인터넷에서 사용되는 여러 기술

FTP 이야기 2

File Transfer Protocol

글
조인준
KBS 미디어기술연구부 수석연구원

지난 편에서는 FTP 프로토콜의 동작 모델과 제어 연결(Control Connection) 및 이를 위한 인증 방식에 대해 설명 드렸습니다. 이번 편에서는 데이터 연결 및 전송에 관해 설명 드리겠습니다. FTP 서버와 클라이언트 간에 제어 연결을 위한 인증 과정이 완료되면 FTP 세션 기간 동안 유지되는 제어 연결이 수립됩니다. 제어 연결에서는 명령, 응답, 상태 정보 등 데이터 전송에 필요한 제어 관련 정보만이 전송되고, 파일이나 데이터는 전송되지 않습니다. 서버와 클라이언트 간에 파일이나 데이터를 전송해야 할 때마다 제어 연결과는 별도로 새 데이터 연결(Data Connection)을 만들며, 전송이 완료되면 해당 데이터 연결을 종료하는 방식으로 FTP는 동작합니다. 데이터 연결은 클라이언트 데이터 전송 프로세스와 서버 데이터 전송 프로세스를 이어주는 것이며, 이를 통해 파일 송수신을 위한 데이터 전송뿐 아니라, 서버 디렉터리의 파일 목록 같은 파일 이외의 데이터 전송도 이루어집니다. FTP 표준은 데이터 연결을 수립하기 위한 능동 모드(Active Mode)와 수동 모드(Passive Mode) 두 가지 방식을 정의하고 있으며, 그 내용은 다음과 같습니다.

능동 모드 데이터 연결

FTP 데이터 연결의 능동 모드는 서버가 데이터를 전송하기 위해 직접 클라이언트에게 먼저 데이터 연결을 시작하는 방식입니다. 앞서 이야기했듯이 FTP는 파일을 주고받을 때 제어 연결과 실제 파일이나 디렉터리 목록 등의 데이터가 오가는 데이터 연결을 따로 운영합니다. 제어 연결은 FTP 세션이 지속되는 동안 항상 연결된 상태로 유지되지만, 데이터 연결은 전송이 필요할 때마다 새로 만들어졌다가 전송이 끝나면 종료됩니다. 능동 모드에서 클라이언트는 서버의 제어 연결을 위한 21번 포트로 제어 연결을 엽니다(이미 제어 연결이 수립된 상태라면 현재의 제어 연결을 그대로 사용합니다). 이 제어 연결을 통해 클라이언트는 서버에게 파일 전송을 명령하거나 디렉터리 내의 파일 목록 전송을 명령합니다. 이때 서버가 실제 데이터를 보내려면 데이터 연결이라는 또 하나의 별도 연결이 필요한데, 이 데이터 연결을 위해 [그림 1]과 같이 서버가 먼저(능동적으로) 클라이언트에게 연결을 시도하는 방식이 능동 모드입니다. [그림 1]에서 클라이언트가 제어 연결을 통해 서버의 21번 포트에 자신이 데이터를 받을 포트 번호 1742번을 알려줍니다. 그러면 서버는 자신의 20번 포트(FTP 데이터 전송용 표준 포트, 제어 포트 21번보다 1 낮은 번호)로부터 클라이언트가 지정한 포트로 데이터 연결 수립을 시도합니다. 연결에 성공하면 이 연결을 통해 실제 파일 데이터나 디렉터리 목록이 오가고, 전송이 끝나면 이 데이터 연결은 닫힙니다.

