Applying Clustering Algorithms as Core Selection Methods for Multiple Core Trees

Francesc Font, Daniel Mlynek
Signal Processing Laboratory - LTS3 - ITS
EPFL - Ecole Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
francesc.font@epfl.ch

Abstract - In the current Internet, there are two main types of intra-domain multicast routing protocols: dense mode and sparse mode. Dense mode protocols construct shortest path trees from the sender to the receivers. Sparse mode protocols construct shared trees rooted at a certain router, called core or rendezvous point (RP), which acts as a meeting point between senders and receivers of the same group. While solving important scalability problems in wide-area networks, shared trees also bring some drawbacks, the most known of which are traffic concentration and delay. In order to minimize these kinds of drawbacks, a variant of sparse mode using several cores per group emerged. Receivers are attached to the closest core and sources send packets to all the cores with attached receivers of the corresponding group. In this paper three clustering algorithms are presented and compared: max-cut, min-cut and min-concentration. They select the minimal set of cores in multiple core trees that maximizes or minimizes a certain objective function while limiting the QoS degradation according to a certain parameter called clustering ratio. These core selection algorithms are used to show the advantages of using several cores per group over the classical method of using only one.

I. INTRODUCTION

Dense-mode protocols, like DVRMP[1] and PIM-DM[2], use a *flood and prune* algorithm to find where the receivers are located and to construct the *shortest path tree* (SPT) from the sender to each of the receivers. The state for each *(Source, Group)* pair is maintained in all routers, independently of whether they are placed or not along the SPT. It allows a new receiver to join (graft) the SPT for a certain (S,G) pair at any time. Otherwise, MOSPF[3] avoids this flooding by maintaining a complete map of where the receivers are in each router. These protocols are suitable for single-sender groups where the receivers are densely distributed along the domain. But this protocol architecture presents serious scalability problems with the number of senders and groups especially in WANs, where the receivers are sparsely distributed along the domain.

In order to improve these scalability problems sparse mode protocols[4][5] use the concept of *rendezvous point*. A RP is a router that acts as a meeting point between all the senders and receivers of the same group. One *shared tree* (ST) per group is built from a certain RP to all the receivers, allocating group state information only in the routers along the ST. All the sources of the same group send the information to the same RP. From this RP, the information reaches the receivers along the ST.

While bringing advantages in scalability issues, sparse mode protocols also have some drawbacks such as traffic concentration around RPs and longer delays than in dense mode[8]. Two proposals arose to improve these drawbacks. The first one constructs bi-directional shared trees (BSTs)[6]. The senders also join the groups, and packets are already distributed to the receivers while traveling towards the RP. The second one uses several RPs per group in a PIM-SM domain[7]. It improves the fault tolerance as it no longer has a single point (RP) of failure. The resulting multicast trees are called *multiple core trees* (MCTs)[8]. In this kind of tree all RPs are used for all groups. Several algorithms have been proposed to choose the RPs in STs[9][10]. However, we have not found any algorithm in the literature to choose the set of cores in MCTs. Moreover, all clustering algorithms in the literature[14][15][16][17][18] work with a pre-determined number of clusters. In this paper we present three clustering algorithms based on *max-sum* and *min-sum* vector partitioning[18] and the minimal *traffic concentration ratio*. Traffic concentration is important because it can dramatically increase the delay due to queuing[9][10]. These algorithms choose the minimal number of clusters that accomplish the corresponding objective function while limiting the QoS degradation according to a certain *clustering ratio*. This degradation is due to having the cores as intermediate routing points between senders and receivers (triangular effect).

The rest of the paper is organized as follows: section II presents the mathematical basis used by the three algorithms presented in the section III. Section IV shows the advantages of MCTs over STs and BSTs based on the core selection of the algorithms. Section V presents some conclusions.

