RFC 2813

Internet Relay Chat: Server Protocol

Abstract

클라이언트-서버 모델을 기반으로 하지만 IRC (Internet Relay Chat) 프로토콜을 사용하면 서버가 서로 효과적으로 연결하여 네트워크를 구성 할 수 있습니다.

이 문서는 서버가 서로 통신하는 데 사용하는 프로토콜을 정의합니다. 원래는 클라이언트 프로토콜의 상위 집합 이었지만 다르게 발전했습니다.

RFC 1459 [IRC]의 일부로 1993 년 5 월에 처음 공식적으로 문서화 되었으며, 그 이후로 가져온 대부분의 변경 사항은 프로토콜 확장을 개선하는 데 중점을 두었기 때문에 이 문서에서 찾을 수 있습니다. 더 나은 확장성 덕분에 기존의 전 세계 네트워크가 계속 성장하고 이전 사양을 무시하는 크기에 도달 할 수 있었습니다.

목차

각 서버는 일반적으로 다른 모든 서버에서 알고 있지만 "호스트 마스크"를 정의하여 이름에 따라 서버를 그룹화 할 수 있습니다. 호스트 마스크 영역 내에서, 모든 서버는 호스트 마스크와 일치하는 이름을 가지며, 호스트 마스크와 일치하는 이름을 가진 다른 서버는 호스트 마스크 영역 외부의 IRC 네트워크에 연결되지 않아야 합니다. 영역 외부에 있는 서버는 영역 내부에 있는 개별 서버에 대한 지식이 없으며 대신 이름에 대한 호스트 마스크가 있는 가상 서버가 제공됩니다.

2.2 Clients

각 클라이언트에 대해 모든 서버는 다음과 같은 정보를 가져야 합니다. 네트워크 전체 고유 식별자 (형식은 클라이언트 유형에 따라 다름) 및 클라이언트가 연결된 서버.

2.2.1 Users

각 사용자는 최대 길이가 9자인 고유 한 별명으로 다른 사용자와 구별됩니다. 닉네임에 사용할 수있는 것과 사용할 수 없는 것에 대해서는 프로토콜 문법 규칙 (섹션 3.3.1)을 참조하십시오. 별명 외에도 모든 서버에는 모든 사용자에 대한 다음 정보가 있어야 합니다. 사용자가 실행중인 호스트의 이름, 해당 호스트의 사용자 이름 및 클라이언트가 연결된 서버.

2.2.2 Services

각 서비스는 별명과 서버 이름으로 구성된 서비스 이름으로 다른 서비스와 구별됩니다. 닉네임의 최대 길이는 9 자 입니다. 닉네임에 사용할 수있는 것과 사용할 수 없는 것에 대해서는 프로토콜 문법 규칙 (섹션 3.3.1)을 참조하십시오. 서비스 이름을 구성하는 데 사용되는 서버 이름은 서비스가 연결된 서버의 이름입니다. 이 서비스 이름 외에도 모든 서버는 서비스 유형을 알아야 합니다. 서비스는 식별자 형식에 따라 사용자와 다르지만 더 중요한 것은 서비스와 사용자가 서버에 대해 동일한 유형의 액세스 권한을 가지고 있지 않습니다. 서비스는 서버가 유지하는 전역 상태 정보의 일부 또는 전체를 요청할 수 있지만 제한된 명령 집합을 사용할 수 있으며 (자세한 내용은 "IRC 클라이언트 프로토콜"[IRC- 클라이언트] 참조) 채널에 참여할 수 없습니다. 마지막으로 서비스는 일반적으로 섹션 5.8에 설명 된 "홍수 통제"메커니즘의 적용을 받지 않습니다.

2.3 Channels

서비스와 마찬가지로 채널에는 [IRC-CHAN] 범위(scope)가 있으며 반드시 모든 서버에 알려지지는 않습니다. 채널 존재가 서버에 알려지면 서버는 채널 멤버와 채널 모드를 추적해야 합니다(keep track).

3. The IRC Server 사양(Specification)

3.1 Overview

여기에 설명 된 프로토콜은 서버 간 연결에 사용하기 위한 것입니다. 클라이언트 대 서버 연결에 대해서는 IRC 클라이언트 프로토콜 사양을 참조하십시오. 그러나 클라이언트 연결 (신뢰할 수 없는 것으로 간주 됨)에는 서버 연결보다 더 많은 제한이 있습니다.

3.2 Character codes (문자 코드)

특정 문자 집합이 지정되지 않았습니다. 프로토콜은 8 비트를 구성하는 8 비트로 구성된 코드 세트를 기반으로합니다. 각 메시지는 이러한 8 진수로 구성 될 수 있습니다. 그러나 일부 옥텟 값은 메시지 구분 기호로 작동하는 제어 코드에 사용됩니다. 8 비트 프로토콜에 관계없이 구분 기호와 키워드는 프로토콜이 대부분 US-ASCII 터미널 및 텔넷 연결에서 사용 가능하도록되어 있습니다. IRC의 스칸디나비아 기원으로 인해 {} | ^ 문자는 각각 [] \ ~ 문자의 소문자로 간주됩니다. 이것은 두 개의 별명 또는 채널 이름의 동등성을 판별 할 때 중요한 문제입니다.

