Network Working Group
Request for Comments: 2201
Category: Experimental

A. Ballardie
Consultant

September 1997

Core Based Trees (CBT) Multicast Routing Architecture

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. This memo does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Abstract

CBT is a multicast routing architecture that builds a single delivery tree per group which is shared by all of the group's senders and receivers. Most multicast algorithms build one multicast tree per sender (subnetwork), the tree being rooted at the sender's subnetwork. The primary advantage of the shared tree approach is that it typically offers more favourable scaling characteristics than all other multicast algorithms.

The CBT protocol [1] is a network layer multicast routing protocol that builds and maintains a shared delivery tree for a multicast group. The sending and receiving of multicast data by hosts on a subnetwork conforms to the traditional IP multicast service model [2].

CBT is progressing through the IDMR working group of the IETF. The CBT protocol is described in an accompanying document [1]. For this, and all IDMR-related documents, see http://www.cs.ucl.ac.uk/ietf/idmr

TABLE OF CONTENTS

1. Background

Shared trees were first described by Wall in his investigation into low-delay approaches to broadcast and selective broadcast [3]. Wall concluded that delay will not be minimal, as with shortest-path trees, but the delay can be kept within bounds that may be acceptable. Back then, the benefits and uses of multicast were not fully understood, and it wasn't until much later that the IP multicast address space was defined (class D space [4]). Deering's work [2] in the late 1980's was pioneering in that he defined the IP multicast service model, and invented algorithms which allow hosts to arbitrarily join and leave a multicast group. All of Deering's multicast algorithms build source-rooted delivery trees, with one delivery tree per sender subnetwork. These algorithms are documented in [2].

After several years practical experience with multicast, we see a diversity of multicast applications and correspondingly, a wide variety of multicast application requirements. For example, distributed interactive simulation (DIS) applications have strict requirements in terms of join latency, group membership dynamics, group sender populations, far exceeding the requirements of many other multicast applications.

The multicast-capable part of the Internet, the MBONE, continues to expand rapidly. The obvious popularity and growth of multicast means that the scaling aspects of wide-area multicasting cannot be overlooked; some predictions talk of thousands of groups being present at any one time in the Internet.

We evaluate scalability in terms of network state maintenance, bandwidth efficiency, and protocol overhead. Other factors that can affect these parameters include sender set size, and wide-area distribution of group members.

2. Introduction

Multicasting on the local subnetwork does not require either the presence of a multicast router or the implementation of a multicast routing algorithm; on most shared media (e.g. Ethernet), a host,

which need not necessarily be a group member, simply sends a multicast data packet, which is received by any member hosts connected to the same medium.

For multicasts to extend beyond the scope of the local subnetwork, the subnet must have a multicast-capable router attached, which itself is attached (possibly "virtually") to another multicast-capable router, and so on. The collection of these (virtually) connected multicast routers forms the Internet's MBONE.

All multicast routing protocols make use of IGMP [5], a protocol that operates between hosts and multicast router(s) belonging to the same subnetwork. IGMP enables the subnet's multicast router(s) to monitor group membership presence on its directly attached links, so that if multicast data arrives, it knows over which of its links to send a copy of the packet.

In our description of the MBONE so far, we have assumed that all multicast routers on the MBONE are running the same multicast routing protocol. In reality, this is not the case; the MBONE is a collection of autonomously administered multicast regions, each region defined by one or more multicast-capable border routers. Each region independently chooses to run whichever multicast routing protocol best suits its needs, and the regions interconnect via the "backbone region", which currently runs the Distance Vector Multicast Routing Protocol (DVMRP) [6]. Therefore, it follows that a region's border router(s) must interoperate with DVMRP.

Different algorithms use different techniques for establishing a distribution tree. If we classify these algorithms into source-based tree algorithms and shared tree algorithms, we'll see that the different classes have considerably different scaling characteristics, and the characteristics of the resulting trees differ too, for example, average delay. Let's look at source-based tree algorithms first.

3. Source-Based Tree Algorithms

The strategy we'll use for motivating (CBT) shared tree multicast is based, in part, in explaining the characteristics of source-based tree multicast, in particular its scalability.

Most source-based tree multicast algorithms are often referred to as "dense-mode" algorithms; they assume that the receiver population densely populates the domain of operation, and therefore the accompanying overhead (in terms of state, bandwidth usage, and/or processing costs) is justified. Whilst this might be the case in a local environment, wide-area group membership tends to be sparsely

distributed throughout the Internet. There may be "pockets" of denseness, but if one views the global picture, wide-area groups tend to be sparsely distributed.

