Core-Selection Algorithms in Multicast Routing

Ayşe Karaman and Hossam Hassanein

School of Computing
Queen's University
Kingston, ON K7L 3N6, Canada
{karaman, hossam}@cs.queensu.ca

Abstract

The Core-based approach is inevitable in multicast routing protocols as it provides efficient management of multicast path in changing group memberships, and scalability and performance. In this paper, we present a comprehensive analysis of this approach with the emphasis on core selection for the first time in literature. We first examine the evolution of multicast routing protocols into the core-based architecture and the motivation for the approach. Then we review the core-selection algorithms in the literature for their algorithmic structure and performance. Our study involves an extensive computational and message complexity analysis of each algorithm, and a classification for their deployment characteristics and algorithmic complexities. To the best of our knowledge ours is the first paper providing such extensive comparative analysis of core-based multicast routing protocols.

1. Introduction

Multicasting is the simultaneous delivery of data stream to multiple receivers in the domain. Its applications are numerous in group communications including numerous forms of audiovisual conferencing (discussion forums, live auctions, negotiation systems, white-board applications, online games, chat-rooms) and broadcasting (online radios, e-learning systems), mass delivery in digital marketing (software upgrade distribution, news delivery, stock quotes), replicated database applications in various forms of cross-query systems (airline systems, banking applications, cross-catalog search), offline virtual communities (newsgroups, mailing lists), as well as the trivial resource discovery feature of Internet routers. Group communications on Internet are increasingly more pervasive, and are inevitable parallel to wider acceptance of ecommerce applications.

The objective of multicast routing, in broad terms, is the efficient delivery of the stream to the recipients. Multicast routing, as opposed to unicast routing, attempts to duplicate copies of the packets only at branching nodes, nodes where there are more than one outgoing paths to multiple destinations. Group communications have varying demands on network resources and transmission capacities. Some applications require reliability (B2B transactions, mailing lists) while others demand for speed (B2C services, online broadcasting) or cost efficiency of transmission (delivery systems in digital marketing). The typical application in QoS concern is delay sensitive and requires high bandwidth for these encompass the general requirements of multicast applications. The prominent problem of QoS in today's applications, therefore, is *delay-constrained cost optimization*, i.e., delivering the stream to the receivers at minimum cost while meeting a given upper bound for end-to-end delay.

Multicast routing protocols evolved parallel to the growth in Internet size and coverage from broadcast routing that transmits to all but not just a subset of domain nodes. Earlier protocols constructed shortest paths from sources to each receiver in multicast group and hence source-based shortest path trees. Shortest path tree (SPT) still is the design problem in tree construction. In today's protocols, design concern is on core selection into optimization of metrics additional to delay for the improvement of scalability of the protocol especially in today's QoS-sensitive applications.

Multicast routing protocols evolved into core-based approach for improved performance during its operations in distributed networks. Location and structure of routing tree in a group application clearly is the dominant determinant on the protocol performance in multicast applications. Core based scheme offered alternative in multicast tree root and thus location to source-rooted tree approach for improved scalability and performance of the protocol. Multi-core trees with multiple cores to serve the group ultimately provided more efficient use of the multicast trees by the group. For group applications especially those involving instant interaction among the participants like audiovisual conferencing, the protocol is particularly sensitive to the degree of approximation of the multipoint communication path to optimal under multiple metrics for serving QoS demands. Selecting the location(s) of the core(s)—the process preliminary to tree construction operations is crucial in this respect for protocol operations, and the lack of en effective core-selection process is likely to degrade the efficiency of the multicast routing protocol to a significant extend.

This paper is a study on core-based approach and the core selection algorithms presented in literature. In the next section, we examine the multicast routing protocols and the motivation for the core-based architecture. Section 3 is devoted to our review and analysis of core selection algorithms. We first present the terminology and preliminaries to this line of research in Section 3.1. In Section 3.2, we describe a classification of core-selection algorithms for their implementational characteristics. We provide the review of these algorithms in Section 3.3. In Section 3.4, we analyze each algorithm for its computational and message complexity. Section 3.5 includes comparative discussion on the core selection algorithms studied. The final section includes our concluding remarks. To the best of our knowledge, our study is unprecedented for its analysis and coverage of core-selection algorithms in literature to date.

2. Multicast Routing Protocols

In this section, we overview multicast routing protocols with emphasizing core-based architecture and the current state of these protocols. We distinguish a "*multicast routing protocol*" from "*multicast routing algorithm*" in that a multicast routing algorithm refers to merely the tree building algorithm, parallel to the terminology in the literature. A multicast routing protocol, on the other hand, is the entire architecture that includes the multicast tree building algorithm as well as the maintenance of the tree structure in the dynamisms of the multicast environment, packet forwarding, failure discovery and recovery, and other features depending on the protocol specifics. The main architectural component of a multicast routing protocol is the development of the multicast tree not only because it is the structure that directly affects the performance of the protocol, but also is the most demanding one for the development and maintenance. Another major concern in multicast protocol design is the feasibility of the deployment while maintaining high performance in the presence of cost of control overhead to operate and maintain the construct to support that performance. These two are often contradictory goals since a better approximation of the multicast tree for improved performance demands for larger amount of information as compared to that available on the nodes, which in turn results in more overhead to adjust and maintain the tree with respect to changes in the network conditions and group membership dynamics.

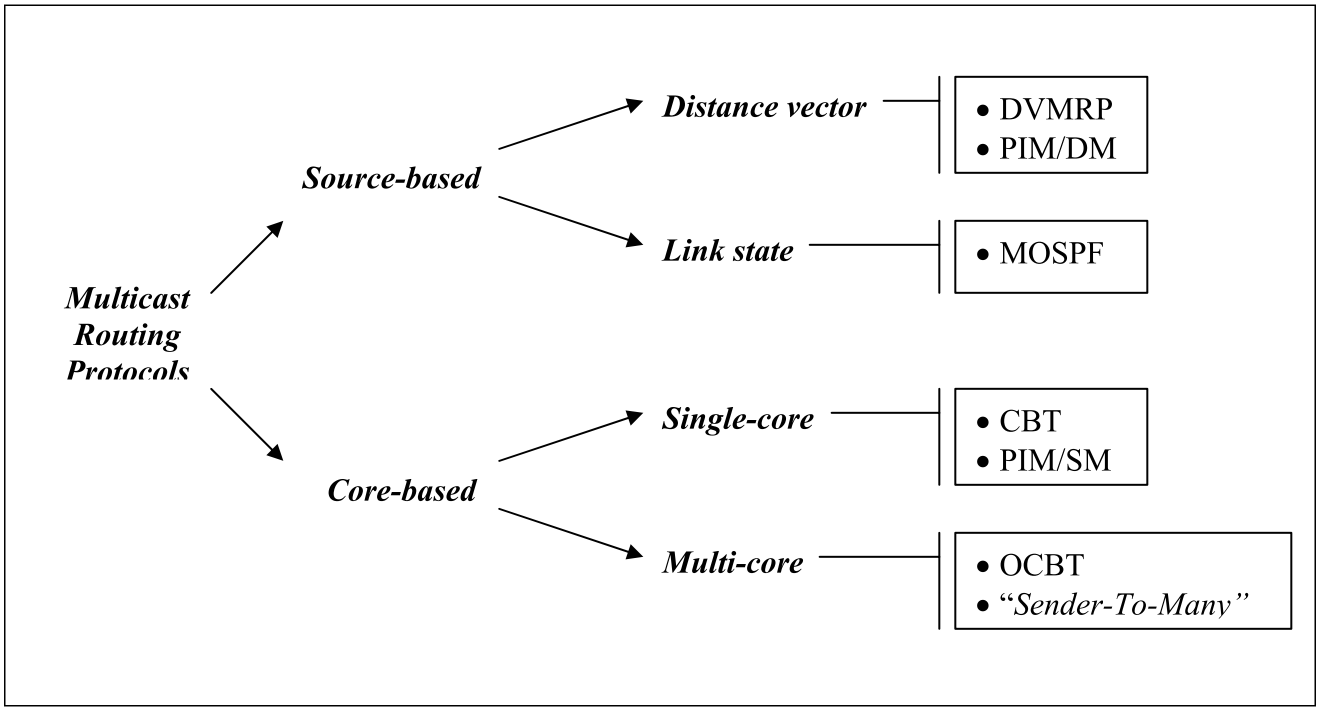

Figure 1. A classification of multicast routing protocols.

In Figure 1, we present a classification of multicast routing protocols with respect to their path construction and maintenance characteristics. The figure to a great extent also indicates the time-sequence of the emergence of the protocols with the exception of PIM/DM which is the dense-mode component of PIM protocol suite appearing in literature along with CBT. Earlier protocols, namely DVMRP and MOSPF assumed source-based multicast trees, i.e., the multicast path is built as a tree rooted as the source of the multicast group. The research later on shifted to core-based trees with the motivation of scalability in sparsely distributed multicast group in a wide domain. Recent trends in multicast protocols include multi-core multicast trees, which involve more than one core in the multicast group for improved tree accessibility and performance. The assumed problem for tree construction in all multicast routing protocols in Figure 1 is SPT on delay metrics.

2.1 Source-Based Protocols

In this group of protocols, the multicast tree is rooted at the source of the group with an attempt to minimize delay between the source and each receiver. As we demonstrate in Figure 1, we further categorize source-based protocols with respect to the unicast scheme they are built on.

Distance vector multicast routing protocol (DVMRP) [WPD88] is the earliest multicast routing protocol on Internet. It is an implementation of distance vector routing for multicasting and is an extension of RIP. DVMRP is a "*broadcast-and-prune*" protocol: it assumes "*every router is a receiver unless it is pruned*". The multicast stream arriving at a node via the shortest path from its source is transmitted to every other link except the incoming link and those links to pruned nodes. It is the protocol one step further from broadcasting in earliest smaller Internet domains towards multicasting to designated nodes. DVMRP is easy to deploy and troubleshoot. Its simplicity has been its major strength in earlier Internet environments with small domain size. Protocol Independent Multicasting–Dense Mode (PIM-DM) [DEFJ94], source-based extension of PIM for dense mode has similar characteristics to DVMRP.

