applied sciences

MDPI

Article

A Multicast Routing Scheme for the Internet: Simulation and Experimentation in Large-Scale Networks

Davide Careglio [1,*], Fernando Agraz [2] and Dimitri Papadimitriou [3]

[1] Department of Computer Architecture, Universitat Politécnica de Catalunya, 08034 Barcelona, Spain
[2] Department of Signal Theory and Communications, Universitat Politécnica de Catalunya, 08034 Barcelona, Spain; fernando.agraz@upc.edu
[3] Department of Mathematics and Computer Science, University of Antwerp, 2000 Antwerp, Belgium; dimitri.papadimitriou@uantwerpen.be
* Correspondence: davide.careglio@upc.edu; Tel.: +34-93-401-6975

Abstract: With the globalisation of the multimedia entertainment industry and the popularity of streaming and content services, multicast routing is (re-)gaining interest as a bandwidth saving technique. In the 1990's, multicast routing received a great deal of attention from the research community; nevertheless, its main problems still remain mostly unaddressed and do not reach the acceptance level required for its wide deployment. Among other reasons, the scaling limitation and the relative complexity of the standard multicast protocol architecture can be attributed to the conventional approach of overlaying the multicast routing on top of the unicast routing topology. In this paper, we present the Greedy Compact Multicast Routing (GCMR) scheme. GMCR is characterised by its scalable architecture and independence from any addressing and unicast routing schemes; more specifically, the local knowledge of the cost to direct neighbour nodes is enough for the GCMR scheme to properly operate. The branches of the multicast tree are constructed directly by the joining destination nodes which acquire the routing information needed to reach the multicast source by means of an incremental two-stage search process. In this paper we present the details of GCMR and evaluate its performance in terms of multicast tree size (i.e., the stretch), the memory space consumption, the communication cost, and the transmission cost. The comparative performance analysis is performed against one reference algorithm and two well-known protocol standards. Both simulation and emulation results show that GCMR achieves the expected performance objectives and provide the guidelines for further improvements.

Keywords: multicast routing; scalable internet; compact routing; adaptability; inter-domain routing

Citation: Careglio, D.; Agraz, F.; Papadimitriou, D. A Multicast Routing Scheme for the Internet: Simulation and Experimentation in Large-Scale Networks. *Appl. Sci.* **2021**, *11*, 8645. https://doi.org/10.3390/app11188645

Academic Editor: Pedro Amaral

Received: 12 August 2021
Accepted: 13 September 2021
Published: 17 September 2021

1. Introduction

Multicast routing is a distributed algorithm for efficient distribution of one-to-many traffic. A multicast source can send packets in one stream to a group of destination nodes (i.e., a multicast group). To enable such a traffic distribution, the multicast routing protocol configures the involved routers to build a (logical) delivery tree, usually known as Multicast Distribution Tree (MDT), between the source and the multicast group. The network is thus responsible for duplicating the multicast packet at given nodes in the MDT; such nodes are commonly referred as to branching node (i.e., the node at which the branches fork). This means that only one copy of the packet is sent by the multicast source and each packet travels only once over any particular link in the network, making multicast trees extremely efficient for distributing the same information to many destinations. Originally defined 30 years ago [1], the potential benefits of multicast routing as a bandwidth saving technique have been verified by studies several times since then [2,3]. In the present context of a cloud-centric environment where multimedia streaming, content and machine-to-machine traffic are increasing their volume, multicast is (re-)gaining world-wide interest [4,5] together with new potential multicast business models [6–9].

Although widely investigated, the problems of multicast faced in the 1990's remain mostly unaddressed. The complete analysis of the deployment issues for the IP multicast routing architecture available in [10], although dated 2000, is still valid today [11]. Indeed, the adoption of multicast routing has failed due to several issues:

- Multicast depends on an overlay routing executed over the unicast routing topology, which suffers in terms of scalability, as it maintains a high number of routing entries and forwarding states at each node;
- It adds a level of indirection as routers forward the multicast traffic to the multicast group, and hosts have to subscribe to that multicast group using a subscription protocol like Internet Group Management Protocol (IGMP) or Multicast Listener Discovery (MLD) in IPv6;
- It uses special address space structure (class D for IPv4 and prefix ff00::/8 for IPv6) which requires that firewalls and Network Address Translation (NAT) routers have to be upgraded and configured to support multicast addresses;
- Economic reasons, as there is a need to determine how upstream providers reimburse downstream providers for installing state and replication of data;
- Management and security concerns about setting up shared trees between domains.

In this paper, we present the Greedy Compact Multicast Routing (GCMR) scheme as a possible solution to some of the above issues. The concept of GCMR was initially proposed in [12]; here, we provide a comprehensive description and performance analysis of GCMR. In particular, the main goal of GCMR is to compact (i.e., to minimise) the routing information stored at each router by taking greedy (i.e., local) decisions. Instead of relying on unicast routing information to derive the multicast entries—as in the case of conventional multicast routing protocols—in GCMR, the information needed to join a given multicast source is acquired when needed by means of a two-stage search process. As a consequence, GCMR:

- Provides a scalable routing architecture because each router stores in its routing table only the (direct) neighbour-related entries, rather than tree structures or network graph information;
- Can work with any addressing space structure like IPv4/IPv6, with identifiers like geometric coordinates or with data/information centric content, as it does not rely on the unicast routing information;
- Can support a variety of metrics to decide the optimal branching path from the multicast source to leaf nodes;
- Has a scope that is not limited to routers and thus can be initiated by any host in such a way that a real end-to-end multicast distribution tree is provided and the additional host-router subscription protocol is not needed anymore.

In summary, GCMR is a name-independent, distributed, dynamic, leaf-initiated, and end-to-end multicast routing algorithm. It is worth mentioning that, although GCMR has been designed to improve scalability of inter-domain multicast routing, it can perform in other environments where only limited topology/routing information is available and routing scalability is a main issue.

In this paper we present the details of GCMR and evaluate its performance. In particular, we are interested in finding the optimal trade-offs between the stretch of the routing scheme, the memory space required to locally store the routing information, the communication cost needed to setup and properly operate the MDT, and the transmission cost to reach all leaf nodes of the multicast group. The comparative performance analysis against state of the art multicast schemes is carried out using both simulation and emulation platform. The simulation platform allows us to evaluate the performance considering large-scale internet topology maps with up to 46k Autonomous Systems (ASes) and up to 4k leaf nodes in the MDT. On the contrary, the GCMR prototype has been developed and tested to experimentally validate its behaviour and performance in a large-scale realistic network environment with more than 200 ASes.

The rest of the paper is organised as follows. Section 2 presents the related work on multicast routing scheme and the contribution of this paper. Section 3 describes the GCMR scheme. In Section 4 we present the prototype of a GCMR routing engine. Section 5 details the performance and comparative analysis using the simulation platform, while in Section 6 we provide and discuss the results of the experimental tests. Finally, Section 7 concludes this paper.

2. Related Work and Our Contribution

2.1. Brief Overview of Multicast Routing in the Internet

The first multicast routing schemes standardised were dependent of the unicast protocols, either distance vector-based such as the Distance-Vector Multicast Routing Protocol (DVMRP) [13] or link state-based such as Multicast Open Shortest Path First (MOSPF) [14]. These schemes were dependent on the underlying unicast routing and presented high scalability limits and complexity. Thus, they have been replaced by the Core-Based Trees (CBT) [15] and the Protocol Independent Multicast (PIM) [16] which perform independently from the unicast scheme.

It worth mentioning that two variants of multicast schemes exist: Dense Mode (DM) and Sparse Mode (SM). Dense mode assumes that a large amount of receivers wants to join the multicast group, and so floods the network with a sort of multicast session setup packets, and prunes back those network subnets that fail to respond or respond with no interest in joining the multicast group. Examples of them are DVMRP and PIM-DM [17]. Sparse mode assumes that very few of the receivers connected to the internet want to join the multicast group, and so waits until a receiver specifically joins the group. In turn, sparse mode defines two different service models, namely Any Source Multicast (ASM) and Single Source Multicast (SSM). ASM explicitly builds unidirectional shared trees rooted at a given router called the rendezvous point (RP) per group which, in turn, establishes the shortest path routes to the specific multicast sources. Receivers subscribe then to the multicast group G but not to any specific source; thus, the multicast channel is defined as $(*, G)$. PIM-SM [16] is an example of an ASM scheme. On the contrary, SSM builds trees that are rooted in the specific multicast source, and receivers subscribe to a specific source S and multicast group G, thus defining the multicast channel as (S, G). PIM-SSM [18] is an example of an SSM scheme. In general, SSM provides better end-to-end performance and network resource utilisation as well as implicit security but the routing state they require increases linearly with the number of sources [19].

In the context of research, prior work on multicast schemes mainly focuses on reducing their complexity and storage requirement. Authors in [20] propose the Yet Another Multicast (YAM) scheme which relies on a limited scope broadcast in the neighbourhood of the joining node to find the closest portion of the MDT to join. Such a search is based on an incremental discovery scheme which expands the limit of the neighbourhood up to find the MDT. During such a search, YAM identifies multiple paths and selects the one that can provide the required QoS. In [21], the authors improve the scalability limits of the YAM scheme and propose the QoS sensitive multicast internet protocol (QoSMIC), which, in case the local search fails, involves the use of the so-called Manager Node. Such nodes are the administrators of specific multicast groups and store the MDT information (i.e., very similar to the RP nodes in the ASM model). Such a scheme implies the use of specific nodes which introduce possible scalability, resilience, and memory concerns. A two-step scheme is proposed in [22]: the first step is carried out to obtain an RP and a candidate MDT; the second step is performed to generate a modified MDT rooted at the RP and the optimal MDT is obtained by comparing the original and the modified MDT. Like QoSMIC, this scheme also uses RP nodes, which again can cause scalability, resilience, and memory issues.

With the introduction of Software Defined Networking (SDN) [23], several recent works propose this new architecture to deploy multicast in the internet. Two surveys of existing and proposed multicast routing algorithms in SDN are available in [24,25]. All

these works claim that SDN enables the use of more complex and efficient algorithms for multicasting as the centralised SDN controller has a global view of the whole topology. An example is the hybrid SDN model proposed in [26] where multicast traffic is managed thanks to a set of OpenFlow flow rules instantiated in network nodes under control of the centralised SDN controller. Nonetheless, any centralised approach suffers of scalability limitations and, in multicast, we have to deal with two dimensions. On the one hand, the number of multicast groups to be managed can be very high: more than 200 millions groups can be created in IPv4, even more if we consider IPv6. On the other hand, hosts can constantly join and leave any multicast group at any time, creating many updates to be sent and managed by the SDN controller to keep all MDTs optimally connected.

Another line of research proposes the use of routing schemes based on the *compact* concept. Compact routing aims at finding the best trade-off between the memory space required to store the routing table entries at each node; the size of the packets headers and the length of the routing paths it produces. Such routing schemes have been extensively studied following the seminal paper in [27]. In the context of multicast routing, prior work is, as far as our knowledge goes, mainly concentrated around the routing schemes proposed in [28]. This scheme enables the construction of multicast routing paths from any source to any set of leaf nodes. Although the author develops an extensive theoretic performance justification and analysis, they do not report any numerical analysis. The upper bounds indicated do not necessarily translate actual performance that can be obtained on graphs underlying large-scale networks such as the internet, which, in addition, shows properties associated to scale-free graphs (small diameter, high-clustering, and power-law degree distribution) [29,30].

