네트워크 워킹그룹
의견 요청: 1459

J. 오이카리넨
D. 리드
1993년 5월

인터넷 릴레이 채팅 프로토콜

이 메모의 상태

이 메모는 인터넷의 실험 프로토콜을 정의합니다.
지역 사회. 개선을 위한 논의와 제안을 요청합니다.
"IAB 공식 프로토콜" 최신판을 참조하세요.
표준"을 참조하여 이 프로토콜의 표준화 상태 및 상태를 확인하세요.
이 메모의 배포는 무제한입니다.

추상적인

IRC 프로토콜은 지난 4년에 걸쳐 개발되었습니다.
처음에는 BBS 사용자가 채팅할 수 있는 수단으로 구현되었습니다.
그들 자신. 이제 전세계적인 서버 네트워크를 지원하고
성장에 대처하기 위해 노력하고 있습니다. 지난 2년 동안,
주요 IRC 네트워크에 연결된 평균 사용자 수는
10배나 성장했습니다.

IRC 프로토콜은 가장 간단한 클라이언트를 갖춘 텍스트 기반 프로토콜입니다.
서버에 연결할 수 있는 소켓 프로그램입니다.

목차

1. 소개

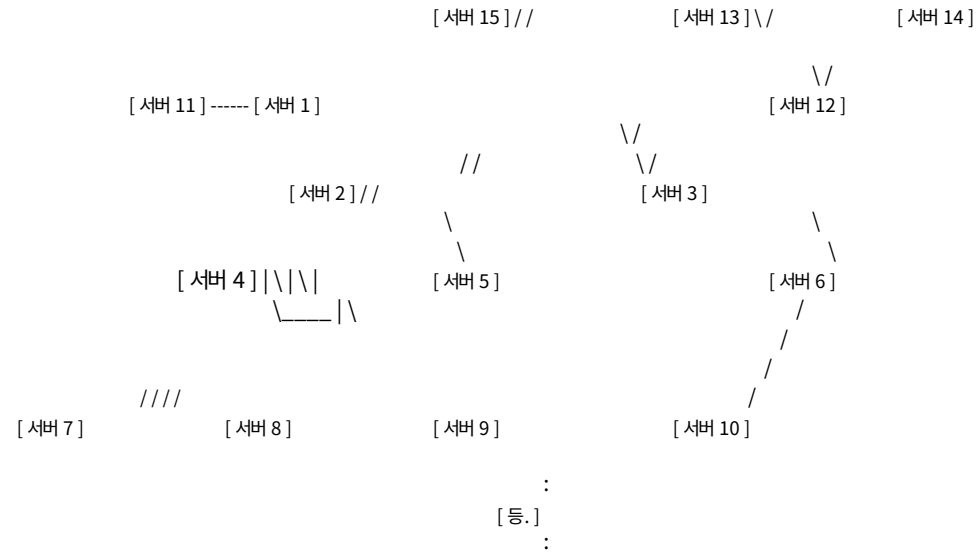
IRC(Internet Relay Chat) 프로토콜은 다음을 통해 설계되었습니다.
텍스트 기반 회의에 사용할 수 있는 연수입니다. 이 문서
현재 IRC 프로토콜을 설명합니다.

IRC 프로토콜은 TCP/IP를 사용하는 시스템에서 개발되었습니다.
네트워크 프로토콜을 유지해야 한다는 요구 사항은 없지만
그것이 작동하는 유일한 영역.

IRC 자체는 원격 회의 시스템입니다.
클라이언트-서버 모델)은 많은 시스템에서 실행하는 데 적합합니다.
분산 방식으로. 일반적인 설정에는 단일 프로세스가 포함됩니다.
(서버) 클라이언트(또는 다른 서버)의 중앙 지점을 형성합니다.
연결하려면 필요한 메시지 전달/다중화를 수행해야 합니다.
그리고 다른 기능.

1.1 서버

서버는 IRC의 백본을 형성하여 다음 지점을 제공합니다.
클라이언트는 서로 대화하기 위해 연결할 수 있으며,
연결하여 IRC 네트워크를 형성하는 서버입니다. 유일한 네트워크
IRC 서버에 허용되는 구성은 스패닝 트리 구성입니다.
그림 1] 각 서버는 나머지 서버의 중앙 노드 역할을 합니다.
그물로 본다.



[그림 1. IRC 서버 네트워크 구성]

1.2 클라이언트

클라이언트는 다른 서버가 아닌 서버에 연결되는 모든 것입니다. 각 클라이언트는 최대 9자의 고유한 별명으로 다른 클라이언트와 구별됩니다. 닉네임에 사용할 수 있는 것과 사용하지 않을 수 있는 것에 대한 프로토콜 문법 규칙을 참조하세요. 별명 외에도 모든 서버에는 모든 클라이언트에 대한 다음 정보가 있어야 합니다. 즉, 클라이언트가 실행 중인 호스트의 실제 이름, 해당 호스트에 있는 클라이언트의 사용자 이름, 클라이언트가 연결된 서버입니다.

1.2.1 연산자

IRC 네트워크 내에서 합리적인 양의 주문이 유지되도록 하기 위해 특별한 클래스의 클라이언트(운영자)가 네트워크에서 일반적인 유지 관리 기능을 수행하도록 허용됩니다. 운영자에게 부여된 권한은 '위험'하다고 간주될 수 있지만 그럼에도 불구하고 필수입니다. 운영자는 잘못된 네트워크 라우팅의 장기간 사용을 방지하기 위해 필요에 따라 서버 연결을 끊고 다시 연결하는 등 기본적인 네트워크 작업을 수행할 수 있어야 합니다. 이러한 필요성을 인식하여, 여기에 논의된 프로토콜은 운영자가 그러한 기능을 수행할 수만 있도록 제공합니다. 섹션 4.1.7(SQUIT) 및 4.3.5(CONNECT)를 참조하세요.

더욱 논란이 되는 운영자의 권한은 '강제로' 연결된 네트워크에서 사용자를 제거할 수 있는 능력입니다. 즉 운영자는 클라이언트와 서버 간의 연결을 닫을 수 있습니다. 남용은 파괴적이고 성가시기 때문에 이에 대한 정당화는 미묘합니다. 이러한 유형의 작업에 대한 자세한 내용은 섹션 4.6.1(KILL)을 참조하세요.

1.3 채널

채널은 해당 채널로 주소가 지정된 메시지를 모두 수신하는 하나 이상의 클라이언트로 구성된 명명된 그룹입니다. 채널은 첫 번째 클라이언트가 참여할 때 암시적으로 생성되고, 마지막 클라이언트가 채널을 떠날 때 채널은 더 이상 존재하지 않습니다. 채널이 존재하는 동안 모든 클라이언트는 채널 이름을 사용하여 채널을 참조할 수 있습니다.

채널 이름은 최대 200자 길이의 문자열('&' 또는 '#' 문자로 시작)입니다. 첫 번째 문자가 '&' 또는 '#'이어야 한다는 요구 사항을 제외하고; 채널 이름에 대한 유일한 제한은 공백(' '), 컨트롤 G(^G 또는 ASCII 7) 또는 프로토콜에서 목록 항목 구분 기호로 사용되는 쉼표(',')를 포함할 수 없다는 것입니다. .

이 프로토콜에서는 두 가지 유형의 채널이 허용됩니다. 하나는 모든 서버에 알려진 분산 채널입니다.

네트워크에 연결되었습니다. 이러한 채널은 채널이 존재하는 서버의 유일한 클라이언트가 채널에 참여할 수 있다는 첫 번째 문자로 표시됩니다. 이는 선행 '&' 문자로 구별됩니다. 이 두 가지 유형 외에도 개별 채널의 특성을 변경하는 데 사용할 수 있는 다양한 채널 모드가 있습니다. 이에 대한 자세한 내용은 섹션 4.2.3(MODE 명령)을 참조하십시오.

새 채널을 만들거나 기존 채널의 일부가 되려면 사용자는 채널에 가입해야 합니다. 가입하기 전에 채널이 존재하지 않으면 채널이 생성되고 생성한 사용자가 채널 운영자가 됩니다. 채널이 이미 존재하는 경우 해당 채널에 대한 JOIN 요청이 수락되는지 여부는 채널의 현재 모드에 따라 다릅니다. 예를 들어 채널이 초대 전용인 경우 (+i) 초대를 받은 경우에만 참여할 수 있습니다. 프로토콜의 일부로 사용자는 동시에 여러 채널의 일부가 될 수 있지만 숙련된 사용자와 초보 사용자 모두에게 충분한 채널 수는 10개로 제한되는 것이 좋습니다. 이에 대한 자세한 내용은 섹션 8.13을 참조하십시오.

두 서버 간의 분할로 인해 IRC 네트워크가 분리되면 각 측면의 채널은 분할된 각 측면의 서버에 연결된 클라이언트로만 구성되며 분할된 한쪽에서는 더 이상 존재하지 않을 수도 있습니다. 분할이 치유되면 연결 서버는 각 채널에 있다고 생각되는 사람과 해당 채널의 모드를 서로에게 알립니다. 채널이 양쪽에 존재하는 경우 JOIN 및 MODE는 포괄적인 방식으로 해석되어 새 연결의 양쪽이 채널에 있는 클라이언트와 채널의 모드에 대해 동의하게 됩니다.

1.3.1 채널 운영자

특정 채널의 채널 운영자("chop" 또는 "chanop"라고도 함)는 해당 채널을 '소유'하는 것으로 간주됩니다. 이러한 상태를 인식하여 채널 운영자에게는 채널에 대한 통제력과 일종의 건전성을 유지할 수 있는 특정 권한이 부여됩니다.

채널 소유자로서 채널 운영자는 자신의 행동에 대한 이유를 요구하지 않습니다. 하지만 그들의 행동이 일반적으로 반사회적이거나 기타 폭력적인 경우 IRC 운영자에게 개입을 요청하거나 사용자가 그냥 나가고 가는 것이 합리적일 수 있습니다. 다른 곳에서 자신의 채널을 형성합니다.

채널 운영자만 사용할 수 있는 명령은 다음과 같습니다.

KICK - 채널에서 클라이언트 퇴출
MODE - 채널 모드 변경
INVITE - 초대 전용 채널에 클라이언트를 초대합니다(모드 +i).
TOPIC - 모드 +t 채널에서 채널 주제 변경

채널 운영자는 채널과 연결될 때마다(예: NAMES, WHO 및 WHOIS 명령에 응답) 별명 옆에 '@' 기호로 식별됩니다.

2. IRC 사양

2.1 개요

여기에 설명된 프로토콜은 서버 대 서버 연결과 클라이언트 대 서버 연결 모두에 사용됩니다. 그러나 클라이언트 연결(신뢰할 수 없는 것으로 간주됨)에는 서버 연결보다 더 많은 제한이 있습니다.

2.2 문자 코드

특정 문자 집합이 지정되지 않았습니다. 프로토콜은 옥텟을 구성하는 8비트로 구성된 코드 세트를 기반으로 합니다. 각 메시지는 이러한 옥텟으로 구성될 수 있습니다. 그러나 일부 옥텟 값은 메시지 구분 기호 역할을 하는 제어 코드에 사용됩니다.

8비트 프로토콜임에도 불구하고 구분 기호와 키워드는 USASCII 터미널과 텔넷 연결에서 프로토콜을 대부분 사용할 수 있도록 되어 있습니다.

IRC는 스칸디나비아 출신이기 때문에 문자 {}| 각각 []\ 문자에 해당하는 소문자로 간주됩니다. 이는 두 별명의 동등성을 결정할 때 중요한 문제입니다.

2.3 메시지

서버와 클라이언트는 응답을 생성할 수도 있고 생성하지 않을 수도 있는 메시지를 서로 보냅니다. 이후 섹션에 설명된 대로 메시지에 유효한 명령이 포함된 경우 클라이언트는 지정된 대로 응답을 기대해야 하지만 응답을 영원히 기다리는 것은 권장되지 않습니다. 클라이언트에서 서버로, 서버에서 서버로의 통신은 본질적으로 비동기적입니다.

각 IRC 메시지는 접두사(선택 사항), 명령 및 명령 매개 변수(최대 15개)의 세 가지 주요 부분으로 구성될 수 있습니다. 접두사, 명령 및 모든 매개변수는 하나 이상의 ASCII 공백 문자(0x20)로 구분됩니다.

접두어의 존재는 메시지 자체의 첫 번째 문자여야 하는 단일 선행 ASCII 콜론 문자(':', 0x3b)로 표시됩니다. 콜론과 접두사 사이에는 공백(공백)이 없어야 합니다. 접두사는 서버에서 실제 내용을 나타내는 데 사용됩니다.

메시지의 출처. 접두사가 메시지에 누락된 경우 접두사는 수신된 연결에서 시작된 것으로 간주됩니다. 클라이언트는 자신에게서 메시지를 보낼 때 접두사를 사용해서는 안 됩니다. 접두사를 사용하는 경우 유일한 유효한 접두사는 클라이언트와 연결된 등록된 별명입니다. 접두어로 식별된 소스를 서버의 내부 데이터베이스에서 찾을 수 없거나 소스가 메시지가 도착한 링크와 다른 링크에서 등록된 경우 서버는 메시지를 자동으로 무시해야 합니다.

명령은 유효한 IRC 명령이거나 ASCII 텍스트로 표시되는 3자리 숫자여야 합니다.

IRC 메시지는 항상 CR-LF(캐리지 리턴 - 라인 피드) 쌍으로 끝나는 문자 라인이며, 이러한 메시지 길이는 후행 CR-LF를 포함한 모든 문자를 포함하여 512자를 초과할 수 없습니다. 따라서 명령과 해당 매개변수에 허용되는 최대 문자 수는 510자입니다. 연속 메시지 라인에 대한 규정은 없습니다. 현재 구현에 대한 자세한 내용은 섹션 7을 참조하세요.

2.3.1 '유사' BNF의 메시지 형식

프로토콜 메시지는 연속적인 옥텟 스트림에서 추출되어야 합니다. 현재 해결 방법은 CR과 LF라는 두 문자를 메시지 구분 기호로 지정하는 것입니다. 빈 메시지는 자동으로 무시되므로 추가 문제 없이 메시지 사이에 CR-LF 시퀀스를 사용할 수 있습니다.

추출된 메시지는 <prefix>, <command> 구성 요소와 <middle> 또는 <trailing> 구성 요소와 일치하는 매개 변수 목록으로 구문 분석됩니다.

이에 대한 BNF 표현은 다음과 같습니다.

<메시지> ::= [':' <접두사> <SPACE>] <명령> <매개변수> <crlf> <접두사> ::= <서버 이름> | <닉네임> ['!' <사용자>] ['@' <호스트>] <명령> ::= <문자> { <문자> } | <번호> <번호> <번호> <SPACE> <매개변수> ::= <SPACE> [':' <후행> | <중간> <매개변수>]

::= ' '{' ' }

<middle> ::= <SPACE를 포함하지 않는 모든 *비어 있지 않은* 옥텟 시퀀스
또는 NUL, CR 또는 LF 중 첫 번째는 ':'이 아닐 수 있습니다.>
<후행> ::= <포함하지 않는 임의의, 아마도 *비어* 있는 옥텟 시퀀스
NUL 또는 CR 또는 LF>

<crlf> ::= CR LF

노트:

1) <SPACE>는 SPACE 문자(0x20)로만 구성됩니다.
특히 TABULATION 및 기타 모든 컨트롤에 유의하세요.
문자는 공백이 아닌 것으로 간주됩니다.

2) 매개변수 목록을 추출한 후 모든 매개변수가 동일하며,
<middle> 또는 <trailing>과 일치하는지 여부. <트레일링>은 단지
매개변수 내에 SPACE를 허용하는 구문적 트릭입니다.

삼) 매개변수 문자열에 CR, LF가 나타날 수 없다는 점은
메시지 프레이밍의 인공물일 뿐입니다. 이는 나중에 변경될 수 있습니다.

4) NUL 문자는 메시지 프레이밍에서 특별하지 않습니다.
기본적으로 매개변수 내부에 포함될 수 있지만
일반적인 C 문자열 처리에 추가 복잡성이 발생합니다. 그러므로
메시지 내에서는 NUL이 허용되지 않습니다.

5) 마지막 매개변수는 빈 문자열일 수 있습니다.

6) 확장 접두사(['!' <user>] ['@' <host>])를 사용해야 합니다.
서버 간 통신에는 사용되지 않으며
제공하기 위해 서버에서 클라이언트로의 메시지를 위한 것입니다.
메시지가 누구인지에 대한 더 유용한 정보를 가진 클라이언트
추가 문의 없이도 가능합니다.