Multicast extension for OSPF (MOSPF) [M94, M94b] is, as the name implies, an extension of the unicast link-state OSPF protocol for multicasting, and is the only link state multicast routing protocol described in the literature. Every MOSPF router keeps a map of the entire domain, and communicates recent changes on links to its immediate neighbors to every other router in the domain. Full availability of domain knowledge to every node makes available centralized processing for multicast path construction: every router uses Dijkstra's [D59] shortest path algorithm to calculate the portion of the tree passing through it. With this, the identical process running on same domain and group information, with identical tie-breakers to provide consistency among the operating nodes in choices made in cases of equally good options throughout the execution of the algorithm generates the same results with local processing on all nodes.

Note that MOSPF is the only suite that assumes a link state approach. In all distance-vector protocols building a multicast tree, path construction is initiated by the new receiver to join the group: the receiver sends a join-request message towards the core or the root of the tree along the shortest path from itself to it. The path the join request travels is remembered by the routers on it, and constitutes the portion of the tree connecting the new receiver to the tree. Thus, the multicast tree constructed by the multicast routing protocols operating in distance-vector platform are reverse-shortest rather than shortest path trees, and correspond to shortest path trees only when the network is symmetric. MOSPF stands out in this respect due to its centralized operation on the availability of full link-state information on all nodes. Every node under this protocol computes the SPT portion on which itself resides using the SPT algorithm with identical tie-breakers as on every other node in the domain. MOSPF routers build and operate on a unique shortest path tree for a multicast stream.

Source-based protocols in general are mostly suitable for small-scale, local-area applications. In sparse mode, the scalability of source-based protocols tend to degrade in terms of network resource consumption. A major incentive in source-based architectures is the simplicity of the protocol as an extension of the underlying unicast platform. As we will be reviewing in the following section, Core-based protocols consider the tree root not necessarily as the multicast source for scalability issues.

2.2 Single-Core Based Protocols

The main incentive behind source-based SPT protocols is cost optimization during multicast forwarding. SPT allows minimum available delay for forwarding the data stream to destinations of the receiver group. However, this comes at the expense of overall bandwidth consumption during transmission and the cost of protocol operation and maintenance.

Core-based trees separate the concept of source from that of the tree root. A shortest path multicast tree is still constructed, but this time rooted at the core. However, the core(s) may or may not be the source. A tree is formed still, rooted at a core this time. The choice of the core(s) can be made according to the topological location of the receiver group considering not only the delay in data forwarding, but also the bandwidth and protocol control overhead. In Figure 2, we present an extreme case when a core-based tree outperforms a source-based tree in terms of total bandwidth consumption, which is a significant advantage of the approach. The example includes a single source in the multicast group. The receivers are located close to one another in a topological neighborhood, and the source is far from their location. The dashed lines indicate a SPT rooted at the source minimizing the delay-distances to each receiver. The alternative is placing a core local to the receiver group and building the tree rooted at this core. The individual delays from the source to the receivers are not necessarily the shortest since each has to pass through the core. However, the data stream travels to the distant location of the receiver group only once, resulting in less bandwidth utilization. For the same reason, core-based tree in this case is easier to maintain than a source-based tree.

There are two single-core based multicast routing protocols in the literature: Protocol Independent Multicasting (PIM) [DEF+96] and Core-Based Tree (CBT) [B97, B97b]. Both protocols build a single tree rooted at a node designated as the core. The core is advertised throughout the domain so that the new receivers to join the tree know whom to contact to initiate the process, and the sources to multicast the group know where to forward their stream for delivery. The multicast tree is constructed/updated as new members join the group.

CBT allows the bidirectional use of the tree so that when the data stream addressed to the core hits the tree at another node on its way, then that node acts as the tree root and relays the stream to every one of its links on the tree. Similarly, an on-tree CBT node, receiving data packets addressed to the multicast group it is serving, relays the stream to every tree link except the incoming one. The bidirectional nature of CBT makes more efficient use of the multicast tree in terms of the network resources. (Note that bi-directional trees are effective to serve this purpose in symmetric network environments. In asymmetric networks where the link cost and delay values are not necessarily equal in both directions, minimal cost is not guaranteed.)

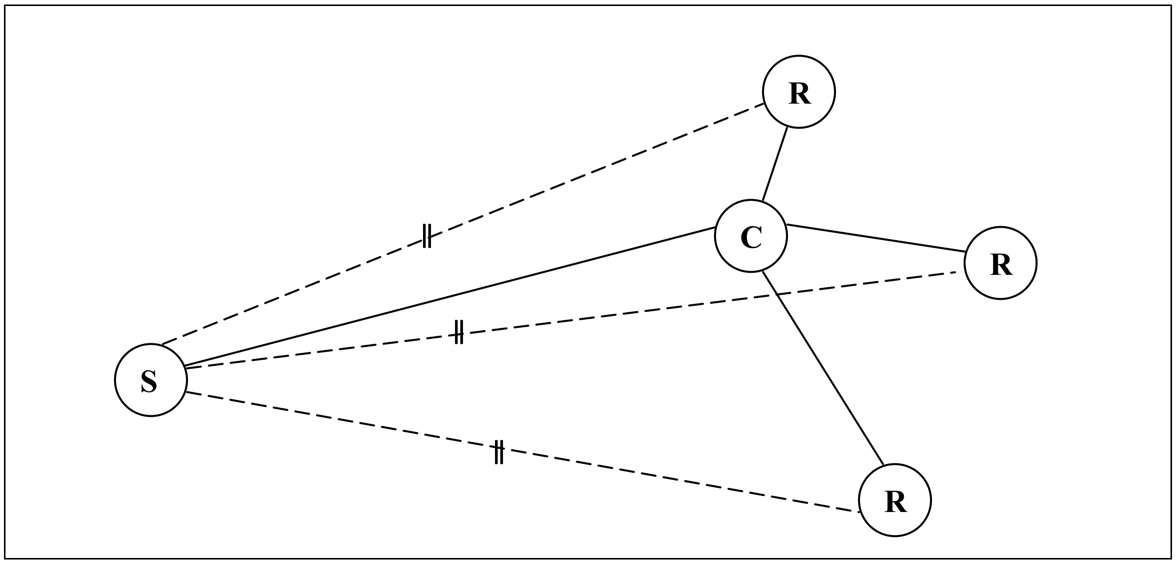

Figure 2. An example for core-based tree outperforming a source-based tree in case of a single source in the multicast group. The receiver group is located close to one another and far from the single source.

PIM is less reliant on the underlying unicast routing protocol in its operation. It refers only to the unicast state database for node distances. All message transmission, including control messages and transmission of the stream from the source to the core of the tree is governed by the protocol itself. PIM also offers individual receivers the option to receive the stream directly from the sender itself if the receiver decides that direct path from the source is more favorable for the path is shorter and/or the data rate warrants direct transmission.

Core-based trees are preferable to source-based trees in case of multiple sources in the multicast group. In this case, source-based schemes result in multiple shortest path trees with each having a source at the root, while a core-based scheme will have a single core based tree serving the entire members of the multicast group. The core-based approach then has the significant advantage of less control traffic of a single shared tree rather than multiple trees and, for the same reason, less state maintenance. Indeed a core-based tree can support a multicast session from more than one sender with only incremental increase in protocol overhead. Results in Wei and Estrin [WE94], and Calvert et al's [CZD95] indicate that the advantages of core-based trees are eminent in sparse mode when the multicast group is distributed widely in a large portion of domain.

2.3 Multi-Core Based Protocols

A core on a multicast tree is a reference on the tree to the multicast tree for the non-member nodes. Since it is among on-tree nodes only the core(s) are advertised throughout the domain so that new receivers can join the tree and sources know where to forward the data stream for delivery. More cores would then mean more references to the tree and thus more efficient use of the tree. More reference points allow the selection of connection points to the tree according to their proximity by the outsider routers, improving the resource consumption and traffic load. Multi-core tree schemes utilize this idea under distance-vector architecture.

There are different multi-core based architectures in the literature. Ordered Core-Based Tree (OCBT) [Sh96, SG97] constructs a unique tree governed by multiple cores to spans the entire receiver group. The senders-to-Many scheme [ZFL02] partitions the receiver groups among the cores and builds exactly one tree per core rooted at that core. Zappala et al [ZFL02] also propose Members-to-Many that supports multiple single-core based trees, each spanning the entire set of receivers. In Members-to-Many, all core-trees are equally available to all new receivers wanting to join and to sources for their choice of the core according to the distance in between.

Ordered core-based tree (OCBT) [Sh96, SG97] constructs and maintains a unique tree consisting of multiple cores with one of them being the root. The main motivation behind OCBT is the development of a loop-free multi-core based tree by assigning level numbers to the cores and the nodes to join the tree to help maintain the tree structure. OCBT ensures a loop-free multicast tree structure even when the underlying unicast domain contains loops. Every on-tree node in OCBT has a unique level number. The tree is rooted at one of the cores. The level number of the root is the highest, and each child has a level number smaller than or equal to that of its parent. The level numbers of the parent and child are not necessarily consecutive. The core nodes do not change level numbers within the tree: a core node's level number is assigned and fixed when it is selected as a core node. Other nodes are assigned level numbers when they join the tree and these numbers are released when they leave the tree. Lower level numbered nodes can move to higher levels, but higher level nodes cannot move down on the tree. The relative accuracy of level numbers is maintained throughout all tree operations. The OCBT tree can be of any depth.

The alternative for a single tree in multi-core approach in the literature is multiple-trees each governed by one core. Zappala, Fabbri and Lo [ZFL02] propose *Senders-to-Many*, an architecture that maintains multiple trees with exactly one tree per core. The receivers are partitioned among the trees so that each receiver is exactly on one core tree at a time. Multicast senders forward their data streams to each core to deliver to the respective receiver partition. Zappala et al [ZFL02] also propose *Members-to-Many,* which consists of multiple core trees each spanning the entire receiver group with the idea that various senders in the multicast group utilize different trees with respect to the proximity of the source to the tree to balance the traffic load and improve performance. Members-to-Many corresponds to multiple single-core trees to serve the same multicast group. The distinct trees are maintained independently in multiple-tree architectures and the cores need not coordinate with one another for their operations unlike in OCBT. All multi-core schemes operate similarly to CBT in protocol specifics other than their tree structures and maintaining these structures.

