Network Working Group
Request for Comments: 947

Ken Lebowitz
David Mankins
BBN Laboratories
June 1985

Multi-network Broadcasting within the Internet

1. Status of this Memo

This RFC describes the extension of a network's broadcast domain to include more than one physical network through the use of a broadcast packet repeater.

The following paper will present the problem of multi-network broadcasting and our motivation for solving this problem which is in the context of developing a distributed operating system. We discuss different solutions to extending a broadcast domain and why we chose the one that has been implemented. In addition, there is information on the implementation itself and some notes on its performance.

It is hoped that the ideas presented here will help people in the Internet who have applications which make use of broadcasting and have come up against the limitation of only being able to broadcast within a single network.

The information presented here is accurate as of the date of publication but specific details, particularly those regarding our implementation, may change in the future. Distribution of this memo is unlimited.

2. The Problem

Communication between hosts on separate networks has been addressed largely through the use of Internet protocols and gateways. One aspect of internetwork communication that hasn't been solved in the Internet is extending broadcasting to encompass two or more networks. Broadcasting is an efficient way to send information to many hosts while only having to transmit a single packet. Many of the current local area network (LAN) architectures directly support a broadcast mechanism. Unfortunately, this broadcast mechanism has a shortcoming when it is used in networking environments which include multiple LANs connected by gateways such as in the DARPA Internet. This shortcoming is that broadcasted packets are only received by hosts on the physical network on which the packet was broadcast. As a result, any application which takes advantage of LAN broadcasting can only broadcast to those hosts on its physical network.

We took advantage of broadcasting in developing the Cronus Distributed Operating System [1]. Cronus provides services and communication to processes distributed among a variety of different

types of computer systems. Cronus is built around logical clusters of hosts connected to one or more high-speed LANs. Communication in Cronus is built upon the TCP and UDP protocols. Cronus makes use of broadcasting for dynamically locating resources on other hosts and collecting status information from a collection of servers. Since Cronus's broadcast capabilities are not intended to be limited to the boundaries of a single LAN, we needed to find some way to extend our broadcasting domain to include hosts on distant LANs in order to experiment with clusters that span several physical networks. Cronus predominantly uses broadcasting to communicate with a subset of the hosts that actually receive the broadcasted message. A multicast mechanism would be more appropriate, but was unavailable in some of our network implementations, so we chose broadcast for the initial implementation of Cronus utilities.

3. Our Solution

The technique we chose to experiment with the multi-network broadcasting problem can be described as a "broadcast repeater". A broadcast repeater is a mechanism which transparently relays broadcast packets from one LAN to another, and may also forward broadcast packets to hosts on a network which doesn't support broadcasting at the link-level. This mechanism provides flexibility while still taking advantage of the convenience of LAN broadcasts.

Our broadcast repeater is a process on a network host which listens for broadcast packets. These packets are picked up and retransmitted, using a simple repeater-to-repeater protocol, to one or more repeaters that are connected to distant LANs. The repeater on the receiving end will rebroadcast the packet on its LAN, retaining the original packet's source address. The broadcast repeater can be made very intelligent in its selection of messages to be forwarded. We currently have the repeater forward only broadcast messages sent using the UDP ports used by Cronus, but messages may be selected using any field in the UDP or IP headers, or all IP-level broadcast messages may be forwarded.

4. Alternatives to the Broadcast Repeater

We explored a few alternatives before deciding on our technique to forward broadcast messages. One of these methods was to put additional functions into the Internet gateways. Gateways could listen at the link-level for broadcast packets and relay the packets to one or more gateways on distant LANs. These gateways could then transmit the same packet onto their networks using the local network's link-level broadcast capability, if one is available. All gateways participating in this scheme would have to maintain tables

of all other gateways which are to receive broadcasts. If the recipient gateway was serving a network without a capacity to broadcast it could forward the messages directly to one or more designated hosts on its network but, again, it would require that tables be kept in the gateway. Putting this sort of function into gateways was rejected for a number of reasons: (a) it would require extensions to the gateway control protocol to allow updating the lists gateways would have to maintain, (b) since not all messages (e.g., LAN address- resolution messages) need be forwarded, the need to control forwarding should be under the control of higher levels of the protocol than may be available to the gateways, (c) Cronus could be put into environments where the gateways may be provided by alternative vendors who may not implement broadcast propagation, (d) as a part of the underlying network, gateways are likely to be controlled by a different agency from that controlling the configuration of a Cronus system, adding bureaucratic complexity to reconfiguration.