데이터 연결 종료 후에도 제어 연결은 여전히 유효한 상태로 유지되어 다음 명령을 기다립니다. 문제는 서버가 먼저 데이터 연결을 시도하는 구조가 네트워크 보안 장비, 라우터 및 NAT(Network Address Translation) 설정에서 종종 문제를 일으킨다는 점입니다. 일반적으로 클라이언트 측에는 외부에서 들어

오는 연결 요청을 무분별하게 받아들이지 않도록 방화벽이나 라우터 설정이 되어 있습니다. 그런데 능동 모드에서는 외부의 서버가 클라이언트 측으로 먼저 연결을 시도하므로, 방화벽 입장에서는 이것이 외부에서 들어오는 위험한 접근처럼 보일 수 있습니다. 그 결과 데이터 연결이 차단되거나 실패하는 경우가 발생할 수 있습니다. 또한 클라이언트가 매번 다른 임시 포트를 지정하기 때문에, 방화벽이 어떤 포트를 열어줘야 할지 미리 아는 것이 사실상 불가능하다는 점도 문제를 복잡하게 만듭니다. 이러한 이유로 능동 모드는 네트워크 환경이 단순하거나 보안 설정이 관대한 경우에는 잘 작동하지만, 기업 네트워크 같은 보안 수준이 높은 환경에서는 연결에 실패하기 쉽습니다.

요약하면, FTP의 능동 모드는 서버가 클라이언트로 데이터 연결을 먼저 시도하여 데이터 전송을 위한 연결을 여는 방식으로 이해할 수 있으며, 제어와 데이터가 서로 다른 연결을 사용한다는 점에서 보통의 서비스와 차별되는 구조를 갖고 있습니다. 오늘날의 보안 수준이 높은 네트워크 환경에서는 이런 외부로부터 시작되는 연결이 보안상 위험으로 간주되기 때문에 많은 FTP 클라이언트와 서버는 수동 모드를 기본으로 동작하는 것이 일반적이라고 합니다.

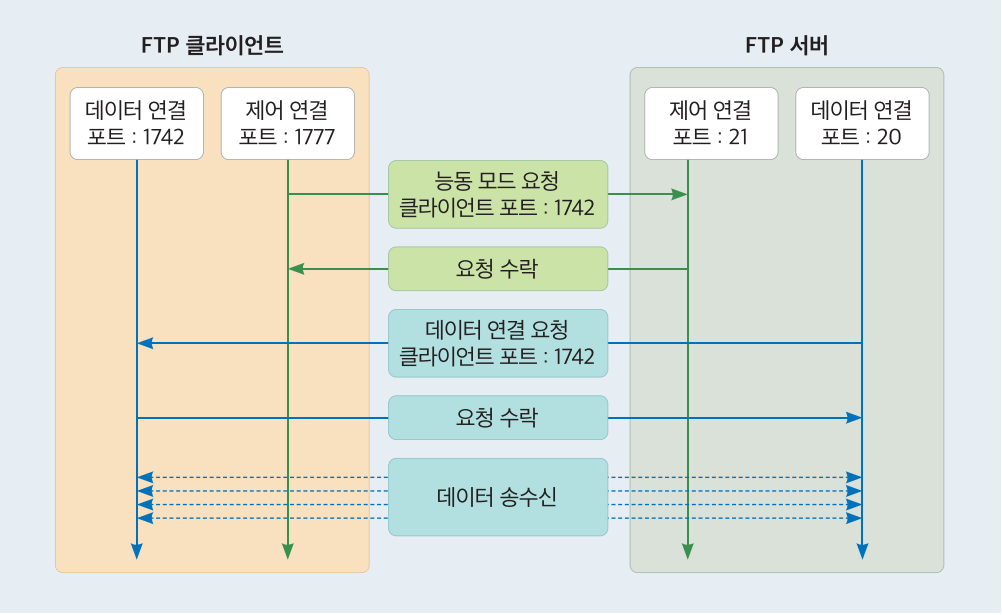


그림 1. 능동 모드 데이터 연결

수동 모드 데이터 연결

FTP의 수동 모드는 서버가 데이터 연결을 시작하는 능동 모드와 달리 클라이언트가 서버에 먼저 데이터 연결을 시작하는 방식입니다. 데이터 전송을 위한 연결을 클라이언트 측에서 먼저 시작한다는 점이 능동 모드와 대비되는 차이입니다. 수동 모드는 앞서 설명된 네트워크 보안으로 인한 능동 모드의 문제점을 해결하기 위해 등장한 방식입니다. 능동 모드에서는 서버가 클라이언트에게 먼저 연결을 시도하기 때문에 클라이언트 측 방화벽이나 라우터가 이를 외부에서 들어오는 비정상적인 연결로 간주하고 차단하는 경우가 많았습니다. 이를 피하기 위해 수동 모드는 [그림 2]와 같이 클라이언트가 연결을 시작하도록 설계되었습니다. 이로써 데이터 연결 수립을 위한 명령이나 응답이 방화벽이나 라우터를 통과하기가 훨씬 수월합니다.