Multi-core trees extend the single-core approach to trees of higher access points (cores) to non-group nodes, while maintaining the advantages of single-core based trees. As well, multi-core trees have better fault tolerance in case of a core failure compared to single-core trees. Note that the core is the single point of failure in a single-core based tree. A back-up core list is maintained within the bootstrap mechanism [EH+97] in both CBT and PIM. Still, the core replacement mechanisms in single-core based protocols incur high delays, as the multicast tree must be rebuilt around a new core. Since different senders are likely to multicast via different cores, it is less likely that the entire traffic will be concentrated around one particular core in the multi-core approach. In single-core trees, traffic concentration is high around the core especially in case of multiple senders in the group. In multi-core architectures, the traffic concentration decreases as the number of cores increase especially when the cores are randomly distributed, since the traffic is spread around the domain on different cores. [CZD95, ZFL02]

Zappala, Fabbri and Lo's results [ZFL02] indicate that multi-core trees with effective sub-optimal core placement mechanisms result in less delay than single-core based trees with optimum placement of the unique core, since the multi-core case allows trees which are likely to be distributed adequately to be closer to a group of sender(s) than a single core. Similarly, multi-core trees incur less total cost of delivering the multicast stream in terms of bandwidth utilization than a single core tree since a new node to join the tree is likely to locate a core among the alternatives, which is potentially closer to it than the only core alternative which is the tree root in the single-core based tree.

3. Core Selection Algorithms

Core selection is the problem of selecting the placement of a core or cores in the domain for the purpose of improving the performance of the tree(s) constructed around these core(s) and thereby the performance of the multicast routing protocol according to the predetermined metrics. In this section, we analyze core selection algorithms in the literature and classify them with respect to certain characteristics based on their deployment. In the following subsection, we provide notations and definitions In Section 3.2, we present our classification scheme. We review the core selection algorithms in Section 3.3. Following this review, in Section 3.4, we analyze computational and message complexity of each algorithm. Section 3.5 includes further discussion on core selection methods.

3.1 Preliminaries

Before we proceed, we review in this section the terminology and preliminary research related to core selection. Wall's study [W80] on single core selection problem for broadcasting is precedent in this line of research in the literature. The terminology we introduce is parallel to his, and adequately covers the generic terminology we use throughout our review since Wall's study in context of broadcasting outlined this line of research, and is the foundation of early contributions to core selection for multicasting.

We denote minimum delay distance of any given node pairs u, v in the domain by $d(u,v)$. For a shortest path tree T_c rooted at node c, $D(s,T_c)$ denotes the maximum of all distances between s and the group members along the tree T_c. According to this definition, $D(s,T_c) = D(c,T_c) + d(s,c)$ where $D(c,T_c)$ is the maximum delay of the tree T_c. Similarly, the average delay of the tree, $A(s,T_s)$ is the average of all delays between the source and the receivers. $A(s,T_c)$ in turn is the average of all delays between the source s and each receiver in the group throughout the tree T_c via the tree root c. *Diameter* of a tree is the longest path on the tree between any two nodes. *D-centered* and *A-Centered* trees respectively represent the trees of which $D(s,T_s)$ and $A(s,T_s)$ are minimum among all such possible trees. A *diameter-centered* tree is the tree with the smallest possible diameter. *MaxD*, *MaxA*, *AveD*, *AveA* respectively denote the maximum delay, maximum of average delays, average delay and average of average delays, which are obtained by considering each node separately as source for its respective shortest path tree, and taking the maximum or the average values across these results. The computations of *MaxA* [W80] and *AveA* [JLK78] are NP-complete while the other values can be computed in polynomial time.

Wall's [W80] findings are theoretical on the delay bounds of the trees rooted at the selected cores compared to the optimal shortest path trees. His results apply for multicasting with certain restrictions. The selection criteria he studied, applying directly to the problem, is adapted for latter solutions in multicasting environment. He considers delay as the single metric in various measures for core placement for the development of a shortest path tree to broadcast to all nodes in the domain. His findings are delay bounds on maximum, average and diameter-centered trees compared to the optimum shortest path tree.

Wall [W80] suggested *D*-center, *A*-center and diameter-centered trees for core node selection based on the use of local/minimal information and computational complexity of the mechanism. Maximum and average delays can be computed by using the local unicast routing information on each router to result in *D*-centered and *A*-centered core selection in the domain. The computation of a diameter-centered tree is somewhat more complex, as the diameter of a tree does not always pass through the root. However, Wall presents a polynomial-time algorithm that runs on the tree root using the unicast routing information to return the diameter value if the diameter is within certain topological proximity to the root [W80].

$D(s,T_c) \leq 2D(s,T_s)$ when T_c is diameter-centered or *D*-centered
$D(s,T_c) \leq 3D(s,T_s)$ when T_c is any other SPT.
$A(s,T_c) \leq 3A(s,T_s)$ when T_c is *A*-centered.

$MaxD(T_c) \leq 2MaxD(T_s)$ when T_c is any SPT.
$MaxA(T_c) \leq 3MaxA(T_s)$ when T_c is any SPT.
$AveD(T_c) \leq 2AveD(T_s)$ when T_c is diameter-centered or *D*-centered.
$AveD(T_c) \leq 3AveD(T_s)$ when T_c is any other SPT.
$AveA(T_c) \leq 2AveA(T_s)$ when T_c is *A*-centered.

Table 1. Delay bounds on *D*-centered, *A*-centered and diameter-centered trees compared to the optimal SPT. The bounds are tight. The missing bounds indicate that the measure can be arbitrarily bad compared to the optimum value.

Let T_c be the SPT rooted at node c for a host group. Let s be any sender node and T_s be the SPT rooted at s for the same group. We present in Table 1 Wall's findings on delay bounds. According to the first of these, for example, the maximum delay between the source and a receiver can be as high as twice the maximum delay of the source-rooted tree when the source is multicasting via a core-based tree of which the core is selected to yield a diameter-centered or *D*-centered tree.

However, Wall's [W80] findings are for broadcast environments and their applicability for multicasting are limited with the necessity of considering the sources and all candidate cores as on-tree alongside the host group members in the calculations for core selection. With this in mind, Wall's findings suggest that although *A*-centered trees may have higher upper-bounds on certain measures, they are more consistent to the average performance than the other types of trees.

3.2 Classification of Core Selection Algorithms

Before proceeding with the review of core-selection schemes, we present a classification for their deployment and structural and operational characteristics (Figure 3).We mainly distinguish core selection algorithms as *single-core* versus *multi-core* for this is the dominant determinant of the complexity of the scheme. Parallel to the emergence of single-core based shared trees before multi-core based architectures, research on single-core selection precede multi-core selection in literature.

Algorithm	***M***	***C***	***A***	***D***	***Metrics***
maxDist [TR97]			+	+	hop-count
aveDist [TR97]			+	+	hop-count
Diam [TR97]			+	+	hop-count
estCost [TR97]			+	+	hop-count
Topology-based [CZD95]			+	+	hop-count
Group-based/Center [CZD95]			+	+	hop-count
Group-based/WA [CZD95]			+	+	delay & cost
Tournament [SBK94]				+	Cost
MinMaxD [Sa96]		+		+	Delay
INITIAL [Sa96]		+		+	Delay
n-dominating set [ZFL02]	+		+		hop-count
k-center [ZFL02]	+		+		hop-count
GREEDY [Sa96]	+	+			Delay
NAÏVE [Sa96]	+	+		+	Delay
NAÏVE/MinMaxD [Sa96]	+	+		+	Delay
NAÏVE/AVGD [Sa96]	+	+		+	Delay
optTree [KH03]	+	+			hop-count & delay

Table 2. A classification of core selection algorithms. M: multi-core (vs. single-core), C: constrained (vs. unconstrained), A: asymmetric (vs. symmetric), D: distributed (vs. centralized). *MinMaxD* and *INITIAL* [Sa96] have additional assumption that all members in the multicast group are both sources and receivers.

The underlying platform assumed in all core selection algorithms is the distance-vector routing environment in which restricted domain information, namely the distances from the router itself to each other domain router are available to a particular router. In our context, we assume the availability of this information in every metric. According to this, the state information of a given router includes the shortest delay, cost and hop-count distances from itself separately to every other domain router along with the outgoing links to lead to the shortest-distance path.

The extent of the use of local information on domain nodes for the operation of the algorithm determines whether the algorithm is centralized or distributed. A *distributed* algorithm is one that operates on multiple nodes using the local information on each node. The solution is obtained as the nodes are traversed, or as the performance results obtained locally from each candidate are gathered at one node by message transmission. In a *centralized* algorithm, the entire processing is carried at one node, and the information required for the process is forwarded to it. We classify an algorithm as centralized if its deployment to operate on one router is more feasible than its distributed implementation, distributed otherwise. Once again, the assumed state information available to all routers in the domain is the distance-vector environment—the distances in each metric from the node that the state table resides on to every other router. As we analyze the algorithms for their computational structure in Section 3.4, we further distinguish a group of distributed algorithms for their operational characteristics as "Compute-and-Report" *Class-CR* algorithms.

We denote a core-selection algorithm as *constrained* if the algorithm considers a given delay-bound for the group during the selection process, and the resulting core set is such that there exists a path between each source-receiver pair in the group which passes through at least one core without violating the delay-bound.

An *asymmetric* network is one in which the link costs in all metrics are not necessarily equal in both directions. It is clear that the algorithmic complexity reduces when the network is symmetric especially in distributed processing. This is because in the assumed distance-vector setting, the information available to each node represents the distance information not only *from* the node itself to every other node in the domain, but also the distances *to* it. Asymmetric algorithms, however, operating on a directed network, have wider applicability.

As we see in Table 2, the set of single-core algorithms overlap with the set of distributed algorithms for the simplicity of their computations in one process operating. The constrained algorithms studied so far are restricted to symmetric networks for considerable increase in the effectiveness of these algorithms upon this restriction at the cost of their applicability also in asymmetric networks.

