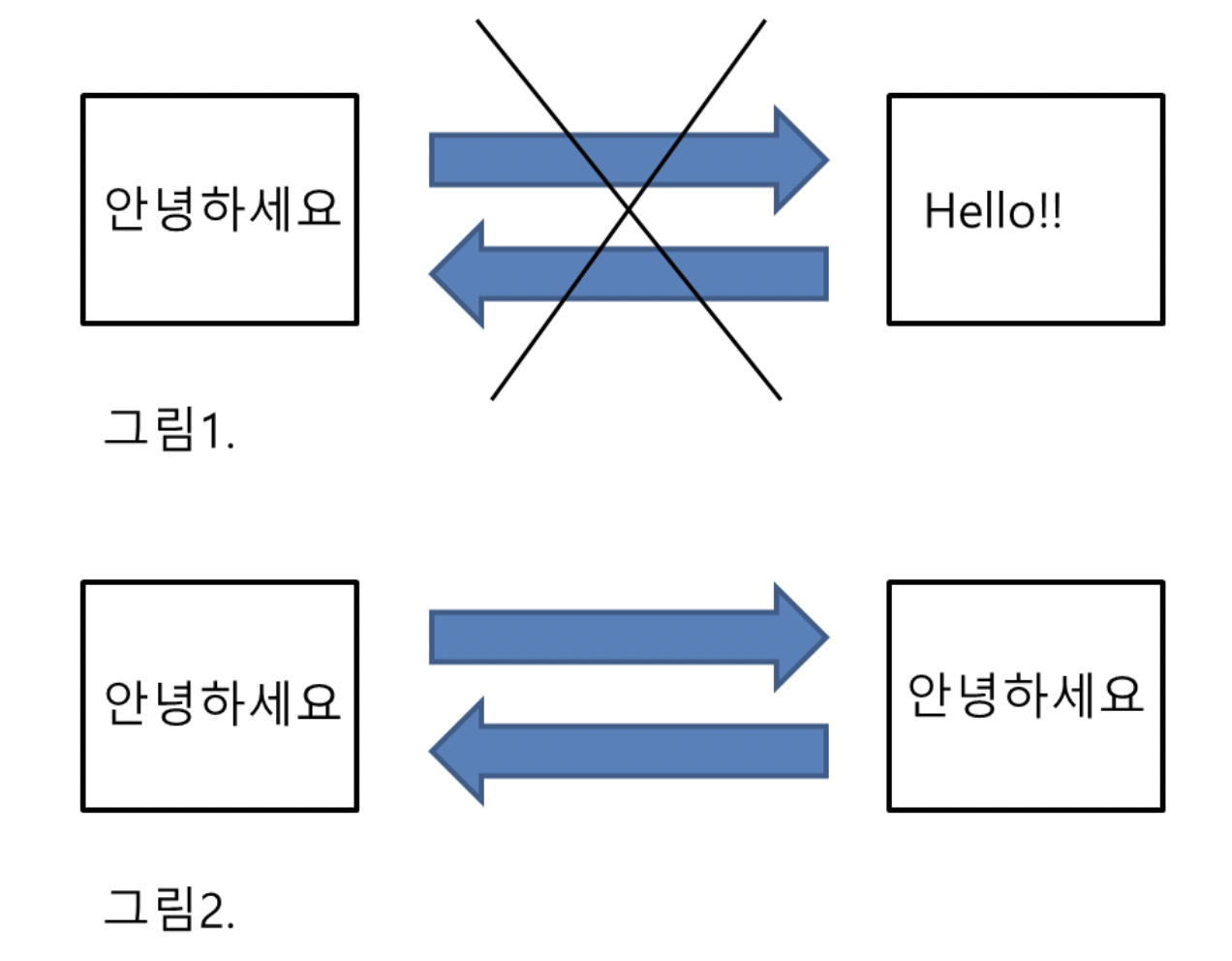
**프로토콜(개념, 종류)**

* 컴퓨터나 원거리 통신 장비 사이에서 메시지를 주고 받는 양식과 규칙의 체계

즉, 통신 규약 및 약속 // 컴퓨터끼리 서로 통신하기 위해서는 양쪽이 동일한 프로토콜을 이해하고 처리할 수 있어야 함.

비유해보자면



* 한국어나 영어를 ‘프로토콜’
* 언어로 커뮤니케이션 하는 것은 ‘통신’
* 대화의 내용을 ‘데이터’

<그림1> 한쪽은 한국어를 한쪽은 영어를 쓴다고 하면 서로 알 수 없는 말이기 때문에 의사소통을 할 수 없다. 언어 프로토콜이 다르기 때문에 서로의 데이터를 전달 할 수 없다

<그림2>

서로 한국어라는 같은 프로토콜을 사용하기 때문에 전달하고 싶은 데이터를 상대방에게 전달할 수 있고 그 결과 통신이 성립하게 된다.