II. DEFINITIONS

A The Laplacian matrix

A network can be represented as an undirected weighted graph $G(V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices or nodes and $E = \{e_1, e_2, \ldots, e_m\}$ is the set of edges or links in the network. This graph can be entirely represented by the *edge-weight matrix* $W = (w_{ij})$, a $n \times n$ symmetric matrix, where $w_{ij} > 0$ is the weight or cost of the edge $(v_i, v_j) \in E$; $w_{ij} = 0$ if no (v_i, v_j) edge exists[11][12]. The degree of a vertex v_i is the sum of the weights of all its incident edges:

$$deg(v_i) = \sum_{j=1}^{n} w_{ij} \tag{II.A.1}$$

The *degree matrix* is the diagonal matrix $D = (d_{ii})$ where $d_{ii} = deg(v_i)$.

Definition 1: the $n \times n$ symmetric matrix $L = D - W$ is the *Laplacian matrix* of G.

Definition 2: an *n-vector* $\vec{\mu}$ is an eigenvector of L with eigenvalue λ if and only if $L\vec{\mu} = \lambda\vec{\mu}$.

The eigenvectors of L are $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_n$ with corresponding eigenvalues $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$. The $n \times d$ *eigenvector matrix* $U_d = (\mu_{ij})$ has as columns the eigenvectors $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_d$ with corresponding eigenvalues $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_d$. The $d \times d$ *eigenvalue matrix* $\Delta = (\lambda_{ii})$ is a diagonal matrix with diagonal entries $\lambda_{ii} = \lambda_i$. We use Δ for Δ_n and U for U_n. The eigenvectors form an orthogonal basis in n-dimensional space[13]. We assume that the eigenvectors are normalized, i.e., $\|\vec{\mu}_i\| = 1$, $\forall (1 \le i \le n)$.

Definition 3: the *weighted eigenvector matrix* is defined as:

$$Y_d = trans(\Delta_d^{1/2} trans(U_d)) \tag{II.A.2}$$

Some interesting properties of this matrix are show in the section C.

B The k-way Partitioning

Definition 4: a k-way partitioning of a vertex set V of a graph G is a set of k-disjoint non-empty clusters $P^k = \{C_1, C_2, \ldots, C_k\}$ such that $C_1 \cup C_2 \cup \ldots \cup C_k = V$, which optimize a certain objective function $f(P^k)$ [14].

Definition 5: the *external degree* or *edge-cut weight* E_h of a cluster C_h is the sum of the weights of the edges connecting vertices in C_h with vertices that do not belong to this cluster:

$$E_h = \sum_{v_i \in C_h} \sum_{v_j \notin C_h} w_{ij} \tag{II.B.3}$$

E_h is the degree of the vertex that results from coarsening[1] all the vertices in C_h [15].

Definition 6: the *Max-cut* (*Min-cut*) k-way partitioning is the clustering method that has as an objective the maximization (minimization) of the function:

$$f(P^k) = \sum_{h=1}^{k} E_h \tag{II.B.4}$$

C The k-way vector partitioning

Definition 7: a k-way vector partitioning of a set of vectors Y is a set of k-disjoint non-empty subsets $S^k = \{S_1, S_2, \ldots, S_k\}$ such that $S_1 \cup S_2 \cup \ldots \cup S_k = Y$, which optimize a certain objective function $g(S^k)$.

Definition 8: the *Max-sum* (*Min-sum*) k-way vector partitioning is the vector clustering method that has as an objective the maximization (minimization) of the function:

$$g(S^k) = \sum_{h=1}^{k} \left\|\vec{Y}_h\right\|^2 \qquad where \qquad \vec{Y}_h = \sum_{\vec{y} \in S_h} \vec{y} \tag{II.C.1}$$

Property 1: if we take as a set of vectors the *weighted eigenvector matrix* Y_d, where $\vec{y}_i^{\,d}$ is the row i of this matrix and $d = n$, the *max-cut* (*min-cut*) graph partitioning and *max-sum* (*min-sum*) vector partitioning objectives are identical [18]:

$$f(P^k) = \sum_{h=1}^{k} E_h = \sum_{h=1}^{k} \left\|\vec{Y}_h\right\|^2 = g(S^k) \tag{II.C.2}$$

Property 2: Each row $\vec{y}_i^{\,n}$ of Y_n is the projection of the vertex $v_i \in V(G)$ over the eigenvectors $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_n$ of L.

