The Client/Server Bulletin Board System (csbss)

Jim Van Peursem – KEOPH
Bob Arasmith – NOARY

Abstract

The current Bulletin Board System (BBS) network is showing signs of age, There are problems with duplicate messages, new users are intimidated by their interface, the channel is becoming congested, etc. By looking at the underlying assumptions of the current systems, significant improvements can be made. Some of them require substantial changes, but the benefits far outweigh the trouble involved. This paper looks at a client/server architecture currently being developed by NOARY, KEOPH, WB6YRU, and N6ZFJ, called the csbbs (client/server bbs). The paper describes the goals of the new system, and some of the ideas being considered.

Introduction

Typical BBS's today are becoming more and more powerful. They are handling an exponentially increasing number of messages every day. They are also offering more features to their users. As compared to BBS systems just a few years ago, today's systems have added features like callsign servers, HF gateways, multiple ports, internet gateways, etc.

These advances are truly great, but because of the extra load, some other changes need to be made in order to keep the system useful and feasible into the future. Two of these changes come in the area of user interface, and channel efficiency.

User interface

BBS technology has recently been progressing in every area except the user interface. Today's BBS systems are still command line driven, just as they have always been, To use any of their features, you need to know the commands. To learn the commands, you need to read the online manual, which consumes bandwidth.

With the powerful Graphical User Interface (GUI) on today's computers, it seems silly to force the command line on all users. Why is it still done? Because that is the easiest interface to program. Besides, what other alternative is there? The BBS cannot assume any one particular architecture because there are many including Macintosh, MS-DOS, MS-Windows, C-64, Apple II, etc. Each one has it's own unique interface, and the users of these systems like to see that type of interface used. It would be best if the BBS could be separated into two pieces: one that handles all of the normal functions and message forwarding, and one that handles the user interface. This is easily done with a client/server model.

Client/Server model

The client/server model is a well known method that is used in many networking situations. It is the cornerstone for many services on intemet such as WAIS, Gopher, Archie, NFS, X-windows, etc. This is an ideal model for today's BBS systems as well. The BBS that you currently know would become the server, and a new piece, the client, would be introduced to handle the user interface. The client will run locally on the user's computer, and talk to the server at a very low level. In this way, the BBS doesn't have to worry about presenting any kind of user interface, and users can use the BBS in the comfort of their own user interface.

To better illustrate how this would work, consider the diagram in Figure 1. As you can see, the server assumes most of the responsibility of today's BBS. It forwards bulletins, stores and forwards all messages, etc. It has no interface to the user, but rather an interface to the client application. The client application is responsible for presenting the information to the user and handling user commands. Don't get this confused with the way things work now. Users won't send commands to the server. The user will interact with the client application by using the mouse and menu commands for example. Only the client application will interact with the server.

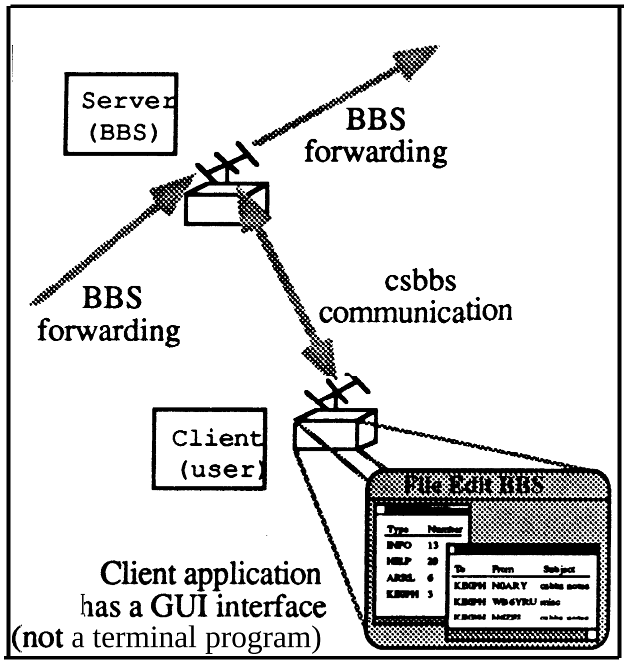

Figure 1: Client/Server illustration

When the user invokes an action that requires the server's attention, the client properly formats that request and sends it to the server. For example, if the user requests to read a bulletin (by clicking the mouse button on a message entry for example), the client will send the appropriate low level command to read it (not "r 1234"). The server will then send that message to the client in a very efficient manner. See "How to improve channel utilization" below.

The client can then store that message locally so that if the user ever wants to read it again, they can without consuming any network bandwidth. It will be listed and displayed in the same way as non-local bulletins so it will be a very familiar interface. It will be a convenient way to archive messages for later reference. This has other advantageous side effects which will be discussed later in "Message Listing".

How to improve channel utilization

Another problem with current systems is that they consume far more bandwidth than they really need to. By taking advantage of some simple techniques, the BBS's impact on the channel utilization can be changed dramatically.

Most of the utilization gains will again come from the client/server model. Since the BBS (server) isn't responsible for providing any kind of user interface, it can transfer the information in a way that only the client can decode. For example, rather than sending the message number 12345 as a text number, which takes 5 bytes, it can be sent as one 16-bit number, which takes 2 bytes. The flag bits could also fit in that same 16-bit number, for a savings of a few more bytes per message listing.