조금 세부적으로 데이터 연결 수립과정을 살펴보면 다음과 같습니다. 우선, 클라이언트는 서버의 21번 포트로 제어 연결을 열고, 로그인을 위한 인증과 같은 기본 절차를 거칩니다. 그 후 데이터 전송이 필요해지면 [그림 2]와 같이 클라이언트는 제어 연결을 통해 서버에게 수동 모드로 동작해 달라는 요청을 보냅니다. 이 요청을 받은 서버는 자신이 데이터 연결에 사용할 임시 포트 번호([그림 2]의 경우 2195)를 정한 뒤 클라이언트에게 해당 포트 번호를 알리는 응답을 보냅니다. 클라이언트는 서버의 응답에 따라 서버의 2195 포트로 데이터 연결을 시작합니다. 서버는 자신의 2195 포트에서 대기하고 있다가 클라이언트의 데이터 연결 요청을 받아들입니다. 이렇게 데이터 연결이 논리적으로 완료되면, 이를 통해 파일 데이터나 디렉터리 내의 파일 목록 등이 실제로 전송되기 시작합니다. 모든 전송이 끝나면 데이터 연결은 닫히고, 제어 연결은 그대로 남아서 다음 명령을 처리할 수 있는 상태를 유지합니다. 이런 구조는 보안 관련 측면에서 유리합니다. 클라이언트 측에서 보면 외부에서 들어오는 연결을 수신할 필요가 없으므로 방화벽은 외부로 나가는 연결만 허용하는 설정으로도 FTP를 정상적으로 사용할 수 있습니다. 대부분의 방화벽과 라우터는 내부에서 외부로 나가는 연결에 대해서는 많이 허용하는 편이기 때문입니다. 이 때문에 수동 모드는 기업 네트워크 등 보안 정책이 엄격하거나 NAT가 필수적인 환경에서도 안정적으로 작동합니다.

물론 수동 모드도 완벽한 해결책은 아닙니다. 서버 측에서 보면 외부로부터 들어오는 클라이언트의 연결을 여러 개 처리해야 하므로 서버의 방화벽은 여러 임시 포트에 대한 수신을 허용해야 하는 상황에 처합니다. 이 문제를 관리하기 위해 서버는 보통 수동 모드용 포트 범위를 미리 정해두고 해당 구간의 포트로 들어오는 메시지들에 대해 방화벽을 여는 네트워크 관련 설정을 합니다. 이렇게 하면 서버는 보안 위험을 통제하면서도 다수의 클라이언트로부터 들어오는 수동 모드 데이터 연결을 유연하게 처리할 수 있습니다. 이렇듯 FTP의 수동 모드는 데이터 연결을 서버가 아닌 클라이언트가 시작하는 방식이며, 제어와 데이터가 분리되어 동작하는 구조는 능동 모드와 동일하지만 연결을 시작하는 측이 반대라는 점이 핵심적인 차이 입니다. 수동 모드는 방화벽과 NAT 환경에서 발생하는 연결 차단 문제를 완화하여 현대 네트워크 환경에서 FTP가 실용적으로 사용될 수 있도록 한다는 점에서 많이 활용되는 모드입니다.