Property 3: as $E_h = \left\|\vec{Y}_h^{\,n}\right\|^2$, when $d = n$, given $\vec{y}_i^{\,n}$ the row i of the matrix Y_n,

$$\left\|\vec{y}_i^{\,n}\right\|^2 = deg(v_i) \tag{II.C.3}$$

and the module of the sum of several vectors gives the *external weight* or *edge-cut weight* (II.B.3) of the cluster formed by the vertices represented by these vectors.

D Traffic concentration

A network is properly defined as an undirected graph $G(V, E)$, which represents the physical network and a *routing*[19], which represents the unicast routing protocol.

Definition 9: The *routing* $\Re$ of a graph G of order n is defined as the set of $n(n-1)$ paths for all the pairs (v_x, v_y) of

1. Eliminate edges between vertices "collapsing" them in one resulting vertex

vertices of G:

$$\Re = \{\Re_{xy} \mid (v_x, v_y \in V, x \neq y)\} \quad \text{(II.D.1)}$$

Definition 10: given a network $(G, \Re)$, the *load of an edge* $e_j \in E(G)$, $\pi(G, \Re, e_j) \geq 0$, is defined as the number of paths $\Re_{xy}$ that contain the edge e_j [20]. The *edge-forwarding index* $\pi(G,\Re)$ is the number of paths in $\Re$ going through the edge $e_j \in E(G)$ with the highest load[19]:

$$\pi(G, \Re) = max\{\pi(G, \Re, e_j) \mid (e_j \in E(G))\} \quad \text{(II.D.2)}$$

The l*oad of an edge* gives a quantitative idea about how much traffic is natively[2] concentrated in the link represented in G by the edge e_j. The *edge-forwarding index* is then a measure of the native *traffic concentration* in the "a priori" most congested link in the network.

Definition 11: the *traffic concentration vector* of a network $(G, \Re)$ with a certain distribution χ of multicast traffic and groups, $\vec{t}(G,\Re,\chi)$, represents the traffic traversing each link or edge in G:

$$\vec{t}(G,\Re,\chi) = [t(G,\Re,\chi,e_1), \ldots, t(G,\Re,\chi,e_m)] \quad \text{(II.D.3)}$$

where $t(G,\Re,\chi,e_j)$ is the traffic load, in bytes, on the link represented by e_j in G. The module of the this vector is then a quantitative measure of the *traffic concentration* all around the network for a certain scenario χ:

$$\left\|\vec{t}(G,\Re,\chi)\right\| = \sqrt{\sum_{i=1}^{m} (t(G,\Re,\chi,e_i))^2} \quad \text{(II.D.4)}$$

Dense mode protocols construct shortest path trees from the senders to the receivers[1][2][3]. This type of multicast routing gives a "perfect" data distribution relative to the unicast *routing* $\Re$. Therefore, the comparison between sparse mode and dense mode gives much more information about *traffic concentration* than the absolute value of the *traffic concentration vector.*

Definition 12: the *traffic concentration ratio* of a network $(G, \Re)$ for a certain distribution of multicast traffic and groups χ, is the relation between the module of the corresponding dense and sparse mode *traffic concentration vectors*:

$$\eta(G,\Re,\chi) = \frac{\left\|\vec{t}(G,\Re,\chi)\right\|_{DM}}{\left\|\vec{t}(G,\Re,\chi)\right\|_{SM}} \quad \text{(II.D.5)}$$

Definition 13: the *intrinsic traffic concentration of a path* $\Re_{ij} \in \Re$ between two nodes $v_i, v_j \in (G,\Re)$ is defined as:

$$t_\pi(v_i,v_j) = \left(\sqrt{\sum_{e \in \Re_{ij}} (\pi(G, \Re, e))^2}\right) / (length(\Re_{ij})) \quad \text{(II.D.6)}$$