The subject, and all other text components can be compressed before transmission which would save even more. Text is very easy to compress, so a 50% savings is not unheard of. In fact, the entire body of the message would likely be sent in compressed binary to give a greater savings.

These techniques will result in some savings of channel bandwidth. Some larger savings will come because the client stores message lists locally. So for example, if a user wants to see the message listing in a different order, no transmissions are necessary. The client will use it's internal list to derive the new listing.

One more large channel utilization improvement will come from sending message listings and messages to multiple recipients at once, If N stations are receiving the information at once, a bandwidth reduction of N times is realized. This too will require some special technique because the AX.25 connection model only allows point-to-point connections [1]. Two alternatives are available.

Either the clients can snoop copies of messages being read by others, or the server can broadcast messages at the less utilized times in the day. This second method would, in effect, offer some load balancing. If users have the messages stored locally, because of the broadcasts at non-peak times, they won't need to connect to the BBS during the high congestion times.

The messages can be sent using the UI protocol, and adding internal message index numbers. For example, at the beginning of sending the body of a message, first send the BID string. This will uniquely identify this message. Then number every frame sent so that a client knows if it has missed a frame. When this situation arises it can either send a request to immediately fill this frame, or wait until the next time the message is sent and grab a copy of the missing frame.

In general, it seems appropriate for the server to send out messages at periodic times throughout the day. This would be done in an effort to maintain some sort of channel load balancing. If messages can be sent from server to clients during the very light utilization times, it will help ease the heavy usage times. This will require a lot of experimentation to determine if it is feasible or not. It might also border on some legal issues. This may be determined to be broadcasting since it is a simplex operation. The term, "dispersion" of information is preferred.

Initially, the message snooping idea seems more feasible. Since there are so many new messages arriving at the BBS's every day, it will take a lot of dispersion for them to get to everyone. However, the more popular messages are read multiple times by users, so the possibility of successfully snooping a copy of them is very likely. The message listing can also be snooped this way. In fact, the clients may even be able to snoop messages during BBS forwarding too.

Server Arbitration

Since BBS's are a. hot spot for many users, they are a cause for great network congestion. While many stations are competing to communicate with them, there is a great potential for collisions. Some sort of arbitration scheme may help out.

Some areas in Germany are using DAMA [2] as an arbitration mechanism for their node stations. We will be experimenting with this technique and probably others in an effort to reduce the hidden transmitter problem and collisions in general.

Message Listings

Another area where BBS systems have not shown much progress is in the area of viewing message listings. The standard viewing technique is still chronological. The listing goes from the most recently received message, to the least recently received message. This is a very cumbersome way for users to view messages. What about viewing them by subject or who they are addressed to or from?

Hams are starting to address their bulletins in a more useful way, such as using the TO: field to give the general category of the bulletin. This is very useful and should be taken advantage of better. The user should be able to view messages by categories. The user should also be able to search the subject field for certain words, or use regular expression searches. NOARY currently provides these powerful capabilities in his BBS software, and they are very popular.

All of these things can be done very efficiently now because the message listing will only be sent from the server to the client once. The client will store the listing and present it to the user in any way that they want without ever using any network bandwidth, because it is totally a local operation. If the user wants to see a different subset or see the messages in a different order, the client application takes care of this request locally.

Conclusion

The client/server idea for BBS's will not take over immediately, but it will be fairly simple to use the existing BBS systems along with the csbbs systems. In fact, current BBS systems can set up another SSID port that acts as the server port. That way, those users that don't have a client application can still use the BBS in the way that it is used today. The BBS and server parts can share the same internal storage and functionality.

The project is currently in the early coding stage. We are still debating many issues. We hope to have some very early results soon. NOARY will be adding the server protocol to his BBS software, and others will be working on client applications for different computers.

We are trying to make the interface between the client and server as general as possible so that it will work for all computer systems. We currently have representatives working on client applications for the Macintosh (KEOPH), MS-DOS (WB6YRU), and MS-Windows (N6ZFJ) architectures. We would like to see the C-64, Apple II, etc. represented as well so that we don't inadvertently make any bad assumptions. If you would like to volunteer, contact NOARY for more details.

Acknowledgments

Many thanks to NOARY for heading this great idea. He has an outstanding BBS program that will be used as the starting point for the server application. Thanks also to the people currently working on the client applications. For more information, feel free to contact them directly.

Bob Arasmith – NOARY – Server
837 Jasmine Dr.
Sunnyvale, CA 94086
NOARY @ NOARY.#NOCAL.CA.USA
internet: n0ary@arasmith.com

Jim Van Peursem – KEOPH – Macintosh
RR#1, Box 83A
Kelley, IA 50134
KEOPH @ NOAN.IA.USA
intemet: jvp@iastate.edu

Gary Mitchell – WB6YRU -MS-DOS
185 1 Laurinda Dr.
San Jose, CA 95124
WB6YRU @ NOARY.#NOCAL.CA.USA

Connie Arasmith – N6ZFJ – MS-Windows
837 Jasmine Dr.
Sunnyvale, CA 94086
N6ZFJ @ NOARY .#NOCAL.CA.USA

References

[1] T.L. Fox – WB4JFI, *AX.25 Amateur Packet-Radio Link-Layer Protocol. Version 2 .O* Newington, CT: ARRL, 1984.

[2] Detlef J. Schmidt – DK4EG, *DAMA – A New Method of Handling Packets?* ARRL Amateur Radio 8th Computer Networking Conference, 1989, pg 203–209.