

Efficient Multicast on a Terabit Router

Punit Bhargava Sriram C. Krishnan Rina Panigrahy
punit@cisco.com srikrish@cisco.com rinap@cisco.com

Abstract— Multicast routing protocols and routers on the internet enable multicast transmission, one-to-many connections, by replicating packets close to the destinations, obviating the need for multiple unicast connections for the same purpose, thereby saving network bandwidth and improving the throughput of the Internet. Similarly, within a router, multicast between linecards is enabled by a multicast capable switch fabric. A multicast cell is sent once from the source linecard to the switch fabric; the switch fabric sends the cells to all the destination linecards obviating the need for, and the waste of linecard to fabric bandwidth that would result from multiple unicast cell transmissions for the same purpose. For high capacity routers (several terabits), the fixed size destination field of the (fixed size) cell is inadequate to specify exactly the subset of the switch ports the multicast cell should be sent to for the ever increasing number of multicast connections to be supported. Therefore, for several connections, we have to supercast, i.e., send the cell to non-subscribing linecards and have them drop the cell. In this paper, we study this problem of assigning destination labels for multicast cells so that the amount of supercast, the amount of wasted bandwidth, is minimized, and thereby the throughput of the router is maximized. We formalize this combinatorial optimization problem and prove it NP-complete and hard to find approximate solutions too. We have devised several effective heuristic algorithms that we have implemented and report experimental results for. Faster heuristics can support a higher multicast connection establishment rate; slower heuristics can be invoked off-line to further optimize multicast label maps. **Keywords:** combinatorics, graph theory, mathematical programming/optimization, system design, routers, multicast.

I. INTRODUCTION

In a unicast connection a source sends to a single destination. In a multicast connection a source sends to multiple destinations. While a multicast can be implemented as multiple unicasts, this is inefficient use of network bandwidth especially if the unicast routes overlap. So routers today are capable of replicating packets and sending them out on multiple outgoing interfaces. At the network level, which routers replicate and what interfaces the packets are sent out on is determined by multicast routing algorithms.

Within a router also, it is desirable to multicast rather than multiply unicast from the ingress linecard to the multiple egress linecards, thereby saving on bandwidth from the linecard to the switch-fabric. This multicast within a router is enabled by the switch fabric that connects the various linecards.

Switch fabrics transport fixed size packets called *cells* between linecards. The header, also a fixed size section of the cell, for each cell, contains amongst other information the destination field. This destination field is interpreted by the switch fabric to yield an outgoing link to send the cell to. If there are n links, for unicast, $\lceil \log_2 n \rceil$ bits would suffice for

the destination address field. For example, if the switch fabric is a single-stage cross-bar connecting sixteen linecards a four bit field for the destination would suffice.

For multicast, we need to send the cell to a subset of the n links. If we allocate greater than $\lceil \log_2 n \rceil$ bits for the destination field, these bits contribute to wasted bandwidth through the router while transmitting unicast. Especially for large terabit routers, such as those that sit in the core of the internet, the number of outgoing links n from the switch-fabric, i.e., the number of linecards in the system, is large and the difference between n and $\lceil \log_2 n \rceil$ gets amplified. Therefore, in order to save on bandwidth, the destination field in the cell header is of a width m between $\lceil \log_2 n \rceil$ and n , i.e., $\lceil \log_2 n \rceil < m < n$. The destination field is translated into an n -bit vector corresponding to the output links the cell should be sent to via a table look-up. Therefore at most 2^m link subsets can be addressed.

However, the number of multicast connections can be much greater than 2^m . For IPv4, the class D address space permits about 2^{27} connections, and these could be interpreted with each source to denote a different set of linecards within the router. With IPv6 many many more connections will be possible. Each of these connections needs to be mapped, to a destination label stamped on the cell that translates to a link-set that *includes* each of the linecards that have recipients to the multicast connection behind them. The volume of multicast traffic, in the enterprise and the wider Internet, is growing and expected to explode in the future. Since there are a limited set of labels, as the number of multicast connections grows, we will eventually have to send cells to linecards that are not subscribers to the multicast group. We call this *supercasting*.