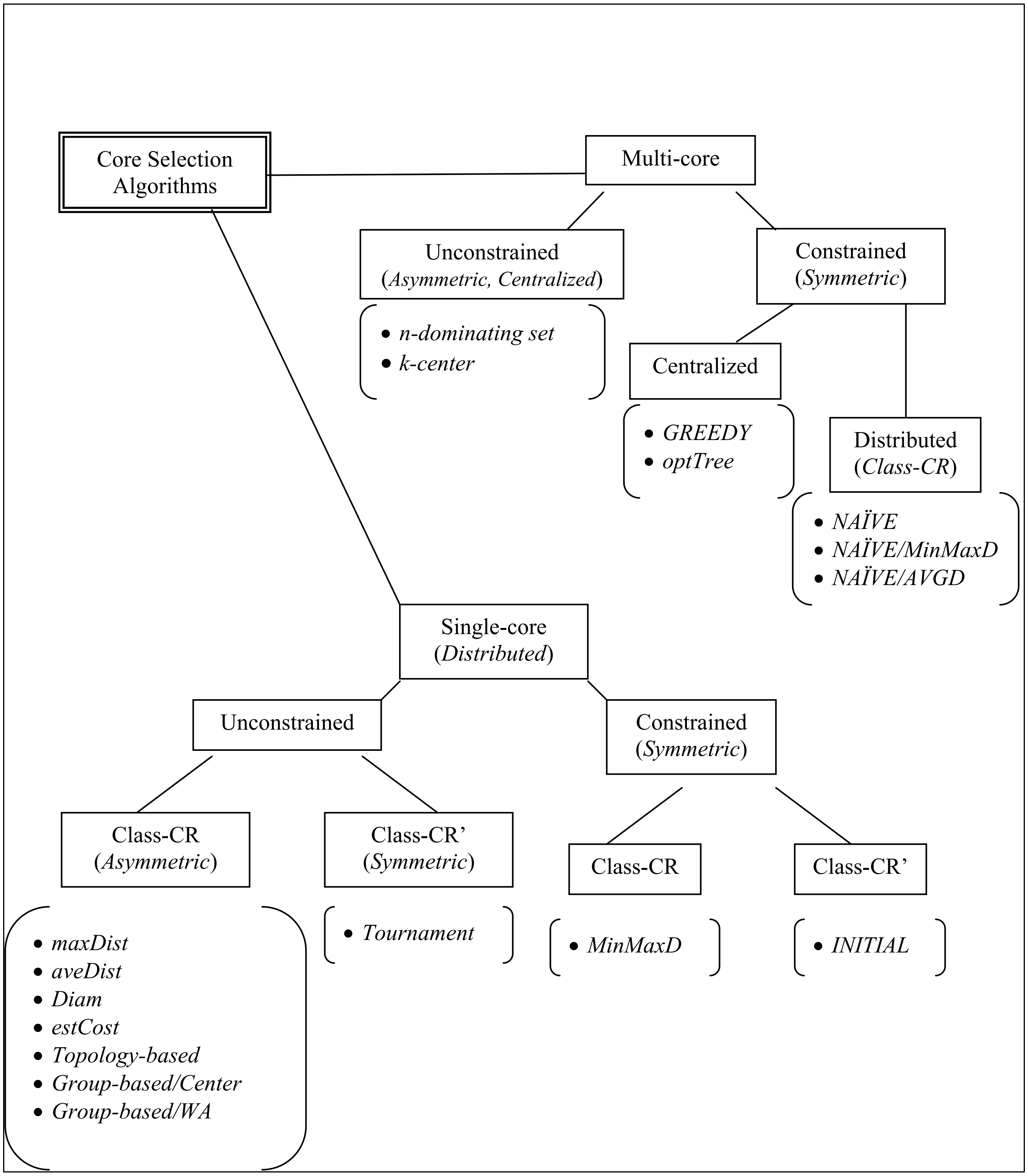

Figure 3. Classification of core selection algorithms. Class-CR' denotes those distributed algorithms which are not in Class-CR.

3.3 Review of Core Selection Algorithms

In this section, we review the core selection schemes according to the selection criterion they apply and their performance. Our analysis in Section 3.4 completes the characterization of these algorithms for their computational properties.

3.3.1 Single-Core Selection Algorithms

As previously stated, earlier protocols applied a single-core scheme for the ease of construction and maintenance of a loop-free multicast path. The relative simplicity of the selection of a single core to serve the entire multicast group allows feasible deployment of these algorithms on local information and moderate message exchange among the participant nodes.

3.3.1.1 Single-Core Selection Algorithms / Unconstrained

Thaler and Ravishankar [TR97] adapt Wall's [W80] broadcast single core selection criteria to multicasting to a subset of nodes in the domain. An additional selection criterion, estimated cost, is also introduced. The estimated cost is the average of minimum and maximum possible tree costs computed. The applicability of the results is restricted to the case where all the link costs in the network are assumed to be equal. However, the core selection algorithms in [TR97] apply for asymmetric networks for they rely on local information on the nodes during the selection process. The criteria considered are:

1.) _maxDist_: maximum distance from the core to any multicast group member. Corresponds to *D*-centered trees described by Wall [W80] for broadcasting environment.
2.) _aveDist_: average distance from the core to all multicast group members. Corresponds to Wall's *A*-centered trees.
3.) _Diam_: diameter of the tree rooted at the core defined as the sum of the distances of the two farthest away members in the group from the core. Note that this definition does not describe the exact diameter of the tree rooted at the core minimizing this measure since the diameter of a shortest path tree does not necessarily pass through the root [W80].
4.) _estCost_: the average of the minimum possible and maximum possible cost of the tree rooted at the candidate node being tested. The minimum cost is estimated to be the sum of the maximum destination cost and the number of duplicate distance nodes in the multicast group, on the assumption that the minimum cost tree is of a chain structure, with the exception of the group members that are of equal distance to the core with one another, adding minimum of one more link each to the tree. The other extreme, tree of highest cost, is of star structure with all multicast group members directly connected to the core. The exception arises when the degree of core is less than the number of multicast group members, in which case multiple group members share the links outgoing from core. The minimum estimated cost, $estCost_{min}$, and the maximum estimated cost, $estCost_{max}$, are given by:

$$estCost_{min} = \max_{u \in S} d(c,u) + (\text{ number of duplicate distance nodes in } S\text{ })$$

$$estCost_{max} = \begin{cases} \sum_{u \in S} d(c,u) & \text{if } |S| \leq degree(c) \\ \sum_{u \in S} d(c,u) - (|S| - \deg ree(c)) & \text{otherwise} \end{cases}$$

$$estCost = (estCost_{min} + estCost_{min})/2$$

where *c* is the candidate core, and *S* is the multicast group being considered for evaluation.

Results indicate that *estCost* and *aveDist* return the best results in terms of the tree cost measured as total number of links on the tree. *estCost*, although not a close approximation of the tree cost, returns trees with lowest costs as the only measure testing directly this metric. For maximum source-to-destination delay, maximum delay and maximum diameter performed higher. Note here that, maximum source-to-destination delay is the maximum of minimum possible delays between any two source and destination pair where maximum distance is the delay distance between the root and an on-tree node. As expected, Maximum diameter, minimizing the distance between any two nodes on the tree, approximat the tree more accurately for the maximum source-to-destination delay.

Calvert, Zegura and Donahoo [CZD95] propose the use of topological center of the domain and the subgraph induced by the multicast group for placement of a single core to serve the group without the consideration of a delay constraint. The core selection methods are tested across different application scenarios. In each of these scenarios, the multicast group is distributed widely throughout the network. The following core choice techniques are tested:

1.) _Topology-based_: This group includes the core choice methods with respect to the proximity of the core to the topological center of the network. Let a Tol-*t* node be a node at which the distance is no more than *t* hops away from the topological center of the domain, where a Tol-*0* node is at the center of the domain. Results indicate that the placement of the core close to domain center improves both delay and total bandwidth consumption compared to average-core choice case, which is obtained by taking each node in the domain as the core, testing the performance of the tree for each case and taking the average of all performances. Performance of topology-based core selection demonstrated steady improvement in delay and very small improvement in bandwidth consumption as the tolerance (Tol value) is reduced.

2.) _Group-based_: This includes the following schemes for core selection among the receivers:

a.) _Center_: according to the proximity to the topological center of the receivers group,

b.) _WA_: weighted average of bandwidth and delay.

Results indicated that group-based core placement is effective only when the multicast group is localized in a topological area, and the performance can be reduced compared to the topology-based core choice if the participants are widely distributed.

Calvert et al [CZD95] provide the performance measurement for multicast data stream flow excluding the core selection and tree construction overhead. Thus their study excludes the specifics of the implementation of the process. A node is a center of a graph if its maximum hop-distance to any other node in the graph is minimum possible across all nodes in the graph. Consider the function $D'(u,T_v)$, a modification of the function $D(u,T_v)$ to operate on hop-distance rather than delay-distance between the domain nodes in case of broadcasting. Then, a node u is a graph center if $D'(u,T_u) = \min_{v \in V} D'(v,T_v)$ where V is the set of all nodes in the domain — corresponding to the optimal core to lead to a D-centered tree with unit link weights in broadcasting to entire domain. Considering the hop-distance to the domain nodes are available to a node itself, topological center of the graph can be computed in a distributed manner by local computation of $D(u,T_u)$ for each node and comparison of the accumulated results of this computation for the minimum value to yield the center. Following a similar argument, the core choice schemes proposed by Calvert et al [CZD95] suggest distributed deployment for the efficiency of the implementation. The computations in all cases are based on local information available on each node, namely the distance (according to various metrics) of the node itself to the other domain nodes. Note that the schemes proposed in [CZD95] are applicable on asymmetric networks.

Shukla et al [SBK94] propose *Tournament*, a core selection algorithm based on the single metric which is the cost distance of the multicast group members to one another in a symmetric network. The selection process is a tournament at each of which phase a candidate core selected with respect to the locations of two candidates of the previous phase is a participant in the next level of tournament.

Initially, the multicast group members are paired so that a sender is matched with a receiver before the remaining members are matched. During the initial matching, the distance of the nodes to one another is considered so that the nodes farthest apart are paired first, and the process is repeated until all nodes are paired. At each step, nodes that are not matched due to an odd number of candidates are elected for the next level. The matching process is coordinated by one of the nodes in the domain designated as the “scheduler”. The distance between the node pairs is compared by one of the nodes leading the pair, and the node closest to the midpoint of the shortest path between the two nodes is selected for the next level. The process terminates when a single core remains at the end of the tournament. At each phase, the “winners” of pairs are reported to the scheduler node where they are paired for the next phase. Tournament is a distributed algorithm, processing at the scheduler for the matching process and at the node pairs for the election between the pairs. Shukla et al [SBK94] propose the distance information between the winners of the second and subsequent phases be ignored during the matching process for the knowledge of entire winners is not available to a winner at these phases and the distance to each other winner can not be communicated to the scheduler. The communication of the winner set to each member in this set, and the transmission of relative distances information between winner pairs back to the scheduler under the coordination of scheduler, however, can be considered to improve performance of the algorithm at the cost of increased message transmission still keeping the operation distributed.