This parameter gives a quantitative measure about how this path is "natively" used by the underlying unicast routing protocol. The lenght of a path is its number of links. The goal for a clustering algorithm called *min-concentration* is then the maximization of the *intrinsic traffic concentration* of the paths from each core to the members of its cluster in order to approximate MCTs to the SPTs constructed by dense mode protocols as much as possible. This approximation has as an objective to minimize *traffic concentration*, reaching values as close to dense mode as possible. The *objective function* to maximize is:

$$f(P^k) = \sum_{h=1}^{k} \sum_{v_j \in C_h} t_\pi(core_h, v_j) \quad \text{(II.D.7)}$$

E The Clustering ratio

Definition 14: the *edge-load matrix* is an *edge-weight matrix* where edges are weighted with its *load*:

$$W_\pi = (w_{\pi_{xy}}) = \left\{ \begin{array}{cc} \pi(G, \Re, (v_x,v_y)) & If((v_x,v_y) \in E) \\ 0 & If((v_x,v_y) \notin E) \end{array} \right\} \quad \text{(II.1)}$$

Definition 15: the *edge-forwarding distance* between two nodes $v_i, v_j \in (G,\Re)$ is the sum of the *load of the edges* along the path $\Re_{ij} \in \Re$:

$$\delta_\pi(v_i,v_j) = \sum_{e \in \Re_{ij}} \pi(G, \Re, e) \quad \text{(II.E.1)}$$

The load of the only edge of a leaf router is $2(n-1)$ [19], where n is the number of nodes of G. We can then normalize the *edge-forwarding distance* with this value:

$$\delta^n_\pi(v_i,v_j) = \sum_{e \in \Re_{ij}} \frac{\pi(G, \Re, e)}{2(n-1)} \quad \text{(II.E.2)}$$

Definition 16: the *clustering ratio* Υ may be defined as the upper limit to the *normalized edge-forwarding distance* between the head of a cluster and the members of its cluster. In our algorithms (section III), all clusters C_i must fulfill:

$$\delta^n_\pi(core_i,v_j) \leq \Upsilon \qquad \forall v_j \in C_i \quad \text{(II.E.3)}$$

We want our clustering algorithms to fulfill two main conditions: (a) have more than one cluster and (b) attach leaf nodes to some cluster. These two conditions are reached by bounding the *clustering ratio*:

2. *Traffic concentration* due to the underlying unicast routing protocol.

$$1 \le \Upsilon < max(\delta_\pi^n) \qquad \text{(II.E.4)}$$

where $max(\delta_\pi^n)$ is the maximal *normalized edge-forwarding distance* of any path $\Re_{ij} \in \Re$ in the network $(G, \Re)$.

The clustering ratio may also be given in terms of the number of links, the *link cost* or the propagation *delay* between cores and the members of their clusters.

III. THE ALGORITHMS

The algorithms have as an input a network $(G, \Re)$ and a *clustering ratio* Υ that verifies (II.E.4). The result is the minimal set of k clusters $S^k = \{S_1, S_2, \ldots, S_k\}$ and their corresponding heads $H^k = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_k\}$ that

(A) maximize the *edge-cut weight* of clusters (II.B.4)
(B) minimize the *edge-cut weight* of clusters (II.B.4)
(C) maximize the *intrinsic traffic concentration* between each head and its cluster members (II.D.7)

and accomplish the *clustering ratio* condition (II.E.3). The algorithms work as follows:

1.Compute W_π.

2.Compute Y_n using W_π as the *edge-weight matrix* $(W = W_\pi)$. Each vector $\vec{y}_i^n$ corresponds to $v_i \in (G,\Re)$.

3.Extract $M \subset Y_n$ of p vertices with minimal module. This is the set of vertices with minimal degree (II.C.3).