Recent work [4] proposes the Bit Index Explicit Replication (BIER) multicast architecture which does not require any explicit tree-building protocol, nor does it require intermediate nodes to maintain any per-flow state. BIER consists of encapsulating packets in BIER headers that contain a string of bits (called bit-string) in which each bit represents exactly one node in the domain; to forward the packet to a given set of destination nodes, the bits corresponding to those nodes are set in the BIER header. This solution completely relies therefore on the unicast routing to forward multicast packets to the destination nodes. Since its first appearance, BIER attracted much attention for its simplicity, and works have been presented to enhance its basic functionality, for instance, with reliable multicast transmission [31], packet caching for fast retransmission [32], and fast reroute in case of failure [33].

Nevertheless, in BIER, all the required information to route the packets to the leaf nodes is encoded in the BIER header, which implies at least three issues. The first one is that the size of this additional header scales linearly to the number of nodes in the network. Currently, the limitation of the bit-string is set to 256 bits (equivalent to the two IPv6 addresses), meaning no more than 256 routers can be identified as a destination in the BIER header. In an internet with 100k ASes, this is to be a significant constraint. There is the possibility to overcome this limitation by creating (up to 256) subsets, meaning that packets belong to different subsets (also known as BIER sub-domains) need to be sent separately, partially losing therefore the benefits of multicast routing as a bandwidth saving technique. Another problem is that each bit in the bit-string requires the execution of a routing process; the results are then ANDed to merge the common next hops and create the new BIER headers. Finally, the resulting MDT constructed in this way is a combination of shortest paths, i.e., a Shortest Path Tree (SPT) per BIER sub-domain. As we will see in the results, SPT does not provide the optimally performing tree.

During the last decade, only PIM-SM and PIM-SSM, have been really deployed and only for the intra-domain multicast case, mostly for the IPTV service within some Internet Service Provider (ISP) network [5,34]. However, multicast has failed to be adopted and does not reach the acceptance level required for its wide deployment, especially for the inter-domain case.

2.2. Our Contributions

In this paper, we present the details of our proposed GCMR scheme which is a SSM sparse mode multicast scheme. The objective is two-fold. On the one hand, we provide a performance analysis of GCMR and compare it with the PIM-SSM, BIER, and Steiner Tree (ST) schemes (see Section 5 for more details). For this performance study, we use an ad-hoc c-based simulator and four large CAIDA maps of the internet topology.

On the other hand, we present the development and implementation of GCMR in a prototype developed using the libraries of the Quagga open source routing suite (http://www.nongnu.org/quagga/ accessed on 15 September 2021). We then use this prototype to experimentally validate GCMR and compare its performance against PIM-SSM in the context of inter-domain multicast routing. These experimental tests are performed in a network comprising 207 Autonomous Systems (ASes) and executed using the iLab.t Virtual Wall (https://doc.ilabt.imec.be/ilabt/virtualwall/index.html accessed on 15 September 2021) platform, which is a large-scale experimental testbed available at the IMEC research centre in Ghent, Belgium.

3. The Greedy Compact Multicast Routing (GCMR) Scheme

3.1. Preliminaries

Consider a network topology modelled by an undirected weighted graph $G = (V, E, \omega)$, where the set $V, |V| = n$ represents the finite set of vertices or nodes (all multicast capable), the set $E, |E| = e$ represents the finite set of edges or links, and ω is a non-negative function $\omega : E \rightarrow \mathbb{R}+$ which associates a non-negative weight or cost $\omega(u, v)$ to each edge $(u, v) \in E$. For $u, v \in V$, the path $p(u, v)$ from vertex u to v is defined as the vertex sequence $[x_0(= u), x_1, \ldots, x_{i-1}, x_i, \ldots, x_p(= v)]$ such that the vertices x_i are all distinct and vertex x_{i-1} is adjacent to $x_i, \forall (x_{i-1}, x_i)_{i=1,\ldots,p} \in E$. The following definitions apply: (i) the cost $c(u, v)$ of a path $p(u, v)$ as the sum of the weights of the edges on the path from u to v, (ii) the length $l(u, v)$ of a path $p(u, v)$ as the number of edges the path traverses from u to v, (iii) the distance $d(u, v)$ between two vertices u, v of the graph G as the cost of the minimum cost path $p(u, v)$ from u to v, and (iv) the diameter $\delta(G)$ of the graph G is defined as the largest distance between any two vertices $u, v \in V$, i.e., $\delta(G) = \max_{u,v \in V}\{d(u, v)\}$.

Let $S, S \subset V$, be the finite set of multicast source nodes and $s \in S$ denote a multicast source node. Let $D, D \subseteq V\ S$, denote the finite set of all possible leaf nodes that can join a multicast source s and let $d \in D$ denote a leaf node. The MDT $T_{s,M} = (V_T, E_T)$ is defined as an acyclic connected sub-graph H of G, i.e., the tree rooted at the multicast source node s with leaf node set $M, M \subseteq D$. The tree $T_{s,M}$ is also referred to as the multicast routing path and its information is usually stored in the TIB table. The set M corresponds to the current set of nodes at a given construction step of the MDT. The size of the tree $T_{s,M}$ is defined as the size of the connected graph of G, i.e., $|T_{s,M}| = h \leq n$.

With the objectives of this work in mind, we define the following performance metrics:

- The stretch of a routing scheme is defined as the maximum ratio between the cost of the path $p(u, v)$ traversed by a packet and the cost of the shortest path considering all node pairs $u, v \in V$. For the multicast case, the definition of the stretch changes to the total cost of the edges of the MDT to reach a given group of leaf nodes is divided by the cost of the minimum tree, which is the one produced by solving the corresponding Steiner Tree (ST) problem for the same leaf group [28].
- The memory space (in bits) takes into account the information required to be stored at each node to properly treat the multicast packets. The storage required by the algorithm is directly related to routing system scalability because the less memory a router needs to store its entries, the more scalable the routing system would be. The following routing information bases are defined [16]:
  - The Tree Information Base (TIB) essentially stores the state of all multicast distribution trees necessary to forward multicast packets at a router;

  - The Unicast Routing Information Base (URIB) is the unicast control message routing and it is used to determine the next shortest hop and/or the path (e.g., like BGP) towards a node of the network;
  - The Multicast Routing Information Base (MRIB) is the multicast control message routing and it is used to determine the next-hop neighbour to which any Join/Leave message is sent (towards the tree root or source).
- The communication cost (in number of messages) is defined as the number of messages exchanged to build the MDT (it is also usually known as the message cost). This metric is directly related to the overhead of the protocol, i.e., the higher the communication cost the longer the time needed to construct and maintain the tree.
- The transmission cost (in Mbytes) is defined as the total cost of transmitting packets through the edges of the MDT to reach a given set of leaf nodes. Concretely, we are counting the number of additional bits required to transmit a single packet to all leaf nodes taking into account a 1500-byte IP packet to be transmitted along a ST as a reference.
- The recovery time is defined as the maximum time needed to receive back a multicast transmission at the leaf nodes once a failure occurs in a link or node of the MDT.

3.2. Basic Functionality

The GCMR scheme seeks to reduce the size of each node's routing table (and hence memory space) at the expense of (i) multicast routing paths with relatively modest deviations compared to the optimal stretch generated by the reference ST algorithm, and (ii) greater communication costs compared to PIM and BIER.

GCMR achieves the minimisation of the memory space because only direct neighbour-related information is needed to be stored locally at each node and thus it scales linearly with the node degree. This information is used during the MDT construction to provide the next hop along the least cost branching path. Thus, GCMR does not need: (i) non-local topology/path information like PIM; (ii) the construction of complex structures such as sparse covers like in compact routing methods [28]; (iii) tree structures like in ST; or (iv) additional labels or headers containing information for computing the routing paths like in BIER. The difficulty is keeping the best possible compromise between the stretch and the memory space while limiting the communication cost, i.e., the amount of messages exchanged during the construction of the MDT.

In GCMR, any node $u \in V, u \notin T_{s,M}$ can become a leaf node of a given multicast source s by creating a least cost branch path to its MDT $T_{s,M}$, i.e., GCMR is leaf-initiated. The information needed to join a MDT is acquired when needed by means of a two-stage search process. Figure 1 illustrates this process schematically. Node u starts with a local search sending a type-R message within its vicinity ball $B(u)$ for any node $v \neq u, v \in T_{s,M}$ (see Figure 1a). The vicinity ball covers the joining node's neighbourhood. The reasoning is that in large networks with a diameter $\delta(G)$ that is logarithmically proportional to the number of nodes n, i.e., $\delta(G) \sim \log(n)$, the likelihood of finding the node v within a few hops distance from the joining node u is high. Furthermore, this probability rises as the MDT grows in size. If the local search over the vicinity ball $B(u)$ fails, the search is resumed across the remaining undiscovered topology without requiring global knowledge of the MDT (see Figure 1b). In order to avoid costly global search in terms of the amount of messages, a variable path budget π is utilised to limit the distance they can travel. As to not repeat the unsuccessful search inside the $B(u)$, the global search starts at the edge nodes of the $B(u)$; such edge nodes are identified during the local search and requested by the joining node u to start the global flooding outside $B(u)$ only.

In any of these two phases, if a node belonging to the MDT is found, it returns an answer (type-A message) to the joining node. If an intermediate node receives more than one answer during a given time frame, it only forwards one answer to the joining node. The decision about which answer depends on the costs/metrics defined and transported in the answer. In a context without QoS, the cost is unique and can represent the number

of hops; the least cost branching path is therefore the minimum hop distance between the joining node and the node belonging to the MDT. In a context with QoS, we can define metrics that accumulate additively along the path (e.g., number of hops, delay, path length, etc.) or metrics that exclude given paths (e.g., not enough bandwidth, too low reliability, etc.), and thus the decision at each intermediate node can be based on the requested QoS level. For sake of simplicity, in this paper we evaluate the GCMR performance in a scenario without QoS.

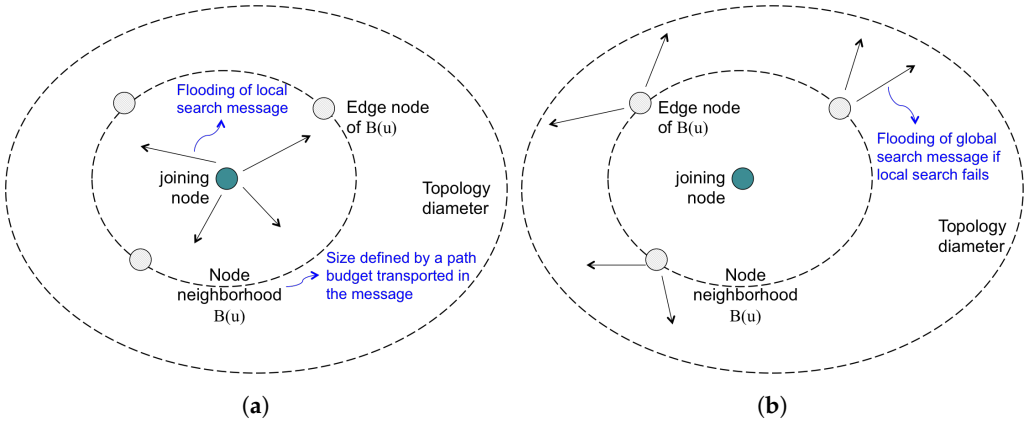


Figure 1. Two-stage search process: (**a**) Local search: joining node searching the MDT in its neighbourhood; (**b**) Global search: edge nodes searching outside the joining node's neighbourhood.