The algorithm is likely to locate a core well-centered to the member set for operating on “midpoints” between distant pairs rather than considering equally the cost distances of all members throughout the entire process. [SBK94]’s studies comparing the performance of the tree rooted at the core selected by the algorithm indicate improvement in the tree cost in case of single and multiple sources in the group, with the difference in this cost gain diminishing as the number of sources increase and the network is highly connected.

3.3.1.2 Single-Core Selection Algorithms / Constrained

Salama [Sa96] studied two single-core selection algorithms for delay-constrained shared trees in symmetric networks. Salama's solutions are restricted to the case in which all receivers are also sources. A typical example of this case is a video-conference application, where all participants can communicate with every other member in the group both ways. Salama tested the following delay-constrained single-core selection methods:

1.) *MinMaxD*: the node that has minimum of maximum delay distance to any multicast group member. This corresponds to the delay-constrained extension of *MaxDist* proposed in [TR97].
2.) *INITIAL*: A random multicast group member is selected to initiate the core selection process. The selected member locates the most distant group member from itself, and constructs a delay-constrained path in between. The node closest to the center of this path is selected to be the core. *INITIAL* applies the main intuition behind the *Tournament* selection algorithm [SBK94] to a lesser extend, considering the delay constraint between only two distant members of the multicast group. *INITIAL*, however, does not guarantee success to locate a core that would result in a tree spanning all multicast group members within the delay bounds, even if such a core existed.

The delay bound to be fed as input to the delay-constrained tree construction algorithm varies depending on which method is used for the selection of the core. Note that with Salama's assumption that all multicast group members are both sources and receivers, the delay-constrained tree construction problem transforms to the construction of a "diameter-constrained" tree, in which the longest on-tree path passing through the core is within the given bound. With this, Salama suggests that the delay bound, Δ', used for *MinMaxD* is at most as great as $\Delta/2$ where Δ is the actual delay bound for the group to ensure that an on-tree path crossing the core between a source-destination pair does not exceed the actual constraint. For *INITIAL*, the on-tree delay constraint value to be considered, however, differs in that the delay from the core to one of the two members on the initially constructed path can exceed half the value of actual value. The on-tree delay constraint value for this latter case, therefore, is computed as $\Delta'=min(\Delta/2, \Delta-max(d_1,d_2))$ where d_1 and d_2 are the delays on paths between the core and the initial multicast group members joining the tree to select the core.

Both selection criteria are based on a single metric — delay. Cost of the trees in both cases improved closer to the cost of the optimal tree as the number of group members increased, verifying the fact that the cost-performance of the tree is higher in dense mode. *INITIAL* outperformed *MinMaxD* in terms of cost especially when the delay bound is tighter. Note that, this result is biased by the fact that the tree cost increase is based on successfully generated trees satisfying the delay bounds by the use of a polynomial-time tree construction algorithm, which lacks well-approximation of cost minimization. Performance results indicate that the difference in costs of the trees generated in *MinMaxD* and *INITIAL* remained within 15% of the cost of the optimal tree, with this difference diminishing as the number of group members increase.

3.3.2 Multi-Core Selection Algorithms

Multi-core algorithms have the common property that their feasibility is eminent in centralized implementations using extensive domain knowledge compared to the local state information on distance-vector protocol routers.

3.3.2.1 Multi-Core Selection Algorithms / Unconstrained

Zappala, Fabbri and Lo [ZFL02] tested two multi-core selection algorithms against average core choice in *Senders-to-Many* and *Members-to-Many* protocol settings:

1.) *n-dominating set*: locates a minimal dominating set as the set of cores, which is a subset of nodes in the domain so that all nodes in the multicast group are within *n*-hops of this set for a given *n*.
2.) *k-center*: places a given number of *k* cores in the domain so that the hop-distance from all nodes in the multicast group to the cores is minimum.

Note that neither the dominating set nor *k*-center algorithms has polynomial-time solutions [GJ00]. Zappala et al [ZFL02] also use the greedy dominating set algorithm with modifications for *k*-center heuristics. A node *r* is said to be dominated by a core *c* if *n* is within at most *n*-hops distance of *c*. We denote by *domination set* of a given node *v* the set of nodes that are dominated by *v*. Dominating set, on the other hand, refers to a set of nodes so that the set of receivers is a subset of the union of domination sets of the members of the dominating set. The algorithm iterates to select the node in domain that currently dominates the maximum number of un-dominated members until all the group members are dominated. The dominating set heuristics is further parameterized with a constant *k* to specify the exact number of cores to be selected for a solution to *k*-center problem. It should be noted that Zappala et al [ZFL02] were not specific on the algorithmic details of *k*-center heuristics.

Zappala, Fabbri and Lo [ZFL02] tested each of these algorithms comparing their performance to random core selection. Both *k*-center and dominating set algorithms demonstrated modest improvement in delay compared to random core selection. *k*-center core placement with multiple cores resulted in lower delays than single-core tree with optimal core placement as the value of *k* increased. Zappala et al's results suggest that polynomial-time heuristics for dominating set, and especially *k*-center core placement contribute to the efficiency of multi-core protocols.

Both *k*-center and dominating set algorithms refer to hop-count as the single metric for core selection. We assume, due to the complexity of especially the *k*-center algorithm for potential heuristics, that both algorithms are implemented as centralized, since the transmission of relevant domain information to the operating node in centralized deployment is likely to match for the advantages of the distributed implementation which involves excessive message exchange among the participant nodes for the same level of approximation of the optimum result as a centralized algorithm provides. Hop-count is an asymmetric metric for it describes equal link weights in both directions by definition, and hence *k*-center and *n*-dominating set algorithms operating on hop-count classify as asymmetric.

3.3.2.2 Multi-Core Selection Algorithms / Constrained

Salama [Sa96] also studied the problem of multiple core selection for Zappala, Fabbri and Lo's [ZFL02] Senders-to-Many architecture for delay-constrained environment. Observe that, for delay-constrained case, the additional motivation for multi-core as opposed to single-core scheme is that multiple cores extend the potential solutions to delay-bound paths between source-receiver pairs passing through a core where a single core could fail to serve the entire group in this manner without violating the delay constraint. Salama's multi-core selection algorithms assume a symmetric network:

1.) *<u>GREEDY</u>*: This algorithm uses a variation of the solution for the dominating set problem in which the domination criterion is the delay bound rather than the hop-count. Note that the problem remains NP-complete in the modified version [Sa96]. *GREEDY* assumes as input, for each node in the domain, the set of receiver nodes that the node is dominating for the delay bound for all sources in the group. The algorithm merely iterates to select the node that dominates the maximum number of un-dominated receivers as the next core until all receivers are dominated. *GREEDY* operates on full domain information and is centralized.

2.) *<u>NAÏVE</u>*: *NAÏVE* applies iterative selection of the core set as the tree is constructed. Initially the selected core set is empty. When a new receiver attempts to join the multicast path, the algorithm checks iteratively the members of the core set to see whether there already exists a core so that the paths from each source to the new receiver via the core is within the delay bound. If the search fails, then the candidate core set is searched for a core to lead to a feasible solution. Salama suggests as the candidate core set either the set of receivers, or an ordered list of a set of nodes with respect to the maximum (*<u>NAÏVE-MinMaxD</u>*), or average (*<u>NAÏVE-AVGD</u>*) delay of the candidate to the multicast group members. When the receiver set constitutes the candidate cores, the next core to be selected is always the new receiver to join itself in case of inexistence of a feasible core among the selected ones. In all versions, *NAÏVE* processes for at most as many cores as the number of sources in the group be selected, and terminates without providing a solution with the idea that source-based trees would be more feasible when the number of cores and hence number of distinct trees exceed the number of source-based trees. *NAÏVE* and its variations operate on the minimum delay distances between source-candidate core and receiver-candidate core pairs which is maintained in the routing state database of these nodes. Thus, *NAÏVE* can be implemented for distributed processing alternatively with the transmission of the delay-routing state information of the group nodes and candidate cores to a coordinator node in the domain to execute the algorithm, or with the transmission of group information to the candidate cores where each candidate evaluates itself against the group members and returns its result back to the coordinator for comparison with the other candidates. We will be comparing commenting these two implementations in Section 3.4 during our complexity analysis.

Both *GREEDY* and *NAÏVE* return a feasible solution whenever such a solution exists, provided that the candidate core set in *NAÏVE* does not exclude some subset that can actually lead to a feasible solution. Both algorithms lack the consideration of the cost within the core selection criterion. In terms of the number of cores generated, *GREEDY*, considering the delay-domination of the group members directly throughout the algorithm, performed very close to the optimal solution. *NAÏVE-MinMaxD* and *NAÏVE-AVGD* underperformed *GREEDY*, generating similar results still close to optimal.

Karaman and Hassanein [KH03] extend *GREEDY* to consider an "optimization criterion" in addition to the "domination criterion", which still is the given delay-bound between any source-destination pair in the group. The algorithm, which we call *optTree*, generates a set of cores each associated with a domination set — the receiver partition including those receivers dominated by that core for all sources in the group. The assumed design problem for *optTree* is the delay-constrained minimization of total number of links on the resulting set of paths for the multicast group for optimization of tree structure on total link count. During its iteration to extend the selected core set until all receivers are dominated, *optTree* updates the existing domination sets according to the optimization criterion to better approximate the optimum set of multicast paths to be constructed. According to this, each receiver dominated by the new core and in the domination set of the already selected core is assigned to the domination set of that core ranking higher with respect to the optimization criterion. The following are considered for optimization criteria:

1.) *Node-count*: highest number of nodes that the node is dominating
2.) *Delay-residue*: highest average delay-residue from the core to the receivers in its domination set which are not yet dominated by a selected core
3.) *aveHop*: highest average hop-count from the core to the receivers in its domination set which are not yet dominated by a selected core
4.) *Degree*: highest node degree

The number of nodes that are not currently dominated by any of the selected cores and dominated by the candidate core, namely item (1) above is the criterion used by Salama, and *optTree/Node-count* turns into the greedy algorithm in [Sa96] when it is used as the entire criterion. Criteria (2) and (4) directly consider the continuous changes in the tree structure due to membership dynamics, with (2) leaving more residue for the delay-bound for potential extensions of the existing paths for new members, and (4) selecting the node of higher degree. We consider these two as additional criteria. The farther the link-distance to the members dominated, the more the paths connecting the core to multiple members are combined on the tree. On the other hand, a core local to a set of receivers is likely to lead to a core-tree of a simpler structure possibly at the cost of extending the corresponding path(s) on the source tree(s). Item (3) is open to be tested for the performance. *optTree* is proposed for multi-point communication groups with multiple sources under dynamic group memberships.