대부분의 프로토콜 메시지는 추가 의미와 구문을 지정합니다.
추출된 매개변수 문자열은
목록. 예를 들어, 많은 서버 명령은 첫 번째
명령 뒤의 매개변수는 대상 목록입니다.
설명:

<대상> <대상> <채널> <서버 이름> ::= <호스트>
::= <대상> ["," <대상>]
::= <채널> | <사용자> '@' <서버 이름> | <닉네임> | <마스크>
::= ('#' | '&') <chstring>
<host> ::= 허용되는 호스트 이름에 대한 자세한 내용은 RFC 952 [DNS:4]를 참조하세요.
<nick> ::= <문자> { <문자> | <번호> | <특수> }
<mask> ::= ('#' | '$') <chstring>
<chstring> ::= <SPACE, BELL, NUL, CR, LF를 제외한 모든 8비트 코드
쉼표(',')>

다른 매개변수 구문은 다음과 같습니다.

<사용자> ::= <비백인> { <비백인> }
<문자> <번호> ::= 'a' ... 'z' | 'A' 'Z' ...
<특수문자> ::= '0' ... '9'
::= '-' | '[' | ']' | '\' | '`' | '^' | '{' | '}'

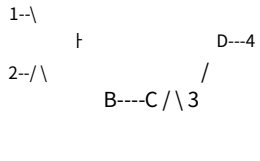
<nonwhite> ::= <SPACE(0x20), NUL(0x0), CR(0xd), LF(0xa)를 제외한 모든 8비트 코드>

2.4 숫자 답장

서버로 전송되는 대부분의 메시지는 일종의 응답을 생성합니다. 가장 일반적인 응답은 오류 및 일반 응답 모두에 사용되는 숫자 응답입니다. 숫자 응답은 보낸 사람 접두사, 세 자리 숫자 및 응답 대상으로 구성된 하나의 메시지로 전송되어야 합니다. 숫자 응답은 클라이언트에서 시작될 수 없습니다. 서버에서 수신한 모든 메시지는 자동으로 삭제됩니다. 다른 모든 측면에서 숫자 응답은 키워드가 일련의 문자가 아닌 3자리 숫자로 구성된다는 점을 제외하면 일반 메시지와 같습니다. 다양한 답변 목록은 섹션 6에 나와 있습니다.

3. IRC 개념.

이 섹션에서는 IRC 프로토콜 구성의 실제 개념과 현재 구현이 다양한 메시지 클래스를 전달하는 방법을 설명합니다.



서버: A, B, C, D, E 클라이언트: 1, 2, 3, 4

[그림 2. 소규모 IRC 네트워크 샘플]

3.1 일대일 커뮤니케이션

일대일 통신은 일반적으로 클라이언트에 의해서만 수행됩니다. 왜냐하면 대부분의 서버 간 트래픽은 서버가 서로 통신한 결과가 아니기 때문입니다. 클라이언트가 서로 통신할 수 있는 안전한 수단을 제공하려면 모든 서버가 클라이언트에 도달하기 위해 스패닝 트리를 따라 정확히 한 방향으로 메시지를 보낼 수 있어야 합니다. 전달되는 메시지의 경로는 스패닝 트리의 두 지점 사이의 최단 경로입니다.

다음 예는 모두 위의 그림 2를 참조합니다.

예 1: 클라이언트 1
과 2 사이의 메시지는 서버 A에서만 볼 수 있으며 서버 A는 이를 클라이언트 2로 직접 보냅니다.

예 2: 클라이언트 1
과 3 사이의 메시지는 서버 A & B와 클라이언트 3에 표시됩니다. 다른 클라이언트나 서버는 메시지를 볼 수 없습니다.

예제 3: 클라이언트 2
와 4 사이의 메시지는 서버 A, B, C, D와 클라이언트 4에서만 볼 수 있습니다.

3.2 일대다

IRC의 주요 목표는 쉽고 효율적인 회의(일대다 대화)가 가능한 포럼을 제공하는 것입니다. IRC는 이를 달성하기 위한 여러 가지 수단을 제공하며 각각은 고유한 목적을 달성합니다.

3.2.1 목록으로

일대다 대화의 가장 효율적인 스타일은 클라이언트가 사용자 '목록'과 대화하는 것입니다. 이 작업이 수행되는 방법은 거의 자명합니다. 클라이언트는 메시지가 전달될 대상 목록을 제공하고 서버는 이를 분할하여 지정된 각 대상에 별도의 메시지 복사본을 전달합니다.

이는 대상 목록이 분할되고 중복 항목이 각 경로로 전송되지 않는지 확인하지 않고 발송되기 때문에 그룹을 사용하는 것만큼 효율적이지 않습니다.

3.2.2 그룹(채널)으로

IRC에서 채널은 멀티캐스트 그룹과 동일한 역할을 갖습니다. 이들의 존재는 동적(사람들이 채널에 가입하고 탈퇴함에 따라 왔다 갔다 함)이며 채널에서 수행되는 실제 대화는 특정 채널에서 사용자를 지원하는 서버로만 전송됩니다. 동일한 채널의 서버에 여러 사용자가 있는 경우 메시지 텍스트는 해당 서버로 한 번만 전송된 다음 채널의 각 클라이언트로 전송됩니다. 그런 다음 원본 메시지가 펼쳐져 채널의 각 구성원에 도달할 때까지 각 클라이언트-서버 조합에 대해 이 작업이 반복됩니다.

다음 예는 모두 그림 2를 참조합니다.

예 4: 클라이언트가 1
개인 모든 채널. 채널에 대한 메시지는 서버로 이동한 후 다른 곳으로 이동하지 않습니다.

예시 5: 채널에 2개의
클라이언트가 있습니다. 모든 메시지는 마치 채널 외부의 두 클라이언트 사이의 비공개 메시지인 것처럼 경로를 통과합니다.

예 6: 채널의 클라이
언트 1, 2, 3. 채널에 대한 모든 메시지는 모든 클라이언트로 전송되며, 단일 클라이언트에 대한 개인 메시지인 경우 메시지가 통과해야 하는 서버에
만 전송됩니다. 클라이언트 1이 메시지를 보내면 클라이언트 2로 돌아가고 서버 B를 통해 클라이언트 3으로 돌아갑니다.

3.2.3 호스트/서버 마스크로

IRC 운영자에게 대규모 관련 사용자에게 메시지를 보내는 메커니즘을 제공하기 위해 호스트 및 서버 마스크 메시지가 제공됩니다. 이 메시지는 마스크의 호
스트 또는 서버 정보와 일치하는 사용자에게 전송됩니다. 메시지는 채널과 유사한 방식으로 사용자가 있는 위치로만 전송됩니다.

3.3 일대다

일대다 유형의 메시지는 모든 클라이언트나 서버 또는 둘 다에 전송되는 브로드캐스트 메시지로 더 잘 설명됩니다. 사용자와 서버로 구성된 대규모
네트워크에서는 단일 메시지로 인해 원하는 모든 대상에 도달하기 위해 네트워크를 통해 많은 트래픽이 전송될 수 있습니다.

일부 메시지의 경우 각 서버가 보유한 상태 정보가 서버 간에 합리적으로 일관되도록 모든 서버에 브로드캐스트하는 것 외에는 옵션이 없습니다.

3.3.1 클라이언트 대 클라이언트

단일 메시지에서 다른 모든 클라이언트로 메시지가 전송되는 메시지 클래스는 없습니다.

3.3.2 클라이언트-서버

상태 정보(예: 채널 멤버십, 채널 모드, 사용자 상태 등)를 변경하는 대부분의 명령은 기본적으로 모든 서버에 전송되어야 하며 이 배포는 클라이언트
에 의해 변경될 수 없습니다.

3.3.3 서버 대 서버.

서버 간 대부분의 메시지는 모든 '다른' 서버에 배포되지만 이는 사용자, 채널 또는 서버에 영향을 미치는 모든 메시지에만 필요합니다. 기본적으로 들
어있는 아이템들이기 때문에

IRC, 서버에서 발생하는 거의 모든 메시지는 연결된 다른 모든 서버로 브로드캐스트됩니다.

4. 메시지 세부정보

다음 페이지에는 IRC 서버와 클라이언트가 인식하는 각 메시지에 대한 설명이 있습니다. 이 섹션에 설명된 모든 명령은 이 프로토콜에 대한 모든 서버에서 구현되어야 합니다.

ERR_NOSUCHSERVER 응답이 나열된 경우 이는 <server> 매개변수를 찾을 수 없음을 의미합니다. 서버는 해당 명령에 대해 이후에 다른 응답을 보내서는 안 됩니다.

클라이언트가 연결된 서버는 전체 메시지를 구문 분석하여 적절한 오류를 반환해야 합니다. 서버가 메시지를 구문 분석하는 동안 치명적인 오류가 발생하면 오류를 클라이언트로 다시 전송하고 구문 분석을 종료해야 합니다. 치명적인 오류는 잘못된 명령, 서버에 알려지지 않은 대상(서버, 별명 또는 채널 이름이 이 범주에 해당), 매개변수가 충분하지 않거나 잘못된 권한으로 간주될 수 있습니다.

전체 매개변수 세트가 제공되면 각 매개변수의 유효성을 확인하고 적절한 응답을 클라이언트로 다시 보내야 합니다. 쉼표를 항목 구분 기호로 사용하는 매개변수 목록을 사용하는 메시지의 경우 항목별로 응답을 보내야 합니다.

아래 예에서 일부 메시지는 전체 형식을 사용하여 표시됩니다.

:이름 COMMAND 매개변수 목록

이러한 예는 서버 간에 전송되는 "이름"의 메시지를 나타내며, 원격 서버가 올바른 경로를 따라 응답을 다시 보낼 수 있도록 메시지의 원래 보낸 사람의 이름을 포함하는 것이 중요합니다.

4.1 연결 등록

여기에 설명된 명령은 IRC 서버와의 연결을 사용자 또는 서버로 등록하고 올바르게 연결을 끊는 데 사용됩니다.

"PASS" 명령은 클라이언트 또는 서버 연결을 등록하는 데 필요하지 않지만 서버 메시지 또는 NICK/USER 조합의 후자 앞에 와야 합니다. 실제 연결에 일정 수준의 보안을 제공하려면 모든 서버 연결에 비밀번호를 사용하는 것이 좋습니다. 클라이언트의 권장 등록 순서는 다음과 같습니다.

1. 패스 메시지 2. 닉 메시지 3. 사
용자 메시지

4.1.1 비밀번호 메시지

명령: 통과
매개변수: <비밀번호>

PASS 명령은 '연결 비밀번호'를 설정하는 데 사용됩니다. 연결 등록을 시도하기 전에 비밀번호를 설정할 수 있으며 설정해야 합니다. 현재 이
를 위해서는 클라이언트가 NICK/USER 조합을 보내기 전에 PASS 명령을 보내야 하며 서버는 *반드시* SERVER 명령 전에 PASS 명령을 보내
야 합니다. 제공된 비밀번호는 C/N 라인(서버의 경우) 또는 I 라인(클라이언트의 경우)에 포함된 비밀번호와 일치해야 합니다. 등록하기 전에 여러 개의 PASS
명령을 보낼 수 있지만 마지막에 보낸 명령만 확인에 사용되며 등록 후에는 변경할 수 없습니다. 숫자 답장:

ERR_NEEDMOREPARAMS ERR_이미 등록됨

예:

여기에 비밀번호를 입력하세요.

4.1.2 닉 메시지

명령: NICK
매개변수: <닉네임> [<홉수>]

NICK 메시지는 사용자에게 닉네임을 부여하거나 이전 닉네임을 변경할 때 사용됩니다. <hopcount> 매개변수는 홈 서버에서 닉이 얼마나 멀리 떨어져 있
는지를 나타내기 위해 서버에서만 사용됩니다. 로컬 연결의 홉 수는 0입니다. 클라이언트가 제공한 경우 무시해야 합니다.

다른 클라이언트의 동일한 닉네임을 이미 알고 있는 서버에 NICK 메시지가 도착하면 닉네임 충돌이 발생합니다.

별명 충돌의 결과로 별명의 모든 인스턴스가 서버의 데이터베이스에서 제거되고 KILL 명령이 실행되어 다른 모든 서버의 데이터베이스에서 별명을 제거합
니다. 충돌을 일으키는 NICK 메시지가 닉네임 변경인 경우 원래(이전) 닉도 제거해야 합니다.

서버가 직접 연결된 클라이언트로부터 동일한 NICK를 수신하면 로컬 클라이언트에 ERR_NICKCOLLISION을 발행하고 NICK 명령을 삭제하
고 종료를 생성하지 않을 수 있습니다.

숫자 답장:

ERR_NONICKNAMEGIVEN ERR_ERRONEUSNICKNAME
ERR_NICKNAMEINUSE ERR_NICKCOLLISION

예:

닉 위즈 ; 새로운 닉네임 '위즈'를 소개합니다.

:WIZ NICK 킬로이 ; WiZ는 별명을 Kilroy로 바꿨습니다.

4.1.3 사용자 메시지

명령: 사용자
매개변수: <사용자 이름> <호스트 이름> <서버 이름> <실제 이름>

USER 메시지는 연결 시작 시 새 사용자의 사용자 이름, 호스트 이름, 서버 이름 및 실제 이름을 지정하는 데 사용됩니다. 또한 USER와 NICK가 클라이언트로부터 수신된 후에만 사용자가 등록되기 때문에 IRC에 도착하는 새 사용자를 나타내기 위해 서버 간 통신에도 사용됩니다.

서버 간 USER 앞에는 클라이언트의 NICKname이 붙어야 합니다.
USER 명령이 직접 연결된 클라이언트에서 올 때(보안상의 이유로) 호스트 이름과 서버 이름은 일반적으로 IRC 서버에 의해 무시되지만 서버 간 통신에는 사용됩니다. 이는 새로운 사용자가 네트워크의 나머지 부분에 소개될 때 동행하는 USER가 전송되기 전에 NICK가 항상 원격 서버로 전송되어야 함을 의미합니다.

realname 매개변수는 공백 문자를 포함할 수 있고 이것이 인식되도록 하려면 콜론(':') 접두어를 붙여야 하기 때문에 마지막 매개변수여야 한다는 점에 유의해야 합니다.

클라이언트가 USER 메시지에만 의존하여 사용자 이름에 대해 거짓말을 하기가 쉽기 때문에 "Identity Server"를 사용하는 것이 좋습니다. 사용자가 연결하는 호스트에 이러한 서버가 활성화되어 있으면 사용자 이름은 "Identity Server"의 응답에서와 같이 설정됩니다.

숫자 답장:

ERR_NEEDMOREPARAMS ERR_이미 등록됨

예:

USER 게스트 tolmoon tolsun :Ronnie Reagan

; 사용자 이름은 "guest"이고 실명은 "Ronnie Reagan"으로 등록된 사용자입니다.

:testnick 사용자 손님 tolmoon tolsun :Ronnie Reagan ; USER 명령이 속한 별명을 가진 서버 간 메시지

4.1.4 서버 메시지

명령: 서버
매개변수: <서버 이름> <홉 수> <정보>

서버 메시지는 새 연결의 다른 쪽 끝이 서버임을 서버에 알리는 데 사용됩니다. 이 메시지는 전체 네트워크를 통해 서버 데이터를 전달하는 데에도 사용됩니다. 새로운 서버가 네트워크에 연결되면 해당 서버에 대한 정보가 전체 네트워크에 방송됩니다. <hopcount>는 모든 서버가 얼마나 멀리 떨어져 있는지에 대한 내부 정보를 모든 서버에 제공하는 데 사용됩니다. 전체 서버 목록을 사용하면 전체 서버 트리의 맵을 구성할 수 있지만 호스트 마스크로 인해 이 작업이 수행되지 않습니다.

SERVER 메시지는 (a) 아직 등록되지 않았으며 서버로 등록을 시도 중인 연결 또는 (b) 다른 서버에 대한 기존 연결에서만 허용되어야 합니다. 이 경우 SERVER 메시지는 새로운 연결을 도입합니다. 그 뒤에 서버

섬기는 사람.

SERVER 명령을 수신할 때 발생하는 대부분의 오류는 대상 호스트(대상 SERVER)에 의해 연결이 종료되는 결과를 낳습니다. 오류 응답은 일반적으로 숫자 대신 "ERROR" 명령을 사용하여 전송됩니다. ERROR 명령에는 여기에서 유용하게 사용되는 몇 가지 유용한 속성이 있기 때문입니다.

SERVER 메시지가 구문 분석되어 수신 서버에 이미 알려진 서버를 도입하려고 시도하는 경우 서버에 대한 중복 경로가 형성되고 비순환적 특성이 있으므로 해당 메시지의 연결을 닫아야 합니다(올바른 절차에 따라). IRC 트리가 깨졌습니다.