3.3 Message

서버와 클라이언트는 응답을 생성하거나 생성하지 않을 수있는 서로 메시지를 보냅니다. 서버는 대부분 클라이언트에 대한 라우팅 작업을 수행하므로 서버 간의 대부분의 통신은 응답을 생성하지 않습니다. 각 IRC 메시지는 접두어 (OPTIONAL), 명령 및 명령 매개 변수 (최대 15 개)의 세 가지 주요 부분으로 구성 될 수 있습니다. 접두사, 명령 및 모든 매개 변수는 각각 하나의 ASCII 공백 문자 (0×20)로 구분됩니다. 접두어의 존재는 단일 선행 ASCII 콜론 문자 (':', 0×3b)로 표시되며 메시지 자체의 첫 번째 문자 여야합니다. 콜론과 접두사 사이에 공백 (공백)이 없어야합니다. 접두사는 서버에서 메시지의 실제 출처를 나타내는 데 사용됩니다. 접두사가 메시지에서 누락 된 경우 해당 접두사가 수신 된 연결에서 시작된 것으로 간주됩니다. 클라이언트는 자신에게서 메시지를 보낼 때 접두사를 사용하지 않아야합니다. 하나를 사용하는 경우 유일한 유효한 접두사는 클라이언트와 관련된 등록 된 별명입니다. 서버는 메시지를 수신 할 때 (최종적으로 가정되는) 접두사를 사용하여 소스를 식별해야합니다. 서버의 내부 데이터베이스에서 접두어를 찾을 수없는 경우 반드시 버려야하며 접두사가 메시지가 (알 수없는) 서버에서 온 것으로 표시되면 메시지를 수신 한 링크를 삭제해야합니다. 이러한 상황에서 링크를 삭제하는 것은 약간 과도하지만 네트워크의 무결성을 유지하고 향후 문제를 방지하기 위해 필요합니다. 또 다른 일반적인 오류 조건은 서버의 내부 데이터베이스에있는 접두사가 다른 소스 (일반적으로 메시지가 도착한 링크와 다른 링크에서 등록 된 소스)를 식별하는 것입니다. 메시지가 서버 링크에서 수신되고 접두사가 클라이언트를 식별하는 경우 클라이언트에 대해 KILL 메시지가 발행되어 모든 서버로 전송되어야합니다. 다른 경우에는 메시지가 도착한 링크가 클라이언트에 대해 삭제되어야하고 서버에 대해 삭제되어야합니다. 모든 경우에 메시지를 버려야합니다. 명령은 유효한 IRC 명령이거나 ASCII 텍스트로 표시되는 세 (3) 자리 숫자 여야합니다. IRC 메시지는 항상 CR-LF (Carriage Return-Line Feed) 쌍으로 끝나는 문자 줄이며, 이러한 메시지는 후행 CR-LF

를 포함한 모든 문자를 계산하여 길이가 512자를 초과하지 않아야합니다. 따라서 명령 및 해당 매개 변수에 허용되는 최대 문자 수는 510 자입니다. 연속 메시지 라인에 대한 규정이 없습니다. 현재 구현에 대한 자세한 내용은 섹션 5를 참조하십시오.

3.3.1 Message format in Augmented BNF (증강 BNF의 메시지 형식)

프로토콜 메시지는 연속적인 옥텟 스트림에서 추출되어야합니다. 현재 해결책은 메시지 구분 기호로 CR과 LF라는 두 문자를 지정하는 것입니다. 빈 메시지는 자동으로 무시되므로 추가 문제없이 메시지 사이에 CR-LF 시퀀스를 사용할 수 있습니다. 추출 된 메시지는 구성 요소 <prefix>, <command> 및 매개 변수 목록 (<params>)으로 구문 분석됩니다. 이에 대한 증강 BNF 표현은 "IRC 클라이언트 프로토콜"[IRC-CLIENT]에 있습니다. 확장 된 접두사 (["!"사용자 "@"host])는 서버 간 통신에 사용해서는 안되며 클라이언트에게 메시지의 출처에 대한보다 유용한 정보를 제공하기 위해 서버 간 메시지에만 사용됩니다. 추가 쿼리가 필요합니다.

3.4 Numeric replies (숫자 응답)

서버로 전송되는 대부분의 메시지는 일종의 응답을 생성합니다. 가장 일반적인 응답은 오류 및 일반 응답 모두에 사용되는 숫자 응답입니다. 숫자 응답은 발신자 접두사, 3 자리 숫자 및 응답 대상으로 구성된 하나의 메시지로 전송되어야합니다. 클라이언트로부터의 숫자 응답은 허용되지 않습니다. 서버에서 수신 한 이러한 메시지는 자동으로 삭제됩니다. 다른 모든 측면에서 숫자 응답은 키워드가 문자열이 아닌 3 자리 숫자로 구성된다는 점을 제외하면 일반 메시지와 같습니다. 다른 응답 목록은 "IRC 클라이언트 프로토콜"[IRC-CLIENT]에 제공됩니다.

4. Message Details (메세지 세부 정보)

IRC 서버와 클라이언트가 인식하는 모든 메시지는 IRC 클라이언트 프로토콜 사양에 설명되어 있습니다. ERR_NOSUCHSERVER 응답이 리턴되면 메시지 대상을 찾을 수 없음을 의미합니다. 서버는 해당 명령에 대해이 오류 후에 다른 응답을 보내면 안됩니다. 클라이언트가 연결된 서버는 전체 메시지를 구문 분석하여 적절한 오류를 반환해야합니다. 메시지를 구문 분석하는 동안 서버에서 치명적인 오류가 발생하면 오류가 클라이언트로 다시 전송되고 구문 분석이 종료되어야합니다. 치명적 오류는 잘못된 명령, 그렇지 않으면 서버에서 알 수없는 대상 (서버, 클라이언트 또는 채널 이름이이 범주에 해당), 매개 변수가 충분하지 않거나 잘못된 권한으로 인해 발생할 수 있습니다. 전체 매개 변수 세트가 제공되면 각 매개 변수의 유효성을 확인하고 적절한 응답을 클라이언트로 다시 보내야합니다. 항목 구분 기호로 쉼표를 사용하는 매개 변수 목록을 사용하는 메시지의 경우 각 항목에 대해 응답을 보내야합니다. 아래 예에서 일부 메시지는 전체 형식을 사용하여 나타납니다. : Name COMMAND 매개 변수 목록 이러한 예는 서버간에 전송되는 "Name"의 메시지를 나타내며, 원격 서버에서 메시지의 원래 보낸 사람 이름을 포함해야합니다. 올바른 경로를 따라 답장을 보낼 수 있습

니다. 클라이언트와 서버 간 통신에 대한 메시지 세부 사항은 "IRC 클라이언트 프로토콜" [IRC-CLIENT]에 설명되어 있습니다. 다음 페이지의 일부 섹션은 이러한 메시지 중 일부에 적용되며 서버 간 통신 또는 서버 구현에만 관련된 메시지 사양에 대한 추가 사항입니다. 여기에 소개 된 메시지는 서버 간 통신에만 사용됩니다.