In this paper, we study how to optimally supercast while minimizing the wastage of bandwidth from the switch fabric to the linecard, thereby increasing the throughput of the switch-fabric/router and therefore the internet. This problem has been previously addressed by Marsan et al [1]. We distinguish our work from theirs in Section VIII.

A. Contributions and organization

We begin in Section II by establishing our notation and terminology and a mathematical formalization of the problem. We characterize the computational complexity of the problem in Section III; we prove the problem to be NP-complete. We note an interesting fact about switch throughput maximization and matchings and bicliques in Section IV. We establish the hardness of finding approximate solutions in Section V. We present effective heuristics in Section VI. We have implemented our heuristics and present experimental results

in Section VII. We also present interesting theoretical analysis that explains and sheds light on our experimental findings. Concluding remarks and comparison with the previous work of Marsan et al [1] is in Section VIII.

II. THE PROBLEM

Assume that there are n links out of the switch-fabric. Each multicast cell has to be sent to some of the n links. A *destination-set* (DS) is defined to be a subset of $\{1, 2, \dots, n\}$. The destination links of each cell that is transported by the switch-fabric is specified by a label of m bits carried in the cell header. For large capacity terabit routers, n is large, and to save bandwidth (since m bits are carried by each cell including unicast cells), m is sized between $\lceil \log_2 n \rceil$ and n , i.e., $\lceil \log_2 n \rceil < m < n$.

The label is mapped into a destination-set by looking up a label-to-destination-set table (LTDT) maintained at (each stage of) the switch-fabric. The LTDT is programmed via an out-of-band connection at a much slower rate than the cells are transmitted (and so programming the LTDT at the same rate as the cells is out of question).

So associated with each multicast connection c we have:

- 1) The destination set D_c —the set of links the cell should be sent out on, and
- 2) The *assigned-set* (AS) A_c —derived from the label and the LTDT.

For the multicast connection to reach all its intended recipients (linecards), the cell has to be *supercast* by the switch-fabric, i.e., the AS A has to be a *superset* of the DS, $A \supseteq D$.

Assume N multicast connections, i.e., N DSs. Let $M = 2^m$. Typically, N , the number of multicast connections is larger than M . Thus, in summary,

$$\log_2 M < n; M < N$$

Therefore, the AS for some DSs has to be the same, and strict supercast has to happen—some DS has to be mapped to a (strictly) larger set. The optimization objective is to map the N DSs to the M ASs so that the amount of *strict supercast* is minimized. The optimization problem is posed as a formal decision problem ala [2] as follows:

MINIMUM CUMULATIVE SUPERCAS (MCS)

INSTANCE: Positive integers n , M , N , and K such that $M \leq N$; a set of N DSs $\mathcal{D} = \{D_1, D_2, \dots, D_N\}$, where each $D_i \subseteq \{1, 2, \dots, n\}$; positive integers w_1, w_2, \dots, w_N .

QUESTION: Does there exist a set of M assigned sets $\mathcal{A} = \{A_1, A_2, \dots, A_M\}$ where each A_i is subset of $\{1, 2, \dots, n\}$ and a map $f : \mathcal{D} \rightarrow \mathcal{A}$, such that:

- 1) for each i , $1 \leq i \leq N$, $f(D_i) \supseteq D_i$, and
- 2) Cost, $\sum_{i=1}^N (|f(D_i) \setminus D_i| * w_i) < K$.