As a consequence of this process, the routing table of each node $v \in T_{s,M}$ includes the following entries (i) one entry in the MRIB table that indicates the upstream neighbour node to which the join message is sent for each multicast source s and (ii) one entry in the TIB table to enable routing of incoming multicast traffic (originated by that source s) from its incoming port to a set of outgoing ports.

Moreover, GCMR is adaptive, meaning that the routing decisions may be modified once there is a change in the information that has led to that decision. This change can be (i) a (node/link) failure in some part of the MDT and GCMR needs to restore the MDT or (ii) a given sequence of join/leave events that lead to a sub-optimal multicast routing path and GCMR needs to optimise the MDT. GCMR treats both cases in the same way and causes the execution of the adaptability mechanism which is based on a modified two-stage search process explained above. Nonetheless, in this paper we only focus on the adaptation of the MDT in case of failure events. In this context, the adaptation is executed by the upstream node of the (node/link) failure in the MDT.

The next section describes three examples to illustrate the behaviour of GCMR when (i) a node joins an MDT with a local search, (ii) a node joins an MDT with the global search (i.e., the local search fails), and (iii) a link failure in one of the edges of the MDT disconnects some leaf nodes.

3.3. Examples

In Figure 2, we report an illustrative example of the local search process. In Figure 2a, we can observe a joining node J with its vicinity ball $B(J)$. In this example, this vicinity ball contains all nodes with at most two hops distant from J (remember that we are not considering QoS in this work, therefore the only cost used is the number of hops). In Figure 2b, node J sends a searching request (type-R message) which floods up to find node(s) belonging to a given MDT. If at least one node is found (Figure 2c), it sends an answer back (type-A message) to the joining node; the accumulated path cost is transported in this answer in such a way that, in case of multiple answers, only the least cost branch path reaches the joining node. It is worth mentioning that, during the flooding of type-R messages, nodes store temporarily the port from which the message has arrived: in this way, they can use this information to forward the type-A messages back to the joining node

without creating entries in the routing tables. In the example, we can see that the answer from node 7 transports an infinite cost, i.e., the MDT has not been found along this path inside $B(J)$. The other two answers transport cost 1 and cost 2 from node 2 and node 5, respectively. Therefore, the joining node sends the join message towards node 2 which is the least cost branch path to the MDT as we can observe in Figure 2d. Note that, in the case that two or more paths have the same cost, the preference goes to the path that is received first (i.e., the oldest one).

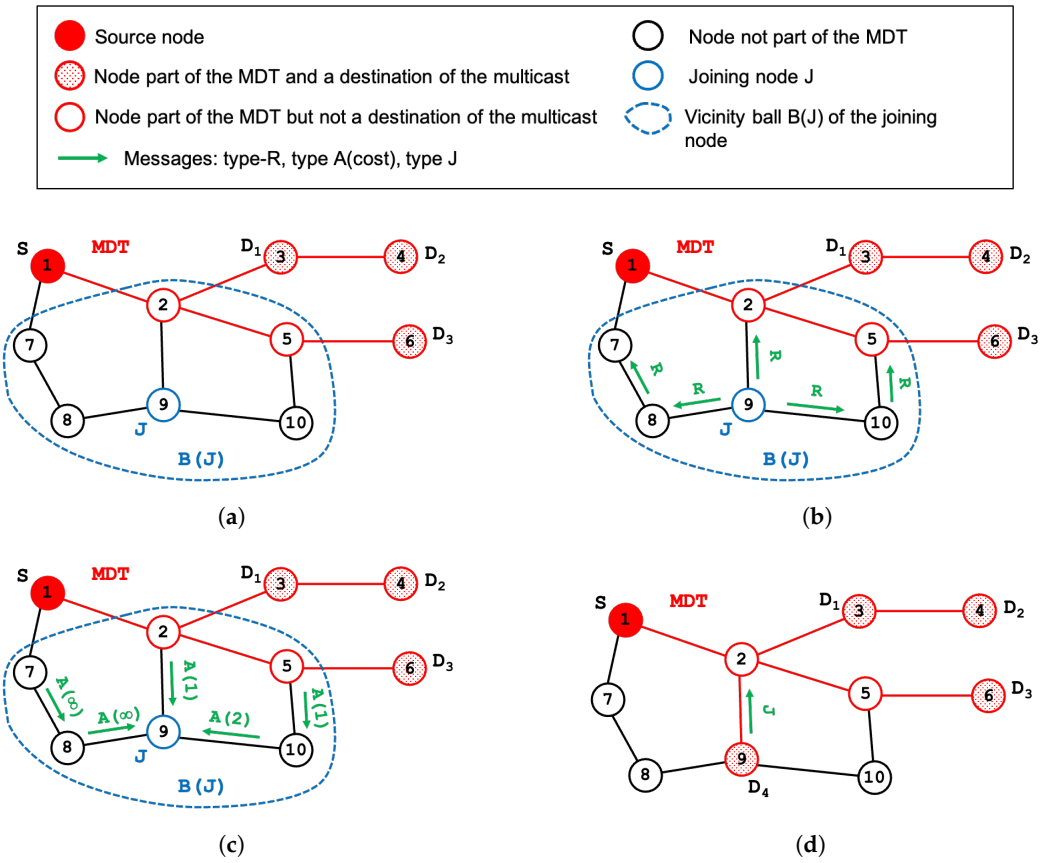


Figure 2. Creation of a branching path between a joining node and the MDT using local search: (**a**) initial situation with joining J and the MDT; (**b**) joining node searching messages within its vicinity ball; (**c**) nodes' answers to the joining node; (**d**) node joins the MDT through the least cost branching path.

Figure 3 presents an illustrative example of the global search process. As in the previous example, node J wants to join a given MDT but this time the local search fails as there are no nodes belonging to that MDT in the vicinity ball $B(J)$ (see Figure 3a). It means all type-R messages sent are answered with type-A messages with infinite cost. For simplicity, we do not depict all messages in the figure but only show the paths followed by the request and answer messages up to the edge nodes of the vicinity ball. Therefore, node J needs now to discover the MDT outside $B(J)$ with the global search. The global search starts at the edge nodes of the $B(J)$; such edge nodes are identified during the local search and requested by the joining node J to start the global flooding outside $B(J)$ only (see Figure 3b). When a node is found (Figure 3c), it sends an answer back (type-A message) to the joining node transporting the accumulated path cost. In the example, we can observe that three answers arrive to node J, which selects the least cost branching path. As in the local search case, if two or more paths have the same cost, the preference goes to the oldest path. In Figure 3d, we can see that node J selects the path to node 2 and it is now connected to the MDT.

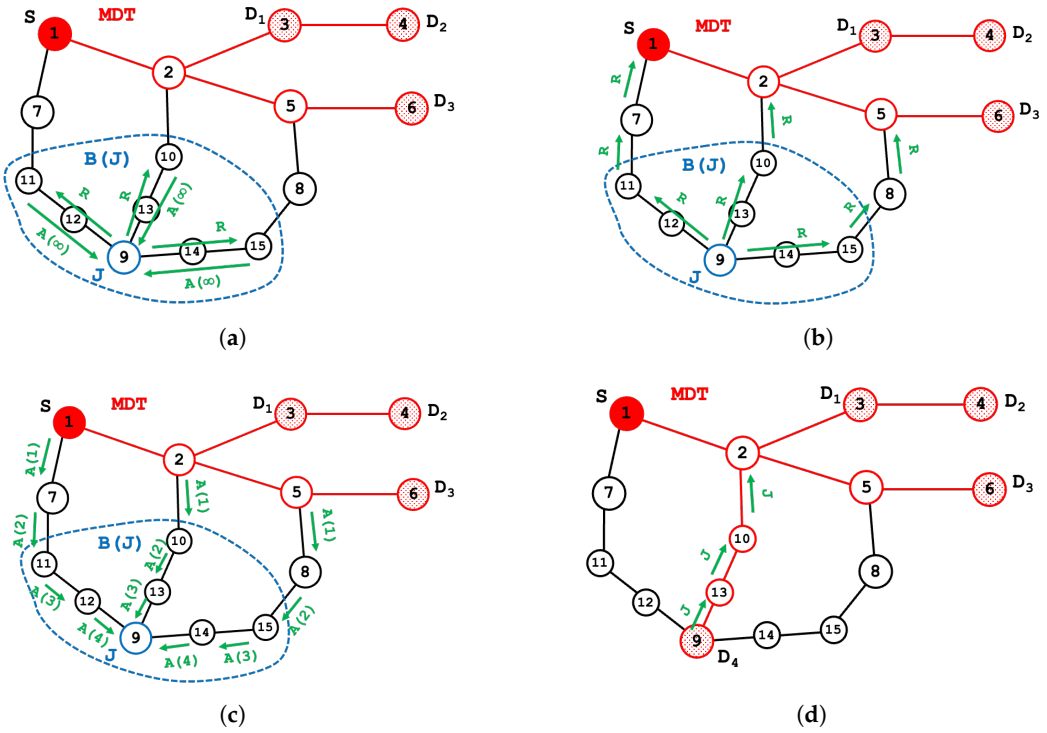


Figure 3. Creation of a branching path between a joining node and the MDT using global search: (**a**) local search fails to find a node of the MDT in $B(J)$; (**b**) joining node J searching messages outside $B(J)$; (**c**) nodes' answers to the joining node; (**d**) node joins the MDT through the least cost branching path.

In Figure 4, we show an illustrative example of an adaptation of the MDT after a link failure event. We assume that the link l fails (Figure 4a) and both upstream node 2 and downstream node 4 (D_1) can detect it, for instance using keepalive messages. In GCMR, the upstream node 2 is in charge of re-connecting the disconnected leaves (Figure 4b). In this example, leaf nodes D_1, D_2, and D_3 are disconnected from the MDT. Node 2 uses type-A messages to the downstream nodes: remember that a type-A message collects the cost of the paths, so then downstream nodes can compare the costs and join the least cost branching paths. At the same time, node 2 sends type-R messages towards the upstream nodes, in such a way that upstream nodes can also start sending type-A messages to reconnect the MDT. In the example in Figure 4b, node 2 sends a type-R message only to the source which in turn sends type-A messages to its neighbours. In Figure 4c, the three disconnected leaf nodes D_1, D_2 and D_3 are reached and they can join the new least cost branching path to node 2 and eventually releases the now unused edges previously part of the MDT. In the example, D_3 receives three messages and compares the cost of the branching paths towards node 2: two paths have the same cost and the preference goes to the path that was received first (the oldest one), i.e., D_3 keeps the path to node 5 which was already part of the MDT. On the contrary, node D_1 needs to explicitly change the direction of the edge to node 7. Finally, node D_2 keeps the same path.