4.Initialize S to p clusters $(S = S^p)$. Add to each cluster a vertex $\vec{y}_i^n \in M$ and remove $\vec{y}_i^n$ from Y_n $(H^k = M)$.

5.**For** each $S_j \in S$ find the $\vec{y}_i^n \in Y_n$ that

(A/B) max-/minimizes the *edge-cut weight* of S_j (II.B.3).
(C) maximizes the *intrinsic traffic concentration* with the head of the cluster (II.D.6).

6.**For** each $S_j \in S$ take the $\vec{y}_i^n$ computed in 5

If $\vec{y}_i^n$ accomplishes the *clustering ratio*[3] condition (II.E.3) **and** it is adjacent to some $\vec{y} \in S_j$ **and**

(A/B) gives the max-/minimal increase in the *edge-cut weight* in this cluster **then**
(C) gives the maximal *intrinsic traffic concentration* with the head in this cluster **then**

6.1.Add $\vec{y}_i^n$ to S_j and remove $\vec{y}_i^n$ from Y_n.

6.2.**If** $(\|\vec{y}_i^n\| > \|\vec{y}_t^n\|) \forall (\vec{y}_t^n \in S_j)$ **then** $\vec{y}_i^n$ is the new head of the cluster S_j $(\vec{h}_j = \vec{y}_i^n)$.

7.Find the $\vec{y}_i^n \in Y_n$ that does not accomplish the *clustering ratio* condition (II.E.3) with any cluster **and**

(A/B) min-/maximizes the average *edge-cut weight* (II.B.4) with all the existing clusters.
(C) minimizes the average *intrinsic traffic concentration* with the heads of all the existing clusters.

Create a new cluster S_j with this $\vec{y}_i^n$ and remove $\vec{y}_i^n$ from Y_n ($\vec{y}_i^n$ is the head $\vec{h}_j$ of the new cluster S_j).

8.Merge the maximum number of clusters that accomplish the *clustering ratio* condition (II.E.3) **and** give the best *objective function* value.

8.1.Check if the nodes of the merged clusters can migrate to another cluster, improving the *objective function*.

9.**If** $Y_n \neq \varnothing$ **then** go back to step 5.

The algorithms are initialized with clusters containing the nodes with less connectivity. Vertices giving the best *objective function* value are progressively added to clusters. New clusters can be generated when it is no longer possible to fulfill the *clustering ratio* with the existing clusters. Clusters "expand" and some of them can be merged when they make contact. Step 8 assures that the solution giving the best *objective function* value for the minimal number of clusters is found. The head or core of each cluster is the vertex with the maximal degree or connectivity in the cluster (II.C.3). As multicast traffic is transferred from sources to cores and from cores to group members, this election minimizes the *traffic concentration* in the incident edges of the core.

IV. RESULTS

As an example, we have generated a graph of 15 nodes (Fig. 1). Each edge e_j is numbered with its *load* $\pi(G, \Re, e_j)$. MCTs have several cores to which some nodes are attached. The core and their corresponding nodes form a cluster. The clustering results of applying the three methods explained in this paper are presented in table I. The number in bold is the core or head of the cluster while the subsequent numbers are the nodes belonging to this cluster. We applied three *clustering ratios* $(\Upsilon = 1.0, 2.0, 3.0)$(II.E.3). The result for all the algorithms is four, three and two clusters respectively. For simplicity, we chose as an example a small graph. So, it is not surprising that the algorithms give similar results. Furthermore, the constraint on the "radius" of the clusters and the election of the "most connected" vertex in the cluster as a core limit the divergence of the results between the different clustering methods.

As well as comparing the MCTs built by the different clustering methods, it is also very important to compare MCTs

3. If $\vec{y}_i^n$ is the new head of the cluster it has to accomplish the *clustering ratio* condition with all the members of this cluster.