The *cluster-map* f partitions the DSs into M clusters and assigns an AS for each cluster while satisfying the supersetting condition and minimizing the cost. We also say that the cluster-map produces a *clustering* of the destination sets. The positive integers w_1, w_2, \dots, w_N are the weights associated with each destination set and represent the bandwidth, i.e., frequency of

transmission, for the corresponding multicast connection. The *cost of a cluster* i is the amount of supercast incurred for DSs in the cluster, i.e.,

$$\sum_{D_j: f(D_j)=A_i} (|A_i \setminus D_j| * w_j)$$

A. Relationship to similar problems

Several clustering/partitioning problems [3], [4], in varied application domains from data mining [5] to image processing [6] to DNA sequencing [7], have been studied. Of these, the problem of clustering points and choosing the centroid of the points in the cluster as their representative seems closest in flavor to MCS. Both the Euclidean and rectilinear (or Hamming distance) versions of the problem have been proven NP-complete [4]. While MCS does use the hamming distance as the cost function, the cluster representative being the superset of the comprising points implies that the representative is not equidistant from the comprising points (i.e., not the centroid). This unique aspect of our cost function, the supersetting requirement, defeated our efforts to reduce any known clustering problems in the literature to MCS.

Before we proceed to the computation complexity of MCS we capture some simple yet key properties of the MCS problem that we use repeatedly in the sequel.

B. Key properties

We seek insight into whether a cluster of DSs is useful or not and how much it reduces the cost of the solution by.

Towards this end, we employ the convenient representation of sets as bit-vectors. A DS D is an n -bit 0-1 row vector; the i^{th} bit is 1 if location $i \in D$, and 0 otherwise. The (row) vectors for the N destination sets are stacked one on top of the other to yield the $N \times n$ *destinations-matrix* (DM) \mathcal{Q} with N rows and n columns. A *row-cluster* is a subset of rows of the DM \mathcal{Q} . A *row-clustering* of the DM \mathcal{Q} is a partition of the rows of \mathcal{Q} into a set of row-clusters. Therefore, the clustering is nothing but the cluster-map; a row-cluster is nothing but a cluster (set of DSs).

Similarly, we define a *column-cluster* of \mathcal{Q} to be a subset of its columns. A *column-clustering* of \mathcal{Q} is a partition of its columns into a set of column-clusters.

The correspondence between sets and bit-vectors turn set operations into Boolean operations on their bit-vectors; we'll abuse notation and also talk of Boolean operations on sets.

Having developed the bit-vector viewpoint of sets, we return to the question of the cost of a cluster. First, observe:

Observation 2.1: The AS for a cluster is the union (Boolean OR) of the DSs in the cluster.

Since the cost of a cluster involves set difference between the AS and the constituent DSs, it is only a zero in a row (DS) that can contribute to cost.

A zero in a row, if dominated by a one in another row in the same cluster, would contribute “weight assigned to the row” units of cost to the cluster. So the *maximum possible* cost of a row in a cluster is the number of zeros in the row multiplied by the weight assigned to the row and called the *weighted number of zeros*. The weighted number of zeros of a cluster

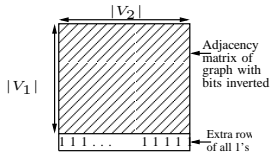


Fig. 1. Reduction from MEB to MCS: DM for MCS instance

is the sum of the weighted number of zeros of each of the rows in the cluster; and this is the maximum possible cost of the cluster. Whether, the *actual* cost of the cluster is the weighted number of zeros or less depends on whether there are columns of zeros in the cluster or not. Although simple, this is an important enough observation that we set it off as a lemma:

Lemma 2.2: Given a cluster, it yields *cost savings*, i.e., a zero in a row of the cluster does not contribute to the cost of the cluster only if the column in the cluster is comprised only of zeros.

III. COMPUTATIONAL COMPLEXITY OF MCS

Theorem 3.1: MCS is NP-complete.

Proof: Clearly MCS is in NP as given a cluster map it is easy to verify its correctness in polynomial time.

We show MCS is NP-hard by reducing the maximum edge biclique (MEB) problem, proved NP-complete in [8], and stated as follows:

Maximum Edge Biclique

INSTANCE: A bipartite graph $G = (V_1 \cup V_2, E)$ and a positive integer K' .