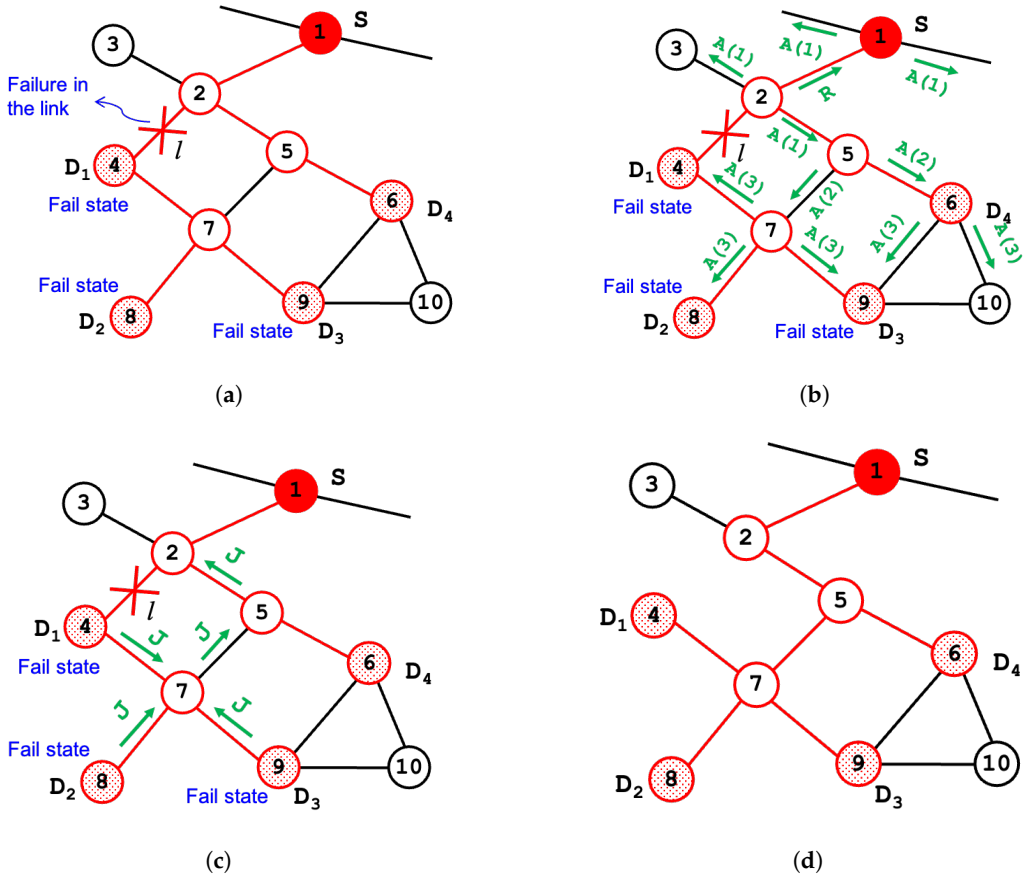
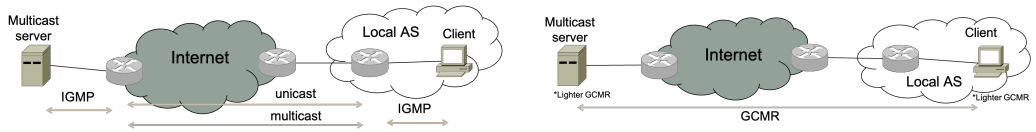


Figure 4. Example of the MDT adaptability: (**a**) a failure in link l disconnects leaves D_1, D_2, and D_3; (**b**) upstream node 2 starts looking for the downstream leaves using a type-A_2 message; (**c**) disconnected leaves join the new least cost branching paths to the upstream node 2; (**d**) leaves are now re-connected and can release previous edges.

3.4. Summary

The GCMR scheme is (i) leaf-initiated, (ii) name-independent, (iii) dynamic, (iv) adaptive, (v) distributed, (vi) protocol-independent, and (vii) QoS-aware. It is leaf-initiated since join/leave requests are initiated by the leaf nodes. It is name-independent as it does not use additional labels or headers containing (partial) information for computing the routing paths like BIER. It is dynamic as nodes can join and leave the MDT during operation without interrupting the service. It is adaptive as it can adapt the MDT to keep it both as optimal as possible and connected in case of a node/link failure. It is distributed since any node can process the incoming messages and independently derive the least cost branching path to the MDT without requiring any centralised or specialised nodes. It is protocol-independent as it is not dependent on any underlying topology constructed by the unicast routing scheme. Finally, it is QoS-aware as different metrics can be consulted to determine the least cost branching path during the search process.

In addition, the GCMR does not need any additional or supporting protocol. Figure 5a shows the different protocols currently involved in the construction and maintenance of an MDT, meaning a multicast protocol like PIM or BIER, a unicast protocol like MP-BGP (Multiprotocol extension Border Gateway Protocol) [35] for addresses discovery, and IGMP for multicast membership subscription between hosts and local routers. In contrast, Figure 5b highlights GCMR as a real end-to-end multicast protocol over the internet: routers run a full featured GCMR while hosts use a lighter version which can initiate or terminate an instance but they cannot forward messages.



(**a**) (**b**)

Figure 5. Protocols required for multicast: (**a**) conventional scheme: a unicast routing, a multicast routing, and a membership subscription protocol; (**b**) GCMR only requires GCMR.

4. Experimental Platform

One of the aims of this work is to experimentally validate GCMR. Hence, we implemented GCMR in a prototype developed using the Quagga open source routing suite and executed it in the iLab.t Virtual Wall emulation testbed platform.

4.1. Quagga Routing Suite

Quagga is an open-source routing protocol suite providing implementations of different routing protocols such as BGP and PIM. It has a significant developer community that includes independent code committers, service providers, and academic institutions. The software is written in the standard C programming language and is compatible with Unix operating systems such as Linux, Solaris, and BSD.

There are just a few other open source routing solutions that enable rapid implementation of additional protocols, features, and functionality. Bird (http://bird.network.cz acccessed on 15 September 2021), and XORP (https://github.com/greearb/xorp.ct accessed on 15 September 2021) are the most popular, and, like Quagga, they all run on conventional PC hardware. We chose Quagga because of its extensive adoption and maturity, whereas Bird's inter-domain routing support is currently limited, and XORP is neither mature nor widely utilised for research. Quagga is also starting to establish itself as the standard reference platform for software-defined routers in production applications.

The Quagga architecture consists of the Zebra daemon (i.e., the core module), which is an abstraction layer to the underlying Unix kernel. The Zebra daemon comes with a collection of client modules named Zserv, each of which implements a different routing protocol. Additionally, Zebra includes the Zserv Application Programming Interface (ZAPI), which allows routing protocol modules to access and send routing modifications to the kernel routing database and network interfaces. The Zebra daemon must be started first and provides the inter-communication between the kernel and the routing protocol daemons we would like to employ in the network. All daemons in Quagga include a command-line interface (named VTY) for configuration, which has a CISCO OS-like syntax. It is also possible to use pre-defined configuration files.

It is worth noting that neither Quagga, Bird, or even XORP are capable of completely implementing a router. They just supply the route engine (algorithms and protocols), so datagram transmission still requires a forwarding engine. The implementation of GCMR as well as the forwarding engine used in our prototype are described in the sections below.

4.2. GCMR Implementation

Figure 6 depicts the architecture and the different modules of a GCMR-capable node. The Routing Core (RC) and the Routing communication Protocol (RP) modules are the two fundamental components of the routing engine. The former is a collection of processes that a GCMR-capable node must implement in order to perform its functions. The latter module is made up of a set of objects and messages that allow GCMR nodes to communicate with one another.

The key operations involved in the GCMR procedure are found in the RC module. This module also contains the database (which includes the data for the TIB and the MRIB tables), the neighbours information, the multicast groups created and two interfaces. One interface is to connect the RC module with the RP module. The other interface is to communication this node with an external client (for instance, an application) which can

request a given GCMR functionality such as creating a new multicast group, joining an existing multicast group, and detaching the node from a multicast group.

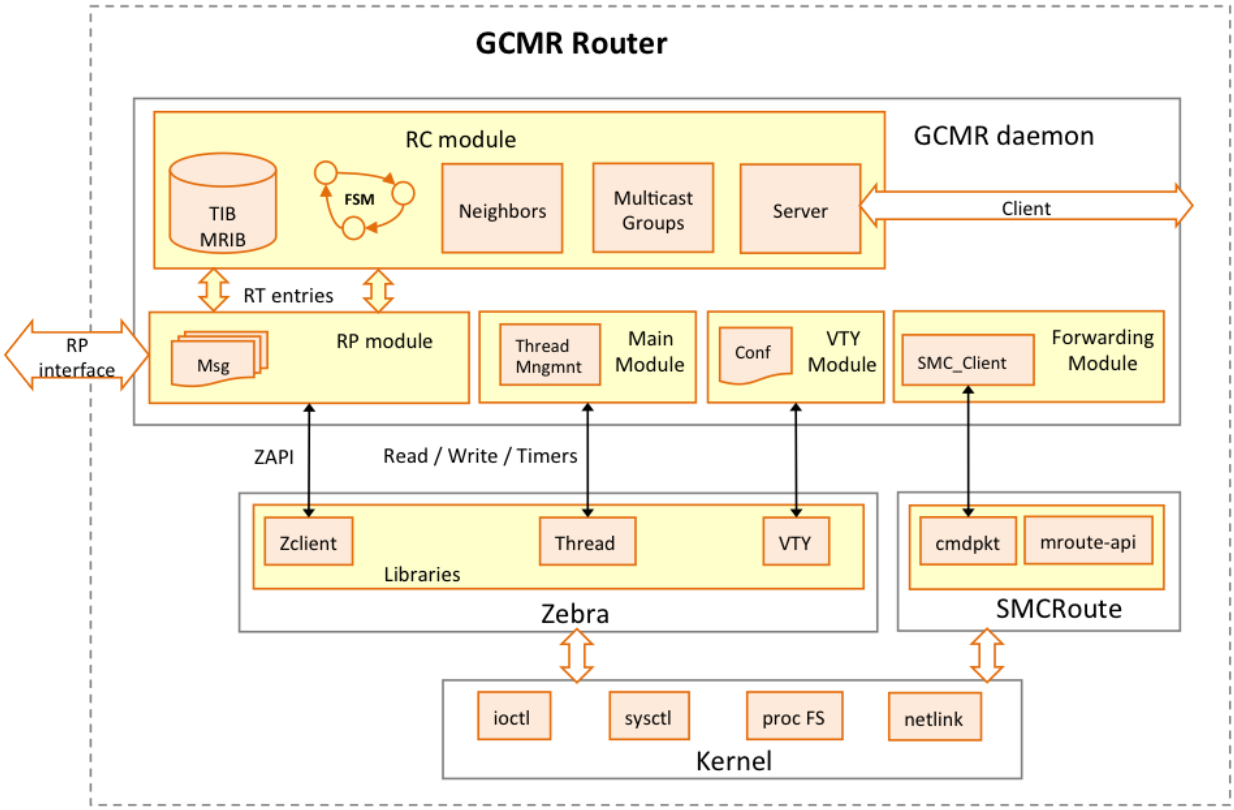


Figure 6. The GCMR node architecture.