Clustering ratio	Max-cut	Min-cut	Min-concentration
1.0	**2** |0, 1 **3** |4, 5 ,6 **8** |7, 9 **11**|10, 12, 13, 14	**2** |0, 1 **3** |4, 5 **8** |6, 7, 9 **11**|10, 12, 13, 14	**2** |0, 1 **3** |4, 5 **8** |6, 7, 9, 10 **11**|12, 13, 14
2.0	**2** |0, 1 **3** |4, 5, 6 **8** |7, 9, 10, 11, 12 13, 14	**2** |0, 1 **3** |4, 5 **8** |6, 7, 9, 10, 11, 12, 13, 14	**2** |0, 1 **3** |4, 5 **8** |6, 7, 9, 10, 11, 12, 13, 14
3.0	**2** |0, 1, 11, 13, 14 **3** |4, 5, 6, 7, 8, 9, 10, 12	**2** |0, 1, 10, 11, 12, 13, 14 **3** |4, 5 ,6, 7, 8, 9	**2** |0, 1, 11, 13, 14 **3** |4, 5, 6, 7, 8, 9, 10, 12

Table I: Clustering results for the *min-cut*, *max-cut* and *min-concentration* algorithms.

with STs, BSTs and SPTs. STs are build by CBT[4] and PIM-SM[5] protocols, BSTs are build by BiDir-PIM[6] and SPTs are built by dense mode protocols[1][2][3]. The results for SPTs are presented as the optimal reference for the other trees. However, dense mode protocols are not feasible in WANs due to their scalability problems. We implemented MCTs and we ran all the simulations in *Network Simulator*[22]. As a scenario χ we generated 23 random groups having between 1 and 3 senders and 2 and 5 receivers. All the sources send two packets of 210 bytes and links have an end-to-end delay of 10 ms. The set of RPs used for STs and BSTs is {2, 3, 8, 11}, which is the set of RPs minimizing the *traffic concentration* in this kind of trees[10]. In Fig. 2 are marked the levels reached by SPTs, BSTs and STs for two important parameters: the *traffic concentration* and the average delay. As well as the *traffic concentration ratio* (II.D.5), the maximal *traffic concentration* in any link in the network is also shown. For the same number of RPs ($\Upsilon = 1.0$) MCTs always give much better results than STs and BSTs, and are really close to SPTs. For $\Upsilon = 2.0$, MCTs are still better than BSTs and for $\Upsilon = 3.0$ the results are similar to BSTs and only worse in the *traffic concentration ratio*. In all cases MCTs fare better than STs. In general, it is observed that MCTs have very good results in terms of delay and maximal *traffic concentration*.

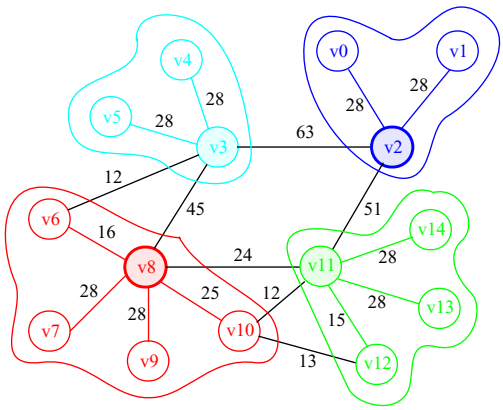

Fig. 1. A graph with its corresponding *edge-loads* and the clustering result applying the *min-concentration* algorithm for $\Upsilon = 1.0$. The shaded nodes are the cores.