4.1 Connection Registration (연결 등록)

여기에 설명 된 명령은 다른 IRC 서버와의 연결을 등록하는 데 사용됩니다.

4.1.1 Password message

Command: PASS
Parameters: <password> <version> <flags> [<options>]

PASS 명령은 '연결 암호'를 설정하는 데 사용됩니다. 연결 등록을 시도하기 전에 암호를 설정해야합니다. 현재 이것은 서버가 SERVER 명령보다 먼저 PASS 명령을 보내야 함을 의미합니다. 하나의 연결에서 하나의 PASS 명령 만 허용됩니다. 클라이언트 (예 : 사용자 또는 서비스)로부터 수신 된 경우 마지막 세 (3) 개의 매개 변수는 무시되어야합니다. 서버에서받은 경우에만 관련됩니다. <version> 매개 변수는 최소 4 자, 최대 14 자의 문자열입니다. 처음 4 개의 문자는 숫자 여야하며 메시지를 발행하는 서버가 알고있는 프로토콜 버전을 나타내야합니다. 이 문서에서 설명하는 프로토콜은 "0210"으로 인코딩 된 버전 2.10입니다. 나머지 OPTIONAL 문자는 구현에 따라 다르며 소프트웨어 버전 번호를 설명해야합니다.

<flags> 매개 변수는 최대 100 자의 문자열입니다. 문자 "|"로 구분 된 두 개의 하위 문자열로 구성됩니다. (% x7C). 존재하는 경우 첫 번째 하위 문자열은 구현의 이름이어야합니다. 참조 구현 (섹션 8, "현재 지원 및 가용성"참조)은 "IRC"문자열을 사용합니다. 식별자가 필요한 다른 구현이 작성된 경우 해당 식별자는 RFC 게시를 통해 등록되어야합니다. 두 번째 하위 문자열은 구현에 따라 다릅니다. 두 부분 문자열 모두 선택 사항이지만 문자 "|" 필수입니다. 문자 "|" 어느 부분 문자열에도 나타나지 않아야합니다.

마지막으로 마지막 매개 변수 인 <options>는 링크 옵션에 사용됩니다. 프로토콜에 의해 정의 된 유일한 옵션은 링크 압축 (문자 "Z"사용)과 남용 방지 플래그 (문자 "P"사용)입니다. 이러한 옵션에 대한 자세한 내용은 섹션 5.3.1.1 (압축 된 서버 간 링크) 및 5.3.1.2 (남용 방지 보호)를 각각 참조하십시오.

Numeric Replies:

ERR_NEEDMOREPARAMS ERR_ALREADYREGISTRED

Example:

PASS moresecretpassword 0210010000 IRC|aBgH$ Z

4.1.2 Server message

Command: SERVER

Parameters: <servername> <hopcount> <token> <info>

SERVER 명령은 새 서버를 등록하는 데 사용됩니다. 새로운 연결은 피어에 대한 서버로 자신을 소개합니다. 이 메시지는 전체 네트워크를 통해 서버 데이터를 전달하는데도 사용됩니다. 새 서버가 네트워크에 연결되면 그에 대한 정보가 전체 네트워크에 브로드 캐스트되어야 합니다.

<info> 매개 변수에는 공백 문자가 포함될 수 있습니다.

<hopcount>는 각 서버가 얼마나 멀리 떨어져 있는지에 대한 내부 정보를 모든 서버에 제공하는 데 사용됩니다. 로컬 피어의 값은 0이고 전달 된 각 서버는 값을 증가시킵니다. 전체 서버 목록을 사용하면 전체 서버 트리의 맵을 구성 할 수 있지만 호스트 마스크는이를 수행하지 못하게합니다.

<token> 매개 변수는 서버에서 식별자로 사용하는 서명되지 않은 숫자입니다. 이 식별자는 이후에 서버간에 전송되는 NICK 및 SERVICE 메시지에서 서버를 참조하는 데 사용됩니다. 서버 토큰은 사용되는 지점 간 피어링에만 의미가 있으며 해당 연결에 대해 고유해야합니다. 그들은 글로벌하지 않습니다. SERVER 메시지는 (a) 아직 등록되지 않았고 서버로 등록을 시도하는 연결 또는 (b) 다른 서버에 대한 기존 연결에서만 수락되어야합니다.이 경우 SERVER 메시지는 새로운 그 서버 뒤에있는 서버.

SERVER 명령을 수신 할 때 발생하는 대부분의 오류는 대상 호스트 (대상 SERVER)에서 연결을 종료합니다. 이러한 이벤트의 심각도 때문에 오류 응답은 일반적으로 숫자가 아닌 "ERROR"명령을 사용하여 전송됩니다.
SERVER 메시지가 구문 분석되고 수신 서버에 이미 알려진 서버를 도입하려고 시도하는 경우 해당 메시지가 도착한 연결을 닫아야합니다 (올바른 절차에 따라) 서버에 대한 중복 경로가 발생했기 때문입니다. 형성되고 IRC 트리의 비순환 적 특성이 깨집니다. 일부 조건에서는 이미 알려진 서버가 등록 된 연결이 대신 닫힐 수 있습니다. 이러한 종류의 오류는 두 번째 실행 서버의 결과 일 수도 있습니다.이 문제는 프로토콜 내에서 수정할 수 없으며 일반적으로 사람의 개입이 필요합니다.

이러한 유형의 문제는 IRC 네트워크의 일부가 매우 쉽게 분리되어 두 서버 중 하나가 각 파티션에 연결되어 두 부분이 통합되는 것을 불가능하게 만들 수 있기 때문에 특히 교활합니다.

Numeric Replies:

ERR_ALREADYREGISTRED

Example:

SERVER test.oulu.fi 1 1 :Experimental server ; 새로운 서버 test.oulu.fi가 자신을 소개하고 등록을 시도합니다.

:tolsun.oulu.fi SERVER csd.bu.edu 5 34 :BU Central Server ; 서버 tolsun.oulu.fi는 5 홉 떨어져있는 csd.bu.edu의 업 링크입니다. 토큰 "34"는 csd.bu.edu에 연결된 새로운 사용자 나 서비스를 소개 할 때 tolsun.oulu.fi에 의해 사용됩니다.