숫자 답장:

ERR_이미 등록됨

예:

SERVER test.oulu.fi 1 :[tolsun.oulu.fi] 실험용 서버 ; 새로운 서버 test.oulu.fi가 자신을 소개하고 등록을 시도하고 있습니다. []
안의 이름은 test.oulu.fi를 실행하는 호스트의 호스트 이름입니다.

:tolsun.oulu.fi 서버 csd.bu.edu 5 :BU 중앙 서버
; 서버 tolsun.oulu.fi는 5홉 거리에 있는 csd.bu.edu에 대한 업링크입니다.

4.1.5 운영

명령: OPER
매개변수: <사용자> <비밀번호>

OPER 메시지는 일반 사용자가 운영자 권한을 얻기 위해 사용됩니다.
운영자 권한을 얻으려면 <user>와 <password>의 조합이 필요합니다.

OPER 명령을 보내는 클라이언트가 지정된 사용자에 대한 올바른 비밀번호를 제공하면 서버는 클라이언트 별명에 대해 "MODE +o"를 발행하여 나머지 네트워크에 새 운영자를 알립니다.

OPER 메시지는 클라이언트-서버 전용입니다.

숫자 답장:

ERR_NEEDMOREPARAMS RPL_YOUROPER
ERR_NOOPERHOST ERR_PASSWDMISMATCH

예:

OPER 푸 바 ; 사용자 이름 "foo"와 "bar"를 비밀번호로 사용하여 운영자 등록을 시도합니다.

4.1.6 종료

명령: 종료
매개변수: [<종료 메시지>]

클라이언트 세션은 종료 메시지와 함께 종료됩니다. 서버는 QUIT 메시지를 보내는 클라이언트에 대한 연결을 닫아야 합니다. "종료 메시지"가 제공되면 기본 메시지인 닉네임 대신 이것이 전송됩니다.

netsplit(두 서버의 연결 끊김)이 발생하면 종료 메시지가 나타납니다.

관련된 두 서버의 이름으로 구성되며 공백으로 구분됩니다. 첫 번째 이름은 아직 연결되어 있는 서버의 이름이고 두 번째 이름은 연결이 끊어진 서버의 이름입니다.

다른 이유로 인해 클라이언트가 QUIT 명령을 실행하지 않고 클라이언트 연결이 닫히면(예: 클라이언트가 종료되고 소켓에서 EOF가 발생하는 경우) 서버는 종료 메시지를 클라이언트의 성격을 반영하는 일종의 메시지로 채워야 합니다. 그 일이 일어나게 만든 사건.

숫자 답장:

없음.

예:

QUIT : 점심 먹으러 갔어요 ; 선호하는 메시지 형식.

4.1.7 서버 종료 메시지

명령: SQUIT
매개변수: <서버> <설명>

SQUIT 메시지는 종료되거나 죽은 서버에 대해 알려주는 데 필요합니다.
서버가 다른 서버에 대한 연결을 끊고 싶다면 다른 서버의 이름을 서버 매개변수로 사용하여 SQUIT 메시지를 다른 서버에 보내야 합니다. 그런 다음 종료하는 서버에 대한 연결을 닫습니다.

이 명령은 IRC 서버 네트워크를 질서정연하게 연결하는 데 도움이 되는 연산자로도 사용할 수 있습니다. 운영자는 원격 서버 연결에 대해 SQUIT 메시지를 발행할 수도 있습니다. 이 경우 SQUIT는 운영자와 원격 서버 사이의 각 서버에서 구문 분석되어 아래 설명과 같이 각 서버가 보유한 네트워크 보기를 업데이트해야 합니다.

<comment>는 다른 운영자가 이 작업의 이유를 알 수 있도록 (현재 있는 서버에 연결되지 않은) 원격 서버에 대해 SQUIT를 실행하는 모든 운영자가 제공해야 합니다. <comment>는 서버에 의해 채워지므로 여기에 오류나 유사한 메시지가 표시될 수 있습니다.

닫히는 연결의 양쪽에 있는 두 서버는 해당 링크 뒤에 있는 것으로 간주되는 다른 모든 서버에 대해 (다른 모든 서버 연결에) SQUIT 메시지를 보내야 합니다.

마찬가지로 해당 링크 뒤에 있는 모든 클라이언트를 대신하여 네트워크의 나머지 연결된 다른 서버로 QUIT 메시지를 보내야 합니다. 또한, 분할로 인해 멤버를 잃은 채널의 모든 채널 멤버에게는 QUIT 메시지를 보내야 합니다.

서버 연결이 조기에 종료된 경우(예: 링크의 다른 쪽 끝에 있는 서버가 종료된 경우), 이러한 연결 끊김을 감지한 서버는 나머지 네트워크에 연결이 종료되었음을 알리고 주석 필드에 적절한 내용을 채워야 합니다. .

숫자 답변:

ERR_NOPRIVILEGES ERR_NOSUCHSERVER

예:

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

:Trillian SQUIT cm22.eng.umd.edu :서버 통제 불능
; 연결을 끊으라는 Trillian의 메시지
인터넷에서 "cm22.eng.umd.edu"
"서버 통제 불능"이기 때문입니다.

4.2 채널 운영

이 메시지 그룹은 채널, 해당 속성(채널 모드) 및 해당 콘텐츠(일반적으로 클라이언트) 조작과 관련됩니다.

이를 구현하는 과정에서 네트워크의 반대쪽 끝에 있는 클라이언트가 명령을 보내면 궁극적으로 충돌하게 되는 여러 가지 경쟁 조건이 불가피합니다. 또한 <nick> 매개변수가 제공될 때마다 서버가 최근에 변경된 경우 해당 기록을 확인할 수 있도록 닉네임 기록을 유지해야 합니다.

4.2.1 가입 메시지

명령: JOIN
매개변수: <채널>{,<채널>} [<키>{,<키>}]

JOIN 명령은 클라이언트가 특정 채널 청취를 시작하는 데 사용됩니다. 클라이언트의 채널 가입 허용 여부는 클라이언트가 연결된 서버에서만 확인됩니다. 다른 모든 서버는 다른 서버로부터 사용자가 수신되면 자동으로 채널에 사용자를 추가합니다. 이에 영향을 미치는 조건은 다음과 같습니다.

1. 채널이 초대 전용인 경우 사용자를 초대해야 합니다.

2. 사용자의 별명/사용자 이름/호스트 이름은 어떤 것과도 일치해서는 안 됩니다.
활성 금지;

3. 설정된 경우 올바른 키(비밀번호)를 입력해야 합니다.

이에 대해서는 MODE 명령에서 더 자세히 설명합니다(참조:
자세한 내용은 섹션 4.2.3 참조).

사용자가 채널에 가입하면 모든 항목에 대한 알림을 받습니다.
서버가 채널에 영향을 미치는 명령을 수신합니다. 이것
MODE, KICK, PART, QUIT는 물론 PRIVMSG/NOTICE도 포함됩니다. 그만큼
JOIN 명령은 모든 서버에 브로드캐스트되어 각 서버가
채널에 있는 사용자를 어디서 찾을 수 있는지 알고 있습니다. 이를 통해
PRIVMSG/NOTICE 메시지를 채널에 최적으로 전달합니다.

JOIN이 성공하면 사용자에게 채널의 주제가 전송됩니다.
(RPL_TOPIC 사용) 및 채널에 있는 사용자 목록(사용
RPL_NAMREPLY), 여기에는 참여하는 사용자가 포함되어야 합니다.

숫자 답장:

ERR_NEEDMOREPARAMS ERR_BANNEDFROMCHAN
ERR_INVITEONLYCHAN ERR_BADCHANNELKEY
ERR_CHANNELISFULL ERR_BADCHANMASK
ERR_NOSUCHCHANNEL ERR_TOOMANYCHANNELS
RPL_TOPIC

예:

#foobar에 참여하세요	; 채널 #foobar에 가입하세요.
JOIN &foo 푸바	; "fubar" 키를 사용하여 채널(&foo)에 가입하세요.
JOIN #foo,&bar 푸바	; "fubar" 키를 사용하여 채널 #foo에 가입하세요. 키를 사용하지 않고 &bar합니다.
JOIN #foo,#bar fubar,foobar	; "fubar" 키를 사용하여 채널 #foo에 가입하세요. "foobar" 키를 사용하여 채널 #bar를 사용합니다.
#foo,#bar에 참여하세요	; 채널 #foo 및 #bar에 가입하세요.
:WiZ JOIN #Twilight_zone	; WiZ의 JOIN 메시지

4.2.2 부품 메시지

명령: PART
매개변수: <채널>{,<채널>}

PART 메시지를 사용하면 메시지를 보내는 클라이언트가 매개변수 문자열에 나열된 모든 특정 채널의 활성 사용자 목록에서 제거됩니다.

숫자 답장:

ERR_NEEDMOREPARAMS ERR_NOSUCHCHANNEL
ERR_NOTONCHANNEL

예:

PART #twilight_zone ; "#twilight_zone" 채널 나가기

부품 #oz-ops,&group5 ; 채널 "&group5" 및 "#oz-ops"를 모두 그대로 둡니다.

4.2.3 모드 메시지

명령: 모드

MODE 명령은 IRC의 이중 목적 명령입니다. 사용자 이름과 채널 모두 모드를 변경할 수 있습니다. 이 선택의 근거는 언젠가는 닉네임이 더 이상 사용되지 않고 이에 상응하는 속성이 채널이 될 것이라는 것입니다.

MODE 메시지를 구문 분석할 때 전체 메시지를 먼저 구문 분석한 다음 그 결과로 발생한 변경 사항을 전달하는 것이 좋습니다.

4.2.3.1 채널 모드

매개변수: <채널> {[+|-]|o|p|s|i|t|n|b|v} [<제한>] [<사용자>] [<마스크 금지>]

MODE 명령은 채널 운영자가 '자신의' 채널의 특성을 변경할 수 있도록 제공됩니다. 또한 채널 운영자가 생성될 수 있도록 서버에서 채널 모드를 변경할 수 있어야 합니다.

채널에 사용할 수 있는 다양한 모드는 다음과 같습니다.

o - 채널 운영자 권한을 부여/취합니다. p - 개인 채널 플래그; s - 비밀 채널 플래그; i - 초대 전용 채널 플래그; t - 채널 운영자 전용 플래그로 설정할 수 있는 주제; n - 외부 클라이언트로부터 채널로 메시지가 전달되지 않습니다. m - 조정된 채널; l - 사용자 제한을 채널로 설정합니다.

b - 사용자를 차단하기 위해 금지 마스크를 설정합니다.
v - 조정된 채널에서 말할 수 있는 능력을 부여/취득합니다.
k - 채널 키(비밀번호)를 설정합니다.

'o', 'b' 옵션 사용 시 총 3개로 제한
모드별 명령이 부과되었습니다. 즉, 'o'의 모든 조합
그리고

4.2.3.2 사용자 모드

매개변수: <닉네임> {[+|-]|i|w|s|o}

사용자 모드는 일반적으로 다음 중 하나에 영향을 미치는 변경 사항입니다.
클라이언트는 다른 사람이 볼 수 있거나 클라이언트가 보낸 '추가' 메시지가 무엇인지 알 수 있습니다.
사용자 MODE 명령은 송신자 모두가 수신한 경우에만 허용됩니다.
메시지와 매개변수로 부여된 닉네임은 모두 동일합니다.

사용 가능한 모드는 다음과 같습니다.

i - 사용자를 보이지 않는 것으로 표시합니다.
s - 사용자에게 서버 알림 수신을 표시합니다.
w - 사용자가 벽랍을 받습니다.
o - 연산자 플래그.

나중에 추가 모드를 사용할 수 있습니다.

사용자가 "+o"를 사용하여 자신을 운영자로 만들려고 시도하는 경우
플래그가 있으면 시도를 무시해야 합니다. 제한은 없으며,
그러나 누구든지 자신을 '더핑'하는 경우("-o" 사용). 숫자
답글:

ERR_NEEDMOREPARAMS RPL_CHANNELMODEIS
ERR_CHANOPRIVSNEEDED ERR_NOSUCHNICK
ERR_NOTONCHANNEL ERR_KEYSET
RPL_BANLIST RPL_ENDOFBANLIST
ERR_UNKNOWNMODE ERR_NOSUCHCHANNEL

ERR_USERSDONTMATCH RPL_UMODEIS
ERR_UMODEUNKNOWNFLAG

예:

채널 모드 사용:

모드 #핀란드어 +im ; #Finnish 채널을 조정하고
'초대 전용'.

모드 #핀란드어 +o 킬로이 ; Kilroy에게 'chanop' 권한을 부여합니다.

채널 #핀란드어.

MODE #핀란드어 +v 위즈 ; WiZ가 #Finnish로 말할 수 있도록 허용하세요.

MODE #Fins -s ; 채널에서 '비밀' 플래그를 제거합니다.
#에게.

모드 #42 +k oulu ; 채널 키를 "oulu"로 설정하세요.

MODE #eu-opers +l 10 ; 사용자 수 제한 설정
채널에서 10으로.

모드 &oulu +b ; 채널에 설정된 금지 마스크를 나열합니다.

MODE &oulu +b *!*@* ; 모든 사용자가 참여하는 것을 방지합니다.

MODE &oulu +b *!*@*.edu ; 호스트 이름에서 사용자를 방지
가입에서 *.edu와 일치합니다.

사용자 모드 사용:

:모드 마법사 -w ; WALLOPS 메시지 수신을 변경합니다.
WiZ를 위해 꺼져.

:엔젤 모드 엔젤 +i ; 스스로를 만들어가라는 천사의 메시지
보이지 않는.

모드 마법사 -o ; WiZ '더핑'(연산자 제거)
상태). 이것의 명백한 반전
명령("MODE WiZ +o")을 사용하면 안 됩니다.
우회하므로 사용자로부터 허용됨
OPER 명령.

4.2.4 주제 메시지

명령: 주제
매개변수: <채널> [<주제>]

TOPIC 메시지는 채널의 주제를 변경하거나 보는 데 사용됩니다.
<topic>이 없으면 <channel> 채널의 주제가 반환됩니다.
주어진. <topic> 매개변수가 있는 경우 해당 주제는
채널 모드에서 이 작업을 허용하는 경우 채널이 변경됩니다.

숫자 답장:

ERR_NEEDMOREPARAMS ERR_NOTONCHANNEL
RPL_NOTOPIC RPL_TOPIC
ERR_CHANOPRIVSNEEDED

예:

:Wiz TOPIC #테스트 :새 주제 ;사용자 마법사가 주제를 설정합니다.

TOPIC #test :다른 주제 ;#test의 주제를 "다른"으로 설정
주제".

주제 #테스트 ; #test 주제를 확인하세요.

4.2.5 이름 메시지

명령: 이름
매개변수: [<채널>{,<채널>}]

NAMES 명령을 사용하면 사용자는 모든 별명을 나열할 수 있습니다.
볼 수 있는 모든 채널에서 볼 수 있습니다. 채널 이름
그들이 볼 수 있는 것은 비공개(+p) 또는 비밀(+s)이 아닌 것입니다.
또는 실제로 켜져 있는 것. <채널> 매개변수
유효한 경우 정보를 반환할 채널을 지정합니다.
잘못된 채널 이름에 대한 오류 응답은 없습니다.

<channel> 매개변수가 제공되지 않으면 모든 채널과 해당 채널의 목록이 표시됩니다.
입주자가 반환됩니다. 이 목록의 끝에는 다음과 같은 사용자 목록이 있습니다.
표시되지만 어떤 채널에도 없거나 표시되는 채널에 없습니다.
'채널' "*"에 있는 것으로 나열됩니다.

숫자:

RPL_NAMREPLY RPL_ENDOFNAMES

예:

이름 #twilight_zone,#42 ; #twilight_zone에 표시되는 사용자 나열
채널이 표시되는 경우 #42
너.

이름 ; 표시되는 모든 채널과 사용자를 나열합니다.

4.2.6 목록 메시지

명령: 목록
매개변수: [<채널>{,<채널>} [<서버>]]

목록 메시지는 채널과 해당 주제를 나열하는 데 사용됩니다. 만약
<channel> 매개변수가 사용되며 해당 채널의 상태만 표시됩니다.
표시됩니다. 비공개 채널이 나열됩니다(해당 채널 제외).
주제)를 쿼리를 생성하는 클라이언트가 아닌 한 채널 "Prv"로
실제로 그 채널에서요. 마찬가지로 비밀 채널은 나열되지 않습니다.

클라이언트가 해당 채널의 회원이 아닌 이상 전혀 그렇지 않습니다.