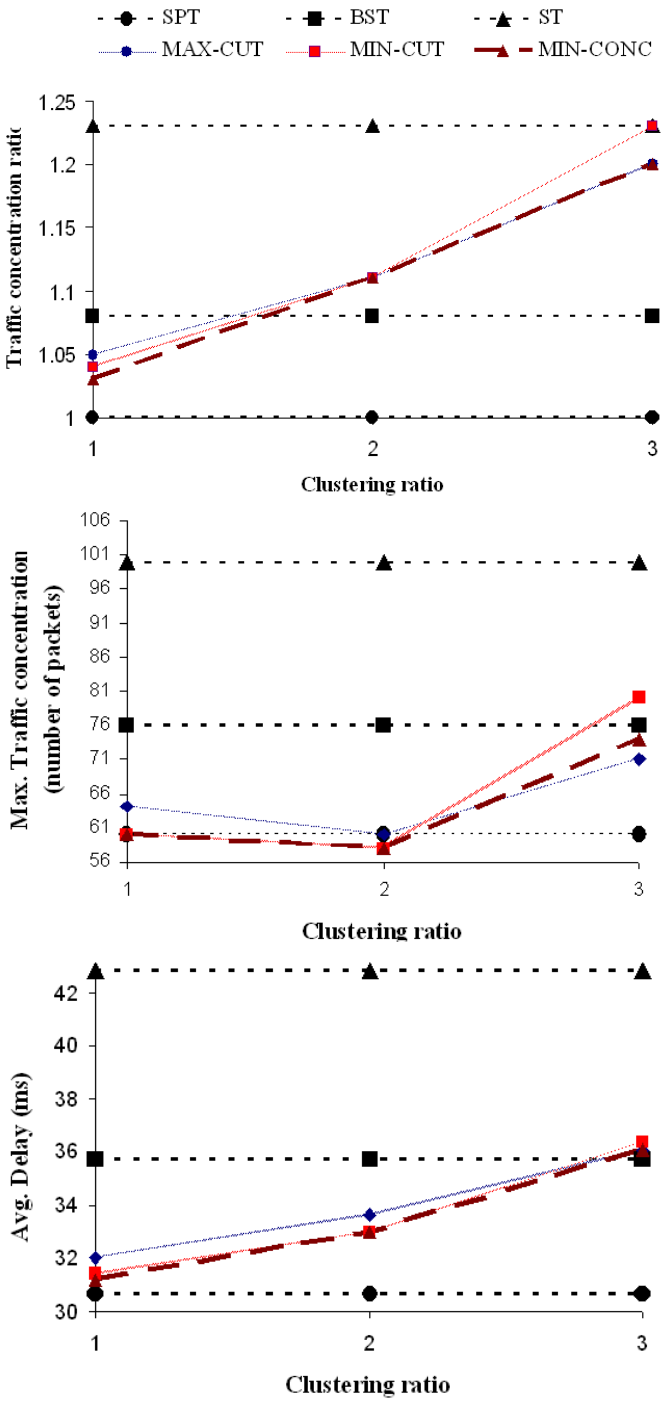

Fig. 2. Traffic concentration ratio, maximal traffic concentration and average delay in SPTs, BSTs, STs and MCTs .

The results for MCTs are similar for the different clustering algorithms. This is mainly due to the small number of nodes in the example. However, in this and other larger examples, the *min-concentration* algorithm reaches better results than the *min-cut* and *max-cut*. This is especially true in the *traffic concentration ratio*, because the algorithm was designed to mini-

mize it. Otherwise, *min-cut* is better than *max-cut* when the number of cores is well adapted to the size of the graph. For a small number of cores, the *max-cut* algorithm shows a better behavior.

V. CONCLUSIONS

This paper presents three clustering algorithms choosing the minimal set of cores in MCTs that optimize a certain *objective function* while keeping the *clustering ratio* condition (II.E.3). The *clustering ratio* is related to the QoS degradation relative to the SPTs built by dense mode protocols. This degradation is due to the triangular effect produced by having the cores as intermediate routers between senders and receivers. The clustering results of these algorithms are used to show the advantages of MCTs over STs and BSTs in terms of *traffic concentration* and delay. In addition, these parameters are close to the values reached in SPTs when the number of cores is equal to those used in STs and BSTs to minimize the traffic concentration[10]. The *min-concentration* algorithm gives the best results. Furthermore, as MCTs re-use all the cores for all the groups, they are robust to core failures while STs and BSTs suffer from a single point (RP) of failure.