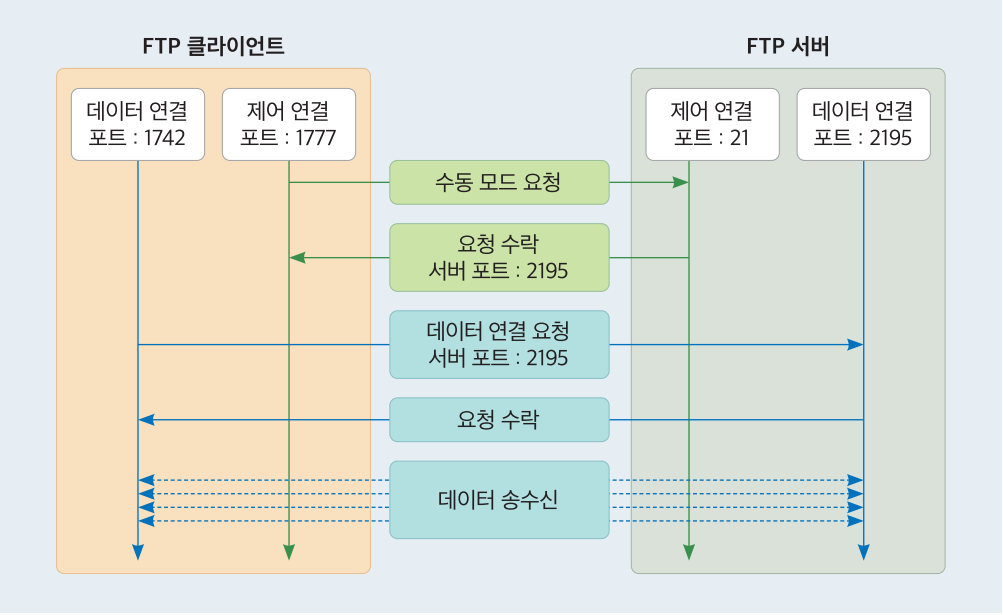


그림 2. 수동 모드 데이터 연결

서버와 클라이언트 사이에 데이터 연결이 설정되면, 특정 명령에 따라 클라이언트에서 서버로 또는 서버에서 클라이언트로 데이터가 전송되기 시작합니다. 제어관련 정보는 제어 연결을 통해 전송되므로 데이터 연결은 순수한 데이터 통신을 위해 사용됩니다. FTP는 데이터 연결을 통해 데이터를 전송하기 위해 스트림 모드(Stream Mode), 블록 모드(Block Mode), 압축 모드(Compressed Mode)의 세 가지 방식을 지원합니다.

☑ 스트림 모드(Stream Mode)

스트림 모드에서는 데이터가 구조화되지 않은 연속적인 바이트로만 전송됩니다. 많은 프로토콜들이 독립된 헤더 필드를 가진 메시지 단위로 데이터를 주고받는 것과 달리, FTP의 스트림 모드는 별도의 메시지 구조를 사용하지 않고 TCP의 신뢰성에만 의존한 데이터 전송을 수행합니다. FTP 데이터 전송의 기본 모드이자 가장 단순한 방식이어서 구현이 쉽고 호환성을 위해 반드시 지원되어야 하는 이유로 세 가지 전송 방식 중 스트림 모드가 실제 FTP 구현에서 가장 널리 사용되는 방식 입니다.

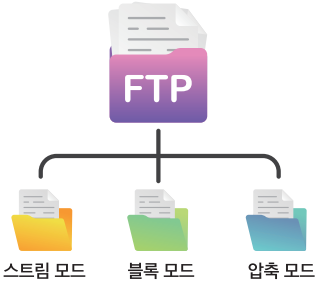
☑ 블록 모드(Block Mode)

블록 모드는 IP 통신의 가장 통상적인 형식을 따르는 방식으로, 데이터를 개별 블록 단위로 나누어 전송합니다. 각 블록은 3바이트의 헤더를 가지며, 이 헤더에는 블록의 길이와 전송되는 데이터 블록에 대한 정보가 포함됩니다. 헤더 정보 등을 통해 전송된 데이터를 추적하고, 전송이 중단된 경우 처음부터 다시 시작하는 것이 아니라 오류가 발생된 부분부터 재전송을 할 수 있도록 설계되어 있습니다.