All GCMR operations requiring a communication between nodes are implemented in the RP module. This includes therefore all messages involved in the construction and maintenance of the MDT described in Section 3.2: request R and answer A used during the two stage search process; join J and detach D a node to/from the MDT; and failure detection S to detect and localise a node/link failure. Finally, the RP module is also in charge of the communication with the Zebra daemon through the ZAPI interface in order to request the configuration of the interfaces when a node is added or removed to/from a multicast group.

As commented previously, Quagga provides a powerful routing engine but does not come with a forwarding engine. Therefore, we implemented a forwarding module called SMC_Client. This module is able to automatically generate requests to add or remove multicast routes and send them to the SMCRoute daemon (https://github.com/troglobit/smcroute accessed on 15 September 2021). The SMCRoute is a tool to manage and monitor multicast routes directly in the UNIX/Linux kernel.

Finally, the Finite State Machine (FSM) implementation defines the sequence of actions to be followed by the GCMR engine during each MDT computation. Its structure is depicted in Figure 7. Five states are defined:

- Idle: where GCMR initialises resources and structures;
- Local search: when a node sends a local *R* message to its neighbours or receives it from a neighbour;
- Global search: when local search fails, a node moves to the global search;
- Join: when a node receives a *J* message and remains in this state as long as a *D* message is received. In this case, the detach procedure is executed and the node is removed from MDT and returns to the idle state.

- Fail: when a node does not receive a *S* message from the source, it enters in the fail state and tries to recover it. It returns to the join state if the *S* message is received back.

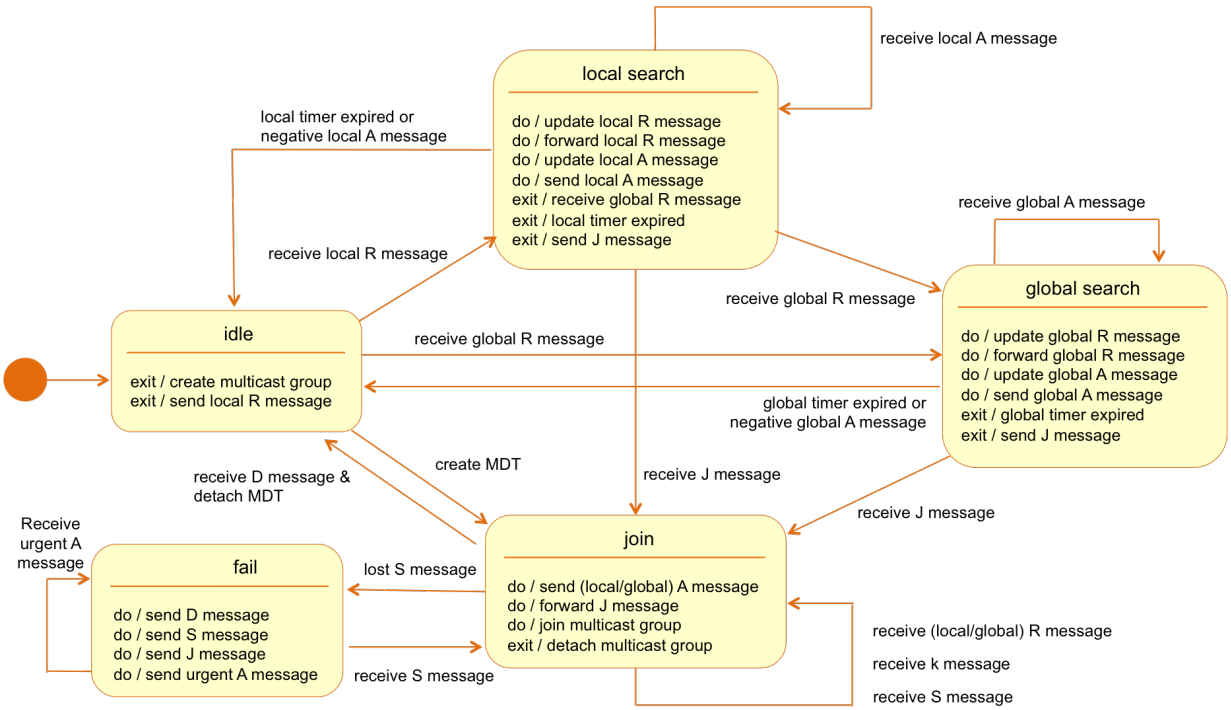


Figure 7. The GCMR state machine.

4.3. iLab.t Virtual Wall Platform

The Virtual Wall at IDLab, an imec research group embedded in Ghent University and the University of Antwerp, is a large-scale generic environment that allows researchers to assess and analyse the performance of network software prototypes by providing computer hardware as well as various software tools. Two facilities are currently available, each one consists of around 100 server blades interconnected by a non-blocking VLAN Ethernet switch. For our experiments, we used one Virtual Wall where each node has the following specifications: 2× dual core CPU, 8 GB RAM, 4 × 80 GB hard disk, four or six Gigabit Ethernet interfaces and a control interface providing remote login.

In Virtual Wall, virtualisation using OpenVZ (http://openvz.org accessed on 15 September 2021) Linux containers is adopted to enable multiple virtual nodes to run on one machine. In this way, large-scale experiments with 10–20 times the number of physical machines are possible. Moreover, it is possible to use virtual interfaces to create an arbitrary number of virtual network links, which can be individually shaped, multiplexed over physical links or used to interconnect virtual nodes within one physical machine.

We used an average of 45 server blades to experimentally validate GCMR. In order to reach a network with more than 200 nodes, each blade was virtualised into five virtual machines. In each node, we installed a Linux distribution, Quagga, SMCRoute and GCMR and used Phyton and bash scripts to configure all nodes and packages according to the experimental setup. During the execution, the information exchanged between the nodes and their status were constantly stored in log files and processed afterwards to validate the experiments and obtain the performance results.

5. Simulation Analysis

5.1. Reference Multicast Schemes

The main objective of these simulation tests is to evaluate the performance of GCMR in the context of inter-domain routing and compare it with the following multicast routing schemes.

The Steiner Tree (ST) algorithm offers the lower bound in terms of stretch, based on the concept of multicast routing stretch described in Section 3.1. We use an ST-Integer Linear Programming (ILP) formulation adapted from Sage's Graph Library (http://www.sagemath.org/ accessed on 15 September 2021) for bi-directional graphs to obtain a near-optimal solution for the ST problem. The formulation is provided in Appendix A. Although the ST scheme uses a centralised approach to create the minimum tree, we assume that the nodes still require exchange of messages during the MDT construction process, just as they would in a distributed environment. Thus, at each stage of the construction of the MDT, the communication cost of ST is calculated by counting the minimum number of messages generated by the nodes to keep the global knowledge of the MDT status. In this way, any node can create and maintain an entry in its MRIB routing table to reach the MDT's nearest node. It means that there is no need for URIB entries as the the MRIB is constructed from the global knowledge of the MDT status. On the contrary, the TIB table is needed at each node of the tree to forward data packets towards the destination nodes of the multicast group. We assume that ST is also the reference for the transmission cost: packets are sent through the minimum tree with their IP headers without the need for any extra overhead.

The currently deployed PIM-SSM routing [18] constructs the MDT using data exchanged via a unicast routing protocol that includes the multicast source s as well as information about the routing path used to reach that source. If we assume that there are is routing policy changing the default behaviour, the resulting routing path is the shortest path from each leaf node to the source s. As a consequence, the MDT constructed in this way is a Shortest Path Tree (SPT). Each node keeps the following entries in its local routing table (i) an entry in the URIB per neighbour node to exchange routing messages, (ii) an entry in the MRIB per selected path to the multicast source s (obtained from the unicast routing table), and (iii) a multicast forwarding entry in the TIB for source s and group G. As in the case of ST, packets do not require any additional overhead besides the standard IP headers. Nonetheless, the stretch of the SPT computed by PIM-SSM should be higher than ST. Hence, we also expect higher transmission cost as packets will traverse additional edges with respect to the ST reference case.

The BIER proposal distributes the multicast packet using the unicast routing scheme. As in the PIM case, without routing policy, the routing paths are the shortest path from the source s to each receiver, meaning that the resulting MDT is an SPT per BIER sub-domain. Each node keeps the following entries in its local routing table: (i) an entry in the URIB per neighbour node to exchange routing messages, (ii) a 16-bit unique number called a BIER Forwarding Router identifier (BFR-id) to identify each router in a BIER sub-domain, and (iii) the Bit Index Routing Table (BIRT) containing an entry per each destination and the route to reach it [4]. Each entry of this BIRT consists of the BFR-id decomposed in the bit-string of 256 bits and its subset of 8 bits, the IP address of the destination and the IP address of the next hop. In addition, BIER adds a header in front of the IP packets containing at least 256 bits for the bit-string and 8 bits for the subset (we assume that the BIER header is 362 bits long as indicated in [36]). Finally, packets belonging to different subsets need to be transmitted separately, meaning that the transmission cost of BIER would probably be higher than the others. On the contrary, BIER does not require the construction of the MDT; it only needs to setup the unique identifiers for each router once at the beginning (or if a change/failure affects these identifiers) and then fully populates the BIRT. Therefore there is no need to interchange messages between the routers during the construction of the MDT; thus, we assume the communication cost only involves the interchange of the BFR-id.

Finally, for sake of simplicity, we do not consider the IGMP protocol required by ST, PIM-SSM, and BIER to subscribe hosts to their local router to join a multicast group. Therefore, any additional communication or transmission cost due to IGMP is not accounted in the following results. Please note that this is a simplification that goes in favour of ST, PIM-SSM and BIER as GCMR does not require IGMP.

5.2. Scenario

The goal is to compare GCMR against ST, PIM-SSM and BIER in terms of stretch, memory size, communication cost, and transmission cost as defined in Section 3.1. Therefore, we developed an ad-hoc simulator written in c in order to be able to execute, in reasonable time, instances of the four protocols in large-scale network topologies.

The simulations are performed using four CAIDA maps of the Internet topology comprising 16k, 26k, 36k, and 46k nodes as of January 2004, August 2008, January 2011 and November 2013 (https://api.asrank.caida.org/v2/docs accessed on 15 September 2021), respectively. The scenario executed simulates the construction of the multicast routing path for a leaf node set of increasing size from 400 to 4000 nodes. Each execution is performed 20 times by considering 20 different random selections of the source and the leaf nodes.

It is worth mentioning that the stretch and the transmission cost of BIER strongly depend on the configuration of its subsets. In fact, each subset can contain 256 nodes at maximum; since we setup MDT ranging from 400 to 4000 leaf nodes, one single BIER sub-domain is not able to contain all leaf nodes. Thus, to be fair, we consider two opposite cases: (i) BIER (best) considers the best possible situation where the minimum number of subsets is required to allocate all leaf nodes and hence the minimum number of both edges in the separate MDTs and packet transmissions are needed; and (ii) BIER (worse) considers the worse case where the maximum number of subsets is required.