3.4 Complexity Analysis

In this section, we analyze the computational and message complexity of the selection criteria at each node. We consider as the core selection process the sequence of events that begins after the triggering of participant nodes to mark the beginning of the process, and ends with the set of core(s) selected among the candidates. We exclude from this cycle the announcement of the selected core(s) to the domain. As we proceed in this section, we will be observing that for some of the algorithms, the candidate core set—the pool of nodes among which the core-choice is to be made—can be restricted externally to the selection algorithm without affecting the modular operation of the algorithm. In such cases, we assume as input to the core selection process the set of candidate cores, and exclude the process of restricting this set from core selection. For distributed algorithms, we generically name the node designated for the gathering of local results from the participant nodes, and processing on this compiled information, where necessary as the *coordinator* node. In the case of *Tournament* it is the "scheduler" [SBK94]. In the centralized case, we use the term "coordinator" to represent the only processing node in the domain. We assume that the coordinator node is already selected and known in the domain.

Algorithm	***Local***	***Local-nodes***	***Coordinator***	***Message***
maxDist [TR97] (*)	$O(\|M\|)$	C; $[\|C\|]$	$O(\|C\|)$	$O(\|C\|)$
aveDist [TR97] (*)	$O(\|M\|)$	C; $[\|C\|]$	$O(\|C\|)$	$O(\|C\|)$
Diam [TR97] (*)	$O(\|M\|)$	C; $[\|C\|]$	$O(\|C\|)$	$O(\|C\|)$
estCost [TR97] (*)	$O(\|M\|)$	C; $[\|C\|]$	$O(\|C\|)$	$O(\|C\|)$
Topology-based [CZD95] (*)	$O(\|V\|)$	V; $[\|V\|]$	$O(\|V\|)$	$O(\|V\|)$
Group-based/Center [CZD95] (*)	$O(\|M\|)$	M; $[\|M\|]$	$O(\|M\|)$	$O(\|M\|)$
Group-based/WA [CZD95] (*)	$O(\|M\|)$	M; $[\|M\|]$	$O(\|M\|)$	$O(\|M\|)$
Tournament [SBK94]	$O(\|V\|)$	V; $[\|M\|-1]$	$O(\|M\|^2)$	$O(\|V\|\|M\|)$
MinMaxD [Sa96] (*)	$O(\|M\|)$	C; $[\|C\|]$	$O(\|C\|)$	$O(\|C\|)$
INITIAL [Sa96]	$O(\|V\|)$	M; [1]	$O(1)$	$O(\|V\|)$
n-dominating set [ZFL02]	-	-	$O(\|M\|^2\|C\|)$	-
k-center [ZFL02]	-	-	$O(\|M\|^2\|C\|h_{diam})$	-
GREEDY [Sa96]	-	-	$O(\|M\|^2\|C\|)$	-
NAÏVE [Sa96]	$O(\|M^2\|)$	M; $[\|M\|^2]$	$O(\|M\|^2)$	$O(\|M\|^2)$
NAÏVE/MinMaxD [Sa96]	$O(\|M\|^2)$	C; $[\|M\|\|C\|]$	$O(\|M\|\|C\|)$	$O(\|M\|\|C\|)$
NAÏVE/AVGD [Sa96]	$O(\|M\|^2)$	C; $[\|M\|\|C\|]$	$O(\|M\|\|C\|)$	$O(\|M\|\|C\|)$
optTree [KH03]	-	-	$O(\|M\|^3\|C\|)$	-

Table 3. Computational and message exchange complexity of core selection algorithms. M, C and V denote respectively the sets of multicast group members, candidate cores and the entire nodes in the domain. h_{diam} is the maximum hop-distance between any two domain nodes. Class-CR algorithms are indicated with an asterix.

Table 3 summarizes our complexity analysis of the algorithms in Section 3.3. The entry of column "Coordinator" is the computational complexity of the process operating on the coordinator node. For distributed algorithms, we also analyze the proportional growth of message exchange throughout the entire algorithm and present our results in respective columns of the table. "Local" indicates the computational complexity of the distributed algorithm processing locally at each node, and "Local-nodes" specify the minimal set of nodes which is a superset of the set of domain routers that operate on local information. We also provide in "Local-nodes" in square-brackets the total number of times the local process executes throughout the entire algorithm. Note that, in the centralized case, only the "Coordinator" column of the table applies and the rest of the row entries are nil. The assumed state information available on a given router in all cases consists of the distance information from the router itself to every other router in domain in each relevant metric. For centralized algorithms, we further comment in Section 3.4.1 on the amount of message transmission to communicate the required information to the coordinator under the availability of restricted domain knowledge as assumed.

We note that a number of a distributed core selection algorithms have similar computational characteristics to locally compute to evaluate the candidate cores and report the results to coordinator for comparison. We group these in a class and denote it as the class of "*Compute-and-Report*", *Class-CR*. In Class-CR algorithms, each candidate node measures itself for the selection criterion based on the local information and reports the result to the coordinator. The coordinator compiles the node results from the candidate cores and compares them to select the best scoring node as the core. All selection algorithms proposed by Thaler and Ravishankar [TR97] and Calvert et al [CZD95], and *MinMaxD* single core selection algorithm [Sa96] are in this class. Message exchange involves the transmission of the group members information from the coordinator to the candidate cores, and the node-results back to the coordinator with the exception of *Topology-based* core choice [CZD95] in which each node testing itself against the values for all domain nodes, the communication of group information to the candidate cores is eliminated. Therefore $O(|C|)$ messages are transmitted for the selection of a particular core in Class-CR algorithms, where C is the candidate core set. Selection of the best core among the candidates on the coordinator node is done by iterating over individual node values with constant-time computation at each iteration, and requires linear time in the size of candidate core set provided that the node results are transmitted to coordinator within a given time-out period. The time-complexity of computation on coordinator is the same as the bound on number of message exchange for all Class-CR schemes.

Local computations of Class-CR algorithms iterate for each node in the multicast group to calculate the node measure on the selection criteria which, in each case, requires to look up the local state table followed by the calculation and comparison with current optimum in constant time. Thus, local computations in Class-CR algorithms take $O(|M|)$ time where M is the set of multicast group members. Specifically, observe that *MinMaxD*, selecting the among the candidates according to the same criterion as *MaxDist* unless no node among the candidates satisfying the delay constraint exists is computationally same as *MinMax* with the resulting core additionally tested for in constant time to satisfy the delay bound. Similarly, *Topology-based* core choice operates the same algorithm as *MaxDist* with the difference that the "multicast group" this time is the entire set of domain nodes, and the link weights are unit for hop-count metric. For *Group-based/center* choice of a core, a similar process takes place, but this time on the modified graph of which the nodes are the receivers and weight of each link connecting a pair of receivers is the total hop-count of the minimum hop-distance path connecting the same two receivers in the original graph. Note that *Topology-based* core choice, operating solely on the domain topology stands out among all others in that it requires only static processing irrespective of the varying multicast groups active in the domain.

If we consider *NAÏVE* operating in a relatively static network environment where routing state updates are unlikely during the multicast session, one-time transmission of multicast group and candidate core delay-routing state tables to the coordinator and maintenance of this information at the coordinator allows local operation for the selection of core set. However, considering the dynamisms of network environment, we assume the core selection algorithm executing on up-to-date network state information for the selection of each core. According to this, we assume the distributed design of the algorithm so that, when a new group member request to join, the coordinator transmits the identification of this node to the candidate cores. Each candidate operates on its routing state database to test itself for the evaluation criterion depending on the *NAÏVE* version, and returns its result back to the coordinator. Thus, at each iteration of *NAÏVE* for the selection of another core, the algorithm operates as a Class-CR algorithm. The message transmission involves, as in other Class-CR algorithms, the transmission of group information to the candidates and the node results back to the coordinator and hence is bounded by $O(|C|)$. For the selection of a single core, the algorithm operates in $O(|C|)$ time at the coordinator for message transmission and comparison of node results. During a local execution, each candidate core processes on its distances to the group members and thus the complexity of computation during a core selection is $O(|M|)$. *NAÏVE* iterates this cycle at most as many times as the number of group members. Therefore, the message complexity, and the complexity of local computations at the coordinator and the candidate nodes are multiplied by $|M|$. The local processes execute at most as many times as $|C||M|$ where the operating local-node set is M in *NAÏVE*, and C in *NAÏVE*/variations.

Tournament suffers from heavy message exchange. Each level of tournament is the sequence of following steps:

i.) pairing the multicast group members at coordinator
ii.) triggering the pair leaders
iii.) each pair finding their midpoint
iv.) each midpoint reported back to the scheduler.

The random pairing process in step (*i*) is linear in time to the number of candidates at each level. During the first level of tournament, the process involves the comparison of distances between the node pairs for the matching process, and takes square time in the group size, i.e., $O(|M|^2)$. Triggering the pair leader requires transmission of pair knowledge to it from the coordinator. The number of messages transmitted in step (*ii*) is half the number of candidates at current tournament level and hence $O(|M|)$. The process of locating the node closest in cost-distance to the midpoint of a pair in step (*iii*) merely requires the knowledge of nodes that lie on the shortest path connecting the pair available to the pair leader, requiring transmission of $O(|V|)$ messages and comparison of node distances of the intermediary nodes in $O(|V|)$ time at the pair leader. In step (*iv*), as many messages as in part (2) are transmitted back to the coordinator.