Source-based multicast trees are either built by a distance-vector style algorithm, which may be implemented separately from the unicast routing algorithm (as is the case with DVMRP), or the multicast tree may be built using the information present in the underlying unicast routing table (as is the case with PIM-DM [7]). The other algorithm used for building source-based trees is the link-state algorithm (a protocol instance being M-OSPF [8]).

3.1. Distance-Vector Multicast Algorithm

The distance-vector multicast algorithm builds a multicast delivery tree using a variant of the Reverse-Path Forwarding technique [9]. The technique basically is as follows: when a multicast router receives a multicast data packet, if the packet arrives on the interface used to reach the source of the packet, the packet is forwarded over all outgoing interfaces, except leaf subnets with no members attached. A "leaf" subnet is one which no router would use to reach the souce of a multicast packet. If the data packet does not arrive over the link that would be used to reach the source, the packet is discarded.

This constitutes a "broadcast & prune" approach to multicast tree construction; when a data packet reaches a leaf router, if that router has no membership registered on any of its directly attached subnetworks, the router sends a prune message one hop back towards the source. The receiving router then checks its leaf subnets for group membership, and checks whether it has received a prune from all of its downstream routers (downstream with respect to the source). If so, the router itself can send a prune upstream over the interface leading to the source.

The sender and receiver of a prune message must cache the <source, group> pair being reported, for a "lifetime" which is at the granularity of minutes. Unless a router’s prune information is refreshed by the receipt of a new prune for <source, group> before its "lifetime" expires, that information is removed, allowing data to flow over the branch again. State that expires in this way is referred to as "soft state".

Interestingly, routers that do not lead to group members are incurred the state overhead incurred by prune messages. For wide-area multicasting, which potentially has to support many thousands of active groups, each of which may be sparsely distributed, this technique clearly does not scale.

3.2. Link-State Multicast Algorithm

Routers implementing a link state algorithm periodically collect reachability information to their directly attached neighbours, then flood this throughout the routing domain in so-called link state update packets. Deering extended the link state algorithm for multicasting by having a router additionally detect group membership changes on its incident links before flooding this information in link state packets.

Each router then, has a complete, up-to-date image of a domain's topology and group membership. On receiving a multicast data packet, each router uses its membership and topology information to calculate a shortest-path tree rooted at the sender subnetwork. Provided the calculating router falls within the computed tree, it forwards the data packet over the interfaces defined by its calculation. Hence, multicast data packets only ever traverse routers leading to members, either directly attached, or further downstream. That is, the delivery tree is a true multicast tree right from the start.

However, the flooding (reliable broadcasting) of group membership information is the predominant factor preventing the link state multicast algorithm being applicable over the wide-area. The other limiting factor is the processing cost of the Dijkstra calculation to compute the shortest-path tree for each active source.

3.3. The Motivation for Shared Trees

The algorithms described in the previous sections clearly motivate the need for a multicast algorithm(s) that is more scalable. CBT was designed primarily to address the topic of scalability; a shared tree architecture offers an improvement in scalability over source tree architectures by a factor of the number of active sources (where source is usually a subnetwork aggregate). Source trees scale O(S * G), since a distinct delivery tree is built per active source. Shared trees eliminate the source (S) scaling factor; all sources use the same shared tree, and hence a shared tree scales O(G). The implication of this is that applications with many active senders, such as distributed interactive simulation applications, and distributed video-gaming (where most receivers are also senders), have a significantly lesser impact on underlying multicast routing if shared trees are used.

In the "back of the envelope" table below we compare the amount of state required by CBT and DVMRP for different group sizes with different numbers of active sources:

Number of groups	10			100			1000		
Group size (# members)	20			40			60		
No. of srcs per group	10%	50%	100%	10%	50%	100%	10%	50%	100%
No. of DVMRP router entries	20	100	200	400	2K	4K	6K	30K	60K
No. of CBT router entries	10			100			1000		

Figure 1: Comparison of DVMRP and CBT Router State

Shared trees also incur significant bandwidth and state savings compared with source trees; firstly, the tree only spans a group's receivers (including links/routers leading to receivers) -- there is no cost to routers/links in other parts of the network. Secondly, routers between a non-member sender and the delivery tree are not incurred any cost pertaining to multicast, and indeed, these routers need not even be multicast-capable -- packets from non-member senders are encapsulated and unicast to a core on the tree.