5.3. Results

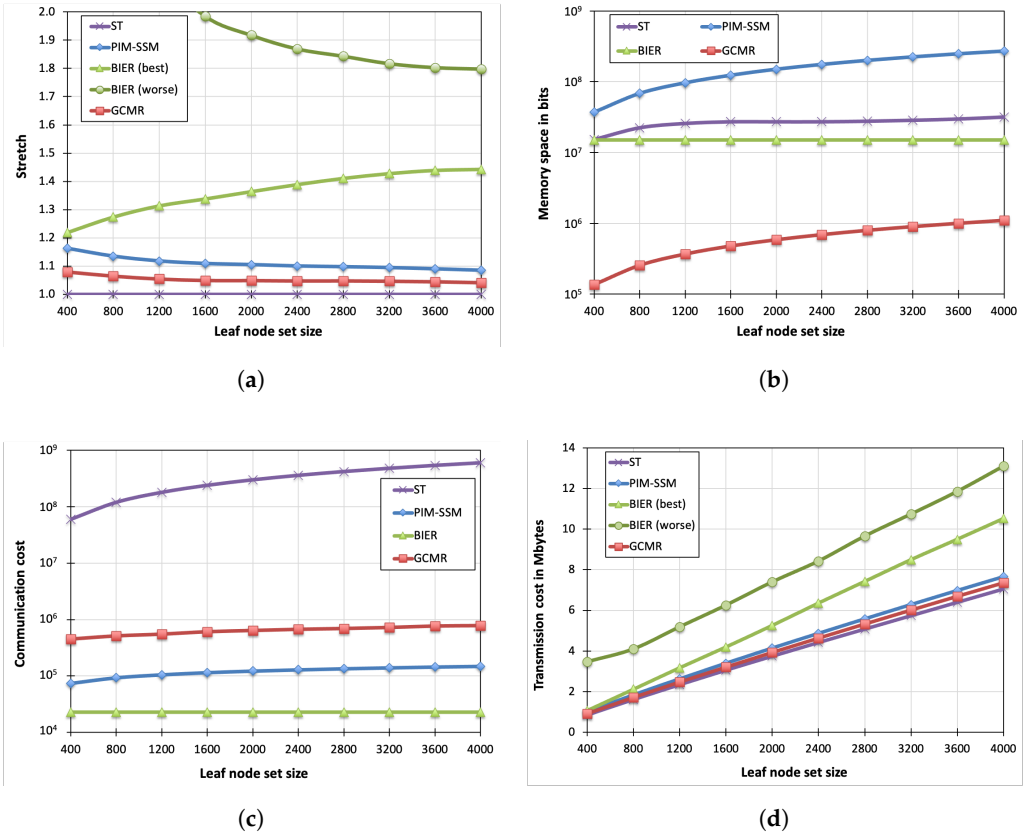
Figure 8 summarises the results we obtained of the four metrics considering the 46k CAIDA map and leaf nodes ranging from 400 to 4000.

Figure 8a compares the stretch of GCMR against ST, PIM-SSM, BIER (best) and BIER (worse). Note that the stretch refers to the *distance* between the MDT created by the protocol and the minimum tree. The minimum tree is obtained with ST and for this reason is one for any leaf node set in this figure. The stretch of GCMR is in between ST and PIM-SSM; with the increase in the size of the MDT from 400 to 4000, GCMR is between 0.1 and 0.05 better than PIM-SSM and between 0.1 and 0.05 worse than ST. In contrast, BIER requires the construction of multiple MDTs to reach all leaf nodes as they need to belong to different subsets (i.e., BIER sub-domains). For instance, BIER (best) requires 2 MDTs for 400 leaf nodes and this number increases up to 16 MDTs to allocate 4000 leaf nodes. The consequence is that the stretch of BIER (best) is higher than the other schemes, reaching almost 1.45 with 4000 leaf nodes. Even worse, the stretch of BIER (worse) is out of scale in the figure when the leaf node set size is below 1600, then it decreases and reaches 1.8 with 4000 leaf nodes.

Figure 8b presents the results in terms of memory space (in bits). It is worth mentioning that for the ST, PIM-SSM and BIER cases we do not consider all routing table entries but only those of the nodes part of the MDT and required by the multicast protocols to route packets through the MDT. We can observe that GCMR outperforms the other schemes. For a leaf set of 400 nodes, the memory space ratio between GCMR and the other schemes is almost 2 orders of magnitude. This difference decreases as the size of the leaf node set increases, reaching one order of magnitude with 4000 leaf nodes.

Simulation performed on the CAIDA map comprising 46k nodes shows that the communication cost of the GCMR scheme is relatively higher compared to BIER and PIM-SSM (see Figure 8c); ST requires a global flooding after each event and thus its communication cost is several times higher than the other schemes. BIER achieves the best performance in this metric as it does not require the construction of the MDT.

Nonetheless, this performance comes with a cost: BIER requires addition of a specific BIER header to each packet and, in order to not have very large header, one packet can be transmitted to 256 leaf nodes at maximum. This means that the transmission cost of BIER (see Figure 8d) is higher than the other schemes. In particular, we can observe that, with the best possible configuration of the BIER sub-domains, BIER (best) can obtain similar performance with 400 leaf nodes. However, with increasing leaf node set size, its transmission cost quickly starts to grow and at 4000 leaf nodes BIER (best) is 40% worse than, for instance, GCMR. The other three schemes achieve comparable results, with ST being the best as its packets traverse the minimal tree.



(**a**) (**b**)

(**c**) (**d**)

Figure 8. Comparison between GCMR, ST, PIM-SSM and BIER in the simulation platform for the 46k map: (**a**) stretch; (**b**) memory size in bits; (**c**) communication cost in number of messages; (**d**) transmission cost in bits.

In Table 1, we compare the stretch and the memory space ratio considering 2000 leaf nodes and the 16k, 26k, 36k and 46k CAIDA maps. We can observe that the stretch of all schemes increases with increased network size compared to the optimal stretch of ST. As expected, BIER (best) is the scheme experiencing the higher increase while GCMR is very close to the optimal stretch and its deviation only slightly increases with the network size.

For the memory space ratio, we used GCMR as the reference as it is the scheme requiring less information in its local routing table. This ratio gives a reasonable indication about the improvements provided by GCMR in terms of the memory space needed to store the routing information and routing table entries compared to the other schemes. We can see that the gain of GCMR increases against the other schemes as the size of the network increases from 16k to 46k nodes: for instance from 146.81 to 270.38 (PIM-SSM) and from 9.54 to 25.7 (BIER).

Table 1. Stretch and memory space consumption considering 2000 leaf nodes and the 16k, 26k, 36k and 46k networks.

	Stretch				Memory Space Ratio			
	16k	**26k**	**32k**	**46k**	**16k**	**26k**	**32k**	**46k**
ST	1	1	1	1	23.51	29.47	32.67	46.5
PIM-SSM	1.058	1.072	1.082	1.105	146.81	160.6	202.2	270.38
BIER (best)	1.083	1.143	1.250	1.364	9.54	14.98	20.17	25.7
GCMR	1.031	1.038	1.044	1.049	1	1	1	1

In summary, we can conclude that the GCMR scheme is very competitive against PIM-SSM and BIER; GCMR obtains the lowest memory space requirements and the best stretch. Nevertheless, GCMR requires around 10 and 40 times more messages than PIM-SSM and BIER to setup the MDT, respectively. BIER presents the lowest communication cost and would probably obtain similar performance in the other metrics with smaller leaf node sets than the one considered in this work. Nonetheless, considering that the internet is continuously growing in the number of ASes, equipment and hosts and we are experiencing a globalisation of the multimedia interests, we do believe that a multicast service should be able to reach as many receivers as possible at the least possible cost.

6. Experimental Analysis

6.1. Reference Multicast Scheme

The primary goal of these experiments is to show that GCMR can operate successfully in the context of inter-domain routing on a large-scale network testbed. In this section, we experimentally compare GCMR only against the standard PIM-SSM since the external daemon named qPIM (http://www.nongnu.org/qpimd/ accessed on 15 September 2021) is available at this moment for Quagga. In contract, the BIER protocol has not been implemented yet.

As described previously, PIM-SSM needs two extra protocols: a unicast routing protocol and a host–router subscriber protocol. During MDT creation, the former is required to create the entries in the routing table, from which PIM-SSM generates the MRIB. For our inter-domain scenario, we used the MP-BGP Quagga daemon. For the latter, the IGMP protocol is utilised to manage multicast group subscription between hosts and routers. As a forwarding engine, qPIM makes use of the Linux kernel's ipmroute daemon to analyse the MRIB and generate the TIB entries accordingly as well as to create the add/delete instructions for the configuration of the network interfaces.

6.2. Scenario

The PIM-SSM and GCMR experimental tests are carried out in a network of 207 Autonomous Systems (ASes) as illustrated in Figure 9. The behaviour of each AS has been simplified as a single router since we are analysing the construction of a multicast service in an inter-domain scenario. Hence, these 207 ASes represent only a section of the internet in which one AS offers this multicast service to the remaining 206 ASes.

We perform ten runs of the identical experiment: initially, one multicast server in one AS is chosen, and then ten receivers from ten distinct ASes are sequentially added to the MDT. The server and nine receivers are selected at random, while one of the receiver is always the client located at the UPC premises in Barcelona and connected to the Virtual Wall by means of an openvpn tunnel. These runs allowed us to evaluate the average stretch, the routing table size, and the communication cost of PIM-SSM and GCMR.

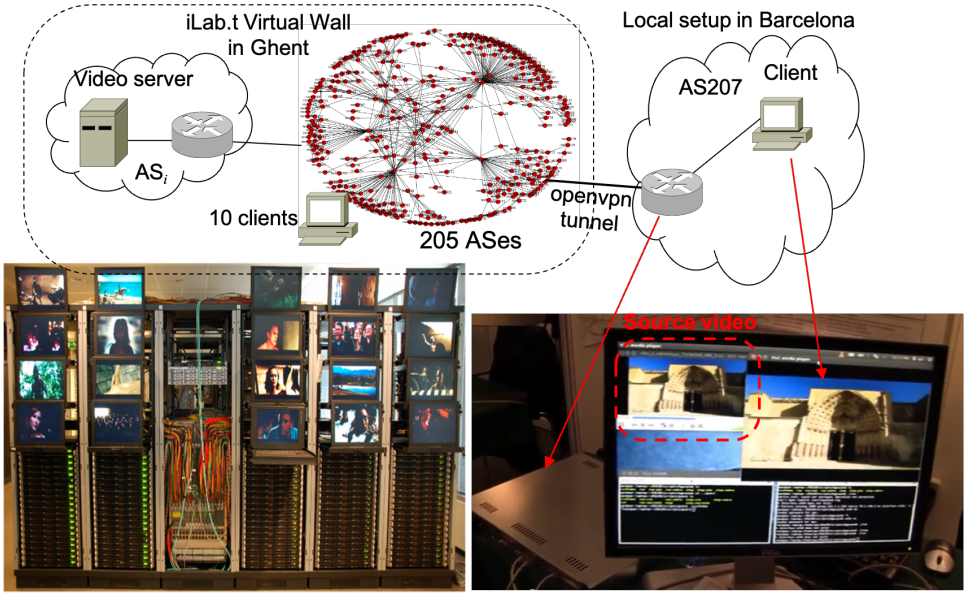


Figure 9. Setup of the experiments.

