Network Working Group
Request for Comments: 2810
Updates: 1459
Category: Informational

C. Kalt

April 2000

Internet Relay Chat: Architecture

Status of this Memo

Copyright Notice

Abstract

The IRC (Internet Relay Chat) protocol is for use with text based conferencing. It has been developed since 1989 when it was originally implemented as a mean for users on a BBS to chat amongst themselves.

First formally documented in May 1993 by RFC 1459 [IRC], the protocol has kept evolving. This document is an update describing the architecture of the current IRC protocol and the role of its different components. Other documents describe in detail the protocol used between the various components defined here.

Table of Contents

1. Introduction

The IRC (Internet Relay Chat) protocol has been designed over a number of years for use with text based conferencing. This document describes its current architecture.

The IRC Protocol is based on the client-server model, and is well suited to running on many machines in a distributed fashion. A typical setup involves a single process (the server) forming a central point for clients (or other servers) to connect to, performing the required message delivery/multiplexing and other functions.

This distributed model, which requires each server to have a copy of the global state information, is still the most flagrant problem of the protocol as it is a serious handicap, which limits the maximum size a network can reach. If the existing networks have been able to keep growing at an incredible pace, we must thank hardware manufacturers for giving us ever more powerful systems.

2. Components

The following paragraphs define the basic components of the IRC protocol.

2.1 Servers

The server forms the backbone of IRC as it is the only component of the protocol which is able to link all the other components together: it provides a point to which clients may connect to talk to

each other [IRC-CLIENT], and a point for other servers to connect to [IRC-SERVER]. The server is also responsible for providing the basic services defined by the IRC protocol.

2.2 Clients

A client is anything connecting to a server that is not another server. There are two types of clients which both serve a different purpose.

2.2.1 User Clients

User clients are generally programs providing a text based interface that is used to communicate interactively via IRC. This particular type of clients is often referred as "users".

2.2.2 Service Clients

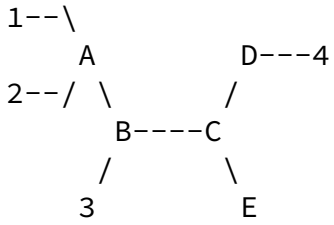
Unlike users, service clients are not intended to be used manually nor for talking. They have a more limited access to the chat functions of the protocol, while optionally having access to more private data from the servers.

Services are typically automatons used to provide some kind of service (not necessarily related to IRC itself) to users. An example is a service collecting statistics about the origin of users connected on the IRC network.

3. Architecture

An IRC network is defined by a group of servers connected to each other. A single server forms the simplest IRC network.

The only network configuration allowed for IRC servers is that of a spanning tree where each server acts as a central node for the rest of the network it sees.



Servers: A, B, C, D, E Clients: 1, 2, 3, 4

[Fig. 1. Sample small IRC network]

The IRC protocol provides no mean for two clients to directly communicate. All communication between clients is relayed by the server(s).

4. IRC Protocol Services

This section describes the services offered by the IRC protocol. The combination of these services allow real-time conferencing.

4.1 Client Locator

To be able to exchange messages, two clients must be able to locate each other.

Upon connecting to a server, a client registers using a label which is then used by other servers and clients to know where the client is located. Servers are responsible for keeping track of all the labels being used.

4.2 Message Relaying

The IRC protocol provides no mean for two clients to directly communicate. All communication between clients is relayed by the server(s).

4.3 Channel Hosting And Management

A channel is a named group of one or more users which will all receive messages addressed to that channel. A channel is

characterized by its name and current members, it also has a set of properties which can be manipulated by (some of) its members.

Channels provide a mean for a message to be sent to several clients. Servers host channels, providing the necessary message multiplexing. Servers are also responsible for managing channels by keeping track of the channel members. The exact role of servers is defined in "Internet Relay Chat: Channel Management" [IRC-CHAN].

5. IRC Concepts

This section is devoted to describing the actual concepts behind the organization of the IRC protocol and how different classes of messages are delivered.

5.1 One-To-One Communication

Communication on a one-to-one basis is usually performed by clients, since most server-server traffic is not a result of servers talking only to each other. To provide a means for clients to talk to each other, it is REQUIRED that all servers be able to send a message in exactly one direction along the spanning tree in order to reach any client. Thus the path of a message being delivered is the shortest path between any two points on the spanning tree.

The following examples all refer to Figure 1 above.

Example 1: A message between clients 1 and 2 is only seen by server A, which sends it straight to client 2.

Example 2: A message between clients 1 and 3 is seen by servers A & B, and client 3. No other clients or servers are allowed see the message.

Example 3: A message between clients 2 and 4 is seen by servers A, B, C & D and client 4 only.

5.2 One-To-Many

The main goal of IRC is to provide a forum which allows easy and efficient conferencing (one to many conversations). IRC offers several means to achieve this, each serving its own purpose.

