소켓과 포트 번호

- 먼저 클라이언트와 서버의 차이점을 한 번 더 정리해보자.
- 데이터를 송 · 수신하는 동작의 관점에서 보면 클라이언트와 서버라는 상태로 역할을 고정시키는 것은 좋은 방법이라고 할 수 없다.
- 현재는 클라이언트에서 서버에 액세스한다는 형태의 애플리케이션이 다수이지만, 다른 형태로 액세스하는 애플리케이션도 있다.
- 왜냐하면 본래 다양한 형태가 있기 때문이다.
- 이와 같이 다양한 형태로 데이터를 송 · 수신하는 구조를 지원하려면 데이터 송 · 수신의 구조에서 클라이언트와 서버라는 식으로 역할을 결정하지 말고 좌우 대칭으로 어디에서나 자유롭게 데이터를 송신할 수 있도록 해두는 방법이 좋다.
- TCP의 배경에는 이러한 개념이 있다.

1. 서버의 개요

서버측의 소켓과 포트 번호

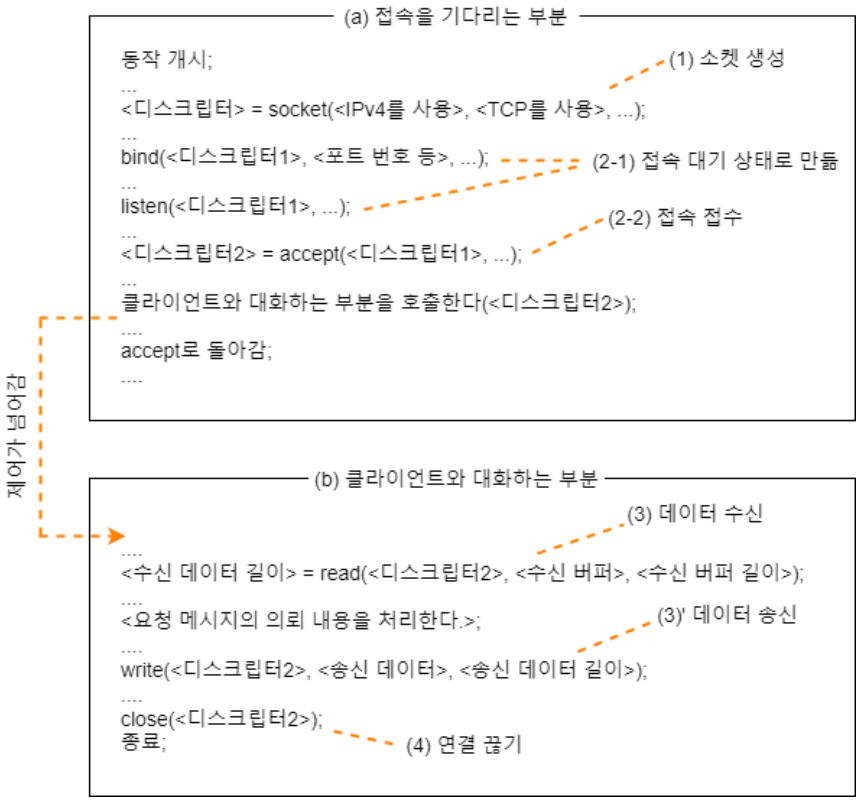
- 어떻게 해도 좌우 대칭으로 만들 수 없는 부분이 있는데, 이것이 접속 동작이다.
- 접속 동작은 한쪽이 기다리고 있는 곳에 또 한쪽을 연결해야 한다.
- 양쪽이 동시에 접속하면 안 되며, 양쪽에서 기다려도 안 되고, 기다리고 있지 않은 곳에 일방적으로 접속하는 것도 안 된다.
- 이 부분만은 접속하는 측과 기다리는 측이라는 역할 분담이 필요하다.
- 데이터 송 · 수신 동작의 시점에서 보았을 때 클라이언트와 서버의 차이점은 여기에 있다.
- 즉 접속하는 측이 클라이언트이고 접속을 기다리는 측이 서버이다.
- 먼저 클라이언트의 데이터 송 수신 동작은 다음의 네 단계로 성립한다.

1) 소켓을 만든다(소켓 작성 단계).
2) 서버측의 소켓과 파이프로 연결한다(접속 단계).
3) 데이터를 송 수신한다(송 · 수신 단계).
4) 파이프를 분리하고 소켓을 말소한다(연결 끊기 단계).

1. 서버의 개요

서버측의 소켓과 포트 번호

- **이에 비해 서버 쪽은 (2)의 접속 부분이 접속을 기다리는 형태가 되므로 다음과 같이 된다.**
  - (1) 소켓을 만든다(소켓 작성 단계).
  - (2-1) 소켓을 접속 대기 상태로 만든다(접속 대기 상태).
  - (2-2) 접속을 접수한다(접속 접수 단계).
  - (3) 데이터를 송 · 수신한다(송 · 수신 단계).
  - (4) 파이프를 분리하고 소켓을 말소한다(연결 끊기 단계).
- **이 서버의 동작을 의사 코드로 나타내면 그림과 같이 됩니다.**



1. 서버의 개요

서버측의 소켓과 포트 번호

- 먼저 socket 함수를 호출하여 소켓을 만드는데 (그림 (1)), 이곳은 클라이언트와 같다.
- 다음에 bind 함수를 호출하여 소켓에 포트 번호를 기록합니다(그림 (2-1)).
- 클라이언트측에서 접속 동작을 실행할 때 서버 측의 소켓에 할당한 포트 번호를 지정하는데, 이것이 포트 번호이다.
- 구체적인 번호는 규칙에 의해 서버 애플리케이션마다 결정되어 있고 웹 서버의 경우는 80번으로 되어 있다.
- 포트 번호를 기록하면 listen 함수를 호출하여 소켓에 접속하기를 기다리는 상태라는 제어 정보를 기록한다(그림 (2- 1)).
- 이렇게 해서 소켓은 클라이언트에서 접속 동작의 패킷이 도착하는 것을 기다리는 상태가 된다.
- 그러면 accept 함수를 호출하여 접속을 접수한다(그림 (2-2)).
- 접속을 접수하는 부분은 서버 애플리케이션을 기동한 후 즉시 실행되므로 이 시점에서는 아직 클라이언트의 접속 패킷이 도착하지 않았을 것이다.