QUESTION: Does G contain a biclique (i.e., a complete subgraph) with at least K' edges? That is, does there exist $G' = (V'_1 \cup V'_2, V'_1 \times V'_2)$ such that:

- $V'_1 \subseteq V_1; V'_2 \subseteq V_2; (V'_1 \times V'_2) \subseteq E$; and, $|V'_1 \times V'_2| \geq K'$.

The MCS instance for the MEB instance is shown in Figure 1. The shaded section is the adjacency matrix of the bipartite graph G with zeros (ones) replaced by ones (zeros). The MCS instance has $|V_2|$ destinations but an extra row of all ones (i.e., going to all the destinations). More formally, the parameters of the MCS instance are as follows:

- Number of destinations, $n = |V_2|$.
- Number of DS, $N = |V_1| + 1$; all weights are 1, i.e., $w_i = 1, 1 \leq i \leq N$.
- Number of permitted clusters, $M = 2$.

The destination sets are defined as follows:

- For $1 \leq i \leq |V_1|$ and $1 \leq j \leq |V_2|$, destination set D_i includes destination j only if $(v_i, v_j) \notin E$.
- For $i = |V_1| + 1$, destination set D_i contains all $|V_2|$ destinations.
- $K = |E| + 1 - K'$.

The reduction is clearly a polynomial time reduction. It is easy to see that the number of zeros in the DM is exactly $|E|$, the number of edges in G . Since weights of the DSs are one, the cost of a one-clustering is less than $|E| + 1$. We show that cost

savings in a two clustering are in one-to-one correspondence with the edge count of bicliques in G .

Yes-instance of MEB maps to yes-instance of MCS: Assume G has a biclique with K' edges. Consider the following clustering of the MCS instance: cluster A is formed by rows corresponding to the vertices from V_1 in the biclique and cluster B has all the other rows. Since the adjacency matrix of the subgraph of G that is the biclique will be all ones, the columns corresponding to vertices from V_2 in the biclique in cluster A will be all zeros. Thus, these zeros in cluster A being all-zero columns will not contribute to the cost (by Lemma 2.2); they reduce the cost from a possible maximum of $|E|$ by K' . Therefore, the exhibited clustering has cost $< K = |E| + 1 - K'$.

Yes-instance of MCS implies yes-instance of MEB: Assume the MCS instance has a clustering with cost $< K = |E| + 1 - K'$. Therefore, the cost reduction obtained by this clustering is at least K' .

Consider the cluster that contains the extra row. Since the extra-row is all ones, the assigned set (bit-vector) for this cluster would have to be all ones. Therefore, the cost savings from this cluster is zero.

So all of the cost savings in the 2-clustering comes from the cluster without the extra row. By Lemma 2.2 there have to be columns of all zeros. Therefore the total number of zeros in the columns with all zeros is at least K' . The rows of this cluster and the zero-columns exactly identify the biclique in G with at least K' edges. ■

IV. NOTE ON SWITCH THROUGHPUT MAXIMIZATION FOR UNICAST TRAFFIC AND MULTICAST LABEL ASSIGNMENT

We can also reduce MCS to MEB in the following sense. Invert the bits of the DM to yield the adjacency matrix of a bipartite graph. It is not difficult to see that optimum M -clusterings for MCS correspond to a set of M maximal edge bicliques such that the “left hand sides” of the bicliques yield a partition of V_1 (the set of DSs) and the right hand sides may overlap (not a partition).

It is interesting to note the similarity/contrast with this result of multicast label assignment for maximizing throughput through a switch that involves a partition into maximal bicliques, and unicast switch throughput maximization that involves maximum bipartite matching[9]—unicast involves edges, multicast cliques.

V. HARDNESS OF APPROXIMATION

As shown in the proof of NP-hardness of MCS in Section III, given an instance of MEB, a bipartite graph G with $|E|$ edges, we obtain an instance of MCS so that there is a solution to that instance with cost $|E| - k$ if and only if there is a biclique with k edges in G .