5.2.1 To A Channel

In IRC the channel has a role equivalent to that of the multicast group; their existence is dynamic and the actual conversation carried out on a channel MUST only be sent to servers which are supporting users on a given channel. Moreover, the message SHALL only be sent once to every local link as each server is responsible to fan the original message to ensure that it will reach all the recipients.

The following examples all refer to Figure 2.

Example 4: Any channel with 1 client in it. Messages to the channel go to the server and then nowhere else.

Example 5: 2 clients in a channel. All messages traverse a path as if they were private messages between the two clients outside a channel.

Example 6: Clients 1, 2 and 3 in a channel. All messages to the channel are sent to all clients and only those servers which must be traversed by the message if it were a private message to a single client. If client 1 sends a message, it goes back to client 2 and then via server B to client 3.

5.2.2 To A Host/Server Mask

To provide with some mechanism to send messages to a large body of related users, host and server mask messages are available. These messages are sent to users whose host or server information match that of the mask. The messages are only sent to locations where users are, in a fashion similar to that of channels.

5.2.3 To A List

The least efficient style of one-to-many conversation is through clients talking to a 'list' of targets (client, channel, mask). How this is done is almost self explanatory: the client gives a list of destinations to which the message is to be delivered and the server breaks it up and dispatches a separate copy of the message to each given destination.

This is not as efficient as using a channel since the destination list MAY be broken up and the dispatch sent without checking to make sure duplicates aren't sent down each path.

5.3 One-To-All

The one-to-all type of message is better described as a broadcast message, sent to all clients or servers or both. On a large network of users and servers, a single message can result in a lot of traffic being sent over the network in an effort to reach all of the desired destinations.

For some class of messages, there is no option but to broadcast it to all servers so that the state information held by each server is consistent between servers.

5.3.1 Client-to-Client

There is no class of message which, from a single message, results in a message being sent to every other client.

5.3.2 Client-to-Server

Most of the commands which result in a change of state information (such as channel membership, channel mode, user status, etc.) MUST be sent to all servers by default, and this distribution SHALL NOT be changed by the client.

5.3.3 Server-to-Server

While most messages between servers are distributed to all 'other' servers, this is only required for any message that affects a user, channel or server. Since these are the basic items found in IRC, nearly all messages originating from a server are broadcast to all other connected servers.

6. Current Problems

There are a number of recognized problems with this protocol, this section only addresses the problems related to the architecture of the protocol.

6.1 Scalability

It is widely recognized that this protocol does not scale sufficiently well when used in a large arena. The main problem comes from the requirement that all servers know about all other servers, clients and channels and that information regarding them be updated as soon as it changes.

6.2 Reliability

As the only network configuration allowed for IRC servers is that of a spanning tree, each link between two servers is an obvious and quite serious point of failure. This particular issue is addressed more in detail in "Internet Relay Chat: Server Protocol" [IRC-SERVER].

6.3 Network Congestion

Another problem related to the scalability and reliability issues, as well as the spanning tree architecture, is that the protocol and architecture for IRC are extremely vulnerable to network congestions. This problem is endemic, and should be solved for the next generation: if congestion and high traffic volume cause a link between two servers to fail, not only this failure generates more network traffic, but the reconnection (eventually elsewhere) of two servers also generates more traffic.

In an attempt to minimize the impact of these problems, it is

strongly RECOMMENDED that servers do not automatically try to reconnect too fast, in order to avoid aggravating the situation.

6.4 Privacy

Besides not scaling well, the fact that servers need to know all information about other entities, the issue of privacy is also a concern. This is in particular true for channels, as the related information is quite a lot more revealing than whether a user is online or not.

7. Security Considerations

Asides from the privacy concerns mentioned in section 6.4 (Privacy), security is believed to be irrelevant to this document.

8. Current Support And Availability

Mailing lists for IRC related discussion:
  General discussion: ircd-users@irc.org
  Protocol development: ircd-dev@irc.org

Software implementations:
  ftp://ftp.irc.org/irc/server
  ftp://ftp.funet.fi/pub/unix/irc
  ftp://coombs.anu.edu.au/pub/irc

Newsgroup: alt.irc

9. Acknowledgements

Parts of this document were copied from the RFC 1459 [IRC] which first formally documented the IRC Protocol. It has also benefited from many rounds of review and comments. In particular, the following people have made significant contributions to this document:

Matthew Green, Michael Neumayer, Volker Paulsen, Kurt Roeckx, Vesa Ruokonen, Magnus Tjernstrom, Stefan Zehl.

10. References

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

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

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

[IRC-SERVER] Kalt, C., "Internet Relay Chat: Server Protocol", RFC 2813, April 2000.

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

11. Author's Address

Christophe Kalt
99 Teaneck Rd, Apt #117
Ridgefield Park, NJ 07660
USA

EMail: kalt@stealth.net

12. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.