4.1.3 Nick

Command: NICK
Parameters: <nickname> <hopcount> <username> <host> <servertoken> <umode> <realname>

이 형식의 NICK 메시지는 사용자 연결에서 허용되지 않아야합니다. 그러나 IRC 네트워크에 가입 한 새로운 사용자를 다른 서버에 알리기 위해 NICK / USER 쌍 대신 사용해야합니다.

이 메시지는 실제로 NICK, USER 및 MODE [IRC-CLIENT]의 세 가지 개별 메시지의 조합입니다.

<hopcount> 매개 변수는 사용자가 홈 서버에서 얼마나 멀리 떨어져 있는지 표시하기 위해 서버에서 사용됩니다. 로컬 연결의 hopcount는 0입니다. hopcount 값은 전달 된 각 서버에서 증가합니다.

<servertoken> 매개 변수는 USER의 <servername> 매개 변수를 대체합니다 (서버 토큰에 대한 자세한 정보는 섹션 4.1.2 참조).

Examples:

NICK syrk 5 kalt millennium.stealth.net 34 +i :Christophe Kalt ; 호스트 "millennium.stealth.net"에서 서버 "34"(이전 예에서는 "csd.bu.edu")로 연결된 별명 "syrk", 사용자 이름 "kalt"를 가진 새 사용자.

:krys NICK syrk ; "IRC 클라이언트 프로토콜"[IRC-CLIENT]에 정의되어 있고 서버간에 사용되는 다른 형식의 NICK 메시지 : krys가 그의 별명을 syrk로 변경했습니다.

4.1.4 Service message

Command: SERVICE
Parameters: <servicename> <servertoken> <distribution> <type> <hopcount> <info>

SERVICE 명령은 새 서비스를 도입하는 데 사용됩니다. 이 형태의 SERVICE 메시지는 클라이언트 (등록되지 않았거나 등록 된) 연결에서 허용되지 않아야합니다. 그러나 IRC 네트워크에 가입하는 새로운 서비스를 다른 서버에 알리기 위해 서버간에 사용해야합니다.

<servertoken>은 서비스가 연결된 서버를 식별하는 데 사용됩니다. (서버 토큰에 대한 자세한 내용은 섹션 4.1.2 참조).

<hopcount> 매개 변수는 서비스가 홈 서버에서 얼마나 멀리 떨어져 있는지 표시하기 위해 서버에서 사용됩니다. 로컬 연결의 hopcount는 0입니다. hopcount 값은 전달 된 각 서버에서 증가합니다.

<distribution> 매개 변수는 서비스의 가시성을 지정하는 데 사용됩니다. 서비스는 배포판과 일치하는 이름을 가진 서버에만 알려질 수 있습니다. 일치하는 서버가 서비스에 대한 정보를 가지려면 해당 서버와 서비스가 연결된 서버 사이의 네트워크 경로는 이름이 모두 마스크와 일치하는 서버로 구성되어야합니다. 제한이없는 경우 일반 "*"가 사용됩니다.

<type> 매개 변수는 현재 향후 사용을 위해 예약되어 있습니다.

Numeric Replies:

ERR_ALREADYREGISTRED ERR_NEEDMOREPARAMS
ERR_ERRONEUSNICKNAME
RPL_YOURESERVICE RPL_YOURHOST
RPL_MYINFO

Example:

SERVICE dict@irc.fr 9 .fr 0 1 : 서버 "9"에 등록 된 프랑스어 사전 r "이 다른 서버에 공지됩니다.이 서비스는 이름이".fr "과 일치하는 서버에서만 사용할 수 있습니다

4.1.5 Quit

Command: QUIT

Parameters: [<Quit Message>]

클라이언트 세션은 종료 메시지와 함께 종료됩니다. 서버는 QUIT 메시지를 보내는 클라이언트와의 연결을 닫아야합니다. "Quit Message"가 제공되면 기본 메시지, 별명 또는 서비스 이름 대신 전송됩니다.

"netsplit"(섹션 4.1.6 참조)이 발생하면 "종료 메시지"는 공백으로 구분 된 두 서버의 이름으로 구성됩니다.

첫 번째 이름은 아직 연결되어있는 서버의 이름이고 두 번째 이름은 연결이 끊어진 서버의 이름이거나 나가는 클라이언트가 연결된 서버의 이름입니다.

<Quit Message> = ":" servername SPACE servername

"Quit Message"는 "netsplits"에 대해 특별한 의미를 갖기 때문에 서버는 클라이언트가 위에 설명 된 형식으로 <Quit Message>를 사용하는 것을 허용해서는 안됩니다.

다른 이유로 인해 클라이언트가 QUIT 명령을 실행하지 않고 클라이언트 연결이 닫히면 (예 : 클라이언트가 죽고 소켓에서 EOF가 발생하는 경우) 서버는 종료 메시지를 다음의 특성을 반영하는 일종의 메시지로 채워야합니다. 그것을 일으킨 이벤트. 일반적으로 이는 시스템 특정 오류를보고하여 수행됩니다.

Numeric Replies:

None

Examples:

:WiZ QUIT :Gone to have lunch ; Preferred message format.

4.1.6 Server quit message

Command: SQUIT
Parameters: <server> <comment>

SQUIT 메시지에는 두 가지 용도가 있습니다.

첫 번째 ("인터넷 릴레이 채팅 : 클라이언트 프로토콜"[IRC-CLIENT]에 설명 됨)는 운영자가 로컬 또는 원격 서버 링크를 끊을 수 있도록합니다. 이 형식의 메시지는 결국 서버에서 원격 서버 링크를 끊는 데 사용됩니다.

이 메시지의 두 번째 사용은 "네트워크 분할"("netsplit"이라고도 함)이 발생할 때 다른 서버에 알리는 데 필요합니다. 즉, 다른 서버에 종료 또는 중지 된 서버에 대해 알리는 것입니다. 서버가 다른 서버와의 연결을 끊으려면 다른 서버의 이름을 서버 매개 변수로 사용하여 다른 서버에 SQUIT 메시지를 보내야합니다. 그러면 종료 서버와의 연결이 닫힙니다.

<comment>는 여기에 오류 또는 유사한 메시지를 표시해야하는 서버로 채워집니다.