In addition, we are also interested in evaluating the recovery time experimentally. For this particular test, once the MDT is complete with one video server and the 10 clients, we start the transmission of a Full High Definition video and, after 5 min, we emulate a failure in the link connecting AS1 and AS6 (see Figure 10). Thus, we are able to calculate the time needed to receive back the video at the client in Barcelona after the failure in the MDT. To stress the so-called path exploration problem of BGP [37], we consider a specific inter-domain scenario, where two blocks of ASes are interconnected by means of the link between AS1 and AS6. Path exploration implies that certain BGP routers may explore a number of transitory paths in reaction to path failures or routing policy changes before adopting a new best path or announcing the destination unreachable. As a result, it may take a long time until the entire network converges on a final choice, resulting in poor routing convergence. When we discuss the results below, we will look into this phenomena.

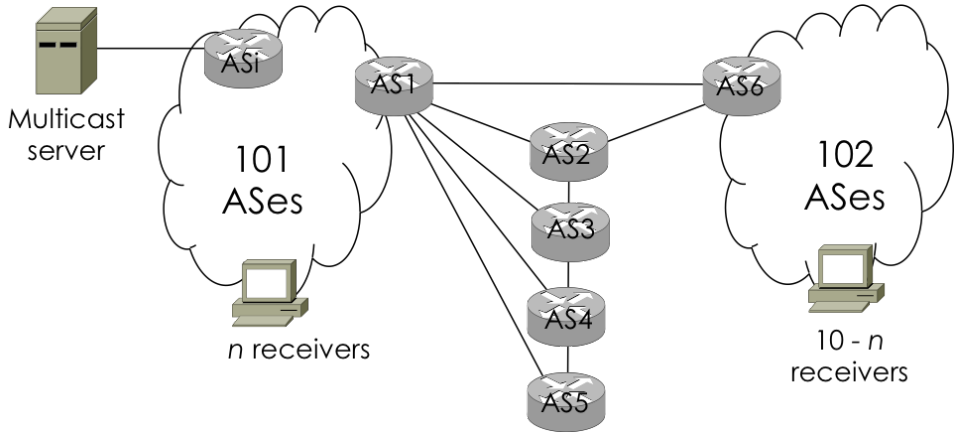


Figure 10. Particularity of the AS1-AS6 connectivity.

6.3. Results

Figure 11a depicts the average GCMR and PIM-SSM stretch of 10 separate runs. In all executions, GCMR has less stretch (0.14 better on average) and some degradation (0.095 on average) when compared to the optimum MDT (stretch-1 is the reference). This finding is consistent with the simulation results presented in Section 5, which show that GCMR achieves about 0.1 better stretch than PIM-SSM in considerably larger networks and a higher number of leafn nodes.

The number of ASes part of the MDT is shown in Figure 11b. GCMR is considerably closer to the optimum MDT than PIM-SSM. From the cost perspective, this is a significant benefit of GCMR since providers charge customers depending on the resources used to deliver the multicast service. Hence, the cost for the costumers should be lower when compared to PIM-SSM since GCMR utilises fewer transit ASes.

Figure 11c compares the two protocols in terms of memory space ratio. We can observe that GCMR requires, on average, around 2.9 times less memory than PIM-SSM. Remember that we only consider the memory space stored in the nodes part of the MDT using the formats defined in Section 5.1: unlike GCMR, which only requires MRIB and TIB entries, PIM-SSM additionally requires certain unicast information from BGP in order to calculate the shortest path to the multicast source.

To analyse the control overhead of the protocols, we determine the communication cost of PIM-SSM and GCMR in Figure 11d. Results shown that GCMR needs an average of 20% more messages than PIM-SSM to set up the MDT.

Again, both the memory space and the communication cost here are consistent with the previous simulation results. For instance, observing the memory space ratio in Table 1, we can see that the ratio for PIM-SSM increases with the size of the network from 146.81 with 16k nodes to 270.38 with 46k nodes. As we are considering a network with 207 nodes in these experimental runs, the gain of GCMR vs. PIM-SSM is reflected by this reduced scenario and drops to 2.9. Similar trends can be seen in communication cost which is around 10 times more when simulating GCMR in a larger network and drops to only 1.2 times more in emulation.

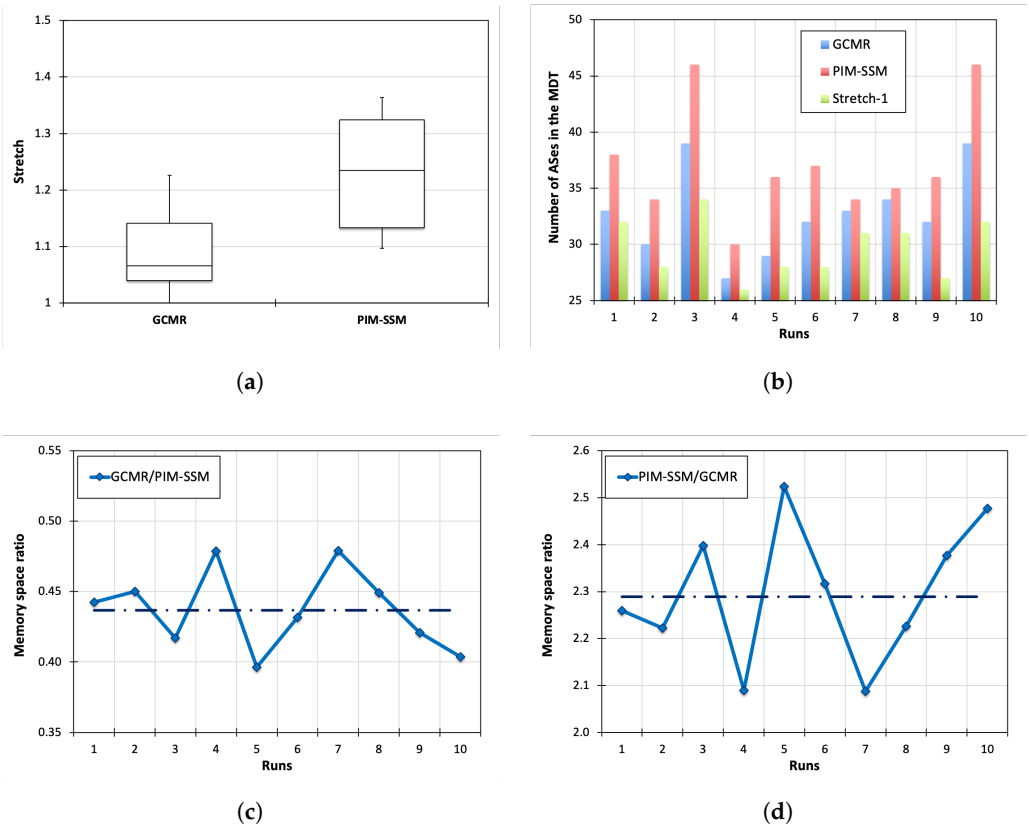


Figure 11. Comparison between GCMR and PIM-SSM in the emulation platform: (**a**) stretch; (**b**) number of ASes in the MDT; (**c**) routing table size ratio; (**d**) communication cost ratio.

The final results concern the recovery time. As commented on above, we count the maximum time elapsed to receive back the multicast service at the receiver in Barcelona after emulating a failure in the link between AS1 and AS6. In the case of GCMR, recovery time includes the time it takes for the failure-detecting node to start a search and receive

responses from its neighbours pointing to the least cost branching path, as well as the time it takes to send a Join message and reconnect the MDT. The experiments show that the recovery time for GCMR in this particular scenario is approximately one second. In the case of PIM-SSM, firstly we need MP-BGP to detect the failure, withdraw the entries of the affected paths to the multicast source, and find the new routes [35], i.e., MP-BGP needs to converge to the new stable state. Secondly, we need PIM-SSM to find the new adjacencies with its Hello messages and exchange the Join message with the new next hops toward the multicast source [19]. Because of the specific nature of the AS1-AS6 interconnection, BGP tends to check all options (an issue known as path exploration [37]) before reaching the convergence. This means that the service interruption experimented with PIM-SSM is significant high, approximately 2 min and half.

7. Conclusions

In this paper, we have presented the GCMR scheme and analysed and compared its performance by simulation against three reference multicast routing algorithms, namely ST, PIM-SSM and BIER. In addition, we have described the implementation of a GCMR prototype using the library of the Quagga open source routing engine and the large experimental tests against PIM-SSM in the iLab.t Virtual Wall testbed. We have proved that GCMR provides a better trade-off between the decrease in the memory space required to locally store the routing information and the increase in the stretch produced. In the experimental platform, we demonstrated the successful operation of GCMR, the consistency between these results and the simulation ones, and the capability of GCMR to considerably reduce the recovery time in case of failure compared to PIM-SSM.

On the contrary, the results obtained for the communication cost ratio between BIER (but also against PIM-SSM) and GCMR show that further improvements are still needed. Future work will focus on this aspect and aims at reducing the communication cost of GCMR at the expense of marginally increasing the memory space consumption. One possible solution is to configure a so-called source ball around the multicast source; nodes at a given path budget distance from the source will be considered inside this source ball and will store a path to the source node. In this way, when a new joining node begins to flood a search message and it reaches one of the nodes within the source ball, this can forward the message directly to the source, preventing the need for further exploration.

Author Contributions: Conceptualisation, D.C. and D.P.; methodology, D.C. and D.P.; software, F.A.; validation, F.A., D.C. and D.P.; formal analysis, D.P.; writing—original draft preparation, D.C.; writing—review and editing, D.C. and D.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially funded by the Spanish Ministry of Economy and Competitiveness under contract FEDER TEC2017-90034-C2 (ALLIANCE project) and supported but not funded by the Generalitat de Catalunya under contract 2017SGR-1037.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AS	Autonomous System
BGP	Border Gateway Protocol
BIER	Bit Index Explicit Replication
BIRT	Bit Index Routing Table
GCMR	Greedy Compact Multicat Routing

IGMP	Internet Group Management Protocol
MDT	Multicast Ditribution Tree
MLD	Multicast Listener Discovery
MP-BGP	Multiprotocol extension Border Gateway Protocol
MRIB	Multicast Routing Information Base
PIM-SSM	Protocol Independent Multicat-Source Specific Mode
SPT	Shortest Path Tree
ST	Steiner Tree
TIB	Tree Information Base
URIB	Unicast Routing Information Base

Appendix A

As commented on in Section 5.1, we consider an ILP formulation to obtain the near optimal solution for an ST algorithm adapting the LP model provided in Sage's Graph Library [38].

Given the same definition provided in Section 3.1: a network topology modelled by an undirected weighted graph $G = (V, E, \omega)$, where the set $V, |V| = n$ represents the finite set of vertices or nodes (all multicast capable), the set $E, |E| = e$ represents the finite set of edges or links, and ω is a non-negative weight or cost function $\omega : E \rightarrow \mathbb{R}+$. Given a set M of vertices, we want to find an acyclic sub-graph of minimum cost, $T_{s,M} = (V_T, E_T)$, linking all of them together. This sub-graph $T_{s,M}$ of G has $\mathcal{V} = |V_T|$ vertices and $\mathcal{E} = |E_T| = |V_T| - 1$ edges and contains each vertex from M. We set e_i and e_o as any incoming and outgoing edge, respectively.