There is evidence that MEB is hard to approximate by reduction from R3SAT [10]. The concept of R3SAT-hardness hinges on the following hypothesis from [10] which asserts that it is hard to differentiate a satisfiable 3CNF formula from a typical random one. Note that in a typical random formula

the maximum number of satisfiable clauses is only a constant fraction of the total.

HYPOTHESIS 2. [10] For every fixed $\epsilon > 0$, for Δ a sufficiently large constant independent of n , there is no polynomial time algorithm that on most 3CNF formulas with n variables and $m = \Delta n$ clauses outputs typical, but never outputs typical on 3CNF formulas with $(1 - \epsilon)m$ satisfiable clauses.

DEFINITION 1. [10] A computational problem is random 3SAT-hard (R3SAT-hard) if having a polynomial time algorithm for the problem contradicts Hypothesis 2.

Although Feige [10] only states that MEB is hard to approximate and provides precise values only for Max balanced Biclique, it is easy to see from his constructions that it is hard to approximate MEB within factor 2. In particular, it is hard to distinguish between graphs that have bicliques with $n^2/16$ edges and those with fewer than $n^2/64$ edges.

Based on this we will show that it is hard to approximate MCS to a factor better than $31/28$.

Theorem 5.1: It is R3SAT-hard to approximate MCS to a factor better than $31/28$.

Proof: Indeed, if we could, that would mean we can tell the difference between graphs with bicliques of size more than $n^2/16$ and less than $n^2/64$.

This is because the ratio of the MCS objective function in the two cases would be $(|E| - n^2/64)/(|E| - n^2/16)$. Since $|E| < n^2/2$, this ratio $> (1/2 - 1/64)/(1/2 - 1/16) > 31/28$.

So if we have an approximation algorithm for MCS with ratio better than $31/28$ we would be able to distinguish between the presence of large and small bicliques. ■

VI. HEURISTIC ALGORITHMS

Since MCS is NP-complete, we focus on devising efficient heuristics. We have devised heuristic algorithms that vary in run time from constant time to $O(M)$ time. The run time of the algorithm determines the maximum connection establishment rate that it can support.

Many switch fabrics provide a mechanism to reprogram the LTDT (label to destinations table, the map from $\{0, 1\}^{\log_2 M}$ to $\{0, 1\}^n$), either with a control register that provides a temporary broadcast mode while the table is being completely reprogrammed or a extra copy of the table with the facility to switch between the tables. This facility can be exploited to periodically spawn an algorithm that runs offline but takes longer to run than the online algorithm and results in less wasted bandwidth.

A. Greedy row-clustering

If the number of rows, N , is less than the allowed range-size, M , we can achieve a zero cost solution with a simple row-clustering consisting of each row in a cluster by itself. Therefore, in practice, as multicast connections are established, DSs become clusters by themselves. After we have established M DSs in clusters by themselves, we add a new DS into a cluster that involves the least cost increase.

There are two elements to the increase in cost resulting from the addition of a DS to an existing row-cluster. The cost for the DS added as well as the increase in the cost of the DSs already in the cluster due to the addition of the new DS. The cost of DSs already in the cluster may increase if the addition of the new DS causes the addition of a one to a column that was previously all zeros (see Lemma 2.2).

To facilitate easy determination of the cost increase due to the addition of a DS to a row-cluster and the cost decrease due to the deletion of a DS from a row-cluster (when a multicast connection is torn down), we maintain for each cluster:

- 1) The number of rows.
- 2) The number of zeros in each column. Since we have the number of rows, we can easily compute the number of ones in each column.
- 3) The sum of the weights of the rows in the cluster.
- 4) The assigned set for the cluster.

1) *Computational complexity:* With the above information maintained per cluster, the cost of deletion of a DS from a cluster and the total cost of adding a DS to a given cluster can be determined in $O(n)$ time, essentially constant time assuming n , the number of destinations, is a constant. For the addition of a DS, we have to consider the cost increment for each of the M clusters and pick the best choice. This takes $O(M)$ time.