The figure below illustrates a core based tree.

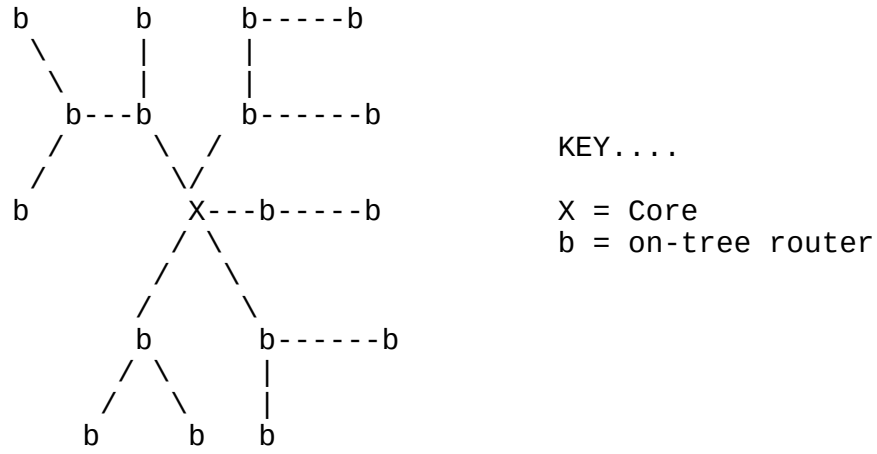

Figure 2: CBT Tree

4. CBT - The New Architecture

4.1. Design Requirements

The CBT shared tree design was geared towards several design objectives:

- scalability - the CBT designers decided not to sacrifice CBT's O(G) scaling characteric to optimize delay using SPTs, as does PIM. This was an important design decision, and one, we think, was taken with foresight; once multicasting becomes ubiquitous, router state maintenance will be a predominant scaling factor. It is possible in some circumstances to improve/optimize the delay of shared trees by other means. For example, a broadcast-type lecture with a single sender (or limited set of infrequently changing senders) could have its core placed in the locality of the sender, allowing the CBT to emulate a shortest-path tree (SPT) whilst still maintaining its O(G) scaling characteristic. More generally, because CBT does not incur source-specific state, it is particularly suited to many sender applications.

- robustness - source-based tree algorithms are clearly robust; a sender simply sends its data, and intervening routers "conspire" to get the data where it needs to, creating state along the way. This is the so-called "data driven" approach -- there is no set-up protocol involved.

It is not as easy to achieve the same degree of robustness in shared tree algorithms; a shared tree's core router maintains connectivity between all group members, and is thus a single point of failure. Protocol mechanisms must be present that ensure a core failure is detected quickly, and the tree reconnected quickly using a replacement core router.

- o simplicity - the CBT protocol is relatively simple compared to most other multicast routing protocols. This simplicity can lead to enhanced performance compared to other protocols.

- o interoperability - from a multicast perspective, the Internet is a collection of heterogeneous multicast regions. The protocol interconnecting these multicast regions is currently DVMRP [6]; any regions not running DVMRP connect to the DVMRP "backbone" as stub regions. CBT has well-defined interoperability mechanisms with DVMRP [15].

4.2. CBT Components & Functions

The CBT protocol is designed to build and maintain a shared multicast distribution tree that spans only those networks and links leading to interested receivers.

To achieve this, a host first expresses its interest in joining a group by multicasting an IGMP host membership report [5] across its attached link. On receiving this report, a local CBT aware router invokes the tree joining process (unless it has already) by generating a JOIN_REQUEST message, which is sent to the next hop on the path towards the group's core router (how the local router discovers which core to join is discussed in section 6). This join message must be explicitly acknowledged (JOIN_ACK) either by the core router itself, or by another router that is on the unicast path between the sending router and the core, which itself has already successfully joined the tree.

The join message sets up transient join state in the routers it traverses, and this state consists of <group, incoming interface, outgoing interface>. "Incoming interface" and "outgoing interface" may be "previous hop" and "next hop", respectively, if the corresponding links do not support multicast transmission. "Previous hop" is taken from the incoming control packet's IP source address, and "next hop" is gleaned from the routing table - the next hop to the specified core address. This transient state eventually times out unless it is "confirmed" with a join acknowledgement (JOIN_ACK) from upstream. The JOIN_ACK traverses the reverse path of the corresponding join message, which is possible due to the presence of the transient join state. Once the acknowledgement reaches the

router that originated the join message, the new receiver can receive traffic sent to the group.