The following two binary variables are defined to know if an edge e or a vertex v is part of the sub-graph $T_{s,M}$:

$$x_e = \begin{cases} 1 & \text{if edge } e \in T_{s,M} \\ 0 & \text{otherwise} \end{cases} \quad x_v = \begin{cases} 1 & \text{if vertex } v \in T_{s,M} \\ 0 & \text{otherwise} \end{cases} \tag{A1}$$

The objective of the optimisation problem is to minimise the total number of edges used to connect all the vertices in M.

$$\min U = \sum_{e \in \mathcal{E}} w_e \times x_e \tag{A2}$$

subject to

$$\sum_{\substack{e_o \in \mathcal{E} \\ e \sim v}} x_e \geq 1, \quad \forall v \in M, \tag{A3}$$

$$\sum_{e_i \in \mathcal{E}} x_e \leq 0, \quad v = s, \tag{A4}$$

$$\sum_{\substack{e_i \in \mathcal{E} \\ e_i \sim v}} x_e \leq 1, \quad \forall v \in V, \forall e_i \in \mathcal{E}, \tag{A5}$$

$$x_e \leq x_v, \quad e \sim v, \forall v \in \mathcal{V}, \forall e \in \mathcal{E}, \tag{A6}$$

$$\sum_{\substack{e_o \in \mathcal{E} \\ e \sim v}} x_e \leq \{x_v + \sum_{\substack{e_i \in \mathcal{E} \\ e \sim v}} x_e\}, \quad \forall v \in \mathcal{V}, \tag{A7}$$

$$\sum_{\substack{e_o \in \mathcal{E} \\ e \sim v}} x_e \geq 1 - \{(1 - x_v) + (1 - \sum_{\substack{e_i \in \mathcal{E} \\ e \sim v}} x_e)\}, \quad \forall v \in \mathcal{V}, \tag{A8}$$

$$\sum_{v \in \mathcal{V}} x_v - \sum_{e \in \mathcal{E}} x_e = 1 \tag{A9}$$

Equation (A3) proves that each node of the multicast group must have at least one of its edges, either incoming or outgoing, as part of the sub-graph. Equation (A4) provides

that the source node can not have any of its incoming links as part of the sub-graph. Equation (A5) defines that only one incoming link of the node can be part of the sub-graph. Equation (A6) means that if an edge is part of the sub-graph so it is the vertex. Equations (A7) and (A8) that guarantee that, if a node is not part of the multicast group and it has one incoming edge, it must have at least one outgoing edge as part of the sub-graph too. Finally, Equation (A9) says that the total number of vertexes is equal to the number of edges plus one.

References

1. Cheriton, D.R.; Deering, S. Host groups: A multicast extension for datagram internetworks. *ACM SIGCOMM Comput. Commun. Rev.* **1985**, *15*, 172–179. [CrossRef]
2. Ratnasamy, S.; Ermolinskiy, A.; Shenker, S. Revisiting IP multicast. In Proceedings of the ACM SIGCOMM 2006 Conference, Pisa, Italy, 11–15 September 2006; pp. 15–26.
3. Holland, J. Why Inter-Domain Multicast Now Makes Sense. *APNIC Blog*. **2020**. Available online: https://blog.apnic.net/2020/07/28/why-inter-domain-multicast-now-makes-sense/ (accessed on 30 July 2021).
4. Wijnands, I.; Rosen, E.; Dolganow, A.; Przygienda, T.; Aldrin, S. Multicast Using Bit Index Explicit Replication (RFC 8279), Internet Engineering Task Force. 2017. Available online: https://tools.ietf.org/html/rfc8279 (accessed on 15 September 2019).
5. Holland, J. IP Multicast: Next steps to make it real. In Proceedings of the NANOG 79, Virtual Meeting, 1–3 June 2020. Available online: https://www.nanog.org/events/nanog-79/ (accessed on 30 July 2021).
6. Chen, M.; Hughes, G.L. Group Based Multicast Streaming Systems and Methods. U.S. Patent US8806522B2, 12 August 2014.
7. Tombes, J. *Multicast ABR, Low-Latency CMAF, CTE, and Optimized Video Playback*; White Paper; Broadpeak's nanoCDN. 2019. Available online: https://broadpeak.tv/solutions/multicast-abr/ (accessed on 30 July 2021).
8. Ramp Holdings. *Enterprise Multicast Is Alive and Well*; White Paper; Ramp Multicast+. 2020. Available online: https://www.rampecdn.com/enterprise-video-delivery/multicast/ (accessed on 30 July 2021).
9. Zarkower, J. How an Evolving Pay TV Infrastructure Can Coexist with OTT. *Streaming Media*, 6 March 2019. Available online: https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=130383 (accessed on 30 July 2021).
10. Diot, C.; Levine, B.; Lyles, B.; Kassem, H.; Balensiefen, D. Deployment issues for the IP multicast service and architecture. *IEEE Netw.* **2000**, *14*, 78–88. [CrossRef]
11. Pardue, L. Some challenges with IP multicast deployment. In Proceedings of the IAB Design Expectations vs. Deployment Reality in Protocol Development Workshop, Helsinki, Finland, 4–5 June 2019.
12. Pedroso, P.; Papadimitriou, D.; Careglio, D. Dynamic compact multicast routing on power-law graphs. In Proceedings of the IEEE Global Communications Conference (GLOBECOM 2011), Houston, TX, USA, 5–9 December 2011.
13. Waitzman, D.; Partridge, C.; Deering, S. Distance Vector Multicast Routing Protocol (RFC 1075), Internet Engineering Task Force. 1988. Available online: https://tools.ietf.org/html/rfc1075 (accessed on 15 September 2019).
14. Moy, J. Multicast Extensions to OSPF (RFC 1584), Internet Engineering Task Force. 1994. Available online: https://tools.ietf.org/html/rfc1584 (accessed on 15 September 2019).
15. Ballardie, A. Core Based Trees (CBT Version 2) Multicast Routing (RFC 2189), Internet Engineering Task Force. 1997. Available online: https://tools.ietf.org/html/rfc2189 (accessed on 15 September 2019).
16. Fenner, B.; Handley, M.; Holbrook, H.; Kouvelas, I.; Parekh, R.; Zhang, Z.; Zheng, L. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification (RFC 7761), Internet Engineering Task Force. 2016. Available online: https://tools.ietf.org/html/rfc7761 (accessed on 15 September 2019).
17. Adams, A.; Nicholas, J.; Siadak, W. Protocol Independent Multicast-Dense Mode (PIM-SM): Protocol Specification (RFC 3973), Internet Engineering Task Force. 2005. Available online: https://tools.ietf.org/html/rfc3973 (accessed on 15 September 2019).
18. Bhattacharyya, S. An Overview of Source-Specific Multicast (SSM) (RFC 3569), Internet Engineering Task Force. 2003. Available online: https://tools.ietf.org/html/rfc3569 (accessed on 15 September 2019).
19. Holbrook, H.; Cain, B. Source-Specific Multicast for IP (RFC 4607), Internet Engineering Task Force. 2006. Available online: https://tools.ietf.org/html/rfc4607 (accessed on 15 September 2019).
20. Carlsberg, K.; Crowcroft, J. Building shared trees using a one-to-many joining mechanism. *ACM SIGCOMM Comput. Commun. Rev.* **1997**, *27*, 5–11. [CrossRef]
21. Yan, S.; Faloutsos, M.; Banerjee, A. QoS-aware multicast routing for the internet: The design and evaluation of QoSMIC. *IEEE/ACM Trans. Netw.* **2002**, *10*, 54–66.
22. Wang, X.; Zhang, J.; Huang, M.; Yang, S. A green intelligent routing algorithm supporting flexible QoS for many-to-many multicast. *Comput. Netw.* **2017**, *126*, 229–245. [CrossRef]
23. Kreutz, D.; Ramos, F.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
24. Islam, S.; Muslim, N.; Atwood, J.W. A Survey on Multicasting in Software-Defined Networking. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 355–387. [CrossRef]
25. Al Saeed, Z.; Ahmad, I.; Hussain, I. Multicasting in software defined networks: A comprehensive survey. *J. Netw. Comput. Appl.* **2018**, *104*, 61–77. [CrossRef]

26. Canonico, R.; Romano, S.P. Leveraging SDN to Improve the Performance of Multicast-Enabled IPTV Distribution Systems. *IEEE Commun. Stand. Mag.* **2017**, *1*, 42–47. [CrossRef]
27. Peleg, D.; Upfall, E. A trade-off between space and efficiency for routing tables. *J. ACM* **1989**, *36*, 43–52. [CrossRef]
28. Abraham, I.; Malkhi, D.; Ratajczak, D. Compact multicast routing. In Proceedings of the International Symposium on Distributed Computing (DISC'09), Elche, Spain, 23–25 September 2009; pp.364–378.
29. Faloutsos, M.; Faloutsos, P.; Faloutsos, C. On power-law relationships of the Internet topology. *ACM Comput. Commun. Rev.* **1999**, *29*, 251–262. [CrossRef]
30. Choromański, K.; Matuszak, M.; Miękisz, J. Scale-free graph with preferential attachment and evolving Internal vertex structure. *J. Stat. Phys.* **2013**, *151*, 1175–1183. [CrossRef]
31. Desmouceaux, Y.; Clausen, T.H.; Fuertes, J.A.C.; Townsley, W.M. Reliable multicast with BIER. *J. Commun. Netw.* **2018**, *20*, 182–197. [CrossRef]
32. Desmouceaux, Y.; Fuertes, J.A.C.; Clausen, T.H. Reliable BIER with peer caching. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1826–1839. [CrossRef]
33. Merling, D.; Lindner, S.; Menth, M. P4-based implementation of BIER and BIER-FRR for scalable and resilient multicast. *J. Netw. Comput. Appl.* **2020**, *169*, 102764. [CrossRef]
34. Zheng, L.; Zhang, J.; Parekh, R. Survey Report on Protocol Independent Multicast-Sparse Mode (PIM-SM) Implementations and Deployments (RFC 7063), Internet Engineering Task Force. 2013. Available online: https://tools.ietf.org/html/rfc7063 (accessed on 15 September 2019).
35. Bates, T.; Chandra, R.; Rekhter, Y.; Katz, D. Multiprotocol Extensions for BGP-4 (RFC 4760), Internet Engineering Task Force. 2007. Available online: https://tools.ietf.org/html/rfc4760 (accessed on 15 September 2019).
36. Wijnands, I.; Rosen, E.; Dolganow, A.; Tantsura, J.; Aldrin, S.; Meilik, I. Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks (RFC 8296), Internet Engineering Task Force. 2018. Available online: https://tools.ietf.org/html/rfc8296 (accessed on 15 September 2019).
37. Oliveira, R.; Zhang, B.; Pei, D.; Zhang, L. Quantifying Path Exploration in the Internet. *IEEE/ACM Trans. Netw.* **2009**, *17*, 445–458. [CrossRef]
38. Cohen, N. Several Graph Problems and their Linear Program formulations. inria-00504914v2. 2019. Available online: https://hal.inria.fr/inria-00504914v2 (accessed on 15 September 2021).