닫히는 연결의 양쪽에있는 두 서버 모두 해당 링크 뒤에있는 것으로 간주되는 다른 모든 서버에 대해 SQUIT 메시지 (다른 모든 서버 연결로)를 보내야합니다.

마찬가지로 QUIT 메시지는 해당 종료 링크 뒤에있는 모든 클라이언트를 대신하여 여전히 연결된 다른 서버로 전송 될 수 있습니다. 또한 "분할"로 인해 회원을 잃은 채널의 모든 채널 회원은 반드시 QUIT 메시지를 보내야합니다. 채널 구성원에게 보내는 메시지는 각 클라이언트의 로컬 서버에서 생성됩니다.

서버 연결이 조기에 종료 된 경우 (예 : 링크의 다른 쪽 끝에있는 서버가 중단 된 경우) 연결이 끊어진 것을 감지 한 서버는 나머지 네트워크에 연결이 닫 혔음을 알리고 주석 필드에 적당한 무언가를 채워야합니다.

SQUIT 메시지의 결과로 클라이언트가 제거되면 서버는 향후 별칭 충돌을 방지하기 위해 일시적으로 사용할 수없는 별칭 목록에 별칭을 추가해야합니다 (SHOULD). 이 절차에 대한 자세한 내용은 섹션 5.7 (최근 사용한 별명 추적)을 참조하십시오.

Numeric replies:

ERR_NOPRIVILEGES ERR_NOSUCHSERVER
ERR_NEEDMOREPARAMS

Example:

SQUIT tolsun.oulu.fi :Bad Link ? ; 서버 링크 tolson.oulu.fi가 "잘못된 링크"로 인해 종료되었습니다.

:Trillian SQUIT cm22.eng.umd.edu :Server out of control ; Trillian에서 "서버가 제어되지 않음"으로 인해 "cm22.eng.umd.edu"를 인터넷에서 분리하라는 메시지가 표시됩니다.

4.2 Channel operations

이 메시지 그룹은 채널, 속성 (채널 모드) 및 콘텐츠 (일반적으로 사용자) 조작과 관련이 있습니다. 이를 구현할 때 네트워크의 반대쪽 끝에있는 사용자가 궁극적으로 충돌 할 명령을 보낼 때 여러 경쟁 조건이 불가피합니다. 또한 서버는 <nick> 매개 변수가 제공 될 때마다 서버가 최근 변경된 경우 기록을 확인하도록 별명 기록을 유지해야합니다.

4.2.1 Join message

Command: JOIN
Parameters: <channel>[%x7 <modes>]
*("," <channel>[%x7 <modes>])

JOIN 명령은 클라이언트가 특정 채널 청취를 시작하는 데 사용됩니다. 클라이언트가 채널에 참여할 수 있는지 여부는 클라이언트가 연결된 로컬 서버에서만 확인됩니다. 다른 모든 서버는 다른 서버에서 명령을 받으면 자동으로 사용자를 채널에 추가합니다.
선택적으로 채널의 사용자 상태 (채널 모드 'O', 'o'및 'v')는 구분 기호로 제어 G (^ G 또는 ASCII 7)를 사용하여 채널 이름에 추가 될 수 있습니다. 메시지가 서버에서 수신되지 않은 경우 이러한 데이터는 무시되어야합니다. 이 형식은 클라이언트로 보내서는 안되며 서버간에 만 사용할 수 있으며 피해야합니다 (SHOULD).
JOIN 명령은 각 서버가 채널에있는 사용자를 찾을 수있는 위치를 알 수 있도록 모든 서버에

브로드 캐스트되어야합니다. 이를 통해 PRIVMSG 및 NOTICE 메시지를 채널에 최적으로 전달할 수 있습니다.

Numeric Replies:

ERR_NEEDMOREPARAMS ERR_BANNEDFROMCHAN
ERR_INVITEONLYCHAN ERR_BADCHANNELKEY
ERR_CHANNELISFULL ERR_BADCHANMASK
ERR_NOSUCHCHANNEL ERR_TOOMANYCHANNELS
ERR_TOOMANYTARGETS ERR_UNAVAILRESOURCE
RPL_TOPIC

Examples:

:WiZ JOIN #Twilight_zone ; JOIN message from WiZ

4.2.2 Njoin message

Command: NJOIN
Parameters: <channel> ["@@" / "@"] ["+"] <nickname> *("," ["@@" / "@"] ["+"] <nickname>)

NJOIN 메시지는 서버간에 만 사용됩니다. 이러한 메시지가 클라이언트로부터 수신되면 무시해야합니다. 두 서버가 서로 연결되어 각 채널의 채널 구성원 목록을 교환 할 때 사용됩니다.

연속 된 JOIN을 사용하여 동일한 기능을 수행 할 수 있지만이 메시지는 더 효율적이므로 대신 사용해야합니다. 접두사 "@@"는 사용자가 "채널 생성자"임을 나타내고, 문자 "@"만 "채널 운영자"를 나타내며, 문자 '+'는 사용자에게 음성 권한이 있음을 나타냅니다.

Numeric Replies:

ERR_NEEDMOREPARAMS ERR_NOSUCHCHANNEL
ERR_ALREADYREGISTRED

Examples:

:ircd.stealth.net NJOIN #Twilight_zone :@WiZ,+syrk,avalon ; ircd.stealth.net의 NJOIN 메시지 : #Twilight_zone 채널에 가입하는 사용자를 알리는 메시지 : 채널 운영자 상태가있는 WiZ, 음성 권한이있는 syrk 및 권한이 없는 avalon.

4.2.3 Mode message

MODE 메시지는 IRC의 이중 목적 명령입니다. 사용자 이름과 채널 모두 모드를 변경할 수 있습니다.

MODE 메시지를 구문 분석 할 때 전체 메시지를 먼저 구문 분석 한 다음 결과적으로 전달 된 변경 사항을 구문 분석하는 것이 좋습니다.

"채널 생성자"및 "채널 운영자"가 생성 될 수 있도록 서버가 채널 모드를 변경할 수 있어야합니다.