2) *Offline version:* In the “online” algorithm, the clusters are seeded by the DSs as the first N connections are established. After several connections are established we can run the greedy row clustering algorithm again, picking random DSs to seed the clusters thus getting rid of the bias of the order of connection establishment, and then reprogram the LTDT with this optimized map.

B. Column-clustering based heuristic

If M , the number of clusters is large, the time to choose the candidate cluster to add the DS could be large. We partition the problem into column-clusters (CCs) to reduce the amount of time it takes to compute the candidate cluster. This however comes at a cost of a possibly suboptimal solution.

For example, assume we split the set of columns into two, resulting in a column-clustering with two CCs with $n/2$ columns each (assume n is a multiple of 2). We then have two problem instances with half the number of columns. The product of the number of row-clusters permitted for each instance would have to be M ; for example, \sqrt{M} per instance. The AS (cluster) for a given DS would be computed in two parts: the first $n/2$ columns and the second $n/2$ columns, requiring $O(\sqrt{M})$ time.

1) *Computational Complexity:* The online version takes $O(M')$ time, where M' is the largest number of clusters assigned to any of the CCs; the offline version takes $O(NM')$ time.

C. Constant time signature methods

We have devised two constant time methods ($O(mn)$ time methods to be precise, essentially constant time). Each of these

methods computes a *signature*. The signature is a proximity measure—two DSs that do not differ by much will likely have the same signature. All the DSs with the same signature get clustered together and the cluster representative is picked (as before) to be the Boolean OR of the DSs in the cluster.

1) *Random Permutation Signatures (RPS)*: We illustrate this method with an example. Assume that $n = 256$ and $m = 20$. Let P_1, P_2 , and P_3 be random permutations of $0 \dots 255$. Given a DS D , let D_1, D_2 , and D_3 be respectively the P_1, P_2 , and P_3 permuted versions of D . Let the index of the first non-zero entry, the *min-index* in D_1, D_2 , and D_3 be respectively l_1, l_2 , and l_3 . The signature of the DS D is the triple $(l_1, l_2, l_3/16)$, where $l_3/16$ is the integer part of the quotient upon dividing l_3 by 16 (the number of distinct signatures possible is $256 \times 256 \times 16 = 2^{20}$, i.e., 2^m).

The min-index may alternately be chosen to be the index of the first zero entry. We denote the zero or one policy by referring to the min-zero-index or the min-one-index respectively.

2) *Subset Intersection Signatures (SIS)*: We pick m subsets of $0 \dots n - 1$; the m bits of the signature of a DS set are determined by the intersection of the DS with the corresponding subset.

Both the above methods can exploit any knowledge of the distribution of the expected DSs. For example, if we expect the DSs to have very few locations, then with the subset intersection method, each of the m subsets can be chosen with exactly that many number of ones. Also, if we expect the sets to have many ones, then we can choose more random permutations and pick fewer bits from each each min-index; for example, m subsets and only the most significant bit from each of the min-indices.

D. Combining methods:

Periodically, we spawn an offline method to compute the LTDT for the current set of connections supported. The question arises as to how to add new connections that arrive while the offline algorithm is still running.

We divide the LTDT into two: the offline and online sections. Periodically, the currently active connections are spawned off to run a more expensive algorithm offline. While the offline algorithm runs, new connections are established in the online section and tear-downs are applied to the section the connection resides in. When the offline algorithm completes the offline section of the LTDT is programmed; the online section is programmed with only the new connections that were established while the offline algorithm ran.

VII. EXPERIMENTAL RESULTS

The focus of our experiments is to investigate if our heuristics perform well on a wide variety of inputs. We evaluate our algorithms on three classes of inputs:

- Whole universe of inputs, i.e., when $N = 2^n$
- Random inputs

n	m	greedy	$\frac{n-m}{2n}$	RPS	2-CCs
14	4	0.34	0.36	0.46	0.35
16	8	0.21	0.25	0.45	0.23
20	10	0.20	0.25	0.46	0.22

TABLE I

ABW OF GREEDY ROW CLUSTERING, COLUMN CLASSIFICATION, RANDOM PERMUTATION SIGNATURES, AND 2 COLUMN-CLUSTERS ON UNIVERSE OF INPUTS

- Preclustered inputs: inputs for which we know there is a good clustering.