1. 서버의 개요

서버측의 소켓과 포트 번호

- 패킷이 도착하지 않았는데도 accept 함수를 호출하여 접속 접수 동작을 실행하는 것이 이상할 수도 있지만 그렇지 않다.
- 접속 패킷이 도착하지 않았으면 도착을 기다리는 상태가 되어 패킷이 도착할 때 접속 접수 동작을 실행하기 때문이다.
- 그러므로 accept 함수를 실행한 시점에서 보통 서버측은 패킷의 도착을 기다리는 상태가 되고 애플리케이션은 쉬는 상태가 된다.
- 이 상태에서 애플리케이션에서 접속 패킷이 도착하면 응답 패킷을 반송하여 접속 접수 동작을 실행한다.
- 그리고 접속 대기의 소켓을 복사하여 새로운 소켓을 만들고, 접속 상대의 정보를 비롯한 제어 정보를 새 소켓에 기록한다.
- 여기까지가 accept 함수를 호출했을 때의 동작으로, 이렇게 해서 새 소켓이 클라이언트측의 소켓과 연결된다.
- accept 함수의 실행이 끝나면 접속을 기다리는 동작은 끝나므로 그 후 접속을 접수하는 부분은 클라이언트와 대화하는 부분을 기동한다.
- 그리고 접속한 새로운 소켓을 클라이언트와 주고 받는 부분에 건네주고 이 부분이 클라이언트와의 대화를 실행한다.
- 이때의 데이터 송 · 수신 동작은 앞에서 설명했듯이 클라이언트와 마찬가지이다.

1. 서버의 개요

서버측의 소켓과 포트 번호

- 이것이 일련의 동작인데 설명에서 빠진 부분이 있다.
- 접속 동작을 할 때 만든 새 소켓이 아니라 원래 있던 접속 대기 상태의 소켓이 어떻게 되는가 하는 점이다.
- 이것은 접속 대기 상태인 채로 계속 존재한다.
- 그리고 다시 accept 함수를 호출하면 클라이언트에서의 접속 패킷이 도착했을 때 접속 접수 동작을 실행한다.
- 이때 앞에서 설명했듯이 접속 대기의 소켓을 복사하여 새 소켓을 만들고, 새 소켓을 클라이언트측의 소켓과 접속한 후 원래 소켓은 그대로 접속 대기 상태인 채로 둔다.
- 이렇게 해서 잇달아 복사하여 새 소켓을 만드는 부분이 요점이다.
- 새 소켓을 만들지 않고 접속 대기의 소켓에 그대로 접속하면 접속 대기의 소켓이 없어져 버리므로 다음에 다른 클라이언트가 접속하면 곤란해진다.
- 이러한 사태를 막기 위해 새 소켓을 만들고 여기에 접속하는 것이다.

1. 서버의 개요

서버측의 소켓과 포트 번호

- 새 소켓을 만들 때의 포트 번호도 요점이다.
- 포트 번호는 소켓을 식별하기 위해 사용하는 것이므로 소켓마다 다른 값을 할당해야 한다는 개념을 고수하려면 곤란한다.
- 이 개념을 고수하려면 접속 대기 동작을 할 때 새로 만드는 소켓에는 원래의 접속 대기 소켓과는 다른 포트 번호를 할당해야 한다.
- 예를 들어 클라이언트에서 80번이라는 포트 번호의 소켓에 접속 패킷을 보냈을 때 이것과는 다른 포트 번호의 소켓에서 회답이 돌아온다.
- 이렇게 되면 접속 패킷을 보낸 상대로부터 회답이 돌아왔는지, 아니면 다른 상대로부터 잘못된 회답의 패킷이 돌아왔는지 판별할 수 없다.
- 그러므로 새로 만든 소켓에도 접속 대기 소켓과 같은 포트 번호를 할당해야 한다.
- 이렇게 하면 다른 문제가 발생한다.
- 즉 포트 번호는 소켓을 지정하기 위한 것인데, 같은 번호를 할당한 여러 개의 소켓이 존재해서 포트 번호로 소켓을 지정할 수 없게 된다는 문제이다.
- 클라이언트에서 패킷이 도착했을 때 TCP 헤더에 기록되어 있는 수신처 포트 번호만 조사해서는 패킷이 어느 소켓에서 대화하고 있는지 판단할 수 없다.

서버측의 소켓과 포트 번호

- 이 문제는 다음과 같이 하면 해결할 수 있다.
- 소켓을 지정할 때 서버측의 소켓에 할당한 포트 번호뿐만 아니라 클라이언트측의 포트 번호도 사용하고, IP 주소도 추가하여 다음의 네 가지 정보를 사용하는 것이다.
  - 클라이언트측의 IP 주소
  - 클라이언트측의 포트 번호
  - 서버측의 IP 주소
  - 서버측의 포트 번호

- 서버측의 소켓에는 같은 포트 번호를 가진 여러 개의 소켓이 존재한다.
- 하지만 클라이언트측의 소켓은 모두 다른 포트 번호를 할당하므로 클라이언트측의 포트 번호에 따라 소켓을 지정할 수 있다.
- 단, 다른 포트를 사용한다는 규칙은 한 대의 클라이언트 내부에서 적용되므로 클라이언트가 복수인 경우에는 각 클라이언트에서 같은 포트 번호를 사용할 수도 있다.
- 그래서 클라이언트의 IP 주소도 소켓을 지정하는 판단 근거에 추가한다.
- 예를 들어 IP 주소가 198.18.203.154인 클라이언트의 포트 번호 1025번과 198.18.142.86의 1025번과는 다른 소켓이라는 것을 알 수 있으므로 여기에 접속되어 있는 서버측의 소켓을 지정할 수 있다.