5. Implementation details

작성 당시이 프로토콜의 유일한 현재 구현은 IRC 서버 버전 2.10입니다. 이전 버전에서는이 문서에서 설명하는 일부 또는 모든 명령을 구현하여 많은 숫자 응답을 대신하는 NOTICE 메시지를 사용할 수 있습니다. 불행히도 이전 버전과의 호환성 요구 사항으로 인해이 문서의 일부 구현은 배치 된 내용에 따라 다릅니다. 주목할만한 차이점은 다음과 같습니다.

- 메시지의 모든 LF 또는 CR이 해당 메시지의 끝을 표시한다는 인식 (CR-LF를 요구하는 대신)

이 섹션의 나머지 부분에서는 서버를 구현하려는 사람들에게 가장 중요하지만 일부 부분은 클라이언트에도 직접 적용되는 문제를 다룹니다.

5.1 Connection 'Liveness'

연결이 끊어 지거나 응답하지 않는시기를 감지하려면 서버가 각 연결을 폴링해야합니다. PING 명령 ("IRC 클라이언트 프로토콜"[IRC-CLIENT] 참조)은 서버가 주어진 시간 동안 피어로부터 응답을받지 못하면 사용됩니다.

연결이 제 시간에 응답하지 않으면 적절한 절차에 따라 연결이 닫힙니다.

5.2 Accepting a client to server connection (클라이언트 대 서버 연결 수락)

5.2.1 Users

서버가 새 사용자 연결을 성공적으로 등록하면 등록 된 사용자 식별자 (RPL_WELCOME), 서버 이름 및 버전 (RPL_YOURHOST), 서버 생성 정보 (RPL_CREATED)를 나타내는 명확한 메시지를 사용자에게 보내야합니다. , 사용 가능한 사용자 및 채널 모드 (RPL_MYINFO), 적절한 것으로 간주 될 수있는 모든 소개 메시지를 보낼 수 있습니다.

특히 서버는 현재 사용자 / 서비스 / 서버 수 (LUSER 응답에 따라)와 마지막으로 MOTD (있을 경우 MOTD 응답에 따라)를 보냅니다.

등록을 처리 한 후 서버는 새로운 사용자의 닉네임 (NICK 메시지), 자체적으로 제공되는 기타 정보 (USER 메시지) 및 서버가 발견 할 수있는 (DNS 서버에서) 다른 서버로 전송해야합니다. 서버는 "IRC 클라이언트 프로토콜"[IRC-CLIENT]에 정의 된대로 한 쌍의 NICK 및

USER 메시지와 함께이 정보를 보내서는 안되며, 대신 섹션 4.1.3에 정의 된 확장 된 NICK 메시지를 이용해야합니다.

5.2.2 Services

새 서비스 연결을 성공적으로 등록하면 서버에는 사용자와 동일한 종류의 요구 사항이 적용됩니다. 서비스가 다소 다르지만 RPL_YOURESERVICE, RPL_YOURHOST, RPL_MYINFO 응답 만 전송됩니다.

이를 처리 한 후 서버는 서비스 (SERVICE 메시지)에서 제공하고 서버가 발견 할 수있는 (DNS 서버에서) 새로운 서비스의 별명 및 기타 정보를 다른 서버 (SERVICE 메시지)로 보내야합니다.

5.3 Establishing a server-server connection. (서버-서버 연결 설정)

문제가 발생할 수있는 영역이 많기 때문에 서버 간 연결을 설정하는 프로세스는 위험합니다.

서버가 유효한 것으로 인식 된 PASS / SERVER 쌍에 따라 연결을 수신 한 후 서버는 해당 연결에 대한 자체 PASS / SERVER 정보와 설명 된대로 알고있는 다른 모든 상태 정보로 응답해야합니다.

시작 서버가 PASS / SERVER 쌍을 수신하면 해당 서버에 대한 연결을 수락하기 전에 응답하는 서버가 올바르게 인증되었는지 확인합니다.

5.3.1 Link options

서버 링크는 공통 프로토콜 (이 문서에 정의 됨)을 기반으로하지만 특정 링크는 PASS 메시지를 사용하여 특정 옵션을 설정할 수 있습니다 (섹션 4.1.1 참조).

5.3.1.1 Compressed server to server links (압축 된 서버 대 서버 링크)

서버가 피어와 압축 된 링크를 설정하려면 옵션 매개 변수의 'Z'플래그를 PASS 메시지로 설정해야합니다. 두 서버 모두 압축을 요청하고 두 서버가 두 개의 압축 된 스트림을 초기화 할 수있는 경우 나머지 통신은 압축됩니다. 서버가 스트림 초기화에 실패하면 압축되지 않은 ERROR 메시지를 피어에 보내고 연결을 닫습니다.

압축에 사용되는 데이터 형식은 RFC 1950 [ZLIB], RFC 1951 [DEFLATE] 및 RFC 1952 [GZIP]에 설명되어 있습니다.

5.3.1.2 Anti abuse protections (남용 방지 보호)

대부분의 서버는 신뢰할 수없는 당사자 (일반적으로 사용자)의 가능한 악의적 행위에 대해 다양한 종류의 보호를 구현합니다. 일부 네트워크에서는 이러한 보호가 필수적이며 다른 네트워크에서는 불필요합니다. 모든 서버가 특정 네트워크에서 이러한 기능을 구현하고 활성화하도록 요구하기 위해 두 서버가 연결될 때 'P'플래그가 사용됩니다. 이 플래그가 있으면

서버 보호가 활성화되어 있고 서버가이를 활성화하기 위해 모든 서버 링크가 필요함을 의미합니다.

일반적으로 발견되는 보호 조치는 섹션 5.7 (최근에 사용한 별명 추적) 및 5.8 (클라이언트의 홍수 제어)에 설명되어 있습니다.

5.3.2 State information exchange when connecting (연결시 상태 정보 교환)

서버간에 교환되는 상태 정보의 순서는 필수적입니다. REQUIRED 순서는 다음과 같습니다.

* 알려진 모든 서버

* 알려진 모든 클라이언트 정보

* 알려진 모든 채널 정보.

서버에 관한 정보는 추가 SERVER 메시지, NICK 및 SERVICE 메시지가있는 클라이언트 정보, NJOIN / MODE 메시지가있는 채널을 통해 전송됩니다.