A. Column classification and universe of inputs—all DSs

The universe of inputs, i.e., all 2^n DSs, while not a very likely input instance, is a good sanity/litmus test for our methods as we show. A special case of the subset intersection signatures method is when all the m subsets are singletons and distinct. In this special case, the subset intersection signatures method reduces to *column classification*, the signature of a DS is just the value of the DS over the chosen m columns. Column classification is not only a simple method for clustering but also an attractive heuristic in several cases as we show below.

For the universe of inputs, classified into clusters based on any m columns, each cluster will have an equal number of DSs, $2^n/M$ (recall $M = 2^m$). The question arises if we can do any better than this:

Theorem 7.1: Consider an MCS instance where all the possible 2^n DSs appear, the number of ASs M is a power of 2, and $\log M \leq n$. Then the cost of any M -clustering is at least $2^n(n/2 - \log M)$.

Proof: The proof is in the Appendix. ■

Theorem 7.1 says the cost savings when all the possible DSs arise (with weight one) is no greater than $2^n \log M$. Whereas, with the $\log M$ columns based classification the savings is $2^n(m/2)$ (on average we save $m/2$ ones per DS; so the total cost is $2^{n-1}(n - m)$). The achieved savings is of the order predicted by the upper bound of Theorem 7.1 (off by a factor of 2).

For our experiments, we measure the performance of all algorithms in terms of the *average bandwidth waste (ABW)*:

$$\text{cost}/Nn \tag{1}$$

To recognize the quantity in (1) as the ABW, recall cost is the total wasted bandwidth, i.e., the total number of zeros that got transformed to ones. Therefore cost/N is the average wasted bandwidth per DS. So, cost/Nn , is the average wasted bandwidth per DS as a fraction of n , i.e., the number of zeros that got transformed to ones as a fraction of n .

For the universe of inputs, on average there are $n/2$ zeros per DS. Column-classification “saves” $m/2$ of these. Therefore, the ABW achieved by column-classification is $(n - m)/2n$.

Table I compares the ABW on the universe of inputs for greedy row clustering, column classification, RPSs, and 2-CCs.

For RPSs, we employ min-zero-indices and as many bits from each min-index till we have m bits of signature. For example, for the third row in Table I where $n = 20$ and $m = 10$, we pick all 5 ($\lceil \log n \rceil$) bits each from the min-zero-indices of the first and second permutations and no bits from the other eight min-zero-indices. Since the average fraction of zeros per DS is $1/2$, the $ABW \leq 0.5$. Greedy row-clustering and 2-column clusters perform better than column-classification; RPSs yield the least improvement.

B. Random inputs with Bernoulli IID destinations

We consider random DSs where each destination is turned on independently with probability $p \in (0 \dots 1)$. We were able to bound the cost achievable by any clustering:

Theorem 7.2: Given N DSs over n destinations with each destination chosen independently and with probability p and M possible clusters. Then

- Any clustering can achieve no more savings over the one-cluster cost than $O(1/p(N \log M + Mn \log n))$ and
- There is a clustering which achieves a cost savings of $\Omega((1-p)(N \log M + Mn))$.

The lower bound of Theorem 7.2 is obtained by picking the better of classification by $\log M$ bits or placing $M - 1$ DSs in clusters by themselves and all other DSs in the remaining cluster ($(1-p)$ is the fraction of zeros out of the n locations in a DS). The upper bound on the cost savings is more difficult to see (proved in extended version of paper). When $N \gg M$ (as might be in practice), the $N \log M$ term will dominate and for (say) $p = 0.5$ the lower and upper bounds will match. Also, notice for lower p , as might be expected the upper bound is higher.