Throughout the tournament, the steps (*i*)..(*iv*) are iterated in n levels where $n = \lceil log_2(|M|) \rceil$. Since the matching process at coordinator operates in time $O(|M|^2)$ at the first level and $O(|M|)$ at subsequent levels, the overall computational complexity of *Tournament* at coordinator is, therefore, $O(|M|^2 + |M| \lceil log_2(|M|) \rceil) = O(|M|^2)$. For a multicast group of size $|M|$, the process at local nodes for finding the midpoint of a pair executes $|M|$-1 times throughout the tournament. (The idea for an inductive proof for this statement is as follows: when the multicast group size is an exact power of 2, i.e., when $|M|=2^n$ for any n, the tournament "schedule tree" to indicate the paired nodes and those elected to the further level of the tournament for all levels is a full binary tree, and the number of pairs depicted on it is $|M|$-1. When one member is removed from the set M, one competing pair involving the removed node turns into an unmatched node at the first tournament level due to an odd number of nodes at that level. Similarly, the removal of each further node from M results in the loss of exactly one pair in one level further set of pairs until exactly one half of the binary tree is lost and the multicast group size is $|M|=2^{n-1}$. The number of times the midpoint-computation process is executed is exactly the number of pairs matched throughout the entire tournament, and is exactly one less than the group size with this argument.) As we stated above, each execution of the local process at step (*iii*) requires message transmission of $O(|V|)$, hence the overall number of messages at this step is bound by $O(|V||M|)$. Additional message transmission which take place at steps (*ii*) and (*iv*) in Tournament amount to $O(|M|)$, and total number of messages exchanged throughout the $O(|V||M|)$.

INITIAL operates similar to *Tournament* at a single step in that one randomly selected group member (*i*) determines the member farthest from itself in delay, and (*ii*) locates the domain node closest to the "midpoint" of the shortest-delay path between the two. Using a similar argument as in *Tournament*, steps (*i*) and (*ii*) are executed respectively in $O(|M|)$ and $O(|V|)$, resulting in time complexity of $O(|V|)$ of the algorithm. The number of messages transmitted in step (*ii*) is bound by $O(|V|)$ and constant in step (*i*).

NAÏVE iteratively adds a new core to the set of selected cores when none of the current cores satisfy to serve a new multicast group member within the given delay bound. For each new group member, the existing core set is searched for a satisfactory core. During this search, each core is examined whether to meet the delay bound for each source in the group to transmit the receiver. Assuming knowledge of the farthest source in delay-distance to each selected core is maintained throughout the algorithm, the time complexity for searching the current core set is $O(|C_{select}|)$ where C_{select} is the set of selected cores and is bound by M—the set of multicast group members. If the search in C_{select} fails, then the new receiver itself is designated as a new core provided that it is in reach within the delay constraint from every source in the group—executing in time linear to the number of sources in the group. Therefore, the entire algorithm computes in time $O(|M|^2)$. In case of *NAÏVE/MinMaxD* and *NAÏVE/AVGD*, the candidate core set, which is the set of domain nodes, is ranked respectively for the maximum and average node delays to the group members. This process executes modularly to the generic NAÏVE algorithm with the modification of *NAÏVE* to operate on an input candidate set, and takes $O(|M||C|)$ time. The computational complexity of variations of *NAÏVE*, therefore, is $O(|M|^2+|M||C|) = O(|M||C|)$.

A greedy dominating set heuristic iterates for the following steps until all the multicast group members are dominated by the core set:

i.) select the candidate core, c, ranking highest according to the "optimization criterion"
ii.) add c to the set of selected cores
iii.) update the domination sets.

As mentioned earlier, we refer by *domination set* to a set associated to a candidate core to return the set of currently un-dominated multicast group members. A typical implementation maintains a two dimensional table with the multicast group members and the candidate cores being separately the indices of these dimensions. Note that, in case of delay-constrained implementation of heuristic, the process to generate the domination sets considers the delay-distances separately between source-core and receiver-core pairs in two different cycles without extending the computational complexity of the process. The greedy algorithm needs as input the set of domination sets, which can be constructed in $O(|M||C|)$ time for all candidates. Updating the domination sets in step (*iii*), following the same procedure on already-constructed sets, have the same time complexity. Step (*i*) operates on domination sets to count the number of members in each set to find the one that has the maximum set size, executing in time proportional to $|M||C|$ in the worst case. Step (*ii*) clearly executes in constant time. Steps (*i*)-(*iii*) are iterated exactly once for each group member in the ultimate case. Therefore, the complexity of the entire algorithm is $O(|M|^2|C|)$. The generic algorithm we described applies for both *n*-dominating set problem [ZFL02] and *GREEDY* [Sa96] with modification on the initial construction of domination sets so that a group member is in domination set of a candidate core if it is in respectively within the *n*-hops or delay-constraint's reach from that candidate.

The optimization criteria suggested for *optTree* [KH03] can be computed for one core in the iteration that the core is examined as a candidate, without changing the structure of the greedy algorithm. If we consider a combination of *Node-count*, *Delay-residue, aveHop* and *Degree* as the optimization criterion of the algorithm, the additional computational complexity to the greedy algorithm is clearly that of the one among four with the highest complexity. In *Node-count*, *Delay-residue* and *aveHop* of *optTree*, each candidate core is examined for its rank depending on the receiver set they are potentially dominating—repeating at most as many times as the number of receivers. *Delay-residue* and *aveHop* also examine the distances of the core from the sources for residue in delay and hop-distances to the receivers. Therefore, the computational complexity of *optTree* is $O(|M|^3|C|)$.

As we mentioned in Section 3.1, Zappala et al [ZFL02] suggest a modification of the dominating set approximation algorithm for a *k*-center heuristic. Assuming repeated execution of *n*-dominating set heuristic for increasing values of *n* until a set of cores of maximum size *k* is obtained to dominate the all members of the multicast group, the complexity of *k*-center approximation for algorithm core selection is $O(|M|^2|C|h_{diam})$ where h_{diam} is the maximum hop-distance considered between a core and a group member and is bound by the diameter of the domain in this metric.

3.4.1 Implementation of Centralized Algorithms in Distributed Environments

As mentioned earlier, local information available to routers in generic setup is the distances of the router the state table is residing on to every other router in the domain in all metrics. Centralized algorithms are not readily supported in this setup for these algorithms demand and operate on broader domain information then that locally available. Our classification of centralized algorithms is based on their higher feasibility compared to their distributed implementation on multiple nodes with message exchange among the nodes, as opposed to processing on one particular node with the required domain information transmitted to it in centralized version. We further examine the amount of message transmission in centralized algorithms to facilitate their deployment in distance-vector routing environment.

We examined the generic dominating set heuristics as iterative steps, for each new multicast group member, of selecting the best candidate core to serve the next new member of the multicast group and updating the domination sets accordingly for the next iteration. If we look into these steps, they all operate on the distance-vector information of the candidate cores to determine the domination set of each candidate for a given multicast group, and examine each candidate with respect to "optimization criterion". Then, an algorithm to approximate the domination set solution following this heuristics simply transmits the distance-vector information of each candidate core in all relevant metrics to the coordinator. Similarly, the coordinator is notified of new members to join the multicast group merely for their domain identification. Therefore, assuming a particular candidate core set and stable network conditions, each candidate transmits its state table once to coordinator in $|C|$ messages, and the coordinator is notified of the members of the multicast group in $|M|$ number of messages, resulting in overall message complexity of $O(|M|+|C|) = O(|C|)$ on assumption that $|C| \geq |M|$.

3.4.2 Bidirectional Implementation of *GREEDY* and *optTree*

In a *bidirectional* routing tree, the first on-tree node receiving the data stream acts as a tree root and the stream is forwarded from this "root" towards all leaves. If the tree is not maintained bidirectional, the stream travels first to the root of tree and then delivered throughout the tree.

Whether the multicast tree is maintained as a bidirectional in a symmetric network environment is especially significant in constrained dominating set heuristics, since a bidirectional tree can describe shorter-delay paths between source-receiver pairs than a non-bidirectional one on which every path between such a pair has to pass through the tree root. Each core has broader potential domination in this case, resulting in better approximation of optimal core selection. The core selection process in this case, however, demands not only for the evaluation of delay-distances between multicast group members and candidate core sets, but also the corresponding information between the members of these sets and any other domain node, for any such node is a potential intermediary on the tree to be developed to act as the root for certain data source(s) on the tree. So, the bidirectional consideration of the multicast tree to be governed by the core set is a trade-off between the optimality of the result and the complexity of the core selection process.

Salama [Sa96] approaches this problem by first computing the domination set of each potential core before proceeding with the iterative selection process to produce a set of nodes to dominate the group. During the computation of the domination sets, each source, receiver and potential core, (s,r,c) triple, is examined for a delay-bound path between s and r on the core-tree rooted at c. Since the tree to be constructed is assumed to be bidirectional, the path between s and r passing through c can be shorter than the concatenation of the paths between (c,s) and (c,r) pairs if these two paths overlap partially at the core end. In other words, there may exist a node v which is on the shortest paths between both the pairs (c,s) and (c,r), and thus the portion between c and v is shared by both paths. The minimum-delay between s and r via the tree governed by c in this case reduces from the sum of the delays between (c,s) and (c,r) to that of (v,s) and (v,r). The corresponding computation to generate domination sets examines each node separately between (c,s) and (c,r) node pairs to locate a overlapping paths. The process then requires as input the knowledge of the entire domain nodes, and the message transmission for the bidirectional implementation of constrained dominating set algorithms is $O(|V|)$. As stated by Salama [Sa96], the process iterates for each source, receiver and potential core triple on all intermediary nodes lying on the examined paths, and the computational complexity of the preliminary process for the generation of domination sets is $O(|M|^2|C||V|)$. The additional component of both algorithms to select the cores among the candidate pool under this approach is computationally less complex than the preliminary phase. Therefore, the complexity of *GREEDY* [Sa96] and *optTree* [KH03] on assumption of bidirectional trees is $O(|M|^2|C||V|)$. However, the effectiveness of bi-directional design of core selection algorithms is restricted to multicast groups in which the source and receiver sets overlap to share paths on the core-trees.

3.5 Discussion

The integration of core-selection algorithms into core-based multicast routing protocols is dominated by whether the components are serving single or multi-core scheme—the single-core selection algorithms are restrictive for the performance of multi-core protocols and the algorithms to generate multiple core can not co-operate with single-core protocols. The remaining characteristics of core-selection algorithms superior to the protocols clearly do not restrict their applicability in coordination with these protocols.