Loops cannot be created in a CBT tree because a) there is only one active core per group, and b) tree building/maintenance scenarios which may lead to the creation of tree loops are avoided. For example, if a router's upstream neighbour becomes unreachable, the router immediately "flushes" all of its downstream branches, allowing them to individually rejoin if necessary. Transient unicast loops do not pose a threat because a new join message that loops back on itself will never get acknowledged, and thus eventually times out.

The state created in routers by the sending or receiving of a JOIN_ACK is bi-directional - data can flow either way along a tree "branch", and the state is group specific - it consists of the group address and a list of local interfaces over which join messages for the group have previously been acknowledged. There is no concept of "incoming" or "outgoing" interfaces, though it is necessary to be able to distinguish the upstream interface from any downstream interfaces. In CBT, these interfaces are known as the "parent" and "child" interfaces, respectively.

With regards to the information contained in the multicast forwarding cache, on link types not supporting native multicast transmission an on-tree router must store the address of a parent and any children. On links supporting multicast however, parent and any child information is represented with local interface addresses (or similar identifying information, such as an interface "index") over which the parent or child is reachable.

When a multicast data packet arrives at a router, the router uses the group address as an index into the multicast forwarding cache. A copy of the incoming multicast data packet is forwarded over each interface (or to each address) listed in the entry except the incoming interface.

Each router that comprises a CBT multicast tree, except the core router, is responsible for maintaining its upstream link, provided it has interested downstream receivers, i.e. the child interface list is not NULL. A child interface is one over which a member host is directly attached, or one over which a downstream on-tree router is attached. This "tree maintenance" is achieved by each downstream router periodically sending a "keepalive" message (ECHO_REQUEST) to its upstream neighbour, i.e. its parent router on the tree. One keepalive message is sent to represent entries with the same parent, thereby improving scalability on links which are shared by many groups. On multicast capable links, a keepalive is multicast to the "all-cbt-routers" group (IANA assigned as 224.0.0.15); this has a

suppressing effect on any other router for which the link is its parent link. If a parent link does not support multicast transmission, keepalives are unicast.

The receipt of a keepalive message over a valid child interface immediately prompts a response (ECHO_REPLY), which is either unicast or multicast, as appropriate.

The ECHO_REQUEST does not contain any group information; the ECHO_REPLY does, but only periodically. To maintain consistent information between parent and child, the parent periodically reports, in a ECHO_REPLY, all groups for which it has state, over each of its child interfaces for those groups. This group-carrying echo reply is not prompted explicitly by the receipt of an echo request message. A child is notified of the time to expect the next echo reply message containing group information in an echo reply prompted by a child's echo request. The frequency of parent group reporting is at the granularity of minutes.

It cannot be assumed all of the routers on a multi-access link have a uniform view of unicast routing; this is particularly the case when a multi-access link spans two or more unicast routing domains. This could lead to multiple upstream tree branches being formed (an error condition) unless steps are taken to ensure all routers on the link agree which is the upstream router for a particular group. CBT routers attached to a multi-access link participate in an explicit election mechanism that elects a single router, the designated router (DR), as the link's upstream router for all groups. Since the DR might not be the link's best next-hop for a particular core router, this may result in join messages being re-directed back across a multi-access link. If this happens, the re-directed join message is unicast across the link by the DR to the best next-hop, thereby preventing a looping scenario. This re-direction only ever applies to join messages. Whilst this is suboptimal for join messages, which are generated infrequently, multicast data never traverses a link more than once (either natively, or encapsulated).

In all but the exception case described above, all CBT control messages are multicast over multicast supporting links to the "all-cbt-routers" group, with IP TTL 1. When a CBT control message is sent over a non-multicast supporting link, it is explicitly addressed to the appropriate next hop.

4.2.1. CBT Control Message Retransmission Strategy

Certain CBT control messages illicit a response of some sort. Lack of response may be due to an upstream router crashing, or the loss of the original message, or its response. To detect these events, CBT

retransmits those control messages for which it expects a response, if that response is not forthcoming within the retransmission-interval, which varies depending on the type of message involved. There is an upper bound (typically 3) on the number of retransmissions of the original message before an exception condition is raised.