☑ 압축 모드(Compressed Mode)

압축 모드는 런렝스 부호화(Run Length Encoding)라는 비교적 단순한 압축 기법을 사용하여 데이터 내의 반복되는 패턴의 중복성을 제거하는 방식을 통해 전체 데이터를 더 적은 바이트로 표현함으로써 전송량을 줄입니다. 압축된 데이터는 블록 모드와 유사하게 헤더 및 페이로드 구조로 전송됩니다.

지금까지 살펴본 세 가지 FTP 데이터 전송 모드는 내부적인 처리 방식과 효율성, 그리고 복구 능력에서 각기 다른 특성을 지닙니다. 이러한 세 가지 모드의 대표적인 장점과 단점을 간단히 정리하면 [표 1]과 같습니다.



전송 모드	특징	장점	단점
스트림 모드	연속적인 바이트 스트림	단순, 호환성 높음, 효율적	중단 시 재개 불가
블록 모드	블록 단위 전송	전송 중단 시 재개 가능, 구조적	오버헤드 증가
압축 모드	런렝스 압축 사용	전송 데이터량 절감 가능	실제 효용 적음

표 1. FTP 데이터 전송 모드 비교

이 세 가지 전송 모드만으로 파일 전송에는 더 이상 고려해야 할 것이 없는 걸까요? 만약 모든 파일을 단순히 바이트들의 집합으로 보고 파일을 한 바이트씩 그대로 한쪽에서 다른 쪽으로 옮기기만 한다면 어떻게 될까요? 일부 파일 형식에서는 이러한 방식이 전혀 문제없이 잘 동작할 수 있습니다. 그러나 시스템마다 다른 내부 표현 방식을 갖는 파일 형식에서는 문제가 발생할 수 있습니다. 이런 문제의 가장 대표적인 사례가 텍스트 파일입니다. ASCII 텍스트 파일은 모두 ASCII 문자 집합을 사용하지만, 줄의 마지막을 표시하는 제어 문자가 시스템마다 다르기 때문입니다. 줄바꿈에 Windows는 CR+LF(Carriage Return+Line Feed) 조합을 사용하고, UNIX 시스템은 LF(Line Feed) 문자를 사용하며, Mac OS는 CR(Carriage Return) 문자를 사용하기 때문에 단순히 파일을 바이트 단위로 전송하면 데이터는 정확히 그대로 옮겨지지만 각 운영 시스템에 따라 프로그램이 줄바꿈 문자를 올바르게 인식하지 못하는 문제가 생길 수 있습니다. 이런 문제를 해결 하려면, FTP는 단순히 모든 파일을 바이트의 집합으로 처리하는 대신 파일의 종류와 시스템의 내부 표현 방식에 따라 스마트한 처리를 할 수 있어야 합니다. 이를 위해 FTP 표준은 파일 전송 전에 파일의 내부 표현 방식과 관련 정보를 데이터 타입(Data Types), 데이터 구조(Data Structures), 포맷 제어(Format Control)를 통해 명시적으로 지정할 수 있도록 정의하고 있습니다.

데이터 타입(Data Types)

데이터 타입은 파일의 기본적인 표현 방식을 정의합니다. 텍스트 파일을 옮길 때는 일반적으로 'ASCII 타입'이 사용되며, 이 경우 전송 과정에서 위에서 언급한 각 시스템의 줄바꿈 방식이 자동으로 변환됩니다. 예를 들어 Mac OS 시스템에서 Windows로 텍스트 데이터가 전송이 될 경우 Mac OS의 CR은 표준 ASCII 표현 LF로 전송되고, 이를 수신한 Windows에서 표준 ASCII 표현 LF가 CR+LF로 변환됩니다.
반면 실행 파일이나 이미지처럼 텍스트가 아닌 데이터를 전송할 때는 '바이너리 타입'이 쓰입니다. 이 모드는 데이터가 가공되지 않은 그대로 전송되므로 손실이 없고, 압축 파일이나 미디어 파일에도 적합합니다. 또한 특수한 하드웨어에서 바이트 크기가 표준과 다를 경우 'Local 타입'으로 바이트 단위를 지정할 수도 있습니다. 과거 IBM 계열 시스템과 호환을 위해 'EBCDIC 타입'도 존재하지만 오늘날에는 거의 사용되지 않습니다.