숫자 답장:

ERR_NOSUCHSERVER RPL_LISTSTART
RPL_LIST RPL_LISTEND

예:

목록 ; 모든 채널을 나열합니다.

목록 #twilight_zone,#42 ; 채널 #twilight_zone 및 #42 나열

4.2.7 초대 메시지

명령: 초대
매개변수: <닉네임> <채널>

INVITE 메시지는 사용자를 채널에 초대하는 데 사용됩니다. 그만큼
매개변수 <nickname>은 초대할 사람의 닉네임입니다.
대상 채널 <채널>. 다음과 같은 요구 사항은 없습니다.
대상 사용자가 초대되는 채널은 존재하거나 유효해야 합니다.
채널. 초대 전용 채널에 사용자를 초대하려면(MODE
+i) 초대를 보내는 클라이언트는
해당 채널의 채널 운영자입니다.

숫자 답장:

ERR_NEEDMOREPARAMS ERR_NOSUCHNICK
ERR_NOTONCHANNEL ERR_USERONCHANNEL
ERR_CHANOPRIVSNEEDED
RPL_INVITING RPL_AWAY

예:

:Angel INVITE 위즈 #더스트 ; 사용자 Angel이 WiZ를 채널에 초대합니다.
#먼지

마법사 초대 #Twilight_Zone ; WiZ를 초대하는 명령
#Twilight_zone

4.2.8 킥 명령

명령: KICK
매개변수: <채널> <사용자> [<설명>]

KICK 명령을 사용하여 사용자를 강제로 제거할 수 있습니다.
채널. 채널에서 '쫓겨납니다'(강제 부분).

채널 운영자만이 다른 사용자를 채널에서 쫓아낼 수 있습니다.
KICK 메시지를 받은 각 서버는 피해자를 채널에서 제거하기 전에 메시지가 유효한지(즉 보낸 사람이 실제로 채널 운영자인지) 확인합니다.

숫자 답장:

ERR_NEEDMOREPARAMS ERR_NOSUCHCHANNEL
ERR_BADCHANMASK ERR_CHANOPRIVSNEEDED
ERR_NOTONCHANNEL

예:

KICK & 멜버른 매튜 ; &Melbourne에서 매튜를 쫓아내세요

KICK #핀란드 존 :Speaking English
; "Speaking English"를 이유(댓글)로 사용하여 #Finnish에서 John을 쫓아냅니다.

:위즈 킥 #핀란드 존 ; 채널 #Finnish에서 John을 제거하라는 WiZ의 KICK 메시지

메모:
KICK 명령 매개변수를 다음으로 확장할 수 있습니다.

<채널>{,<채널>} <사용자>{,<사용자>} [<설명>]

4.3 서버 쿼리 및 명령

서버 쿼리 명령 그룹은 네트워크에 연결된 모든 서버에 대한 정보를 반환하도록 설계되었습니다. 연결된 모든 서버는 이러한 쿼리에 응답하고 올바르게 응답해야 합니다. 유효하지 않은 응답(또는 응답 없음)은 서버가 손상되었다는 신호로 간주되어야 하며 상황이 해결될 때까지 가능한 한 빨리 연결을 끊거나 비활성화해야 합니다.

이러한 쿼리에서 매개변수가 "<server>"로 표시되는 경우 일반적으로 이는 별명, 서버 또는 일종의 와일드카드 이름일 수 있음을 의미합니다. 그러나 각 매개변수에 대해 하나의 쿼리와 응답 세트만 생성됩니다.

4.3.1 버전 메시지

명령: 버전
매개변수: [<서버>]

VERSION 메시지는 서버 프로그램의 버전을 쿼리하는 데 사용됩니다. 선택적 매개변수 <server>는 클라이언트가 직접 연결되지 않은 서버 프로그램의 버전을 쿼리하는 데 사용됩니다.

숫자 답장:

ERR_NOSUCHSERVER RPL_VERSION

예:

:위즈 버전 *.se ; "*.se"와 일치하는 서버 버전을 확인하라는 Wiz의 메시지

버전 tolsun.oulu.fi ; "tolsun.oulu.fi" 서버 버전을 확인하세요.

4.3.2 통계 메시지

명령: 통계
매개변수: [<쿼리> [<서버>]]

통계 메시지는 특정 서버의 통계를 조회하는 데 사용됩니다. <server> 파라미터를 생략하면 통계 응답의 끝 부분만 다시 보냅니다. 이 명령의 구현은 응답하는 서버에 따라 크게 달라집니다. 하지만 서버는 아래 쿼리에 설명된 대로(또는 이와 유사한) 정보를 제공할 수 있어야 합니다.

쿼리는 대상 서버(<server> 매개변수로 제공되는 경우)에서만 확인하고 중간 서버에 의해 무시되고 변경되지 않은 채 전달되는 단일 문자로 제공될 수 있습니다.

다음 쿼리는 현재 IRC 구현에서 발견되는 쿼리이며 해당 서버에 대한 설정 정보의 상당 부분을 제공합니다. 다른 버전에서는 동일
한 방식으로 지원되지 않을 수 있지만 모든 서버는 현재 사용되는 응답 형식 및 쿼리 목적과 일치하는 STATS 쿼리에 유효한 응답을 제공할 수 있어야 합니다.

현재 지원되는 쿼리는 다음과 같습니다.

c - 서버가 연결할 수 있는 서버 목록을 반환합니다.
연결을 허용하거나 허용합니다.
h - 강제로 리프로 처리되거나 허브 역할을 할 수 있는 서버 목록을 반환합니다. i - 서버가 클라이언트를 허용하는 호스트 목록을 반환합니다.

연결하다;
k - 금지된 사용자 이름/호스트 이름 조합 목록을 반환합니다.
해당 서버의 경우;
l - 서버 연결 목록을 반환하며, 방법을 보여줍니다.

각 연결이 설정되고 트래픽이
해당 연결에 대한 바이트 및 메시지 단위
방향;
m - 서버에서 지원하는 명령 목록을 반환합니다.
사용 횟수가 0이 아닌 경우 각각에 대한 사용 횟수입니다.
o - 일반 클라이언트가 액세스할 수 있는 호스트 목록을 반환합니다.
운영자가 되십시오.
y - 서버 구성 파일의 Y(클래스) 행을 표시합니다.
u - 서버가 가동된 시간을 표시하는 문자열을 반환합니다.

숫자 답장:

ERR_NOSUCHSERVER
RPL_STATSCLINE RPL_STATSNLINE
RPL_STATSILINE RPL_STATSKLINE
RPL_STATSQLINE RPL_STATSLLINE
RPL_STATSLINKINFO RPL_STATSUPTIME
RPL_STATSCOMMANDS RPL_STATSOLINE
RPL_STATSHLINE RPL_ENDOFSTATS

예:

통계m ; 서버의 명령 사용법을 확인하십시오.
당신은 연결되어 있습니다

:Wiz STATS c eff.org ; C/N 회선에 대한 WiZ의 요청
서버 eff.org의 정보

4.3.3 링크 메시지

명령: 링크
매개변수: [[<원격 서버>] <서버 마스크>]

LINKS를 사용하면 사용자는 서버에 알려진 모든 서버를 나열할 수 있습니다.
질의에 응답합니다. 반환된 서버 목록은 다음과 일치해야 합니다.
마스크 또는 마스크가 제공되지 않으면 전체 목록이 반환됩니다.

<서버 마스크> 외에 <원격 서버>가 주어지면 LINKS
명령은 해당 이름과 일치하는 첫 번째 서버로 전달됩니다.
(있는 경우) 해당 서버는 쿼리에 응답해야 합니다.

숫자 답장:

ERR_NOSUCHSERVER
RPL_LINKS RPL_ENDOFLINKS

예:

링크 *.au ; 이름이 있는 모든 서버 나열
*.au와 일치합니다.

:WiZ 링크 *.bu.edu *.edu ; WiZ에서 첫 번째 메시지로 LINKS 메시지
*.edu와 일치하는 서버 목록
*.bu.edu와 일치하는 서버.

4.3.4 시간 메시지

명령: TIME
매개변수: [<서버>]

시간 메시지는 지정된 시간에서 현지 시간을 쿼리하는 데 사용됩니다.
섬기는 사람. 서버 매개변수가 제공되지 않으면 서버가 해당 작업을 처리합니다.
명령은 쿼리에 응답해야 합니다.

숫자 답장:

ERR_NOSUCHSERVER RPL_TIME

예:

시간 tolsun.oulu.fi ; 서버의 시간을 확인하세요
"tolson.oulu.fi"

엔젤타임 *.au ; 사용자 천사가 시간을 확인하고 있습니다.
서버 일치 "*.au"

4.3.5 연결 메시지

명령: 연결
매개변수: <대상 서버> [<포트> [<원격 서버>]]

CONNECT 명령을 사용하여 서버가 강제로 연결을 시도하도록 할 수 있습니다.
즉시 다른 서버에 새로운 연결이 가능합니다. 커넥트는
권한 있는 명령이며 IRC 운영자만 사용할 수 있습니다. 만약에
원격 서버가 주어지면 해당 서버에 의해 CONNECT 시도가 이루어집니다.
서버를 <대상 서버> 및 <포트>로 연결합니다.

숫자 답장:

ERR_NOSUCHSERVER ERR_NOPRIVILEGES
ERR_NEEDMOREPARAMS

예:

연결 tolsun.oulu.fi ; 서버에 연결을 시도합니다.
tolsun.oulu.fi

:WiZ CONNECT eff.org 6667 csd.bu.edu

; CONNECT는 WiZ가 eff.org 및 csd.bu.edu 서버를 포트 6667에 연결하도록 시도합니다.

4.3.6 추적 메시지

명령: TRACE
매개변수: [<서버>]

TRACE 명령은 특정 서버에 대한 경로를 찾는 데 사용됩니다. 이 메시지를 처리하는 각 서버는 "traceroute"를 사용하여 얻은 것과 유사한 응답 체인을 형성하는 통과 링크임을 나타내는 응답을 보내 보낸 사람에게 이에 대해 알려야 합니다. 이 응답을 다시 보낸 후 해당 서버에 도달할 때까지 다음 서버로 TRACE 메시지를 보내야 합니다. <server> 매개변수가 생략된 경우 TRACE 명령을 통해 현재 서버가 어느 서버에 직접 연결되어 있는지 알려주는 메시지를 발신자에게 보내는 것이 좋습니다.

"<서버>"가 제공한 대상이 실제 서버인 경우 대상 서버는 연결된 모든 서버와 사용자를 보고해야 하지만 운영자만 현재 사용자를 볼 수 있습니다. <server>가 제공한 대상이 닉네임인 경우 해당 닉네임에 대한 응답만 제공됩니다.

숫자 답장:

ERR_NOSUCHSERVER

TRACE 메시지가 다른 서버를 대상으로 하는 경우 모든 중간 서버는 TRACE가 해당 서버를 통과했으며 다음 위치로 이동함을 나타내기 위해 RPL_TRACELINK 응답을 반환해야 합니다.

RPL_TRACELINK TRACE
응답은 다음 숫자 응답으로 구성될 수 있습니다.

RPL_TRACECONNECTING
RPL_TRACEHANDSHAKE
RPL_TRACEUNKNOWN
RPL_TRACEOPERATOR
RPL_TRACEUSER
RPL_TRACESERVER
RPL_TRACESERVICE
RPL_TRACENEWTYPE
RPL_TRACECLASS

예:

추적 *.oulu.fi ; *.oulu.fi와 일치하는 서버에 대한 TRACE

:WIZ TRACE 엔젤더스트 ; WiZ가 AngelDust라는 별명을 위해 발행한 TRACE

4.3.7 관리자 명령

명령: 관리자
매개변수: [<서버>]

관리자 메시지는 관리자의 이름을 찾는 데 사용됩니다.
주어진 서버 또는 <server> 매개변수가 생략된 경우 현재 서버입니다.
각 서버에는 ADMIN 메시지를 다른 서버로 전달할 수 있는 기능이 있어야 합니다.
서버.

숫자 답장:

ERR_NOSUCHSERVER
RPL_ADMINME RPL_ADMINLOC1
RPL_ADMINLOC2 RPL_ADMINEMAIL

예:

관리자 tolsun.oulu.fi ; ADMIN 답장을 요청하세요.
tolsun.oulu.fi

:WiZ 관리자 *.edu ; 먼저 WiZ의 ADMIN 요청
*.edu와 일치하는 서버가 발견되었습니다.

4.3.8 정보 명령

명령: 정보
매개변수: [<서버>]

INFO 명령은 다음을 설명하는 정보를 반환하는 데 필요합니다.
서버: 해당 버전, 컴파일된 시기, 패치 수준, 시기
시작되었으며 기타 기타 정보는
관련이 있는 것으로 간주됩니다.

숫자 답장:

ERR_NOSUCHSERVER
RPL_INFO RPL_ENDOFINFO

예:

정보 csd.bu.edu csd.bu.edu ; INFO 답변을 요청하세요

:아발론 정보 *.fi ; Avalon에서 처음으로 INFO 요청
*.fi와 일치하는 서버가 발견되었습니다.

정보 엔젤 ; Angel이 연결된 서버에 정보를 요청합니다.

4.4 메시지 보내기

IRC 프로토콜의 주요 목적은 클라이언트가 서로 통신할 수 있는 기반을 제공하는 것입니다. PRIVMSG 및 NOTICE는 실제로 한 클라이언트에서 다른 클라이언트로 문자 메시지를 전달하는 데 사용할 수 있는 유일한 메시지입니다. 나머지는 이를 가능하게 하고 안정적이고 구조화된 방식으로 발생하도록 합니다.

4.4.1 비공개 메시지

명령: PRIVMSG
매개변수: <수신자>{,<수신자>} <전송할 텍스트>

PRIVMSG는 사용자 간에 비공개 메시지를 보내는 데 사용됩니다. <수신자>는 메시지 수신자의 별명입니다. <receiver>는 쉼표로 구분된 이름 또는 채널 목록일 수도 있습니다.

<receiver> 매개변수는 호스트 마스크(#mask) 또는 서버 마스크($mask)일 수도 있습니다. 두 경우 모두 서버는 마스크와 일치하는 서버나 호스트가 있는 사람에게만 PRIVMSG를 보냅니다. 마스크에는 "."이 하나 이상 있어야 합니다. 마지막 "." 뒤에는 와일드카드가 없습니다. 이 요구 사항은 사람들이 모든 사용자에게 브로드캐스팅되는 "#*" 또는 "$*"로 메시지를 보내는 것을 방지하기 위해 존재합니다. 경험상 이것은 책임감 있고 적절하게 사용되는 것보다 더 많이 남용됩니다.

와일드카드는 '*' 및 '?'입니다. 문자. PRIVMSG 명령에 대한 이 확장은 운영자만 사용할 수 있습니다.

숫자 답장:

ERR_NORECIPIENT ERR_NOTEXTTOSEND
ERR_CANNOTSENDTOCHAN ERR_NOTOPLEVEL
ERR_WILDTOPLEVEL ERR_TOOMANYTARGETS
ERR_NOSUCHNICK
RPL_AWAY

예:

:Angel PRIVMSG Wiz :안녕하세요 이 메시지를 받으셨나요?
; 엔젤이 위즈에게 보내는 메시지.

PRIVMSG Angel :네, 받고 있어요 !받고 있어요 !'u>(768u+1n) .br ;
천사에게 보내는 메시지.

PRIVMSG jto@tolsun.oulu.fi: 안녕하세요!
; 서버의 클라이언트에게 보내는 메시지

사용자 이름이 "jto"인 tolsun.oulu.fi.

PRIVMSG $*.fi :서버 tolsun.oulu.fi 재부팅 중입니다.
; 이름이 *.fi와 일치하는 서버의 모든 사람에게 메시지를 보냅니다.

PRIVMSG #*.edu :NSFNet이 작업 중입니다. 중단될 것으로 예상됩니다. 이름이 *.edu와 일치하는 호스트에서 온 모든 사용자에게 보내는 메시지입니다.

4.4.2 공지

명령: 공지
매개변수: <닉네임> <텍스트>

NOTICE 메시지는 PRIVMSG와 유사하게 사용됩니다. NOTICE와 PRIVMSG의 차이점은 NOTICE 메시지에 대한 응답으로 자동 회신을 보내서는 안 된다는 것입니다. 이 규칙은 서버에도 적용됩니다. 알림을 받은 경우 클라이언트에 오류 응답을 다시 보내서는 안 됩니다. 이 규칙의 목적은 수신된 항목에 대한 응답으로 클라이언트가 자동으로 무언가를 보내는 간의 루프를 방지하는 것입니다. 이는 일반적으로 다른 자동 장치와 루프에 빠지지 않도록 항상 응답하는 것으로 보이는 자동 장치(AI 또는 자신의 작업을 제어하는 다른 대화형 프로그램을 사용하는 클라이언트)에 의해 사용됩니다.