For example, the exception procedure for lack of response to an ECHO_REQUEST is to send a QUIT_NOTIFICATION upstream and a FLUSH_TREE message downstream for the group. If this is router has group members attached, it restarts the joining process to the group's core.

4.2.2. Non-Member Sending

If a non-member sender's local router is already on-tree for the group being sent to, the subnet's upstream router simply forwards the data packet over all outgoing interfaces corresponding to that group's forwarding cache entry. This is in contrast to PIM-SM [18] which must encapsulate data from a non-member sender, irrespective of whether the local router has joined the tree. This is due to PIM's uni-directional state.

If the sender's subnet is not attached to the group tree, the local DR must encapsulate the data packet and unicast it to the group's core router, where it is decapsulated and disseminated over all tree interfaces, as specified by the core's forwarding cache entry for the group. The data packet encapsulation method is IP-in-IP [14].

Routers in between a non-member sender and the group's core need not know anything about the multicast group, and indeed may even be multicast-unaware. This makes CBT particulary attractive for applications with non-member senders.

5. Interoperability with Other Multicast Routing Protocols

See "interoperability" in section 4.1.

The interoperability mechanisms for interfacing CBT with DVMRP are defined in [15].

6. Core Router Discovery

Core router discovery is by far the most controversial and difficult aspect of shared tree multicast architectures, particularly in the context of inter-domain multicast routing (IDMR). There have been many proposals over the past three years or so, including advertising core addresses in a multicast session directory like "sdr" [11], manual placement, and the HPIM [12] approach of strictly dividing up

the multicast address space into many "hierarchical scopes" and using explicit advertising of core routers between scope levels.

There are currently two options for CBTv2 [1] core discovery; the "bootstrap" mechamism, and manual placement. The bootstrap mechanisms (as currently specified with the PIM sparse mode protocol [18]) is applicable only to intra-domain core discovery, and allows for a "plug & play" type operation with minimal configuration. The disadvantage of the bootstrap mechanism is that it is much more difficult to affect the shape, and thus optimality, of the resulting distribution tree. Also, it must be implemented by all CBT routers within a domain.

Manual configuration of leaf routers with <core, group> mappings is the other option (note: leaf routers only); this imposes a degree of administrative burden - the mapping for a particular group must be coordinated across all leaf routers to ensure consistency. Hence, this method does not scale particularly well. However, it is likely that "better" trees will result from this method, and it is also the only available option for inter-domain core discovery currently available.

6.1. Bootstrap Mechanism Overview

It is unlikely at this stage that the bootstrap mechanism will be appended to a well-known network layer protocol, such as IGMP [5] or ICMP [13], though this would facilitate its ubiquitous (intra-domain) deployment. Therefore, each multicast routing protocol requiring the bootstrap mechanism must implement it as part of the multicast routing protocol itself.

A summary of the operation of the bootstrap mechanism follows. It is assumed that all routers within the domain implement the "bootstrap" protocol, or at least forward bootstrap protocol messages.

A subset of the domain’s routers are configured to be CBT candidate core routers. Each candidate core router periodically (default every 60 secs) advertises itself to the domain’s Bootstrap Router (BSR), using "Core Advertisement" messages. The BSR is itself elected dynamically from all (or participating) routers in the domain. The domain’s elected BSR collects "Core Advertisement" messages from candidate core routers and periodically advertises a candidate core set (CC-set) to each other router in the domain, using traditional hopby-hop unicast forwarding. The BSR uses "Bootstrap Messages" to advertise the CC-set. Together, "Core Advertisements" and "Bootstrap Messages" comprise the "bootstrap" protocol.

When a router receives an IGMP host membership report from one of its directly attached hosts, the local router uses a hash function on the reported group address, the result of which is used as an index into the CC-set. This is how local routers discover which core to use for a particular group.

Note the hash function is specifically tailored such that a small number of consecutive groups always hash to the same core. Furthermore, bootstrap messages can carry a "group mask", potentially limiting a CC-set to a particular range of groups. This can help reduce traffic concentration at the core.

If a BSR detects a particular core as being unreachable (it has not announced its availability within some period), it deletes the relevant core from the CC-set sent in its next bootstrap message. This is how a local router discovers a group's core is unreachable; the router must re-hash for each affected group and join the new core after removing the old state. The removal of the "old" state follows the sending of a QUIT_NOTIFICATION upstream, and a FLUSH_TREE message downstream.