데이터 구조(Data Structures)

데이터 구조는 파일이 전송 중에 어떤 형태로 나뉘어 표현되는지를 규정합니다. 기본값은 '파일 구조'로 파일을 단순한 연속된 바이트 시퀀스로 봅니다. 대부분의 현대 FTP 구현은 이 구조를 사용합니다. 반면 '레코드 구조'는 파일을 여러 줄이나 행처럼 구분된 레코드 단위의 집합으로 봅니다. 이 방식은 데이터베이스나 텍스트 레코드 기반 시스템처럼 줄 단위 정보가 중요한 경우 유용합니다. 세 번째인 '페이지 구조'는 파일을 페이지 단위로 구성하며 각 페이지에 제목이나 길이 같은 부가 정보를 포함시켜줍니다. 이 구조는 특정 부분만 재전송하거나 임의 접근이 필요할 때 장점이 있지만, 구현이 복잡해 실제로는 거의 쓰이지 않는다고 합니다.

포맷 제어(Format Control)

포맷 제어(Format Control)는 특히 텍스트 파일의 줄바꿈이나 제어 문자 등을 정의합니다. 일반 텍스트 파일은 'Non-Print' 형식으로 전송되며, 이는 인쇄 제어 문자를 포함하지 않고 수신 측에서 자체적으로 줄바꿈을 처리하게 합니다. 'Telnet 포맷 제어'는 텔넷 프로토콜에서 사용하는 제어 문자를 이용해 줄바꿈과 공백을 처리하며, 과거 텍스트 터미널과의 호환성을 위해 사용되었습니다. 'Fortran 캐리지 제어' 방식은 오래된 포트란(Fortran) 환경에서 문서 인쇄 시 사용되던 규칙으로 각 레코드의 첫 글자가 인

쇄 동작을 나타냅니다. 예를 들어 공백은 줄바꿈, '0'은 빈 줄 추가, '1'은 새 페이지 시작 등을 의미합니다. FTP의 포맷 제어에서 프린트(인쇄) 관련 줄바꿈 및·공백 등의 정보가 들어가는 이유는 FTP가 처음 표준화된 시점(1970~80년대 초반)에는 파일 교환의 주요 목적 중 하나가 텍스트 보고서나 과학 계산 결과를 프린터로 출력하기 위한 데이터 전송이었기 때문이라고 합니다. 기술 환경이 완전히 바뀐 오늘날에는 사실상 거의 전부 Non-Print 형식을 사용한다고 합니다.

이렇게 FTP는 단순히 데이터를 이동시키는 도구가 아니라, 서로 다른 운영체제와 시스템이 사용하는 파일 표현 방식의 차이를 극복하기 위한 정교한 측면도 갖도록 설계되었습니다. 일반적인 텍스트 파일 전송에는 'ASCII 타입'과 '파일 구조' 조합이, 바이너리 파일에는 '바이너리 타입'과 '파일 구조' 조합이 가장 널리 쓰입니다. 이러한 설계 덕분에 FTP는 다양한 환경 간에도 데이터의 내용적 일관성을 유지하며 파일을 교환할 수 있는 기반을 마련했습니다.

지금까지 FTP 프로토콜의 제어 연결 및 데이터 연결에 관해 설명해 드렸습니다. 다음 편에서는 FTP의 여러 동작에 사용되는 구체적 명령과 응답에 대해서 알아보도록 하겠습니다.

File Transfer Protocol

P.S.

C군이 여러분께 전하는 내용 중 전문적 성격이 짙은 것은 엄밀한 언어를 사용하여 설명하기에는 한계가 있습니다. 본 내용은 설명하는 대상에 대한 전체적 맥락의 이해에만 이용하시고, 그 이상은 권위 있는 전문자료를 참고하시기 바랍니다.