Another idea which was rejected was to put broadcast functionality into the Cronus kernel. The Cronus kernel is a process which runs on each host participating in Cronus, and has the task of routing all messages passed between Cronus processes. The Cronus kernel is the only program in the Cronus system which directly uses broadcast capability (other parts of Cronus communicate using mechanisms provided by the kernel). We could either entirely remove the Cronus kernel's dependence on broadcast, or add a mechanism for emulating broadcast using serially-transmitted messages when the underlying network does not provide a broadcast facility itself. Either solution requires all Cronus kernel processes to know the addresses of all other participants in a Cronus system, which we view as an undesirable limit on configuration flexibility. Also, this solution would be Cronus-specific, while the broadcast-repeater solution is applicable to other broadcast-based protocols.

5. Implementation

The broadcast repeater is implemented as two separate processes - the forwarder and the repeater. The forwarder process waits for broadcast UDP packets to come across its local network which match one or more specific port numbers (or destination addresses). When such a packet is found, it is encapsulated in a forwarder-repeater message sent to a repeater process on a foreign network. The repeater then relays the forwarded packet onto its LAN using that network's link-level broadcast address in the packet's destination field, but preserving the source address from the original packet.

When the forwarder process starts for the first time it reads a

configuration file. This file specifies the addresses of repeater processes, and selects which packets should be forwarded to each repeater process (different repeaters may select different sets of UDP packets). The forwarder attempts to establish a TCP connection to each repeater listed in the configuration file. If a TCP link to a repeater fails, the forwarder will periodically retry connecting to it. Non-repeater hosts may also be listed in the configuration file. For these hosts the forwarder will simply replace the destination broadcast address in the UDP packet with the host's address and send this new datagram directly to the non-repeater host.

If a repeater and a forwarder co-exist on the same LAN a problem may arise if the forwarder picks up packets which have been rebroadcast by the repeater. As a precaution against rebroadcast of forwarded packets ("feedback" or "ringing"), the forwarder does not connect to any repeaters listed in its configuration file which are on the same network as the forwarder itself. Also, to avoid a broadcast loop involving two LANs, each with a forwarder talking to a repeater on the other LAN, forwarders do not forward packets whose source address is not on the forwarder's LAN.

6. Experience

To date, the broadcast repeater has been implemented on the VAX running 4.2 BSD UNIX operating system with BBN's networking software and has proven to work quite well for our purposes. Our current configuration includes two Ethernets which are physically separated by two other LANs. For the past few months the broadcast repeater has successfully extended our broadcast domain to include both Ethernets even though messages between the two networks must pass through at least two gateways. We were forced to add a special capability to the BBN TCP/IP implementation which allows privileged processes to send out IP packets with another host's source address.

The repeater imposes a fair amount of overhead on the shared hosts that currently support it due to the necessity of waking the forwarder process on all UDP packets which arrive at the host, since the decision to reject a packet is made by user-level software, rather than in the network protocol drivers. One solution to this problem would be to implement the packet filtering in the system kernel (leaving the configuration management and rebroadcast mechanism in user code) as has been done by Stanford/CMU in a UNIX packet filter they have developed. As an alternative we are planning to rehost the implementation of the repeater function as a specialized network service provided by a microcomputer based

real-time system which is already part of our Cronus configuration. Such a machine is better suited to the task since scheduling overhead is much less for them than it is on a multi-user timesharing system.

7. Reference

[1] "Cronus, A Distributed Operating System: Phase 1 Final Report", R. Schantz, R. Thomas, R. Gurwitz, G. Bono, M. Dean, K. Lebowitz, K. Schroder, M. Barrow and R. Sands, Technical Report No. 5885, Bolt Beranek and Newman, Inc., January 1985. The Cronus project is supported by the Rome Air Development Center.

8. Editors Note

Also see RFCs 919 and 940 on this topic.