7. Summary

This document presents an architecture for intra- and inter-domain multicast routing. We motivated this architecture by describing how an inter-domain multicast routing algorithm must scale to large numbers of groups present in the internetwork, and discussed why most other existing algorithms are less suited to inter-domain multicast routing. We followed by describing the features and components of the architecture, illustrating its simplicity and scalability.

8. Security Considerations

Security considerations are not addressed in this memo.

Whilst multicast security is a topic of ongoing research, multicast applications (users) nevertheless have the ability to take advantage of security services such as encryption or/and authentication provided such services are supported by the applications.

RFCs 1949 and 2093/2094 discuss different ways of distributing multicast key material, which can result in the provision of network layer access control to a multicast distribution tree.

[19] offers a synopsis of multicast security threats and proposes some possible counter measures.

Beyond these, little published work exists on the topic of multicast security.

Acknowledgements

Special thanks goes to Paul Francis, NTT Japan, for the original brainstorming sessions that brought about this work.

Clay Shields' work on OCBT [17] identified various failure scenarios with a multi-core architecture, resulting in the specification of a single core architecture.

Others that have contributed to the progress of CBT include Ken Carlberg, Eric Crawley, Jon Crowcroft, Mark Handley, Ahmed Helmy, Nitin Jain, Alan O'Neill, Steven Ostrowsksi, Radia Perlman, Scott Reeve, Benny Rodrig, Martin Tatham, Dave Thaler, Sue Thompson, Paul White, and other participants of the IETF IDMR working group.

Thanks also to 3Com Corporation and British Telecom Plc for funding this work.

References

[1] Ballardie, A., "Core Based Trees (CBT version 2) Multicast Routing: Protocol Specification", RFC 2189, September 1997.

[2] Multicast Routing in a Datagram Internetwork; S. Deering, PhD Thesis, 1991; ftp://gregorio.stanford.edu/vmtp/sd-thesis.ps.

[3] Mechanisms for Broadcast and Selective Broadcast; D. Wall; PhD thesis, Stanford University, June 1980. Technical Report #90.

[4] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

[5] Internet Group Management Protocol, version 2 (IGMPv2); W. Fenner; Work In Progress.

[6] Distance Vector Multicast Routing Protocol (DVMRP); T. Pusateri; Work In Progress.

[7] Protocol Independent Multicast (PIM) Dense Mode Specification; D. Estrin et al; ftp://netweb.usc.edu/pim, Work In Progress.

[8] Moy, J., "Multicast Extensions to OSPF", RFC 1584, March 1994.

[9] Reverse path forwarding of broadcast packets; Y.K. Dalal and R.M. Metcalfe; Communications of the ACM, 21(12):1040--1048, 1978.

[10] Some Issues for an Inter-Domain Multicast Routing Protocol; D. Meyer; Work In Progress.

[11] SDP: Session Description Protocol; M. Handley and V. Jacobson; Work In Progress.

[12] Hierarchical Protocol Independent Multicast; M. Handley, J. Crowcroft, I. Wakeman. Available from: http://www.cs.ucl.ac.uk/staff/M.Handley/hpim.ps and ftp://cs.ucl.ac.uk/darpa/IDMR/hpim.ps Work done 1995.

[13] Postel, J., "Internet Control Message Protocol (ICMP)", STD 5, RFC 792, September 1981.

[14] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[15] CBT - Dense Mode Multicast Interoperability; A. Ballardie; Work In Progress.

[16] Performance and Resource Cost Comparisons of Multicast Routing Algorithms for Distributed Interactive Simulation Applications; T. Billhartz, J. Bibb Cain, E. Farrey-Goudreau, and D. Feig. Available from: http://www.epm.ornl.gov/~sgb/pubs.html; July 1995.

[17] The Ordered Core Based Tree Protocol; C. Shields and J.J. Garcia- Luna-Aceves; In Proceedings of IEEE Infocom'97, Kobe, Japan, April 1997; http://www.cse.ucsc.edu/research/ccrg/publications/info-comm97ocbt.ps.gz

[18] Estrin, D., et. al., "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2117, June 1997.

[19] Multicast-Specific Security Threats and Counter-Measures; A. Ballardie and J. Crowcroft; In Proceedings "Symposium on Network and Distributed System Security", February 1995, pp.2-16.

Author Information

Tony Ballardie,
Research Consultant

EMail: ABallardie@acm.org