* 프로토콜의 기본 요소

구문(syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정

의미(semantics) : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정

시간(timing) : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정

* 종류

|  |  |
| --- | --- |
| 계층 | 프로토콜 |
| 응용 | HTTP, SMTP, FTP, Telnet |
| 표현 | ASCII, MPEG, JPEG, MIDI |
| 세션 | NetBIOS, SAP, SDP, NWLink |
| 전송 | TCP, UDP, SPX |
| 네트워크 | IP, IPX |
| 데이터 링크 | Ethernet, Token Ring, FDDI, Apple Talk |
| 물리 | 없음 |

* 기능

**단편화(Fragmentation)와 재합성(Assembly)**  
단편화 : 송신 측에서는 긴 데이터 블록을 손쉽게 전송할 수 있도록 크기가 똑같은 작은 블록으로 나누어 전송  
재합성 : 수신 측에서 쪼개진 작은 데이터 블록을 재합성하여 원래의 메시지로 복원하는 기능

**캡슐화(Encapsulation)**  
각 프로토콜에 적합한 데이터 블록을 만들려고 데이터에 정보를 추가하는 것  
플래그, 주소, 제어 정보, 오류 검출 부호 등을 부착하는 기능

**연결 제어(Connection Control)**  
비연결 데이터 전송(데이터그램)과 연결 위주 데이터 전송(가상회선)을 위한 통신로를 개설·유지·종결하는 기능  
  
**흐름 제어(Flow Control)**  
데이터양이나 통신속도 등이 수신 측의 처리 능력을 초과하지 않도록 조정하는 기능  
  
**오류 제어(Error Control)**  
데이터 전송 중 발생할 수 있는 오류나 착오 등을 검출하고 정정하는 기능  
  
**순서 결정(Sequencing)**  
연결 위주의 데이터를 전송할 때 송신 측이 보내는 데이터 단위 순서대로 수신 측에 전달하는 기능

**주소 설정(Addressing)**  
발생지, 목적지 등의 주소를 명기하여 데이터를 정확하게 전달하는 기능  
  
**동기화(Synchronization)**  
두 통신 객체의 상태(시작, 종류, 검사 등)를 일치시키는 기능

**HTTP란?**

* 텍스트 기반의 통신 규약으로 인터넷에서 데이터를 주고 받을 수 있는 프로토콜

이렇게 규약을 정해두었기 때문에 모든 프로그램이 이 규약에 맞춰 개발해서 서로 정보를 교환할 수 있게 되었다.

* 동작

클라이언트 즉, 사용자가 브라우저를 통해서 어떠한 서비스 url을 통하거나 다른 것을 통해서 요청(request)를 하면 서버에서는 해당 요청사항에 맞는 결과를 찾아서 사용자에게 응답(response)하는 형태

* 요청 : client -> server
* 응답 : server -> client
* 특징
* HTTP 메시지는 HTTP서버와 HTTP 클라이언트에 의해서 해석된다
* TCP/IP를 이용하는 응용 프로토콜이다
* 연결 상태를 유지하지 않는 비연결성 프로토콜이다 (이를 해결하기 위해 Cookie 와 Session이 등장)
* 연결을 유지하지 않는 프로토콜이기 때문에 요청/응답 방식으로 동작한다.

**HTTP Request & Response**

* Request (요청)

클라이언트가 서버에게 연락하는 것. 요청에 대한 정보를 담아 서버로 보냄

* Ex) 서버가 주문서를 받아 클라이언트가 어떤것을 원하는지 파악할 수 있게 함. 식당에서 주문서를 작성하는 것과 같다
* Response (응답)

서버가 요청에 대한 답변을 클라이언트에게 보내는 것

**HTTP 메소드**

* Request Method
  + GET : 자료를 요청할 때 사용
  + POST : 자료의 생성을 요청할 때 사용
  + PUT : 자료의 수정을 요청할 때 사용
  + DELETE : 자료의 삭제를 요청할 때 사용
* Request HTTP 예시

텍스트이(가) 표시된 사진

자동 생성된 설명

* + 시작줄

메서드 구조 버전으로 구성

GET : HTTP Method

사이트 주소

HTTP/1.1 : HTTP 버전

* + 헤더

두번째 줄 부터는 헤더이며 요청에 대한 정보를 담고 있다. User-Agent(사용자 웹 브라우저 종류 및 버전 정보), Upgrade-Insecure-Requests등등이 헤더에 해당되며 헤더의 종류는 매우 많다

* + 본문

본문은 요청할 때 함께 보낼 데이터를 담는 부분. 현재 예시에는 단순히 주소로만 요청을 보내고 있고 따로 데이터를 담아 보내지 않기 때문에 본문이 비어있음.

**HTTP 상태코드**

* 종류가 많음. 모두 세자리로 이루어져있으며 크게 다섯부류로 나눌 수 있음
  + 1XX (조건부 응답) : 요청을 받았으며 작업을 계속한다.
  + 2XX (성공) : 클라이언트가 요청한 동작을 수신하여 이해했고 승낙했으며 성공적으로 처리했음을 가리킨다.
  + 3XX (리다이렉션 완료) : 클라이언트는 요청을 마치기 위해 추가 동작을 취해야 한다.
  + 4XX (요청 오류) : 클라이언트에 오류가 있음을 나타낸다.
  + 5XX (서버 오류) : 서버가 유효한 요청을 명백하게 수행하지 못했음을 나타낸다.
* Response HTTP 메시지 예시

텍스트이(가) 표시된 사진

자동 생성된 설명

* + 시작줄

버전 상태코드 상대메시지로 구성. 200은 성공적인 요청이었다는 뜻

* + 헤더

두번째 줄부터는 헤더로 응답에 대한 정보를 담고 있음

* + 본문

응답에는 대부분의 경우 본문이 있음. 보통 데이터를 요청하고 응답 메시지에는 요청한 데이터를 담아서 보내주기 때문. 응답 메시지에 HTML이 담겨있는데 이 HTML을 받아 브라우저가 화면에 렌더링한다.