참고 : TOPIC 명령이 이전 주제 정보를 덮어 쓰므로 채널 주제는 여기에서 교환하지 않아야합니다. 따라서 기껏해야 연결의 양측이 주제를 교환 할 수 있습니다.
먼저 서버에 대한 상태 정보를 전달하면 특정 별명을 도입하는 두 번째 서버로 인한 별명 충돌 전에 이미 존재하는 서버와의 충돌이 발생합니다. IRC 네트워크는 비순환 그래프로만 존재할 수 있기 때문에 네트워크가 이미 다른 위치에서 다시 연결되었을 수 있습니다. 이 경우 서버 충돌이 발생한 위치는 네트워크가 분할되어야하는 위치를 나타냅니다.

5.4 Terminating server-client connections

클라이언트 연결이 예기치 않게 닫히면 클라이언트가 연결된 서버에서 클라이언트 대신 QUIT 메시지가 생성됩니다. 다른 메시지는 생성되거나 사용되지 않습니다.

5.5 Terminating server-server connections

SQUIT 명령 또는 "자연적"원인을 통해 서버-서버 연결이 닫히면 연결된 IRC 네트워크의 나머지 부분은 닫힘을 감지 한 서버에 의해 업데이트 된 정보를 가져야합니다. 그런 다음 종료 서버는 SQUIT 목록 (해당 연결 뒤에있는 각 서버에 대해 하나씩)을 보냅니다. (섹션 4.1.6 (SQUIT) 참조).

5.6 Tracking nickname changes

모든 IRC 서버는 최근 닉네임 변경 기록을 유지해야합니다. 이는 명령을 조작하는 명령으로 닉 체인지 경쟁 조건이 발생할 때 서버가 사물을 계속 파악할 수 있도록하는 데 중요합니다. 닉 변경을 추적해야하는 메시지는 다음과 같습니다.

* KILL (연결이 끊어지는 닉)
* 모드 (채널에서 +/- o, v)

* KICK (채널에서 제거되는 닉)

별명 변경을 확인하는 데 필요한 다른 명령은 없습니다.

5.7 Tracking recently used nicknames

이 메커니즘은 일반적으로 "닉네임 지연"으로 알려져 있으며, "네트워크 분할"/ 재 연결 및 남용으로 인한 닉네임 충돌의 수를 크게 줄이는 것으로 입증되었습니다.

닉네임 변경을 추적하는 것 외에도 서버는 최근에 사용되어 "네트워크 분할"또는 KILL 메시지의 결과로 해제 된 닉네임을 추적해야합니다. 이러한 별명은 서버 로컬 클라이언트에서 사용할 수 없으며 특정 기간 동안 (현재 사용 중이 아니더라도) 재사용 할 수 없습니다.

닉네임을 사용할 수없는 기간은 IRC 네트워크의 크기 (사용자 측면)와 일반적인 "네트워크 분할"기간을 고려하여 설정해야합니다 (SHOULD). 주어진 IRC 네트워크의 모든 서버에서 동일해야합니다.

5.8 Flood control of clients

상호 연결된 IRC 서버의 대규모 네트워크를 사용하면 네트워크에 연결된 단일 클라이언트가 네트워크를 플러딩 할뿐만 아니라 다른 사람에게 제공되는 서비스 수준을 저하시키는 연속적인 메시지 스트림을 제공하는 것이 매우 쉽습니다. 모든 '피해자'가 자신의 보호를 제공하도록 요구하는 대신, 홍수 방지는 서버에 기록되어 서비스를 제외한 모든 클라이언트에 적용됩니다. 현재 알고리즘은 다음과 같습니다.

- 클라이언트의 '메시지 타이머'가 현재 시간보다 작은 지 확인하십시오 (그렇다면 동일하게 설정).
- 클라이언트에서 존재하는 모든 데이터를 읽습니다.
- 타이머가 현재 시간보다 10 초 미만인 동안 현재 메시지를 구문 분석하고 각 메시지에 대해 2 초씩 클라이언트에 페널티를줍니다.
- 네트워크를 통해 많은 트래픽을 생성하는 특정 명령에 대해 추가 페널티를 사용할 수 있습니다.

이는 본질적으로 클라이언트가 부정적인 영향을받지 않고 2 초마다 하나의 메시지를 보낼 수 있음을 의미합니다. 서비스에도이 메커니즘이 적용될 수 있습니다.

5.9 Non-blocking lookups

실시간 환경에서는 모든 클라이언트가 공정하게 서비스를받을 수 있도록 서버 프로세스가 가능한 한 대기 시간을 최소화하는 것이 중요합니다. 분명히 모든 네트워크 읽기 / 쓰기 작업에서 비 차단 IO가 필요합니다. 정상적인 서버 연결의 경우 이는 어렵지 않았지만 서버를 차

단할 수있는 다른 지원 작업 (예 : 디스크 읽기)이 있습니다. 가능한 경우 이러한 활동은 짧은 시간 제한으로 수행해야합니다.

5.9.1 Hostname (DNS) lookups

Berkeley 및 기타의 표준 리졸버 라이브러리를 사용하면 응답 시간이 초과되는 경우에 상당한 지연이 발생했습니다. 이를 방지하기 위해 현재 구현에 대해 별도의 DNS 루틴 세트가 작성되었습니다. 루틴은 로컬 캐시를 사용하여 비 차단 IO 작업을 위해 설정 한 다음 메인 서버 IO 루프 내에서 폴링되었습니다.

5.9.2 Username (Ident) lookups

다른 프로그램에 사용하고 포함 할 수있는 수많은 ident 라이브러리 ("Identification Protocol"[IDENT] 구현)가 있지만 동기식으로 작동하고 잦은 지연이 발생하기 때문에 문제가 발생했습니다. 다시 한 번 해결책은 나머지 서버와 협력하고 비 차단 IO를 사용하여 작업하는 루틴 세트를 작성하는 것이 었습니다.

6. Current problems

이 프로토콜에는 여러 가지 인식 된 문제가 있으며, 모두 재 작성하는 동안 가까운 장래에 해결 될 것으로 기대됩니다. 현재 이러한 문제에 대한 해결책을 찾기 위한 작업이 진행 중입니다.