2. 서버의 수신 동작

수신 신호를 디지털 데이터로 변환

- 서버의 전체 모습을 보았으므로 클라이언트가 보낸 패킷이 서버에 도착한 부분부터 다시 시작한다.
- 서버에 도착하는 패킷의 실체는 전기나 빛의 신호이며, 이것을 수신하는 동작은 클라이언트와 같다.
- 수신 동작은 패킷의 신호를 LAN 어댑터에서 수신하고 디지털 데이터로 바꾸는 부분에서 시작된다.
- LAN을 흐르는 패킷의 신호는 1과 0으로 이루어진 디지털 데이터의 신호와 타이밍을 나타내는 클록 신호를 합성한 것이다.
- 여기에서 클록 신호를 추출하고 클록 신호에서 타이밍을 계산하면서 신호를 읽어오면, 1과 0의 디지털 데이터로 바꿀 수 있다(그림).

2. 서버의 수신 동작

수신 신호를 디지털 데이터로 변환

- 다음에 패킷의 맨 마지막에 있는 프레임 체크 시퀀스(FCS)라는 오류 검사용 데이터를 이용하여 오류 유무를 검사한다.
- 즉, 수신하여 디지털 데이터로 되돌리는 것을 오류 검사의 계산식에 따라 계산하고, 패킷의 맨 끝에 있는 FCS 필드의 값과 비교하는 것이다.
- 패킷의 맨 끝에 들어있는 값은 송신할 때 전기 신호로 변환하기 전의 디지털 데이터를 바탕으로 계산한 것이므로 신호에서 되돌린 디지털 데이터가 송신 전과 같으면 계산한 값과 맨 끝에 있는 값은 일치할 것이다.
- 양쪽이 일치하지 않으면 잡음 등의 영향으로 신호가 도중에 변형되어 데이터가 변하고 수신한 패킷은 제 역할을 하지 못하므로 버린다.
- FCS가 일치하고 오류가 없는 것을 확인한 다음에는 맨 앞의 MAC 헤더에 있는 수신처 MAC 주소를 조사하여 패킷이 자신을 수신처로 하여 보낸 것인지 판단한다.
- 이더넷의 기본 동작은 먼저 신호를 LAN 전체로 흘리고 해당자만 신호를 수신한다는 방법을 취한다.
- 그렇기 때문에 다른 기기로 갈 패킷의 신호가 흘러오는 경우도 있다.
- 그러므로 수신처가 자신으로 되어 있는 패킷만 남기고 다른 패킷은 버린다.

2. 서버의 수신 동작

수신 신호를 디지털 데이터로 변환

- 이로써 신호를 수신하여 디지털 데이터로 되돌리는 동작은 끝났으므로 디지털 데이터로 되돌린 것을 LAN 어댑터 내부의 버퍼 메모리에 저장한다.
- 여기까지는 LAN 어댑터의 MAC 부분이 실행한다.
- 그 사이에 서버의 CPU는 패킷의 도착을 감시하고 있는 것이 아니라 다른 일을 실행하고 있으므로 CPU는 패킷을 알아차리지 못한다.
- 따라서 수신 처리가 진행되지 않으므로 인터럽트라는 방법을 사용하여 LAN 어댑터에서 CPU로 패킷의 도착을 알린다.
- 이 시점에서 CPU는 실행하고 있던 작업을 중단하고 LAN 드라이버로 실행을 전환한다.
- 이렇게 해서 LAN 드라이버가 동작하기 시작하여 LAN 어댑터의 버퍼 메모리에서 수신한 패킷을 추출한다.
- 그리고 MAC 헤더의 타입 필드의 값에 따라 프로토콜을 판별하고 프로토콜을 처리하는 소프트웨어를 호출한다.
- 타입 필드의 값은 IP 프로토콜인 것을 나타내는 값으로 되어 있으므로 TCP/IP의 프로토콜 스택을 호출하고 여기에 패킷을 건네준다.