답변과 예시에 대한 자세한 내용은 PRIVMSG를 참조하세요.

4.5 사용자 기반 쿼리

사용자 쿼리는 주로 특정 사용자 또는 그룹 사용자에 대한 세부 정보를 찾는 데 관련된 명령 그룹입니다. 이러한 명령에 와일드카드를 사용할 때 일치하는 경우 '표시'되는 사용자에 대한 정보만 반환합니다. 사용자의 가시성은 사용자 모드와 둘 다 사용 중인 공통 채널 집합의 조합으로 결정됩니다.

4.5.1 누가 질의하는가

명령: WHO
매개변수: [<이름> [<o>]]

WHO 메시지는 클라이언트가 제공한 <name> 매개변수와 '일치'하는 정보 목록을 반환하는 쿼리를 생성하기 위해 클라이언트에서 사용됩니다. <name> 매개변수가 없으면 표시되는 모든 사용자(보이지 않는 사용자(사용자 모드 +i) 및 요청 클라이언트와 공통 채널이 없는 사용자)가 나열됩니다. <name>에 "0"을 사용하거나 와일드카드를 사용하여 동일한 결과를 얻을 수 있습니다.

가능한 모든 항목과 일치하게 됩니다.

WHO에 전달된 <name>은 사용자의 호스트, 서버, 실제와 일치합니다.
<name> 채널을 찾을 수 없는 경우 이름과 닉네임.

"o" 매개변수가 전달되면 연산자만 반환됩니다.
제공된 이름 마스크에.

숫자 답장:

ERR_NOSUCHSERVER
RPL_WHOREPLY RPL_ENDOFWHO

예:

누구 *.fi ; 일치하는 모든 사용자를 나열합니다.
"*.BE".

WHO jto* o ; 다음과 일치하는 모든 사용자를 나열합니다.
연산자인 경우 "jto*"입니다.

4.5.2 Whois 쿼리

명령: 후이즈
매개변수: [<서버>] <닉마스크>[,<닉마스크>[,...]]

이 메시지는 특정 사용자에 대한 정보를 쿼리하는 데 사용됩니다. 그만큼
서버는 여러 숫자 메시지로 이 메시지에 응답합니다.
닉마스크와 일치하는 각 사용자의 다양한 상태를 나타냅니다.
(볼 자격이 있는 경우) 와일드카드가 없는 경우
<nickmask>, 허용된 해당 닉에 대한 정보
참조하여 제시됩니다. 쉼표(',')로 구분된 별명 목록은 다음과 같습니다.
주어진.

후자 버전은 쿼리를 특정 서버로 보냅니다. 그것은
문제의 사용자가 얼마나 오랫동안 사용했는지 알고 싶을 때 유용합니다.
로컬 서버로만 유휴 상태(즉, 사용자가 직접 접속하는 서버)
연결됨)은 해당 정보를 알고 있지만 다른 모든 정보는
세계적으로 알려져 있습니다.

숫자 답장:

ERR_NOSUCHSERVER ERR_NONICKNAMEGIVEN
RPL_WHOISUSER RPL_WHOISCHANNELS
RPL_WHOISCHANNELS RPL_WHOISSERVER
RPL_AWAY RPL_WHOISOPERATOR
RPL_WHOISIDLE ERR_NOSUCHNICK
RPL_ENDOFWHOIS

예:

후이즈 마법사 ; 사용 가능한 사용자 정보 반환
닉 위즈에 대해서

WHOIS eff.org trillian ; 서버 eff.org에 사용자를 요청하세요
트릴리안에 대한 정보

4.5.3 누구였는가

명령: WHOWAS
매개변수: <별명> [<개수> [<서버>]]

Whowas는 더 이상 존재하지 않는 별명에 대한 정보를 요청합니다.
이는 닉네임 변경이나 사용자가 IRC를 떠나기 때문일 수 있습니다.
이 쿼리에 대한 응답으로 서버는 해당 별명을 통해 검색합니다.
역사에서 어휘적으로 동일한 별명을 찾습니다(야생은 아님).
여기에 카드 매칭). 기록을 역방향으로 검색하여 다음을 반환합니다.
가장 최근 항목이 먼저 표시됩니다. 항목이 여러 개인 경우 최대
<count>개의 응답이 반환됩니다(또는 <count>개가 없는 경우 모든 응답이 반환됩니다).
매개변수가 제공됩니다). 양수가 아닌 숫자가 전달되는 경우
<count>이면 전체 검색이 수행됩니다.

숫자 답장:

ERR_NONICKNAMEGIVEN ERR_WASNOSUCHNICK
RPL_WHOWASUSER RPL_WHOISSERVER
RPL_ENDOFWHOWAS

예:

WHOWAS 위즈 ; 닉의 모든 정보를 반환합니다.
"WiZ"라는 별명에 대한 역사;

인어 9는 누구였나요? ; 가장 최근의 9개를 반환합니다.
닉 기록 항목
"인어";

WHOWAS Trillian 1 *.edu ; 에 대한 가장 최근 기록을 반환합니다.
발견된 첫 번째 서버의 "Trillian"
"*.edu"와 일치합니다.

4.6 기타 메시지

이 카테고리의 메시지는 위 카테고리에 해당되지 않습니다.
그럼에도 불구하고 여전히 프로토콜의 일부이며 프로토콜에서 요구됩니다.

4.6.1 종료 메시지

명령: 죽여라
매개변수: <별명> <설명>

KILL 메시지는 실제 연결이 있는 서버에 의해 클라이언트-서버 연결이 닫히도록 하는 데 사용됩니다. KILL은 유효한 별명 목록에서 중복된 항목을 발견할 때 서버에서 사용되며 두 항목을 모두 제거하는 데 사용됩니다. 운영자도 이용 가능합니다.

자동 재연결 알고리즘을 갖춘 클라이언트는 연결 끊김이 매우 짧기 때문에 이 명령을 효과적으로 쓸모 없게 만듭니다. 그러나 이는 데이터 흐름을 중단하고 대량의 남용을 막는 데 사용될 수 있습니다. 모든 사용자는 다른 사용자가 '감시'할 수 있도록 생성된 KILL 메시지를 수신하도록 선택할 수 있습니다.

닉네임은 항상 전역적으로 고유해야 하는 분야에서는 '중복'이 감지될 때마다(즉, 동일한 닉네임으로 두 명의 사용자를 등록하려는 시도) KILL 메시지가 전송됩니다. 1이 다시 나타납니다.

제공된 설명은 KILL의 실제 이유를 반영해야 합니다. 서버에서 생성된 KILL의 경우 일반적으로 두 개의 충돌하는 별명의 출처에 관한 세부 정보로 구성됩니다. 이를 보는 다른 사람들을 만족시킬 수 있는 적절한 이유를 제공하는 것은 사용자의 몫입니다. KILLer의 신원을 숨기기 위해 가짜 KILL이 생성되는 것을 방지/단념시키기 위해 댓글에는 통과하는 각 서버에서 업데이트되는 'kill-path'도 표시되며, 각각의 이름은 경로 앞에 추가됩니다.

숫자 답장:

ERR_NOPRIVILEGES ERR_NEEDMOREPARAMS
ERR_NOSUCHNICK ERR_CANTKILLSERVER

데이비드를 죽여라(csd.bu.edu <- tolsun.oulu.fi)
; csd.bu.edu와 tolson.oulu.fi 간의 닉네임 충돌

참고: 운영
자만 KILL 메시지로 다른 사용자를 죽일 수 있도록 허용하는 것이 좋습니다. 이상적인 세계에서는 운영자도 이 작업을 수행할 필요가 없으며 서버에서 처리하도록 남겨집니다.

4.6.2 핑 메시지

명령: 핑
매개변수: <서버1> [<서버2>]

PING 메시지는 연결의 다른 쪽 끝에 활성 클라이언트가 있는지 테스트하는 데 사용됩니다. 연결에서 다른 활동이 감지되지 않으면 PING 메시지가 정기적으로 전송됩니다. 연결이 설정된 시간 내에 PING 명령에 응답하지 않으면 해당 연결이 닫힙니다.

PING 메시지를 수신하는 모든 클라이언트는 <server1>(PING 메시지를 보낸 서버)에 가능한 한 빨리 적절한 PONG 메시지로 응답하여 클라이언트가 아직 거기에 있고 살아 있음을 나타내야 합니다.

서버는 PING 명령에 응답해서는 안 되며 연결의 다른 쪽 끝에서 PING을 사용하여 연결이 활성 상태임을 나타냅니다.

<server2> 매개변수가 지정된 경우 PING 메시지가 해당 위치로 전달됩니다.

숫자 답장:

ERR_NOORIGIN ERR_NOSUCHSERVER

예:

핑 tolsun.oulu.fi ; 서버가 다른 서버에 PING 메시지를 보내 아직 살아 있음을 나타냅니다.

핑 위즈 ; nick WiZ에게 PING 메시지가 전송되고 있습니다.

4.6.3 퐁 메시지

명령: 퐁
매개변수: <데몬> [<데몬2>]

PONG 메시지는 ping 메시지에 대한 응답입니다. <daemon2> 매개변수가 제공되면 이 메시지는 지정된 데몬으로 전달되어야 합니다. <daemon> 매개변수는 PING 메시지에 응답하고 이 메시지를 생성한 데몬의 이름입니다.

숫자 답장:

ERR_NOORIGIN ERR_NOSUCHSERVER

예:

PONG csd.bu.edu tolsun.oulu.fi ; csd.bu.edu에서 다음으로 보내는 PONG 메시지

tolsun.oulu.fi

4.6.4 오류

명령: 오류
매개변수: <오류 메시지>

ERROR 명령은 심각하거나 치명적인 오류를 운영자에게 보고할 때 서버에서 사용됩니다. 또한 한 서버에서 다른 서버로 전송될 수도 있지만 알 수 없는 일반 클라이언트에서는 허용되어서는 안 됩니다.

ERROR 메시지는 서버 간 링크에서만 발생하는 오류를 보고하는 데 사용됩니다. ERROR 메시지는 상대방 서버(연결된 모든 운영자에게 전송)와 현재 연결된 모든 운영자에게 전송됩니다. 서버에서 받은 경우 서버에서 다른 서버로 전달되지 않습니다.

서버가 수신된 ERROR 메시지를 운영자에게 보낼 때 해당 메시지는 클라이언트가 오류에 대한 책임이 없음을 나타내는 NOTICE 메시지 내에 캡슐화되어야 합니다.

숫자:

없음.

예:

오류: 서버 *.fi가 이미 존재합니다. 다른 서버로 보내는 ERROR 메시지
이 오류가 발생했습니다.

참고 WiZ: csd.bu.edu의 오류 - *.fi 서버가 이미 존재합니다. 위와 동일한 ERROR 메시지가 다른 서버의 WiZ 사용자에게 전송되었습니다.

5. 선택 사항

이 섹션에서는 OPTIONAL 메시지에 대해 설명합니다. 여기에 설명된 프로토콜의 작업 서버 구현에는 필요하지 않습니다. 옵션이 없으면 오류 응답 메시지가 생성되거나 알 수 없는 명령 오류가 발생해야 합니다. 메시지가 다른 서버가 응답하도록 되어 있는 경우 전달되어야 합니다(기본 구문 분석 필요). 이에 대해 할당된 숫자는 아래 메시지와 함께 나열됩니다.

5.1 어웨이

명령: 멀리
매개변수: [메시지]

AWAY 메시지를 사용하면 클라이언트는 자신에게 전달된 모든 PRIVMSG 명령에 대해 자동 응답 문자열을 설정할 수 있습니다(자신이 있는 채널이 아님).

자동 응답은 서버에서 PRIVMSG 명령을 보내는 클라이언트로 전송됩니다. 유일한 응답 서버는 보내는 클라이언트가 연결된 서버입니다.

AWAY 메시지는 하나의 매개변수(AWAY 메시지 설정) 또는 매개변수 없이(AWAY 메시지 제거) 사용됩니다.

숫자 답장:

RPL_UNAWAY RPL_NOWAWAY

예:

AWAY : 점심 먹으러 갔어요. 5로 돌아옴; "점심 먹으러 가세요."라는 메시지를 보내세요.
5"로 돌아왔습니다.

:위즈 어웨이 ; WiZ의 부재중 표시를 해제하세요.

5.2 재해시 메시지

명령: REHASH
매개변수: 없음

운영자는 재해시 메시지를 사용하여 서버가 구성 파일을 다시 읽고 처리하도록 할 수 있습니다.

숫자 답장:

RPL_재해싱 ERR_NOPRIVILEGES

예:

다시 만들다 ; 운영자 상태의 클라이언트에서 서버로 구성 파일을 다시 읽도록 요청하는 메시지입니다.

5.3 재시작 메시지

명령: 다시 시작
매개변수: 없음

다시 시작 메시지는 운영자가 서버 자체를 강제로 다시 시작하는 데만 사용할 수 있습니다. 이 메시지는 임의의 사람이 운영자로서 서버에 연결하고 이 명령을 실행하도록 허용하여 (적어도) 서비스를 중단시키는 위험으로 간주될 수 있으므로 선택 사항입니다.

RESTART 명령은 항상 송신 클라이언트가 연결된 서버에서 완전히 처리되어야 하며 연결된 다른 서버로 전달되어서는 안 됩니다.

숫자 답장:

ERR_NOPRIVILEGES

예:

재시작 ; 매개변수가 필요하지 않습니다.

5.4 소환 메시지

명령: 소환
매개변수: <사용자> [<서버>]

SUMMON 명령은 IRC 서버를 실행하는 호스트에 있는 사용자에게 IRC에 가입하라는 메시지를 보내는 데 사용할 수 있습니다. 이 메시지는 대
상 서버가 (a) SUMMON을 활성화하고, (b) 사용자가 로그인하고, (c) 서버 프로세스가 사용자의 tty(또는 이와 유사한)에 쓸 수 있는 경우에만 전송됩니다.

<server> 매개변수가 제공되지 않으면 클라이언트가 연결된 서버에서 <user>를 소환하려고 시도하며 대상으로 간주됩니다.

서버에서 소환이 활성화되지 않은 경우 ERR_SUMMONDISABLED 숫자를 반환하고 소환 메시지를 계속 전달해야 합니다.

숫자 답장:

ERR_NORECIPIENT ERR_FILEERROR
ERR_NOLOGIN ERR_NOSUCHSERVER
RPL_소환

예:

SUMMON 외 ; 서버 호스트에서 사용자 jto를 소환합니다.

jto tolsun.oulu.fi 소환 ; "tolsun.oulu.fi"라는 서버가 실행 중인 호스트에 사용자 jto를 소환합니다.

5.5 사용자

명령: 사용자
매개변수: [<서버>]

USERS 명령은 who(1), rusers(1) 및 Finger(1)와 유사한 형식으로 서버에 로그인한 사용자 목록을 반환합니다. 일부 사람들은 보안 관련 이유로 자신의 서버에서 이 명령을 비활성화할 수 있습니다. 비활성화된 경우 이를 나타내기 위해 올바른 숫자가 반환되어야 합니다.

숫자 답장:

ERR_NOSUCHSERVER ERR_FILEERROR
RPL_USERSSTART RPL_USERS
RPL_NOUSERS RPL_ENDOFUSERS
ERR_USERSDISABLED

비활성화된 답변:

ERR_USERSDISABLED

예:

사용자 eff.org ; eff.org 서버에 로그인한 사용자 목록을 요청합니다.

:존 사용자 tolsun.oulu.fi ; tolsun.oulu.fi 서버에 로그인한 사용자 목록을 John에게 요청합니다.

5.6 Operwall 메시지

명령: WALLOPS
매개변수: 현재 온라인인 모든 운영자에게 전송될 텍스트

현재 온라인인 모든 운영자에게 메시지를 보냅니다. WALLOPS
를 사용자 명령으로 구현한 후에는 WALLOPS가 많은 사람들에게 메시지를 보내는 수단(WALL과 매우 유사함)으로 자주 악용되는 것으로 나타났습니다. 이로 인해 서버만 WALLOPS의 발신자로 허용하고 인식하여 WALLOPS의 현재 구현을 예로 사용하는 것이 좋습니다.

숫자 답장:

ERR_NEEDMOREPARAMS

예:

:csd.bu.edu WALLOPS :Joshua의 '*.uiuc.edu 6667'을 연결합니다. Joshua로부터 수신되어 실행된 CONNECT 메시지를 알리는
csd.bu.edu의 WALLOPS 메시지입니다.