ACKNOWLEDGMENT

We would like to acknowledge useful discussions about the algorithm with Brice Tsakam and text revisions of Marc Epalza, both of LTS3-EPFL.

REFERENCES

[1] D. Waitzman, C. Partridge and S. Deering. "Distance Vector Multicast Routing Protocol". RFC 1075, November 1988.

[2] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer and L. Wei. "Protocol Independent Multicast version 2 Dense Mode Specification". draft-ietf-idmr-pim-dm-06.txt, Aug 6, 1997.

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

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

[5] D.Estrin, D.Farinacci, A.Helmy, D.Thaler, S.Deering, M.Handley, V.Jacobson, C.Liu, P.Sharma and L.Wei. "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification". RFC 2362, June 1998.

[6] Mark Handley, Isidor Kouvelas, Tony Speakman, Lorenzo Vicisano. “Bi-directional Protocol Independent Multicast (BIDIR-PIM)“. <draft-ietf-pim-bidir-04.txt>, June 2002.

[7] D. Kim, D. Meyer, D. Farinacci. “Anycast Rendezvous Point (RP) mechanism using Protocol Independent Multicast (PIM) and Multicast Source Discovery Protocol (MSDP)”. RFC3446, January 2003.

[8] D. Zappala, A. Fabbri, and V. Lo. “An Evaluation of Shared Multicast Trees with Multiple Cores”. Journal of Telecommunication Systems, Kluwer, March 2002.

[9] K.Calvert and E.Zegura, M.Donahoo. "Core Selection Methods for Multicast Routing". In Proceedings of IC3N ’95, 1995.

[10] F. Font, D. Mlynek. “Choosing the Set of Rendezvous Points in Shared Trees Minimizing Traffic Concentration“. IEEE International Conference on Communications in Anchorage, Alaska, May 2003.

[11] G. Chartrand, Western Michigan University. “Introductory Graph Theory”. Dover Publications, Inc. New York, 1977.

[12] R. Diestel, University of Hamburg. “Graph Theory“, Second Edition. Springer-Verlang New York 1997, 2000.

[13] B. Mohar. “The laplacian spectrum of graphs“. In Proceedings of the 6th International Conference on the Theory and Applications of Graphs, pages 871-898, 1988.

[14] C. J. Alpert and A. B. Kahng. "Multiway Partitioning Via Geometric Embeddings, Orderings, and Dynamic Programming". IEEE Transaction Computer-Aided Design, vol. 14, pp. 1342-1357, Nov, 1995.

[15] G. Karypis and V. Kumar. "Multilevel k-way Partitioning Scheme for Irregular Graphs". Journal of Parallel and Distributed Computing, vol. 48, pp. 86--129, 1998.

[16] H. Zha, C. Ding, M. Gu, X. He, and H.D. Simon. “Spectral relaxation for k-means clustering”. Proc. Neural Info. Processing Systems (NIPS 2001).

[17] P. K. Chan, M. D. F. Schlag and J. Zien. “Spectral k-way ratio cut partitioning and clustering“. IEEE Trans. on CAD (1994) 1088-1096.

[18] C. J. Alpert and S-Z Yao. "Spectral Partitioning: The More Eigenvectors, the Better". Proc. ACM/IEEE Design Automation Conf., 1995.

[19] M. C. Heydemann, J-C. Meyer, and D. Sotteau. “On forwarding indices of networks”. Discrete Applied Mathematics, 23:103123, 1989. RR n# 3452 16 Bruno Beauquier.

[20] B. Mohar. “Some applications of laplace eigenvalues of graphs”. In G. Hahn and G. Sabidussi, editors, Graph Symmetry: Algebraic Methods and Applications, volume 497 of NATO ASI Series C, pages 227-275. Kluwer, Dordrecht, 1997.

[21] G. Gauyacq. “Edge-forwarding index of star graphs and other Cayley graphs”. Discrete Appl. Math. 80 (1997), 149--160.

[22] Network Simulator: *http://www-mash.cs.berkeley.edu/ns*