2. 서버의 수신 동작

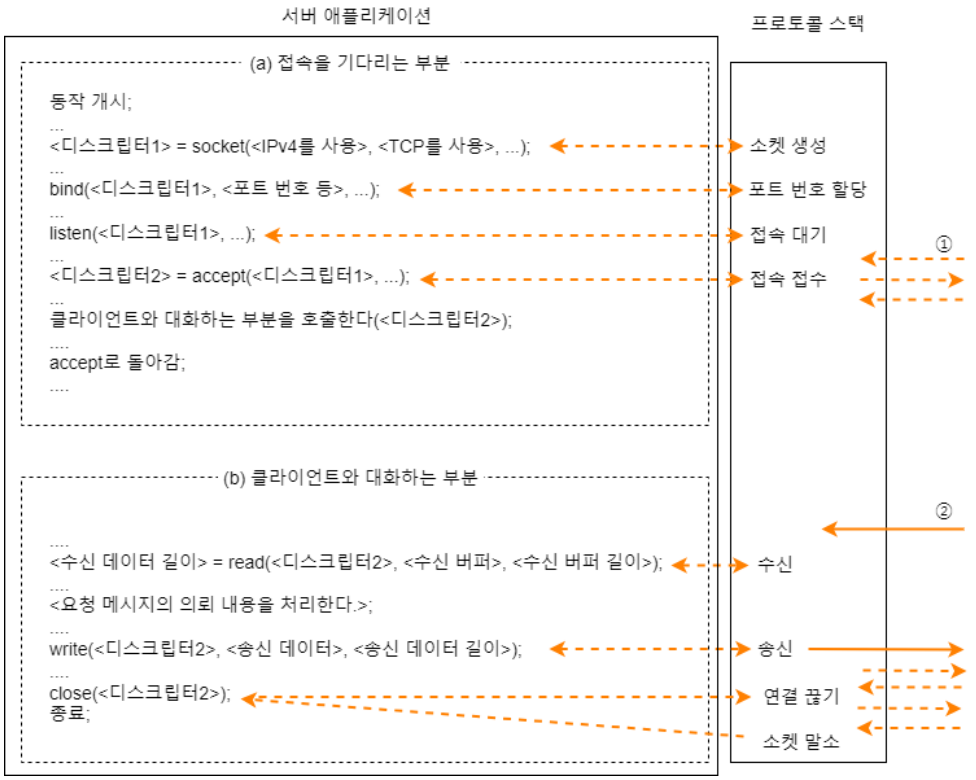
IP 담당 부분의 수신 동작

- 프로토콜 스택에 패킷이 전달되면 우선 IP 담당 부분이 동작하여 IP 헤더를 점검한다.
- IP 헤더의 내용이 규칙에 따라 올바르게 만들어 졌는지부터 점검한 후 수신처 IP 주소가 자신을 대상으로 하는지 조사한다.
- 서버에서 라우터와 같이 패킷을 중계하는 기능이 유효하게 된 경우에는 자신을 대상으로 하지 않은 패킷이 도착할 수 있다.
- 그러면 라우터와 같이 경로표에서 중계 대상을 조사하고 그것에 패킷을 중계한다.
- 패킷이 자신을 대상으로 한 것임을 확인할 수 있으면 다음에는 조각 나누기에 의해 패킷이 분할되었는지 조사한다.
- IP 헤더를 조사하면 분할되었는지 알 수 있으므로 분할되어 있는 경우에는 패킷을 일시적으로 메모리에 저장해둔다.
- 그리고 분할된 패킷의 조각이 전부 도착한 시점에서 패킷의 조각을 조립하여 원래 패킷으로 복원한다.
- 패킷이 분할되어 있지 않으면 받은 패킷은 원래의 모습 그대로여서 이 패킷 조립 동작은 필요 없으므로 패킷을 받은 것이 된다.
- 그러면 IP 헤더의 프로토콜 번호 항목을 조사하여 해당하는 담당 부분에 패킷을 건네준다.

2. 서버의 수신 동작

TCP 담당 부분이 접속 패킷을 수신했을 때의 동작

- 지금까지는 모든 패킷이 같았지만, 앞으로 TCP 담당 부분의 동작은 도착한 패킷의 내용에 따라 달라진다.
- 패킷의 TCP 헤더에 있는 SYN이라는 컨트롤 비트가 1로 되어 있으면 접속 동작의 패킷이다(그림 ①).
- 이 경우 접속을 접수하는 동작을 실행하는데, 그 전에 도착한 패킷의 수신처 포트 번호를 조사하여 이 번호와 같은 번호를 할당한 접속 대기 상태의 소켓이 있는지 확인한다.
- 만약 수신처 포트 번호와 같은 포트 번호의 접속 대기 소켓이 없으면 무언가 잘못된 것이므로 오류 통지 패킷을 클라이언트에 반송한다.



2. 서버의 수신 동작

TCP 담당 부분이 접속 패킷을 수신했을 때의 동작

- 해당하는 접속 대기 소켓이 있으면 패킷을 복사하여 새 소켓을 만들고, 여기에 송신처의 IP 주소나 포트 번호, 시퀀스 번호의 초기값, 윈도우의 값 등 필요한 정보를 기록한다.
- 동시에 송신 버퍼나 수신 버퍼로 사용하는 메모리 영역을 확보한다.
- 그리고 패킷을 받았음을 나타내는 ACK 번호, 서버에서 클라이언트에 보내는 데이터에 관한 시퀀스 번호의 초기값, 클라이언트에서 서버로 보내온 데이터를 받기 위한 수신 버퍼의 빈 용량을 나타내는 윈도우 값 등의 항목을 기록한 TCP 헤더를 만들고 이것을 IP 담당 부분에 의뢰하여 클라이언트에 반송한다.
- 이 패킷이 클라이언트에 도착하면 클라이언트에서 패킷을 받았음을 나타내는 ACK 번호가 돌아올 것이다.
- 이것이 돌아오면 접속 동작은 완료된다.
- 이때 서버측의 애플리케이션은 accept 함수를 호출하여 실행을 쉬는 상태일 것이다.
- 여기에 새로 만든 소켓의 디스크립터를 전달하여 서버 애플리케이션의 동작을 재개한다.

2. 서버의 수신 동작

TCP 담당 부분이 접속 패킷을 수신했을 때의 동작

- 다음은 데이터 송 · 수신 단계에 들어가서 데이터 패킷이 도착한 경우의 동작이다(그림 ②).
- 우선 TCP 담당 부분은 도착한 패킷이 어느 소켓에 해당하는지 조사한다.
- 접속이 끝난 소켓의 경우 서버측의 포트 번호로서 같은 값을 할당한 여러 개의 소켓이 존재할지도 모르므로 수신처 포트 번호만으로는 소켓을 지정할 수 없다.
- 그래서 IP 헤더의 송신처 IP 주소와 수신처 IP 주소, TCP 헤더의 수신처 포트 번호와 송신처 포트 번호라는 4개의 정보가 모두 합치되는 소켓을 찾는다.
- 이들 4개의 정보가 합치되는 해당 소켓을 발견하면 패킷에 기록되어 있는 데이터 송 · 수신의 진행 상황과 도착한 패킷의 TCP 헤더의 정보를 결합하여 데이터 송 · 수신 동작이 올바르게 진행되고 있는지 점검한다.
- 구체적으로는 소켓에 기록된 지난 번 시퀀스 번호나 지난 번 데이터 조각의 길이로부터 다음 시퀀스 번호의 값을 계산하고 도착한 패킷의 TCP 헤더에 기록된 시퀀스 번호와 합치되는지 조사한다.
- 둘이 합치되면 패킷이 도중에 없어지지 않고 제대로 서버까지 도착한 것이다.
- 그러면 패킷에서 데이터 조각을 추출하여 수신 버퍼에 저장한다.
- 이때 지난 번 패킷에서 수신한 데이터 조각의 다음에 연결되는 식으로 해서 데이터를 분할하기 전의 상태로 되돌린다.

2. 서버의 수신 동작

TCP 담당 부분이 접속 패킷을 수신했을 때의 동작

- 과정을 통해 수신한 데이터를 수신 버퍼에 저장하면 수신 확인 응답용 TCP 헤더를 만든다.
- 그리고 여기에 수신 패킷의 시퀀스 번호와 데이터 조각의 길이로부터 계산한 ACK 번호를 기록하고 IP 담당 부분에 의뢰하여 클라이언트에 반송한다.
- 데이터 패킷을 수신했을 때의 동작은 수신 버퍼에 데이터 조각을 보관한 부분에서 일단 끝난다.
- 이후 애플리케이션이 Socket 라이브러리의 read 함수를 호출하여 수신한 데이터를 받아온 부분에서 애플리케이션에 건네준다.
- 애플리케이션이 왕래하기까지는 수신 버퍼에 보관한 상태 그대로이다.
- 보통은 데이터 패킷이 도착하기 전에 애플리케이션이 read 함수를 호출하여 데이터의 도착을 기다리는 상태가 된다.
- 이 경우 TCP 담당 부분의 수신 동작이 끝나는 것과 동시에 애플리케이션에 데이터를 건네주는 동작이 시작된다고 생각하면 된다.