5.7 사용자 호스트 메시지

명령: USERHOST
매개변수: <닉네임>{<스페이스><닉네임>}

USERHOST 명령은 각각 공백 문자로 구분된 최대 5개의 별명 목록을 가져와 찾은 각 별명에 대한 정보 목록을 리턴합니다. 반환된 목록에는 공백으로 구분된 각 응답이 있습니다.

숫자 답장:

RPL_USERHOST ERR_NEEDMOREPARAMS

예:

USERHOST Wiz Michael Marty p ;USERHOST 별명 "Wiz", "Michael", "Marty" 및 "p"에 대한 정보 요청

5.8 Ison 메시지

명령: ISON
매개변수: <닉네임>{<스페이스><닉네임>}

ISON 명령은 주어진 별명이 현재 IRC에 있는지 여부에 대한 응답을 얻는 빠르고 효율적인 수단을 제공하기 위해 구현되었습니다. ISON은 하나의 매개변수(공백으로 구분된 닉네임 목록)만 사용합니다. 존재하는 목록의 각 별명에 대해 서버는 이를 응답 문자열에 추가합니다. 따라서 응답 문자열은 빈 문자열(주어진 별명이 하나도 없음), 매개변수 문자열의 정확한 복사본(모두 존재함) 또는 매개변수에 제공된 별명 집합의 다른 하위 집합을 반환할 수 있습니다. 확인할 수 있는 닉 수에 대한 유일한 제한은 결합된 길이가 서버에서 잘려 512자에 맞도록 너무 커서는 안 된다는 것입니다.

ISON은 명령을 보내는 클라이언트의 로컬 서버에서만 처리되므로 추가 처리를 위해 다른 서버로 전달되지 않습니다.

숫자 답장:

RPL_ISON ERR_NEEDMOREPARAMS

예:

ISON 전화 조 WiZ jarlek Avalon Angel Monstah
; 7개 닉에 대한 샘플 ISON 요청입니다.

6. 답글

다음은 생성된 숫자 응답 목록입니다.
위에 주어진 명령에 대한 응답. 각 숫자는 해당 숫자와 함께 제공됩니다.
번호, 이름, 응답 문자열.

6.1 오류 응답.

401 ERR_NOSUCHNICK
"<닉네임> :해당 닉네임/채널이 없습니다."

- 제공된 별명 매개변수를 나타내는 데 사용됩니다.
명령은 현재 사용되지 않습니다.

402 ERR_NOSUCHSERVER
"<서버 이름> : 해당 서버가 없습니다."

- 현재 주어진 서버 이름을 나타내는 데 사용됩니다.
존재하지 않습니다.

403 ERR_NOSUCHCHANNEL
"<채널 이름> :해당 채널이 없습니다."

- 주어진 채널 이름이 유효하지 않음을 나타내는 데 사용됩니다.

404 ERR_CANNOTSENDTOCHAN
"<채널 이름> :채널로 보낼 수 없습니다."

- (a) 채널에 없는 사용자에게 전송됩니다.
모드 +n이거나 (b) chanop(또는 모드 +v)이 아닙니다.
모드 +m이 설정되어 있고 전송을 시도하는 채널
해당 채널에 PRIVMSG 메시지를 보냅니다.

405 ERR_TOOMANYCHANNELS
"<채널 이름> :너무 많이 가입하셨습니다 \
채널"

- 최대 인원에 가입한 사용자에게 전송됩니다.
허용된 채널의 수와 참여를 시도합니다.
다른 채널.

406 ERR_WASNOSUCHNICK
"<닉네임> :그런 별명은 없었어요"

- 기록이 없음을 나타내기 위해 WHOWAS에서 반환함
해당 닉네임에 대한 정보입니다.

407 ERR_TOOMANYTARGETS
"<대상> :수신자가 중복되었습니다. 메시지가 없습니다. \

배달됐어"

- 전송을 시도하는 클라이언트로 반환됨
user@host 대상 형식을 사용하는 PRIVMSG/NOTICE
그리고 여러 번 발생하는 user@host의 경우입니다.

409 ERR_NOORIGIN
":원산지가 지정되지 않았습니다."

- 발신자 매개변수가 누락된 PING 또는 PONG 메시지
이 명령이 작동해야 하므로 필수입니다.
유효한 접두사가 없습니다.

411 ERR_NORECIPIENT
":받는 사람이 지정되지 않았습니다(<명령>)"

412 ERR_NOTEXTTOSEND
":보낼 문자가 없습니다."

413 ERR_NOTOPLEVEL
"<마스크> : 최상위 도메인이 지정되지 않았습니다."

414 ERR_WILDTOPLEVEL
"<마스크> :최상위 도메인의 와일드카드"

- 412 - 414는 PRIVMSG에 의해 반환되어 다음을 나타냅니다.
어떤 이유로 메시지가 전달되지 않았습니다.
ERR_NOTOPLEVEL 및 ERR_WILDTOPLEVEL은 다음과 같은 오류입니다.
잘못된 사용 시 반환됩니다.
"PRIVMSG $<서버>" 또는 "PRIVMSG #<호스트>"가 시도되었습니다.

421 ERR_UNKNOWN명령
"<명령> :알 수 없는 명령"

- 등록된 고객에게 반환되어
전송된 명령을 서버에서 알 수 없습니다.

422 ERR_NOMOTD
":MOTD 파일이 없습니다"

- 서버의 MOTD 파일을 서버에서 열 수 없습니다.

423 ERR_NOADMININFO
"<서버> : 사용 가능한 관리 정보가 없습니다."

- ADMIN 메시지에 대한 응답으로 서버에서 반환됨
적절한 것을 찾는 데 오류가 있는 경우
정보.

424 ERR_FILEERROR
":<파일>에서 <파일 작업>을 수행하는 동안 파일 오류가 발생했습니다."

- 실패한 파일을 보고하는 데 사용되는 일반 오류 메시지
메시지 처리 중 작업.

431 ERR_NONICKNAMEGIVEN
":닉네임이 지정되지 않았습니다."

- 닉네임 매개변수가 예상되는 경우 반환됩니다.
명령을 찾을 수 없습니다.

432 ERR_ERRONEUSNICKNAME
"<닉네임> :잘못된 별명"

- 다음을 포함하는 NICK 메시지를 수신한 후 반환됨
정의된 세트에 속하지 않는 문자입니다. 보다
유효한 별명에 대한 자세한 내용은 섹션 xxx를 참조하세요.

433 ERR_NICKNAMEINUSE
"<닉네임> :닉네임이 이미 사용 중입니다."

- 결과적으로 NICK 메시지가 처리될 때 반환됩니다.
현재 존재하는 것으로 변경하려고 시도 중입니다.
별명.

436 ERR_NICKCOLLISION
"<닉네임> :닉네임 충돌 KILL"

- 서버가 클라이언트를 감지하면 서버에서 클라이언트로 반환합니다.
닉네임 충돌(NICK에 등록됨)
다른 서버에 이미 존재합니다.)

441 ERR_USERNOTINCHANNEL
"<닉네임> <채널> :그 채널에는 없어요"

- 대상이 있음을 나타내기 위해 서버에 의해 반환됨
명령 사용자가 해당 채널에 없습니다.

442 ERR_NOTONCHANNEL
"<채널> :당신은 그 채널에 없습니다"

- 클라이언트가 시도할 때마다 서버에 의해 반환됩니다.
채널 영향 명령을 수행합니다.
고객이 회원이 아닙니다.

443 ERR_USERONCHANNEL
"<사용자> <채널> :이미 채널에 있습니다."

- 클라이언트가 사용자를 초대하려고 할 때 반환됩니다.
이미 채널에 있습니다.

444 ERR_NOLOGIN
"<사용자> :사용자가 로그인하지 않았습니다."

- SUMMON 명령 이후 소환에 의해 반환됩니다.
사용자가 수행할 수 없습니다.
로그인.

445 ERR_SUMMONDISABLED
":SUMMON이 비활성화되었습니다."

- SUMMON 명령에 대한 응답으로 반환됩니다. 반드시
이를 구현하지 않은 서버에서 반환됩니다.

446 ERR_USERSDISABLED
":USERS가 비활성화되었습니다."

- USERS 명령에 대한 응답으로 반환됩니다. 반드시
이를 구현하지 않은 서버에서 반환됩니다.

451 ERR_NOTREGISTERED
":등록하지 않으셨습니다"

- 클라이언트가 이를 나타내기 위해 서버에 의해 반환됨
서버에서 허용하려면 먼저 등록해야 합니다.
자세히 분석할 예정입니다.

461 ERR_NEEDMOREPARAMS
"<명령> :매개변수가 부족합니다."

- 수많은 명령을 통해 서버에 의해 반환됨
고객에게 공급이 충분하지 않았음을 나타냅니다.
매개변수.

462 ERR_이미 등록됨
":재등록이 불가능합니다."

- 서버가 다음을 시도하는 링크로 반환합니다.
등록된 내용 중 일부를 변경하는 경우(예:
두 번째 USER 메시지의 비밀번호 또는 사용자 세부 정보).

463 ERR_NOPERMFORHOST
":당신의 호스트는 특권을 가진 사람이 아닙니다."

- 등록을 시도하는 클라이언트에게 반환됨
허용하도록 설정되지 않은 서버
호스트로부터의 연결 시도된 연결
시도됩니다.

464 ERR_PASSWDMISMATCH
":비밀번호 오류"

- 등록 시도가 실패했음을 나타내기 위해 반환됨
비밀번호가 필요한 연결 및
제공되지 않았거나 잘못되었습니다.

465 ERR_YOUREBANNEDCREEP
":당신은 이 서버에서 금지되었습니다."

- 연결 및 등록 시도 후 반환됨
다음과 같이 설정된 서버를 사용하세요.
귀하와의 연결을 명시적으로 거부합니다.

467 ERR_KEYSET
"<채널> :채널 키가 이미 설정되었습니다."

471 ERR_CHANNELISFULL
"<채널> :채널(+l)에 참가할 수 없습니다."

472 ERR_UNKNOWNMODE
"<char>:나에게는 알 수 없는 모드 문자입니다."

473 ERR_INVITEONLYCHAN
"<채널> :채널(+i)에 참여할 수 없습니다."

474 ERR_BANNEDFROMCHAN
"<채널> :채널(+b)에 참여할 수 없습니다."

475 ERR_BADCHANNELKEY
"<채널> :채널에 참여할 수 없습니다(+k)"

481 ERR_NOPRIVILEGES
":권한이 거부되었습니다. - 귀하는 IRC 운영자가 아닙니다."

- 작동을 위해 운영자 권한이 필요한 모든 명령
시도가 실패했음을 나타내려면 이 오류를 반환해야 합니다.
실패했습니다.

482 ERR_CHANOPRIVSNEEDED
"<채널> :채널 운영자가 아닙니다."

- 'chanop' 권한이 필요한 모든 명령(예:
MODE 메시지) 클라이언트가 다음과 같은 경우 이 오류를 반환해야 합니다.
시도하는 것은 지정된 chanop이 아닙니다.
채널.

483 ERR_CANTKILLSERVER
":서버를 죽일 수는 없습니다!"

- 서버에서 KILL 명령을 사용하려는 모든 시도
거부되며 이 오류가 직접 반환됩니다.
클라이언트에게.

491 ERR_NOOPERHOST
": 호스트를 위한 O-라인이 없습니다."

- 클라이언트가 OPER 메시지를 보내고 서버가
연결을 허용하도록 구성되지 않았습니다.
클라이언트의 호스트를 운영자로 사용하는 경우 이 오류는 다음과 같아야 합니다.
돌아왔다.

501 ERR_UMODEUNKNOWNFLAG
":알 수 없는 MODE 플래그"

- MODE임을 나타내기 위해 서버에 의해 반환됨
닉네임 매개변수와 함께 메시지가 전송되었으며 그 내용은 다음과 같습니다.
전송된 a 모드 플래그가 인식되지 않았습니다.

502 ERR_USERSDONTMATCH
":다른 사용자를 위해 모드를 변경할 수 없습니다."

- 오류를 보거나 변경하려는 모든 사용자에게 전송됨
자신이 아닌 다른 사용자를 위한 사용자 모드입니다.

6.2 명령 응답.

300 RPL_NONE
더미 응답 번호. 사용되지 않습니다.

302 RPL_USERHOST
":[<답장>{<스페이스><답장>}]"

- USERHOST가 응답을 나열하는 데 사용하는 응답 형식
쿼리 목록. 응답 문자열은 다음과 같이 구성됩니다.
다음과 같습니다:

<답장> ::= <닉네임>['*'] '=' <'+'|'-'><호스트 이름>

'*'는 클라이언트가 등록되었는지 여부를 나타냅니다.
운영자로서. '-' 또는 '+' 문자는
클라이언트가 AWAY 메시지를 설정했는지 여부
각기.

303 RPL_ISON
":[<별명> {<스페이스><별명>}]"

- ISON이 응답을 나열하는 데 사용하는 응답 형식
쿼리 목록.

301 RPL_AWAY
"<닉네임> :<부재 메시지>"

305 RPL_UNAWAY
":더 이상 부재중으로 표시되지 않습니다."

306 RPL_NOWAWAY
":부재중으로 표시되었습니다."

- 이 응답은 AWAY 명령과 함께 사용됩니다.
허용된). RPL_AWAY는 다음을 보내는 모든 클라이언트에게 전송됩니다.
멀리 있는 클라이언트에게 PRIVMSG를 보냅니다. RPL_AWAY는
클라이언트가 연결된 서버에서 전송됩니다.
RPL_UNAWAY 및 RPL_NOWAWAY 응답은 다음과 같은 경우에 전송됩니다.
클라이언트는 AWAY 메시지를 제거하고 설정합니다.

311 RPL_WHOISUSER
"<닉네임> <사용자> <호스트> * :<실명>"

312 RPL_WHOISSERVER
"<닉네임> <서버> :<서버 정보>"

313 RPL_WHOISOPERATOR
"<닉네임> : IRC 운영자입니다."

317 RPL_WHOISIDLE
"<닉네임> <정수> :초 유휴"

318 RPL_ENDOFWHOIS
"<닉네임> :/WHOIS 목록 끝"

319 RPL_WHOISCHANNELS
"<닉네임> :{[@|+]<채널><스페이스>}"

- 답글 311~313, 317~319는 모두 답글입니다.
WHOIS 메시지에 대한 응답으로 생성됩니다. 을 고려하면
충분한 매개변수가 존재하며 응답은 다음과 같습니다.
서버는 위의 응답 중 하나를 공식화해야 합니다.
숫자(쿼리 별명이 발견된 경우) 또는
오류 답변. RPL_WHOISUSER의 '*'는 다음과 같습니다.
와일드카드가 아닌 문자 그대로의 문자입니다. 을 위한
각 응답 세트에는 RPL_WHOISCHANNELS만 나타날 수 있습니다.
한 번 이상(긴 채널 이름 목록의 경우)
채널 이름 옆의 '@' 및 '+' 문자
클라이언트가 채널 운영자인지 또는
중재에 관해 발언할 수 있는 권한이 부여되었습니다.
채널. RPL_ENDOFWHOIS 응답은 표시하는 데 사용됩니다.
WHOIS 메시지 처리가 종료됩니다.

314 RPL_WHOWASUSER
"<닉네임> <사용자> <호스트> * :<실명>"

369 RPL_ENDOFWHOWAS
"<닉네임> :WHOWAS의 끝"

- WHOWAS 메시지에 응답할 때 서버는 다음을 사용해야 합니다.
RPL_WHOWASUSER, RPL_WHOISSERVER 또는 응답
제시된 각 별명에 대한 ERR_WASNOSUCHNICK

목록. 모든 응답 일괄 처리가 끝나면 다음이 있어야 합니다.
RPL_ENDOFWHOWAS이어야 합니다(답장이 하나뿐인 경우에도 마찬가지입니다).
그리고 그것은 오류였습니다).

321 RPL_LISTSTART
"채널:사용자 이름"

322 RPL_LIST
"<채널> <# 표시> :<주제>"

323 RPL_LISTEND
":/LIST의 끝"

- RPL_LISTSTART, RPL_LIST, RPL_LISTEND 표시 응답
시작, 실제 데이터 응답 및 끝
LIST 명령에 대한 서버의 응답. 만일 거기에
반환할 수 있는 채널이 없으며 시작만 가능합니다.
그리고 최종 답장을 보내야 합니다.

324 RPL_CHANNELMODEIS
"<채널> <모드> <모드 매개변수>"

331 RPL_NOTOPIC
"<채널> :주제가 설정되지 않았습니다."

332 RPL_TOPIC
"<채널> :<주제>"

- TOPIC 메시지를 보내어 결정하는 경우
채널 주제에 대한 응답은 두 개 중 하나가 전송됩니다. 만약에
주제가 설정되고 RPL_TOPIC이 다시 전송됩니다.
RPL_NOTOPIC.