6.1 확장성

이 프로토콜은 대규모 경기장에서 사용할 때 충분히 확장되지 않는다는 것이 널리 알려져 있습니다. 주된 문제는 모든 서버가 다른 모든 서버와 클라이언트에 대해 알고 있고 이에 관한 정보가 변경되는 즉시 업데이트되어야한다는 요구 사항에서 비롯됩니다. 또한 두 지점 사이의 경로 길이를 최소화하고 스패닝 트리가 가능한 한 강력하게 분기되도록 서버 수를 낮게 유지하는 것이 바람직합니다.

6.2 라벨

현재 IRC 프로토콜에는 별명, 채널 이름, 서버 이름 및 서비스 이름의 4 가지 유형의 레이블이 있습니다. 네 가지 유형 각각에는 고유 한 도메인이 있으며 해당 도메인 내에서는 중복이 허용되지 않습니다. 현재 사용자가 처음 세 개 중 하나에 대한 레이블을 선택하여 충돌이 발생할 수 있습니다. 이것은 충돌하지 않는 닉에 대한 고유 한 이름에 대한 계획과 순환 트리를 허용하는 솔루션과 함께 재 작업이 필요하다는 것이 널리 알려져 있습니다.

6.2.1 별명

IRC의 닉네임 아이디어는 사용자가 채널 밖에서 서로 대화 할 때 사용하기 매우 편리하지만, 한정된 닉네임 공간 만 존재하고있는 그대로이기 때문에 여러 사람이 같은 것을 사용하고 싶어하는 경우가 드물지 않습니다. 새긴 금. 이 프로토콜을 사용하는 두 사람이 닉네임을 선택하면 둘 중 하나가 성공하지 않거나 둘 다 KILL을 사용하여 제거됩니다 ("IRC 클라이언트 프로토콜"[IRC-CLIENT]의 섹션 3.7.1 참조).

6.2.2 채널

현재 채널 레이아웃에서는 모든 서버가 모든 채널, 해당 거주자 및 속성에 대해 알고 있어야합니다. 잘 확장되지 않는 것 외에도 프라이버시 문제도 우려됩니다. 채널 충돌은 닉네임 충돌을 해결하는 데 사용되는 것과 같은 배타적 인 이벤트가 아니라 포괄적 인 이벤트 (공통 이름을 가진 채널의 두 네트에있는 사람이 그 구성원으로 간주 됨)로 처리됩니다.

이 프로토콜은 채널 충돌의 대상이 될 가능성이 거의없는 "안전 채널"을 정의합니다. 다른 채널 유형은 이전 버전과의 호환성을 위해 유지됩니다.

6.2.3 서버

서버 수는 일반적으로 사용자 및 채널 수에 비해 적지 만 현재는 서버 수가 각각 개별적으로 또는 마스크 뒤에 숨겨져 전체적으로 알려야합니다.

6.3 알고리즘

서버 코드 내의 일부 위치에서는 클라이언트 집합의 채널 목록을 확인하는 것과 같은 N ^ 2 알고리즘을 피할 수 없었습니다.
현재 서버 버전에서는 데이터베이스 일관성 검사가 거의 없으며 대부분의 경우 각 서버는 인접 서버가 올바르다 고 가정합니다. 이것은 연결 서버에 버그가 있거나 기존 네트워크에 모순을 일으키려고 할 때 큰 문제에 대한 문을 열어줍니다.

현재 고유 한 내부 및 글로벌 레이블이 없기 때문에 수많은 경쟁 조건이 존재합니다. 이러한 경쟁 조건은 일반적으로 메시지가 IRC 네트워크를 통과하고 영향을 미치는 데 시간이 걸리는 문제로 인해 발생합니다. 고유 한 레이블로 변경하더라도 채널 관련 명령이 중단되는 문제가 있습니다.

7. Security Considerations

7.1 Authentication

서버에는 들어오는 연결을 인증하는 두 가지 방법 (일반 텍스트 암호 및 DNS 조회) 만 있습니다. 이러한 방법은 약하고 안전하지 않은 것으로 널리 알려져 있지만 과거에는 그 조합으로 충분하다는 것이 입증되었습니다.

* 공용 네트워크는 일반적으로 정확한 인증없이 몇 가지 제한 사항만으로 사용자 연결을 허용합니다.

* 통제 된 환경에서 작동하는 사설 네트워크는 종종 인터넷에서 사용할 수없는 자체 개발 인증 메커니즘 (신뢰할 수있는 ID 서버 [IDENT] 또는 기타 독점 메커니즘)을 사용합니다.

IRC 운영자의 인증에도 동일한 설명이 적용됩니다.

또한 수년 동안 더 강력한 인증에 대한 실제 수요가 없었고 사용자를 안전하게 인증하기위한 더 나은 수단을 제공하려는 실제 노력이 없었지만 현재 프로토콜은 기반으로 외부 인증 방법을 쉽게 플러그인 할 수있을만큼 충분히 제공합니다. 연결시 클라이언트가 서버에 제출할 수 있는 정보 : 닉네임, 사용자 이름, 비밀번호.

7.2 Integrity (무결성)

IRC 프로토콜의 PASS 및 OPER 메시지는 일반 텍스트로 전송되기 때문에 스트림 계층 암호화 메커니즘 (예 : "TLS 프로토콜"[TLS])을 사용하여 이러한 트랜잭션을 보호 할 수 있습니다.

10. References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[IRC] Oikarinen, J. and D. Reed, "Internet Relay Chat Protocol", RFC 1459, May 1993.

[IRC-ARCH] Kalt, C., "Internet Relay Chat: Architecture", RFC 2810, April 2000.

[IRC-CLIENT] Kalt, C., "Internet Relay Chat: Client Protocol", RFC 2812, April 2000.

[IRC-CHAN] Kalt, C., "Internet Relay Chat: Channel Management", RFC 2811, April 2000.

[ZLIB] Deutsch, P. and J-L. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.
[DEFLATE] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[GZIP] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, May 1996.

[IDENT] St. Johns, M., "The Identification Protocol", RFC 1413, February 1993.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol", RFC 2246, January 1999.