2. 서버의 수신 동작

TCP 담당 부분의 연결 끊기 동작

- 데이터 송・수신이 끝나면 연결 끊기 동작에 들어간다.
- TCP 프로토콜의 규칙에 따르면 연결 끊기 동작은 클라이언트와 서버 중 어느 쪽이 먼저 실행해도 상관없다.
- 애플리케이션이 어디에서부터 연결 끊기 동작을 실행하는지 결정해야 한다.
- 웹의 경우 HTTP 프로토콜의 버전에 따라 다르므로 HTTP 1.0이라면 서버에서 연결 끊기 동작을 시작한다.
- 이 경우 서버 측에서 애플리케이션이 Socket 라이브러리의 close를 호출하고, TCP 담당 부분이 FIN이라는 컨트롤 비트에 1을 설정한 TCP 헤더를 만든 후 IP 담당 부분에 의뢰하여 클라이언트에 보낸다.
- 이것이 클라이언트에 도착하면 클라이언트는 ACK 번호를 반송한다.
- 클라이언트가 close 함수를 계속 호출하고, FIN을 1로 한 TCP 헤더를 서버에 보낸 후 서버가 ACK 번호를 반송하면 연결 끊기 동작은 끝난다.
- HTTP 1.1은 클라이언트에서 먼저 연결 끊기 동작을 시작할 수도 있지만, 이 경우에는 클라이언트와 서버의 순번만 반대가 된다.
- 어느 경우든지 연결 끊기 동작이 끝나면 잠시 기다렸다가 소켓을 말소한다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

URI의 변환

- 그림에서 서버 애플리케이션의 동작을 보았지만, 이 동작은 웹 서버 뿐만 아니라 여러 가지 서버 애플리케이션에 공통이다.
- 데이터를 송 · 수신하는 부분의 동작은 모든 애플리케이션에서 비슷하다.
- 서버 애플리케이션의 종류에 따라 달라지는 것은 read 함수를 호출한 뒤에 있는 다음과 같은 부분이다.

```
<리퀘스트 메시지의 의뢰 내용을 처리함>;
```

- 웹 서버의 경우 read 함수에서 받은 데이터의 내용이 HTTP 요청 메시지가 된다.
- 그리고 받은 요청 메시지에 기록되어 있는 내용에 따라 적절한 처리를 실행하여 응답 메시지를 만들고 write 함수를 통해 이것을 클라이언트에 반송한다는 형태로 작동한다.
- 요청 메시지에는 메소드라는 일종의 명령과 데이터 출처를 나타내는 URI라는 파일의 경로명 같은 것이 쓰여있으며, 내용에 따라 데이터를 클라이언트에 반송한다.
- 이 경우 메소드나 URI의 내용에 따라 웹 서버 내부의 동작이 달라진다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

URI의 변환

- 가장 간단한 것은 GET 메소드와 HTML 문서의 파일명을 나타내는 URI를 기록한 요청 메시지이다.
- 이 경우 파일에서 HTML 문서의 데이터를 읽어들이고 이것을 응답 메시지로 반송한다.
- 그러나 단순히 URI에 기록되어 있는 파일을 디스크에서 읽는 것은 아니다.
- 단순히 URI에 기록되어 있는 경로명의 파일을 읽어오면, 디스크의 파일에 전부 액세스할 수 있게 되므로 웹 서버의 디스크가 무방비 상태로 노출되어 위험하기 때문이다.
- 그래서 한 가지 대책이 있다.
- 웹 서버에서 공개하는 디렉토리는 디스크의 실제 디렉토리가 아니라 가상으로 만든 디렉토리이고 이 가상의 디렉토리 구조에서의 경로 이름을 URI에 써야 한다.
- 그러므로 파일을 읽어올 때는 가상의 디렉토리와 실제 디렉토리의 대응 관계를 조사하고 실제 디렉토리의 경로 이름으로 변환한 후 파일을 읽어 데이터를 반송한다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

URI의 변환

- 예를 들어 가상 디렉토리가 만들어진 부분에서 URI에 다음과 같이 쓰여있는 요청 메시지가 도착하면,

```
/~user2/sub-user2/sample.html
```

- /~user2/.... 는 실제 디렉토리인 /home/user2/.... 에 대응하므로 URI를 아래와 같은 경로 이름으로 변환한다.

```
/home/user2/sub-user2/sample.html
```

- 이 경로 이름의 파일을 디스크에서 읽어온 후 해당 데이터를 반송하는 형태이다.
- 이 파일 이름 변환에는 특별한 예가 있다.
- 즉 URI에 쓰여있는 경로 이름이 파일 이름을 생략한 형태인 경우에는 미리 설정한 파일 이름이 쓰여있는 것으로 간주하는 것이다.
- 브라우저에서 다음과 같이 파일 이름을 생략한 URL을 입력하면,

```
http://www.naver.com/tone/
```

- 맨 끝에 파일명이 추가되어 다음과 같은 페이지가 화면에 표시될 수 있다.

```
http://www.naver.com/tone/index.html
```

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

URI의 변환

- 이 예의 경우 'index.html' 이라는 파일 이름이 서버에 설정되어 있어서, 이것이 디렉토리 이름의 뒤에 쓰여있는 것으로 간주한다.
- 파일 이름을 바꿔 쓰는 규칙을 서버측에 설정하고 규칙에 따라 파일 이름을 바꿔 쓰고 나서 파일에 접근하는 웹 서버 애플리케이션도 있다.
- URI에 쓴 경로 이름이 설정되어 있던 패턴에 합치하면 경로 이름을 바꿔쓴 후 디스크에 접근하는 것이다.
- 어떤 이유로든지 웹 서버측에서 디렉토리 이름이나 파일 이름을 변경했지만, 원래 URL에서 그대로 접속하려는 경우에는 이 기능이 편리하다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