341 RPL_INVITING
"<채널> <닉네임>"

- 서버가 반환한 내용은 다음과 같습니다.
INVITE 메시지 시도가 성공했으며
최종 클라이언트로 전달됩니다.

342 RPL_소환
"<사용자> :사용자를 IRC에 소환합니다"

- SUMMON 메시지에 응답하는 서버에 의해 반환됨
해당 사용자를 소환하고 있음을 나타냅니다.

351 RPL_VERSION
"<버전>.<디버그 수준> <서버> :<설명>"

- 버전 세부 정보를 표시하는 서버의 응답입니다.
<version>은 현재 소프트웨어의 버전입니다.

사용된(모든 패치 수준 개정 포함) 및
<debuglevel>은 서버가
"디버그 모드"에서 실행 중입니다.

"설명" 필드에는 다음에 대한 설명이 포함될 수 있습니다.
버전 또는 추가 버전 세부정보.

352 RPL_WHOREPLY
"<채널> <사용자> <호스트> <서버> <닉네임> \
<H|G>[*][@|+] :<홉 수> <실명>"

315 RPL_ENDOFWHO
"<이름> :/WHO 목록 끝"

- RPL_WHOREPLY 및 RPL_ENDOFWHO 쌍이 사용됩니다.
WHO 메시지에 응답하기 위해. RPL_WHOREPLY는
WHO와 적절하게 일치하는 항목이 있는 경우 전송됩니다.
질문. 제공된 매개변수 목록이 있는 경우
WHO 메시지와 함께 RPL_ENDOFWHO를 전송해야 합니다.
<name>이 포함된 각 목록 항목을 처리한 후
그 아이템.

353 RPL_NAMREPLY
"<채널> :[[@|+]<별명> [[@|+]<별명> [...]]]"

366 RPL_ENDOFNAMES
"<채널> :/NAMES 목록 끝"

- NAMES 메시지에 응답하려면 다음으로 구성된 응답 쌍이 필요합니다.
RPL_NAMREPLY 및 RPL_ENDOFNAMES는 다음에서 전송됩니다.
서버가 클라이언트로 다시 돌아갑니다. 채널이 없는 경우
쿼리에서와 같이 발견되면 RPL_ENDOFNAMES만 나타납니다.
돌아왔다. 이에 대한 예외는 NAMES
메시지는 매개변수 없이 전송되며 모두 표시됩니다.
채널과 콘텐츠는 일련의 방식으로 다시 전송됩니다.
표시할 RPL_ENDOFNAMES가 있는 RPL_NAMEREPLY 메시지
끝.

364 RPL_LINKS
"<마스크> <서버> :<홉수> <서버 정보>"

365 RPL_ENDOFLINKS
"<마스크> : /LINKS 목록 끝"

- LINKS 메시지에 대한 응답으로 서버는 다음을 보내야 합니다.
RPL_LINKS 숫자를 사용하여 응답하고
RPL_ENDOFLINKS 응답을 사용하여 목록 끝.

367 RPL_BANLIST
"<채널> <배니드>"

368 RPL_ENDOFBANLIST

"<채널> :채널 금지 목록 끝"

- 특정 채널에 대한 활성 '금지' 목록을 나열할 때,
서버는 다음을 사용하여 목록을 다시 보내야 합니다.
RPL_BANLIST 및 RPL_ENDOFBANLIST 메시지. 분리 된
RPL_BANLIST는 각 활성 banid에 대해 전송됩니다. 후
banids가 나열되었습니다(또는 존재하지 않는 경우).
RPL_ENDOFBANLIST를 전송해야 합니다.

371 RPL_INFO
":<문자열>"

374 RPL_ENDOFINFO
":/INFO 목록 끝"

- INFO 메시지에 응답하는 서버는 다음을 수행해야 합니다.
일련의 RPL_INFO 메시지로 모든 '정보'를 보냅니다.
RPL_ENDOFINFO 응답으로 종료를 나타냅니다.
답글.

375 RPL_MOTDSTART
":- <서버> 오늘의 메시지 -
"

372 RPL_MOTD
":- <텍스트>"

376 RPL_ENDOFMOTD
":/MOTD 명령의 끝"

- MOTD 메시지 및 MOTD 파일에 응답하는 경우
파일이 발견되면 파일이 한 줄씩 표시됩니다.
각 줄은 80자를 넘지 않아야 합니다.
RPL_MOTD 형식으로 응답합니다. 이것들은 둘러싸야 해
RPL_MOTDSTART(RPL_MOTD 이전) 및
RPL_ENDOFMOTD(이후).

381 RPL_YOUROPER
":당신은 이제 IRC 운영자입니다"

- RPL_YOUREOPER가 클라이언트로 다시 전송됩니다.
방금 OPER 메시지를 성공적으로 발행하여 얻었습니다.
운영자 상태.

382 RPL_재해싱
"<구성 파일> :재해싱"

- REHASH 옵션을 사용하여 운영자가 전송하는 경우
REHASH 메시지, RPL_REHASHING이 다시 전송됩니다.
운영자.

391 RPL_TIME

"<서버> :<서버의 현지 시간을 표시하는 문자열>"

- TIME 메시지에 응답할 때 서버는 다음을 보내야 합니다.
위의 RPL_TIME 형식을 사용한 응답입니다. 문자열
시간을 표시하려면 올바른 날짜만 포함하면 됩니다.
거기 시간이야. 에 대한 추가 요구 사항은 없습니다.
시간 문자열.

392 RPL_USERSSTART
":UserID 터미널 호스트"

393 RPL_USERS
":%-8s %-9s %-8s"

394 RPL_ENDOFUSERS
":사용자 종료"

395 RPL_NOUSERS
":아무도 로그인하지 않았습니다."

- USERS 메시지가 서버에 의해 처리되는 경우
RPL_USERSTART, RPL_USERS, RPL_ENDOFUSERS를 응답하고
RPL_NOUSERS가 사용됩니다. RPL_USERSSTART를 보내야 합니다.
먼저 RPL_USERS 시퀀스가 옵니다.
또는 단일 RPL_NOUSER. 다음은
RPL_ENDOFUSERS.

200 RPL_TRACELINK
"링크 <버전 및 디버그 수준> <대상> \
<다음 서버>"

201 RPL_TRACECONNECTING
"해 보세요. <클래스> <서버>"

202 RPL_TRACEHANDSHAKE
"HS <클래스> <서버>"

203 RPL_TRACEUNKNOWN
"???? <클래스> [<점 형식의 클라이언트 IP 주소>]"

204 RPL_TRACEOPERATOR
"오퍼 <클래스> <닉네임>"

205 RPL_TRACEUSER
"사용자 <클래스> <닉네임>"

206 RPL_TRACESERVER
"Serv <클래스> <int>S <int>C <서버> \
<nick!user|*!*>@<호스트|서버>"

208 RPL_TRACENEWTYPE
"<새 유형> 0 <클라이언트 이름>"

261 RPL_TRACELOG
"파일 <로그파일> <디버그 수준>"

- RPL_TRACE*는 모두 서버에서 반환됩니다.
TRACE 메시지에 대한 응답입니다. 몇 개야?
반환된 값은 TRACE 메시지에 따라 달라지며

교환원이 보낸 것인지 아닌지. 거기
먼저 발생하는 미리 정의된 순서가 없습니다.
RPL_TRACEUNKNOWN, RPL_TRACECONNECTING 및 응답
RPL_TRACEHANDSHAKE는 모두 연결에 사용됩니다.
완전히 확립되지 않았으며 다음 중 하나입니다.
알 수 없음, 여전히 연결을 시도 중이거나
'서버 핸드셰이크'를 완료하는 과정입니다.
RPL_TRACELINK는 처리하는 모든 서버에서 전송됩니다.
TRACE 메시지를 다른 사람에게 전달해야 합니다.
섬기는 사람. 전송된 RPL_TRACELINK 목록
IRC를 통과하는 TRACE 명령에 대한 응답
네트워크는 실제 연결을 반영해야 합니다.
해당 경로를 따라 서버 자체.
RPL_TRACENEWTYPE은 모든 연결에 사용됩니다.
다른 카테고리에는 적합하지 않지만
어쨌든 표시됩니다.

211 RPL_STATSLINKINFO
"<링크 이름> <sendq> <보낸 메시지> \
<보낸 바이트> <받은 메시지> \
<수신 바이트> <열린 시간>"

212 RPL_STATSCOMMANDS
"<명령> <개수>"

213 RPL_STATSCLINE
"C <호스트> * <이름> <포트> <클래스>"

214 RPL_STATSNLINE
"N <호스트> * <이름> <포트> <클래스>"

215 RPL_STATSILINE
"나는 <호스트> * <호스트> <포트> <클래스>"

216 RPL_STATSKLINE
"K <호스트> * <사용자 이름> <포트> <클래스>"

218 RPL_STATSYLINE
"Y <클래스> <핑 빈도> <연결 \
빈도> <최대 전송량>"

219 RPL_ENDOFSTATS
"<통계 문자> :/STATS 보고서 끝"

241 RPL_STATSLLINE
"L <호스트마스크> * <서버 이름> <최대 깊이>"

242 RPL_STATSUPTIME
":서버 가동 %d일 %d:%02d:%02d"

243 RPL_STATSOLINE
"O <호스트마스크> * <이름>"

244 RPL_STATSHLINE
"H <호스트마스크> * <서버 이름>"

221 RPL_UMODEIS
"<사용자 모드 문자열>"

- 클라이언트 자신의 모드에 대한 질의에 답변하기 위해,
RPL_UMODEIS가 다시 전송됩니다.

251 RPL_LUSERCLIENT
":<정수> 사용자와 <정수>\가 있습니다.
<integer> 서버에서는 보이지 않음"

252 RPL_LUSEROP
"<정수> :연산자 온라인"

253 RPL_LUSERUNKNOWN
"<정수> :알 수 없는 연결"

254 RPL_LUSERCHANNELS
"<정수> :채널이 형성됨"

255 RPL_LUSERME
":<정수> 클라이언트와 <정수>\가 있습니다.
서버"

- LUSERS 메시지를 처리하는 과정에서 서버는
RPL_LUSERCLIENT에서 일련의 응답을 보냅니다.
RPL_LUSEROP, RPL_USERUNKNOWN,
RPL_LUSERCHANNELS 및 RPL_LUSERME. 언제
응답하면 서버가 다시 보내야 합니다.
RPL_LUSERCLIENT 및 RPL_LUSERME. 다른
개수가 0이 아닌 경우에만 응답이 다시 전송됩니다.
그들을 위해 발견되었습니다.

256 RPL_ADMINME
"<서버>:관리 정보"

257 RPL_ADMINLOC1
":<관리자 정보>"

258 RPL_ADMINLOC2
":<관리자 정보>"

259 RPL_ADMINEMAIL
":<관리자 정보>"

- ADMIN 메시지에 대한 응답 시 서버는
RLP_ADMINME 응답을 사용할 것으로 예상됩니다.
RPL_ADMINEMAIL을 통해 텍스트를 제공하세요.
메시지를 각각 보내세요. RPL_ADMINLOC1의 경우
어떤 도시, 주, 국가에 대한 설명
서버가 예상됩니다.
대학 및 학과 세부정보
(RPL_ADMINLOC2) 그리고 마지막으로 관리
서버 연락처(여기에 이메일 주소를 입력하세요.
필수) RPL_ADMINEMAIL에 있습니다.

6.3 예약된 숫자.

이 숫자는 다음 중 하나에 속하므로 위에서 설명하지 않습니다.
다음 카테고리:

1. 더 이상 사용되지 않습니다.

2. 향후 계획된 사용을 위해 보류됩니다.

3. 현재 사용 중이지만 다음의 일반적이지 않은 '기능'의 일부입니다.
현재 IRC 서버.

209	RPL_TRACECLASS	217	RPL_STATSQLINE
231	RPL_SERVICEINFO	232	RPL_ENDOFSERVICES
233	RPL_SERVICE	234	RPL_SERVLIST
235	RPL_SERVLISTEND		
316	RPL_WHOISCHANOP	361	RPL_KILLDONE
362	RPL_CLOSING	363	RPL_CLOSEEND
373	RPL_INFOSTART	384	RPL_MYPORTIS
466	ERR_YOUWILLBEBANNED	476	ERR_BADCHANMASK
492	ERR_NOSERVICEHOST		

7. 클라이언트 및 서버 인증

클라이언트와 서버는 모두 동일한 수준의 적용을 받습니다.
입증. 두 경우 모두 호스트 이름 조회를 위한 IP 번호(및
이에 대한 역방향 확인)은 모든 연결에 대해 수행됩니다.
섬기는 사람. 그러면 두 연결 모두 비밀번호 확인을 받게 됩니다(해당되는 경우).
해당 연결에 비밀번호가 설정되어 있습니다.) 이러한 수표는
비밀번호 확인만 가능하지만 모든 연결에서 가능합니다.
일반적으로 서버와 함께 사용됩니다.

점점 더 일반화되고 있는 추가 검사는 다음과 같습니다.
연결을 담당하는 사용자 이름입니다. 찾기
일반적으로 연결 반대편 끝의 사용자 이름에는 다음이 포함됩니다.
에 설명된 대로 IDENT와 같은 인증 서버에 연결합니다.
RFC 1413.

비밀번호가 없으면 누가 누구인지 확실하게 판단하기가 쉽지 않습니다.
네트워크 연결의 반대편에 있는 경우 비밀번호를 사용하는 것은
서버 간 연결에 강력히 권장됩니다.
ID 서버 사용과 같은 기타 조치.

8. 현재 구현

이 프로토콜의 현재 구현은 IRC 서버뿐입니다.
버전 2.8. 이전 버전에서는 다음 중 일부 또는 전부를 구현할 수 있습니다.
이 문서에서 설명하는 명령은 NOTICE 메시지로 대체됩니다.

숫자로 답하는 경우가 많습니다. 불행하게도 이전 버전과의 호환성 요구 사항으로 인해 이 문서의 일부 구현은 설명된 내용에 따라 다릅니다. 주목할만한 차이점은 다음과 같습니다.

* 메시지의 어느 위치에 있는 LF나 CR이
해당 메시지의 끝(CR-LF를 요구하는 대신)

이 섹션의 나머지 부분에서는 서버를 구현하려는 사람들에게 가장 중요한 문제를 다루지만 일부 부분은 클라이언트에도 직접 적용됩니다.

8.1 네트워크 프로토콜: TCP - 이것이 여기서 가장 잘 사용되는 이유.

IRC는 TCP가 이러한 규모의 회의에 매우 적합한 안정적인 네트워크 프로토콜을 제공하기 때문에 TCP 위에 구현되었습니다.

멀티캐스트 IP를 사용하는 것도 대안이지만 현재로서는 광범위하게 사용 가능하거나 지원되지 않습니다.

8.1.1 유닉스 소켓 지원

Unix 도메인 소켓이 수신/연결 작업을 허용하는 경우 현재 구현은 Unix 도메인 소켓에서 클라이언트와 서버 연결을 모두 수신하고 수락하도록 구성할 수 있습니다. 이는 호스트 이름이 '/'로 시작하는 소켓으로 인식됩니다.

Unix 도메인 소켓의 연결에 대한 정보를 제공할 때 서버는 실제 소켓 이름을 요청하지 않는 한 경로 이름 대신 실제 호스트 이름을 대체해야 합니다.

8.2 명령 구문 분석

클라이언트와 서버에 유용한 '비버퍼' 네트워크 IO를 제공하기 위해 각 연결에는 가장 최근의 읽기 및 구문 분석 결과가 보관되는 자체 전용 '입력 버퍼'가 제공됩니다. 1개의 전체 메시지를 보관하기 위해 512바이트의 버퍼 크기가 사용되지만 일반적으로 여러 명령이 보관됩니다. 개인 버퍼는 유효한 메시지에 대한 모든 읽기 작업 후에 구문 분석됩니다. 버퍼에 있는 한 클라이언트의 여러 메시지를 처리할 때 클라이언트가 '제거'되는 경우가 있으므로 주의해야 합니다.

8.3 메시지 전달

네트워크 링크가 포화되었거나 데이터를 보낼 수 없는 호스트나 데이터를 보내는 호스트를 찾는 것이 일반적입니다. Unix는 일반적으로 TCP 창과 내부 버퍼를 통해 이를 처리하지만 서버에는 전송할 데이터 양이 많은 경우가 많습니다(특히 새 서버 간 링크가 형성되는 경우).