Hop-count assumed as the only metric in most core selection algorithms restricts the potential results of the implementing protocol for all protocols described use delay-distance for exact measure. Hop-count, however, can be replaced by delay as the metric in the algorithms proposed by Thaler et al [TR97] except for *estCost*—in *maxDist*, *aveDist* and *Diam*, hop-count is used as a measure for point-to-point distance and can be replaced by delay without affecting the algorithm structure. In case of *estCost*, hop-count is measuring the overall tree structure and is the metric in context. Likewise, the description and measurement of topology center can be extended by allowing variable link weights during the process for computing the topology and group center—modifying *Topology-based* and *Group-based/Center* [CZD95] core selection algorithms for delay metric. The problem of *k*-center is defined for optimization on average hop-distance of the multicast group to the core set for a given size of core set, and can be modified as a problem to optimize on average delay-distance with an extended heuristics. *n*-dominating set problem has a different characteristics in that the hop-count is used as the domination criterion to be referred to rule in/out domain nodes into the candidate-core pool to be further filtered by the optimization criterion which in case of the applied heuristics by Zappala et al [ZFL02] is the number of multicast group members dominated by the particular node. A certain delay-distance to be given as the dominating criterion rather than the hop-distance imposes delay as the more precise metric during the selection process. However, the delay-constrained dominating set algorithm, namely *GREEDY* [Sa96] readily provides a broader solution in terms of potential performance by considering the given delay bound from source(s) to receivers via core(s) rather than separately between source-core and receiver-core pairs with a process of similar complexity. Cost in *Tournament* [SBK94] can be replaced by delay with no change in its algorithmic structure since both metrics are additive and have similar characteristics. Note that hop-count mainly serves for measuring the tree structure through measuring the number of on-tree links as it is perceived in [KH03] as additional metric to delay, and clearly has positive correlation with delay.

The core-based protocols studied so far assumed symmetric networks only for they all initiate the path construction process at the new member to join and measure the path distance from receiver to core rather than the other direction in which the stream flows. The asymmetry in path construction and maintenance can be accomplished by delegating the path construction to the core from the receiver-to-join simply by traversing the path to be added to the tree as the acknowledgement travels towards the receiver—rather than as the join request travels the core. Providing asymmetry in protocol features to realize the tree structure is far less challenging than the algorithmic components—including core selection—to specify the tree structure, and existing protocols apply with straightforward modifications to asymmetric networks. Note here that, hop-count is an asymmetric metric and characterizes the algorithm as asymmetric regardless of its symmetry characteristics. Those distributed algorithms we concluded to be applicable also for delay metric above, however, maintain their asymmetry since they all utilize the link values in their exact direction during the computations.

All core selection methods proposed by Salama [Sa96] are constrained, designed to meet the given delay bound on the tree to be constructed rooted at the selected core(s). The core selection algorithms studied by Salama [Sa96] can be described into protocol suites by their extension with tree maintenance and failure recovery components. Salama is leaving the cost optimization to the phase of tree construction and is not considering it as an additional design goal for core selection. The suggested *optTree* [KH03] heuristics take the constrained multi-core scheme one step further by bringing in additional metric for the refinement of ultimate protocol performance.

An interesting result as part of [SBK94]'s studies is that the tree rooted at core selected in consideration of the source group incurred less cost than the tree rooted at the core selected close to center of the receiver set, with the gain in tree cost increasing as the number of sources increase. This result validates the fact that a multicast tree rooted at a single core to optimize the paths to the receivers rather than those from the sources provides more gain in efficiency for optimizing the shared paths to the receivers rather than the unique paths from each source to the core. Note that this result is less significant for multi-core architectures which allow the placement of cores according to the proximity of multicast group members partitions in the domain, thereby optimizing the incurred cost locally on the tree portion covering the member partition.

Calvert et al [CZD95]'s topological center results are significant for single-core selection in homogeneously distributed applications for separating the selection process from the multicast group information and allowing the use of a core selected statically in the domain. Topological center contributes especially for the case the frequency of changes in group membership is high in a well-distributed group, reducing the necessity for the core migration to the instance of core failures.

The candidate core set processed by each algorithm is indicative for the message exchange and processing overhead during the selection process. In certain algorithms the selection method inherently restricts the candidate core set to a particular set of nodes in the domain. *Tournament* [SBK94] and *INITIAL* [Sa96], examining only the nodes which are in the subgraph "bordered" by the receiver group are in this group. Group-based core choices [CZD95] restrict the candidate core set to the set of receivers. Candidate core set of a smaller size can also be imposed externally to the ultimate selection module by restricting the candidate domain through a less excessive and less dynamic process to reduce the overhead still keeping the coverage of the set well-distributed throughout the network.

Core selection is a trade-off between the amount of processing and message transmission during the selection process, and the performance of the protocol in significant metrics. The performance results improved as the group size increased validating that as the size increases, the receivers are well-distributed over the network making the locations of the core less significant [TR97, CZD95]. According to Calvert et al [CZD95], group information does not necessarily improve performance. When the multicast group is widely distributed, the group-based core choice can degrade the performance relative to the topology-based methods. A certain core selection mechanism demonstrates different results for various applications. However a core selection method effective for one application is very likely to be effective in a similar manner for other types of applications compared to other selection mechanisms. There is no single-best core choice method across all metrics.

4. Conclusions

In this paper, we reviewed and analyzed the core-based approach for core selection algorithms in multicast routing protocols. We started with a brief overview of multicast routing protocols and the current trend into core-based approach. We reviewed the core selection algorithms studied in literature for their algorithmic structures, application issues and performance results as tested by their respective authors. Our review resulted in classification of core-selection schemes for their operational characteristics. We then analyzed the computational and message complexity of each algorithm in availability of local distance-vector state information at domain nodes. Our discussion included remarks on the applicability of these algorithms into the available multicast routing protocols.

Multicast routing protocols are mainly designed for their feasible deployment in distributed domains. Development and maintenance of the multicast tree structure above a certain level in the measures described in its design goal is determinant on a protocol performance. The existing architectures can be enriched with components for selection and management of core set for improved approximation of the generated tree and thus protocol operations. In this paper, we examined core selection—placement of core(s) to govern the multicast path for its placement in domain, and its construction especially in multi-core case. Other areas of similar concern are measurement of the tree deterioration from the optimal or approximated structure over the course of multicast session, and core migration for keeping the tree quality above a certain level according with its design description.

The design goals fulfilled by the core-based algorithms are governed by those of the routing protocols, and the assumed design problem in a multicast routing protocol existing in literature is the development of shortest paths between the tree root(s), namely the core(s) in core-based architecture, and the receivers. Multicast routing protocols and most core selection algorithms assume and operate on a unique metric. However, today's QoS-demanding applications require consideration of multiple metrics in optimization, and the typical design problem is the minimization of overall resource consumption during multicast session while meeting the delay-bound of the application. The research field is open to sophisticated architectures to support QoS provisions to increasingly demanding Internet applications, still maintaining the feasibility in actual deployments.

References

[B97] Ballardie, A., *Core Based Trees (CBT version 2) Multicast Routing – Protocol Specification*, RFC 2198, September 1997

[B97b] Ballardie, A., *Core Based Trees* (*CBT*) *Multicast Routing Architecture*, RFC 2201, September 1997

[CZD95] Calvert. K. L., Zegura, E. W., Donahoo, M. L., *Core Selection Methods for Multicast Routing,* Proceedings of ICCCN '95, 1995

[DEFJ94] Deering, S., Estrin, D., Farinacci, D., and Jacobson, V. *Protocol independent multicasting* (PIM), dense mode protocol specification. Technical report, IETF-IDMR, March 1994

[DEF+96] Deering, S., Estrin, D.L., Farinacci, D., Jacobson, V., Liu, C. and Wei, L., *The PIM Architecture for Wide-Area Multicast Routing*, IEEE/ACM Transactions on Networking, Vol.4, No.2, pp.153..162, April 1996

[D59] Dijkstra, E., *A Note on Two Problems in Graphs*, Numerische Mathematik, Vol. 1, pp. 269..271, 1959

[EH+97] Estrin, D., Handley, M., Helmy, A., Huang, P. and Thaler, D., *A Dynamic Bootstrap Mechanism for Rendezvous-Based Multicast Routing*, Proceedings of INFOCOM'97, 1997

[GJ79] Garey, M., Johnson, D., *Computers and Intractability*. W.H. Freeman, 2000

[JLK78] Johnson, D.S., Lenstra, J.K. and Kan, A.H.G.R, *The Complexity of The Network Design Problem*, Networks, Vol.8, No.4, pp.279..285, 1978

[KH03] Karaman, A. and Hassanein, H.S., *DCMC – Delay-Constrained Multipoint Communication with Multiple Sources*, submitted for publication

[M94] Moy, J., *Multicast Extensions to OSPF*, RFC 1584, March 1994

[M94b] Moy, John, *Multicast Routing Extensions for OSPF*, Communications of the ACM, August 1994, Vol.37, No.8, pp.61-114

[Sa96] Salama, H. F., *Multicast Routing for Real-Time Communication on High-Speed Networks*, Ph.D. Thesis, Department of Electrical and Computer Engineering, North Caroline State University, 1996

[Sh96] Shields, Clay, *Ordered Core Based Trees*, M.Sc. Thesis, University of California, Santa Cruz, June 1996

[SG97] Shields, C., Garcia-Luna-Acevez, J.J., *The Ordered Core Based Tree Protocol*, Proceedings of INFOCOM '97, Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, 1997

[SBK94] Shukla, S.B., Boyer, E.B. and Klinker, J.E., *Multicast Tree Construction in Network Topologies with Asymmetric Link Loads*, TR NPS-EC-94-012, Naval Postgraduate School, September, 1994

[TR97] Thaler, D. G. and Ravishankar, C.V., *Distributed Center-Location Algorithms*, IEEE Journal on Selected Areas in Communication, 15(3), pp. 291-303, 1997

[WPD88] Waitzman, D., Partridge, C., & Deering, S., *Distance Vector Multicast Routing Protocol*, RFC1075, November 1988

[W80] Wall, D. W., *Mechanisms for Broadcast and Selective Broadcast*, Ph.D. Thesis, Stanford University, California, U.S.A., 1980

[WE94] Wei, L. and Estrin, D., *The Trade-offs of Multicast Trees and Algorithms*, Proceedings of 1994 International Conference on Computer Communications Networks, September 1994

[ZFL02] Zappala, D., Fabbri, A. and Lo, V., *An Evaluation of Multicast Trees with Multiple Active Cores*, Journal of Telecommunication Systems, Kluwer, March 2002