CGI 프로그램을 작동하는 경우

- URI에 쓴 파일의 내용이 HTML 문서나 화상 데이터인 경우에는 파일의 내용을 그대로 응답 메시지로 클라이언트에 반송한다.
- 하지만 URI에 쓰는 파일의 내용은 HTML 문서뿐만 아니라 프로그램 파일의 이름을 URI에 쓸 수도 있다.
- 이 경우에는 파일의 내용을 그대로 반송하지 않고 해당 프로그램을 작동시켜서 프로그램이 출력하는 데이터를 클라이언트에 반송한다.
- 웹 서버에서 작동시키는 프로그램은 몇 가지 유형이 있고, 각 유형에 따라 구체적인 동작이 달라지지만 CGI(Common Gateway Interface)라는 타입의 프로그램은 다음과 같이 동작한다.
- 웹 서버에서 프로그램을 작동시키는 경우 프로그램에서 처리하는 데이터를 HTTP의 요청 메시지 안에 넣어 브라우저에서 웹 서버로 보내는 것이 일반적이다.
- 데이터에는 여러 가지가 있는데, 쇼핑 사이트에서 주문 양식에 품명, 개수, 배송처 등을 입력하고 보내는 것이 대표적인 예이다.
- 또한 검색 엔진에서 키워드를 입력하고 보내는 것도 흔히 보는 예이다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

CGI 프로그램을 작동하는 경우

- 아무튼 무언가의 데이터를 요청 메시지의 안에 넣어 브라우저에서 웹 서버로 보낸다.
- 이와 관련된 동작은 두 가지 방법을 이용한다.
- 하나는 HTML 문서의 양식 부분에 'method= "GET"'이라고 쓰고 HTTP의 GET 메시지를 사용하여 URI의 뒤에 입력 데이터를 내장해서 서버에 보내는 방법이다.
- 또 하나는 HTML 문서의 양식 부분에 'method= "POST"'라고 쓰고 HTTP의 요청 메시지의 메시지 본문이라는 부분에 데이터를 내장해서 서버에 보내는 방법이다.
- 요청 메시지를 보낸 웹 서버는 다음과 같이 작동한다.
- 웹 서버는 먼저 URI의 부분에 쓰여 있는 파일 이름을 조사하여 이것이 프로그램인지 판단한다.
- .cgi, .php 확장자를 등록해 두고 파일 이름의 확장자가 이것과 일치하면 프로그램으로 간주하는 것이다.
- 프로그램용 디렉토리 이름을 설정해두고 디렉토리에 저장한 파일을 전부 프로그램으로 간주하는 방법도 있다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

CGI 프로그램을 작동하는 경우

- 파일이 프로그램인 것을 알고 있으면 웹 서버는 이 프로그램을 작동시키도록 OS에 의뢰한다.
- 그리고 요청 메시지에서 데이터를 추출하여 작동시킨 프로그램에 건네준다.
- 메소드가 GET이었으면 URI 뒤에 내장된 데이터를 추출하여 건네주고 POST였으면 메시지 본문에 내장된 데이터를 추출하여 건네준다.
- 그러면 작동시킨 프로그램이 받은 데이터를 처리하여 무언가의 출력 데이터를 웹 서버에 되돌려준다.
- 주문을 접수했음을 나타내는 안내문이나 데이터베이스에서 키워드를 검색한 결과 등 내용은 매우 다양하다.
- 내용이 무엇이든지 데이터를 처리한 결과를 클라이언트에 반송하기 위해 출력 데이터를 웹 서버에 되돌려주는 것이다.
- 출력 데이터는 보통 HTML 태그를 내장한 HTML 문서로 되어 있으므로 웹 서버는 이것을 그대로 응답 메시지로 클라이언트에 반송한다.
- 출력 데이터의 내용은 작동시킨 프로그램을 만드는 방법에 따라 달라서 웹 서버에서는 알 수 없으므로 웹 서버는 내용에 관여하지 않는다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

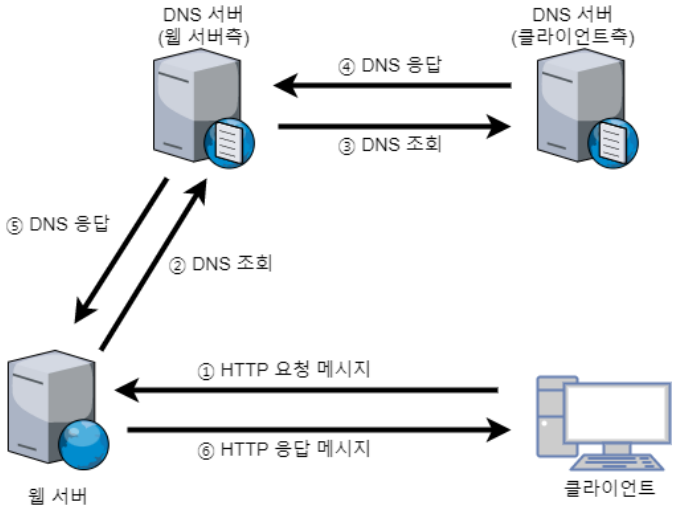
서버로 수행하는 접근 제어

- 요청 메시지의 내용에서 데이터 출처를 판단하고, 그곳에서 데이터를 얻어 클라이언트에 반송한다는 것이 웹 서버의 기본 동작이다.
- 하지만 그 동작을 실행할 때 사전에 설정해 둔 조건에 해당하는지 조사하고, 조건에 해당하는 경우 그 동작을 금지하거나 조건에 해당하는 경우만 동작을 실행한다는 기능도 있다.
- 이와 같이 조건에 따라 액세스 동작 여부를 설정하는 기능을 액세스 제어라고 하며 회원제의 정보 제공 서비스 등에서 특정 사용자에게만 액세스를 허가하는 것과 같은 경우에 액세스 제어를 사용한다.
- 사내에서 운용하는 웹 서버에 특정 부서의 액세스만 허가한다는 사용법도 있다.
- 웹 서버에서 설정하는 조건은 주로 다음의 세 가지이다.
  - 클라이언트의 주소
  - 클라이언트의 도메인명
  - 사용자명과 패스워드
- 이 조건을 데이터 출처가 되는 파일이나 디렉토리와 대응해서 설정한다.
- 그리고 클라이언트에서 요청 메시지를 받고 URI에서 데이터 출처를 판단하면, 여기에 액세스 가능 여부의 조건이 설정되어 있는지 조사하고 액세스가 허가된 경우에만 파일을 읽거나 프로그램을 실행한다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