나가는 대기열에 커널이 충분하지 않습니다. 이 문제를 완화하기 위해 전송될 데이터에 대한 FIFO 대기열로 "전송 대기열"이 사용됩니다.

일반적인 "전송 대기열"은 새 서버가 연결될 때 네트워크 연결이 느린 대규모 IRC 네트워크에서 200KB까지 커질 수 있습니다.

연결을 폴링할 때 서버는 먼저 들어오는 모든 데이터를 읽고 구문 분석하여 전송할 데이터를 대기열에 넣습니다. 사용 가능한 모든 입력이 처리되면 대기 중인 데이터가 전송됩니다. 이렇게 하면 write() 시스템 호출 수가 줄어들고 TCP가 더 큰 패킷을 만드는 데 도움이 됩니다.

8.4 연결 '활성'

연결이 끊기거나 응답하지 않는 시기를 감지하려면 서버는 지정된 시간 내에 응답을 받지 못하는 각 연결을 ping해야 합니다.

연결이 시간 내에 응답하지 않으면 적절한 절차를 사용하여 연결을 닫습니다. 서버 프로세스 블록을 두는 것보다 느린 연결을 닫는 것이 더 낫기 때문에 sendq가 허용된 최대값을 초과하는 경우에도 연결이 삭제됩니다.

8.5 서버-클라이언트 연결 설정

IRC 서버에 연결되면 클라이언트는 MOTD(있는 경우)와 현재 사용자/서버 수(LUSER 명령에 따라)를 보냅니다. 서버는 또한 이름과 버전을 명시하는 명확한 메시지와 적절하다고 간주되는 기타 소개 메시지를 클라이언트에게 제공해야 합니다.

이 문제를 처리한 후 서버는 자체적으로 제공되고(USER 명령) 서버가 검색할 수 있는 대로(DNS/인증 서버에서) 새 사용자의 별명과 기타 정보를 보내야 합니다.

서버는 이 정보를 먼저 NICK, 그 다음 USER로 보내야 합니다.

8.6 서버-서버 연결 설정.

서버 간 연결을 설정하는 프로세스에는 문제가 발생할 수 있는 영역이 많이 있기 때문에 위험이 따릅니다. 그 중 경쟁 조건은 가장 적습니다.

서버가 유효한 것으로 인식된 PASS/SERVER 쌍에 따른 연결을 수신한 후, 서버는 설명된 대로 알고 있는 다른 모든 상태 정보뿐만 아니라 해당 연결에 대한 자체 PASS/SERVER 정보로 응답해야 합니다. 아래에.

시작 서버가 PASS/SERVER 쌍을 수신하면 서버도 마찬가지입니다.

해당 서버에 대한 연결을 수락하기 전에 응답하는 서버가 올바르게 인증되었는지 확인합니다.

8.6.1 연결 시 서버 상태 정보 교환

서버 간에 교환되는 상태 정보의 순서는 필수적입니다. 필수 순서는 다음과 같습니다.

* 알려진 다른 모든 서버;

* 알려진 모든 사용자 정보;

* 알려진 모든 채널 정보.

서버 관련 정보는 추가 SERVER 메시지, 사용자 정보는 NICK/USER/MODE/JOIN 메시지, 채널은 MODE를 통해 전송됩니다.

메시지.

참고: TOPIC 명령이 이전 주제 정보를 덮어쓰기 때문에 채널 주제는 여기에서 교환되지 *않습니다*. 따라서 기껏해야 연결의 양측이 주제를 교환합니다.

서버에 대한 상태 정보를 먼저 전달함으로써 특정 닉네임을 도입한 두 번째 서버로 인해 닉네임 충돌이 발생하기 전에 이미 존재하는 서버와의 충돌이 먼저 발생합니다. IRC 네트워크는 비순환 그래프로만 존재할 수 있기 때문에 네트워크가 이미 다른 위치, 즉 충돌이 발생하는 곳에서 네트워크가 분할되어야 하는 위치에 다시 연결되었을 수도 있습니다.

8.7 서버-클라이언트 연결 종료

클라이언트 연결이 닫히면 클라이언트가 연결된 서버가 클라이언트를 대신하여 QUIT 메시지를 생성합니다. 다른 메시지는 생성되거나 사용되지 않습니다.

8.8 서버 간 연결 종료

서버 간 연결이 원격으로 생성된 SQUIT 또는 '자연스러운' 원인을 통해 닫히면 연결된 IRC 네트워크의 나머지 부분은 닫힘을 감지한 서버에 의해 업데이트된 정보를 가져야 합니다. 그런 다음 서버는 SQUIT 목록(해당 연결 뒤에 있는 각 서버에 하나씩)과 QUIT 목록(다시 말하면 해당 연결 뒤에 있는 각 클라이언트에 하나씩)을 보냅니다.

8.9 닉네임 변경 추적

모든 IRC 서버는 최근 닉네임 변경 내역을 보관해야 합니다. 이는 닉네임 변경 경쟁 조건이 해당 항목을 조작하는 명령으로 발생할 때 서버가 해당 항목에 계속 연락할 수 있도록 하는 데 필요합니다. 닉 변경을 추적해야 하는 명령은 다음과 같습니다.

* KILL (닉이 죽임을 당함)

* 모드(+/- o,v)

* KICK (킥되는 닉네임)

다른 명령에서는 닉네임 변경 사항을 확인할 수 없습니다.

위의 경우 서버는 먼저 닉네임이 있는지 확인한 다음 해당 닉네임이 현재 누구에게 속해 있는지(누구인지!) 기록을 확인해야 합니다. 이렇게 하면 경쟁 조건이 발생할 가능성이 줄어들지만 서버가 잘못된 클라이언트에 영향을 미치면서 여전히 발생할 수 있습니다. 위 명령에 대한 변경 추적을 수행할 때 시간 범위를 지정하고 너무 오래된 항목은 무시하는 것이 좋습니다.

합리적인 기록을 위해 서버는 모든 클라이언트가 변경하기로 결정했는지 알고 있는 모든 클라이언트에 대해 이전 별명을 유지할 수 있어야 합니다. 이 크기는 다른 요인(예: 메모리 등)에 의해 제한됩니다.

8.10 클라이언트의 홍수 통제

상호 연결된 IRC 서버의 대규모 네트워크를 사용하면 네트워크에 연결된 단일 클라이언트가 연속적인 메시지 스트림을 제공하기가 매우 쉽습니다. 이로 인해 네트워크가 넘칠 뿐만 아니라 다른 사용자에게 제공되는 서비스 수준이 저하됩니다. 모든 '피해자'에게 자신의 보호 기능을 제공하도록 요구하는 대신 홍수 방지 기능이 서버에 기록되어 서비스를 제외한 모든 클라이언트에 적용됩니다. 현재 알고리즘은 다음과 같습니다.

* 클라이언트의 '메시지 타이머'가 현재 시간보다 짧은지 확인합니다(그렇다면 동일하게 설정).

* 클라이언트로부터 존재하는 모든 데이터를 읽습니다.

* 타이머가 현재 시간보다 10초 미만인 동안 현재 메시지를 구문 분석하고 각 메시지에 대해 클라이언트에 2초씩 페널티를 줍니다.

이는 본질적으로 클라이언트가 2분마다 1개의 메시지를 보낼 수 있음을 의미합니다.

부정적인 영향을 받지 않고 초.

8.11 비차단 조회

실시간 환경에서는 모든 클라이언트가 공평하게 서비스를 받을 수 있도록 서버 프로세스가 가능한 한 대기 시간을 최소화하는 것이 중요합니다. 분명히 이를 위해서는 모든 네트워크 읽기/쓰기 작업에 대한 비차단 IO가 필요합니다. 일반적인 서버 연결의 경우 어렵지 않았지만 서버를 차단할 수 있는 다른 지원 작업(예: 디스크 읽기)이 있습니다. 가능하다면 그러한 활동은 짧은 시간 초과로 수행되어야 합니다.

8.11.1 호스트 이름(DNS) 조회

Berkeley 및 다른 곳의 표준 확인자 라이브러리를 사용하면 응답 시간이 초과되는 경우에 큰 지연이 발생했습니다. 이를 방지하기 위해 비차단 IO 작업을 위해 설정된 다음 기본 서버 IO 루프 내에서 폴링되는 별도의 DNS 루틴 세트가 작성되었습니다.

8.11.2 사용자 이름(Ident) 조회

다른 프로그램에 사용하고 포함할 수 있는 ID 라이브러리가 많지만 동기식으로 작동하고 빈번한 지연이 발생하므로 문제가 발생했습니다. 이번에도 해결책은 서버의 나머지 부분과 협력하고 비차단 IO를 사용하여 작동하는 일련의 루틴을 작성하는 것이었습니다.

8.12 구성 파일

서버를 설정하고 실행하는 유연한 방법을 제공하려면 다음 사항에 대한 서버 지침이 포함된 구성 파일을 사용하는 것이 좋습니다.

* 클라이언트 연결을 수락할 호스트;

* 서버로 연결을 허용할 호스트;

* 연결할 호스트(활성 및
수동적으로);

* 서버 위치에 대한 정보(대학, 도시/주, 회사 등)

* 서버 및 이메일 주소에 대한 책임은 누구에게 있습니까?
연락 가능한 곳

* 제공을 원하는 클라이언트의 호스트 이름과 비밀번호

제한된 운영자 명령에 대한 액세스.

호스트 이름을 지정할 때 도메인 이름과 '점' 표기법(127.0.0.1) 사용이 모두 허용되어야 합니다. 나가고 들어오는 모든 연결에 대해 사용/수락할 비밀번호를 지정할 수 있어야 합니다(나가는 연결만 다른 서버에 대한 연결임에도 불구하고).

위 목록은 다른 서버와 연결을 원하는 서버의 최소 요구 사항입니다. 사용할 수 있는 기타 항목은 다음과 같습니다.

* 다른 서버가 도입할 수 있는 서버를 지정합니다.

* 서버 분기가 얼마나 깊어질 수 있는지;

* 클라이언트가 연결할 수 있는 시간.

8.12.1 클라이언트 연결 허용

서버는 시작 시 읽고 클라이언트가 서버에 연결하는 데 사용할 수 있는 호스트를 결정하는 데 사용되는 일종의 '액세스 제어 목록'(구성 파일이나 다른 곳에 있음)을 사용해야 합니다.

호스트 액세스 제어에 필요한 유연성을 제공하려면 '거부'와 '허용'을 모두 구현해야 합니다.

8.12.2 연산자

파괴적인 사람에게 운영자 권한을 부여하는 것은 그들에게 부여된 권한으로 인해 일반적으로 IRC 넷의 안녕에 심각한 결과를 초래할 수 있습니다. 따라서 그러한 권한을 획득하는 것이 그리 쉽지는 않습니다. 현재 설정에서는 두 개의 '비밀번호'를 사용해야 하지만 그 중 하나는 일반적으로 쉽게 추측할 수 있습니다. 구성 파일에 운영자 비밀번호를 저장하는 것이 하드 코딩하는 것보다 바람직하며 쉽게 도난당하지 않도록 암호화된 형식(예: Unix의 crypt(3) 사용)으로 저장해야 합니다.

8.12.3 서버 연결 허용

서버의 상호 연결은 사소한 문제가 아닙니다. 잘못된 연결은 IRC의 유용성에 큰 영향을 미칠 수 있습니다. 따라서 각 서버에는 연결할 수 있는 서버 목록과 연결할 수 있는 서버 목록이 있어야 합니다. 어떤 상황에서도 서버는 임의의 호스트가 서버로 연결되는 것을 허용해서는 안 됩니다. 어떤 서버가 연결될 수 있고 연결되지 않을 수 있는지 외에도 구성 파일은 해당 링크의 비밀번호와 기타 특성도 저장해야 합니다.

8.12.4 관리

ADMIN 명령(섹션 4.3.7 참조)에 정확하고 유효한 응답을 제공하려면 서버가 구성에서 관련 세부 정보를 찾아야 합니다.

8.13 채널 멤버십

현재 서버에서는 등록된 로컬 사용자가 최대 10개의 다른 채널에 참여할 수 있습니다. 서버가 채널 멤버십 기반으로 다른 모든 사용자와 (합리적으로) 일관성을 유지하도록 로컬이 아닌 사용자에게 제한이 없습니다.

9. 현재의 문제

이 프로토콜에는 여러 가지 인식된 문제가 있으며, 모두 재작성 중에 가까운 시일 내에 해결될 수 있기를 바랍니다. 현재 이러한 문제에 대한 효과적인 해결책을 찾기 위한 작업이 진행 중입니다.

9.1 확장성

이 프로토콜은 대규모 경기장에서 사용될 때 충분히 확장되지 않는다는 것이 널리 알려져 있습니다. 주요 문제는 모든 서버가 다른 모든 서버와 사용자에 대해 알고 있어야 하고 이들에 관한 정보가 변경되는 즉시 업데이트되어야 한다는 요구 사항에서 비롯됩니다. 또한 두 지점 사이의 경로 길이를 최소화하고 스패닝 트리가 최대한 강력하게 분기되도록 서버 수를 낮게 유지하는 것이 바람직합니다.

9.2 라벨

현재 IRC 프로토콜에는 닉네임, 채널 이름, 서버 이름의 세 가지 유형의 레이블이 있습니다. 세 가지 유형 각각에는 자체 도메인이 있으며 해당 도메인 내에서는 중복이 허용되지 않습니다.

현재는 사용자가 세 가지 중 하나의 라벨을 선택하여 충돌이 발생할 수 있습니다. 순환 트리를 허용하는 솔루션뿐만 아니라 충돌하지 않는 채널 및 닉에 대한 고유한 이름에 대한 계획을 세우는 것이 바람직하다는 점은 재작업이 필요하다는 점을 널리 인식하고 있습니다.

9.2.1 별명

IRC의 닉네임 아이디어는 사용자가 채널 외부에서 서로 대화할 때 사용하기 매우 편리하지만 닉네임 공간은 한정되어 있고 그대로 사용하기 때문에 여러 사람이 동일한 것을 사용하는 경우가 많습니다. 건강 상태. 이 프로토콜을 사용하여 두 사람이 닉네임을 선택하면 둘 중 하나가 성공하지 못하거나

둘 다 KILL(4.6.1)을 사용하여 제거됩니다.

9.2.2 채널

현재 채널 레이아웃에서는 모든 서버가 모든 채널, 해당 채널의 거주자 및 속성에 대해 알아야 합니다. 확장성이 좋지 않은 것 외에도 개인 정보 보호 문제도 우려됩니다. 채널 충돌은 닉네임 충돌을 해결하는 데 사용되는 것과 같은 단독 이벤트가 아닌 포괄적 이벤트(새 채널을 생성하는 두 사람 모두 해당 채널의 구성원으로 간주됨)로 처리됩니다.

9.2.3 서버

일반적으로 서버 수는 사용자 및 채널 수에 비해 적지만 현재는 두 서버를 각각 별도로 또는 마스크 뒤에 숨겨 전역적으로 알려야 합니다.

9.3 알고리즘

서버 코드 내의 일부 위치에서는 클라이언트 집합의 채널 목록을 확인하는 것과 같은 N^2 알고리즘을 피하는 것이 불가능했습니다.

현재 서버 버전에는 데이터베이스 일관성 검사가 없으며 각 서버는 인접 서버가 올바른 것으로 가정합니다. 연결 서버에 버그가 있거나 기존 네트워크에 모순이 발생하려고 하면 큰 문제가 발생할 수 있습니다.

현재 고유한 내부 및 글로벌 레이블이 없기 때문에 다양한 경쟁 조건이 존재합니다. 이러한 경쟁 조건은 일반적으로 메시지가 IRC 네트워크를 통과하여 영향을 미치는 데 시간이 걸리는 문제로 인해 발생합니다. 고유한 라벨로 변경하더라도 채널 관련 명령이 중단되는 문제가 있습니다.

10. 현재 지원 및 가용성

IRC 관련 토론을 위한 메일링 리스트:
향후 프로토콜: ircd-3-request@eff.org 일반 토론: operlist-request@eff.org

소프트웨어 구현 cs.bu.edu:/irc nic.funet.fi:/
pub/irc coombs.anu.edu.au:/
pub/irc

뉴스그룹: alt.irc

보안 고려 사항

보안 문제는 섹션 4.1, 4.1.1, 4.1.3, 5.5 및 7에서 논의됩니다.

12. 저자 주소

자르코 오이카리넨
Tuirantie 17 9로
90500 오울루
핀란드

이메일: jto@tolsun.oulu.fi

대런 리드
4 페이트먼 스트리트
왓슨니아, 빅토리아 3087
호주

이메일: avalon@coombs.anu.edu.au