서버로 수행하는 접근 제어

- 다음으로 이 조건이 설정되어 있는 경우의 동작을 살펴보자.
- 처음에는 클라이언트의 IP 주소가 조건으로 설정되어 있는 경우이지만 이것은 간단하다.
- Accept 함수로 접속을 접수했을 때 클라이언트의 IP 주소를 알 수 있으므로 이것을 점검하기만 한다.
- 클라이언트의 도메인 이름이 조건으로 설정되어 있는 경우에는 클라이언트의 IP 주소에서 도메인 이름을 조사하는데, 이때 DNS 서버를 이용한다.
- 보통 DNS 서버를 사용하는 것은 도메인 이름에서 IP 주소를 조사해야 하는데, 반대로 IP 주소에서 도메인 이름을 조사할 때도 DNS 서버를 사용한다.
- 구체적으로는 다음과 같이 동작한다.



3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

서버로 수행하는 접근 제어

- 클라이언트에서 요청 메시지를 받은 웹 서버는(그림 ①) 프로토콜 스택에 의뢰하여 패킷의 송신처 IP 주소를 조사하고, IP 주소에 대응하는 이름을 조회하는 메시지를 만들어 가장 가까운 DNS 서버에 보낸다(그림 ②).
- 그러면 DNS 서버는 이 IP 주소가 등록된 DNS 서버를 찾아 거기에 조회를 보냅니다(그림 ③).
- 여기에서 도메인 이름을 알 수 있을 것이므로 도메인 이름의 회답이 돌아옵니다(그림④).
- 회답이 돌아왔으면 웹 서버측의 DNS 서버는 이것을 웹 서버에 전송한다(그림⑤).
- 이를 통해 송신처 IP 주소에서 도메인 이름을 판명하고 만일에 대비하여 도메인 이름에서 IP 주소를 조사한 후 송신처 IP 주소와 일치하는 것을 확인한다(그림 ⑥).
- 도메인 이름을 위조하여 DNS 서버에 등록하는 공격 방법도 있으므로 이것을 방지하기 위해 이중으로 점검하는 것이다.
- 양쪽이 일치하면 도메인 이름을 설정한 조건을 대조하여 접속할 수 있는지 판단한다.
- 그림에서 알 수 있듯이 이 방법은 DNS 서버의 조회 메시지가 왕래하는 만큼 시간이 걸리며, 그만큼 웹 서버의 응답 시간이 길어진다.

3. 웹 서버 소프트웨어의 요청 메시지에 대한 응답

응답 메시지

- 이렇게 해서 요청 메시지에 대해 적절하게 처리하고 처리가 완료되면 응답 메시지를 반송한다.
- 이때의 개념은 최초에 클라이언트가 요청 메시지를 웹 서버에 보내는 동작과 같다.
- 먼저 웹 서버가 Socket 라이브러리의 write 함수를 호출하여 응답 메시지를 프로토콜 스택에 건네준다.
- 이때 응답 메시지를 어디에 보내야 할지를 프로토콜 스택에 알려주어야 하는데 통신 상대의 클라이언트를 직접 통지하는 것이 아니라 어느 소켓을 사용하여 통신하고 있는지를 나타내는 디스크립터를 통지하여 상대를 지정해야 한다.
- 소켓에는 통신의 상태가 전부 기록되어 있고, 여기에 통신 상대의 정보도 있으므로 디스크립터만 통지하면 된다.
- 그러면 프로토콜 스택은 데이터를 한 개의 패킷에 들어가는 길이로 분할하고 헤더를 붙여서 패킷을 송출한다.
- 이 패킷에는 수신처로 클라이언트의 주소가 기록되어 있기 때문이다.
- 그렇다면 이 패킷은 스위치나 라우터를 경유하여 인터넷 속을 통해 최종적으로 클라이언트에게 도착한다.

4. 웹 브라우저의 응답 메시지 출력

응답 데이터의 형식

- 웹 서버가 보낸 응답 메시지는 다수의 패킷으로 나뉘어 클라이언트에 도착하는데, 이렇게 되면 클라이언트측에서 이것을 수신한다.
- 이 동작은 먼저 LAN 어댑터가 신호로부터 디지털 데이터로 되돌리고, 프로토콜 스택이 분할된 패킷을 모아서 데이터 부분을 추출하여 원래의 응답 메시지로 되돌린 후 이것을 브라우저에 건네준다.
- 이에 해당하는 동작은 서버의 수신 동작과 같고, 나중에 브라우저의 화면 표시 동작으로 진행한다.
- 화면 표시 동작은 응답 메시지에 저장된 데이터가 어떤 종류인지를 조사하는 곳부터 시작한다.
- 웹에서 취급하는 데이터는 문장, 화상, 음성, 영상 등 다양하고, 종류에 따라 표시 방법이 다르므로 최초에 종류를 조사해야 한다.
- 그렇지 않으면 표시 동작을 올바르게 실행할 수 없다.

4. 웹 브라우저의 응답 메시지 출력

응답 데이터의 형식

- 데이터의 종류를 판단하는 근거는 몇 가지가 있는데, 응답 메시지의 맨 앞부분에 있는 'Content-Type' 이라는 헤더 파일의 값으로 판단하는 것이 원칙이다.
- 여기에는 다음과 같은 형식으로 데이터의 종류를 써야 한다.

```
Content-Type: text/html
```

- /의 왼쪽 부분은 '주 타입'이라 하고 이것을 통해서 데이터 종류의 대분류를 나타낸다.
- 그리고 오른쪽의 '서브 타입'이라는 부분에 실제 데이터의 종류를 나타낸다.
- 예를 들어 위의 예라면 주 타입은 text이고, 서브 타입은 html이다.
- 결국 위의 예에서 데이터의 종류는 HTML 사양에 따라 태그를 내장한 HTML 문서이다.
- 데이터의 종류가 텍스트인 경우에는 어떤 문자 코드를 사용하는지 판단해야 한다.
- 이 경우에는 다음과 같이 charset으로 문자 코드의 정보를 부가하므로 이것을 통해 판단한다.

```
Content-Type: text/html; charset = utf-8
```

- 이곳이 utf-8이면 문자 코드는 UNICODE, euc-kr이면 EUC 코드, iso-2022-jp이면 JIS 코드, shift_jis이면 시프트 JIS 코드이다.

4. 웹 브라우저의 응답 메시지 출력

응답 데이터의 형식

- Content-Type으로 데이터의 종류를 조사하면 'Content- Encoding' 이라는 헤더 필드의 값도 조사한다.
- 압축 기술이나 부호화 기술에 따라 원래 데이터를 변환하고 나서 메시지에 저장한 경우에는 어떤 변환을 했는지를 Content- Encoding 필드에 기록해야 한다.
- 그러므로 이 필드를 조사하여 필요에 따라 원래대로 되돌려야 한다.
- Content-Type 필드를 사용하여 데이터의 종류를 나타내는 방법은 MIME라는 사양에 규정되어 있는데 이것은 웹뿐만 아니라 메일 등에서도 사용하는 일반적인 방법이다.
- 그러나 이 방법은 어디까지나 원칙에 불과한다.
- Content-Type으로 판단하는 방법을 올바르게 동작시키려면 웹 서버가 Content-Type의 값을 정확하게 설정해야 하는데, 현실은 그렇지 않은 경우도 있다.
- 웹 서버의 운영 관리자가 익숙하지 않아서 설정이 부적절하고 Content-Type에 정확한 값이 설정되지 않는 상황도 일어나기 때문이다.
- 그러므로 원칙에 따라 Content-Type을 조사만 해서는 데이터의 종류를 정확하게 판단할 수 없는 경우도 있다.

4. 웹 브라우저의 응답 메시지 출력

응답 데이터의 형식

- 다른 판단 근거를 사용하여 종합적으로 데이터의 종류를 판단하는 경우도 있다.
- 요청한 파일의 확장자나 데이터 내용의 포맷 등에서 종합적으로 판단하는 것이다.
- 예를 들어 파일명의 확장자를 조사하고, 이것이 .html이나 .htm이라면 HTML 문서로 간주하거나 데이터의 내용을 조사하여 맨 앞부분에 <html> 이라는 태그가 쓰여있으면 HTML 문서로 간주하는 식이다.
- HTML 문서와 같은 텍스트 데이터뿐만 아니라 화상 데이터도 마찬가지이다.
- 화상 데이터는 압축된 바이너리 데이터이지만 맨 앞부분에 내용을 나타내는 정보가 기록되어 있으므로 내용을 보고 데이터의 내용을 판단할 수 있다.
- 단, 이 부분은 사양으로 정해진 것이 아니므로 브라우저의 종류나 버전에 따라 동작이 달라진다.

4. 웹 브라우저의 응답 메시지 출력

브라우저 화면에 웹 페이지를 표시

- 데이터의 종류가 판명되면 드디어 종점이 눈앞에 다가왔다.
- 데이터의 종류에 따라 화면 표시의 프로그램을 호출하여 데이터를 표시하면 된다.
- HTML 문서, 일반 텍스트, 화상이라는 기본적인 데이터는 브라우저 자체가 화면 표시 기능을 가지고 있으므로 브라우저가 자체에서 화면 표시 동작을 실행한다.
- HTML 문서에는 문장의 레이아웃이나 글꼴의 종류 등을 기록한 태그가 내장되어 있으므로 태그의 의미를 해석하여 문장을 배치하면서 화면에 표시한다.
- 실제 화면 표시 동작은 OS가 담당하므로 OS에 대해 화면의 어떤 위치에 어떤 문자를 어떤 글꼴로 표시할지 지시하는 것이다.
- 웹 페이지에는 영상 등을 내장한 것도 있다.
- 이 경우 HTML 문서 데이터와 영상 데이터를 별도의 파일에 저장하고 HTML의 문장 데이터 안에 화상이 내장되었음을 나타내는 태그를 써야 한다.

4. 웹 브라우저의 응답 메시지 출력

브라우저 화면에 웹 페이지를 표시

- 문장 데이터를 읽어와서 태그를 발견했으면 브라우저는 화상 데이터의 파일을 서버에서 읽어온다.
- 이 동작은 HTML의 문장 데이터 파일을 읽어오는 동작과 같다.
- HTTP 요청 메시지의 URI 부분에 화상 데이터의 파일명을 쓰기만 하면 된다.
- 이렇게 해서 리퀘스트 메시지를 웹 서버에 보내면 웹 서버에서 화상 데이터가 반송될 것이다.
- 그러면 태그가 쓰여있던 장소에 화상 데이터를 내장시킨다.
- JPEG나 GIF 형식의 화상 데이터는 압축되어 있으므로 압축을 풀고 나서 데이터를 OS에 건네주어 표시하도록 지시한다.
- 물론 그 장소에 문장을 표시하면 겹쳐버리므로 이 부분을 화상의 크기만큼 비워두고 문장을 표시한다.

4. 웹 브라우저의 응답 메시지 출력

브라우저 화면에 웹 페이지를 표시

- HTML 문서나 화상과 같이 브라우저가 자체에서 표시 기능을 가지고 있는 경우에는 이렇게 해서 OS에 지시를 내리며 화면에 표시한다.
- 그러나 웹 서버에 읽어오는 것에는 워드프로세서나 프레젠태이션 소프트웨어라는 애플리케이션의 데이터도 있다.
- 이 경우에는 자체에서 표시할 수 없으므로 해당 애플리케이션을 호출한다.
- 해당 애플리케이션은 브라우저에 플러그인으로 편입된 형태일 수도 있고 독립된 프로그램일 수도 있다.
- 어느 경우든지 데이터의 종류에 따라 호출하는 프로그램이 결정되며, 이것이 브라우저에 설정되어 있을 것이다.
- 그러므로 설정에 따라 프로그램을 호출하여 데이터를 건네면 화면에 호출한 프로그램이 표시된다.
- 이렇게 해서 브라우저는 표시 동작을 마치고 사용자가 다음 행동을 하기를 기다린다.
- 그리고 표시한 페이지 안의 링크를 클릭하거나 URL 입력 상자에 새로운 URL을 입력하면 다시 웹 서버에 대한 접속 동작이 시작된다.

Thank You