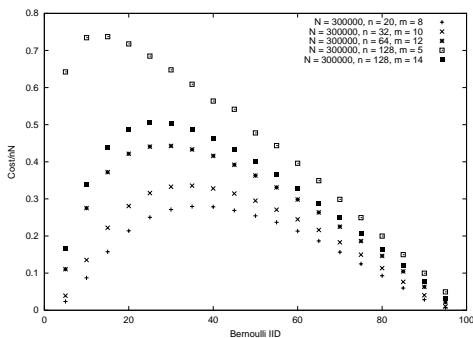


Fig. 2. Plot of ABW on Y-axis versus Bernoulli probability $p * 100$ on X-axis

Figure 2 shows a plot of the ABW obtained by running greedy row clustering versus the Bernoulli probability. Note that $ABW \leq (1-p)$ —the fraction of n that are zeros. When $(1-p)$ is high there is more at stake and the ABW is higher. When the number of sets per cluster is high (e.g., $n = 128$, $m = 5$), the curve is higher also.

$$\frac{\text{cost}}{Nn} \quad (2)$$

$$1 - p - \frac{(1-p)m}{n}$$

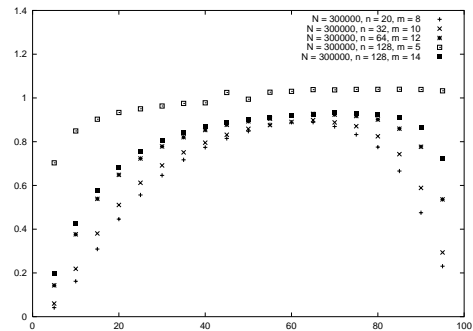


Fig. 3. Plot of (2) on Y-axis versus Bernoulli probability $p * 100$ on X-axis

prob	greedy	2-CCs	RPS	SIS
0.1	0.241	0.325	0.846	0.441
0.3	0.427	0.471	0.688	0.563
0.5	0.354	0.379	0.499	0.427
0.7	0.220	0.247	0.300	0.269
0.9	0.061	0.070	0.100	0.095

TABLE II

ABW OF GREEDY ROW CLUSTERING (COL II), 2 COLUMN CLUSTERS (COL III), RANDOM PERMUTATION SIGNATURES (COL IV), AND SUBSET INTERSECTION SIGNATURES (COL V) VS PROBABILITY (COL I) FOR IID DISTRIBUTED DSs: $N = 100,000$, $n = 64$, AND $m = 12$

The formula in (2) represents the ratio of the ABW to the ABW obtained by column classification with m columns. Recall, that the ABW is the number of zeros as a fraction of n that are turned into ones. For Bernoulli IID inputs, $1-p$ is the fraction of n that are zeros. $1-p - \frac{(1-p)m}{n}$ is the fraction of n that are the zeros not turned into ones by column classification with m columns.

The graph in Figure 3 shows plots of the quantity in (2) versus the Bernoulli probability p for various values of the number of: DSs (N), destinations (n), and clusters ($M = 2^m$). It is noteworthy that the greedy algorithm outperforms column-classification; when the average number of sets per cluster, N/M , is high, we can do no better than column-classification (as predicted by Theorem 7.2).

Finally, in Table II, we compare the ABW obtained with various methods. Greedy row-clustering performs best; 2-column clusters performs next best. With SIS: we generate each signature set with $1/p$ ones where p is the probability of one in the DS so that the probability of intersection with a DS is roughly a constant. For RPSs, for lower probabilities, we pick more bits from each min-one-index and for higher probabilities, fewer bits from each min-one-index. RPSs yield almost no savings; SISs perform slightly better.

C. Preclustered inputs

We generate clustered inputs— M ASs with destinations turned on Bernoulli IID with probability p_1 and N/M DSs per AS with destinations in the AS picked in the DS IID

I	II	III	IV	V	VI	VII	VIII	IX
30	30	0.206	0.395	0.232	0.546	0.828	0.615	0.91
30	70	0.088	0.171	0.082	0.601	0.743	0.656	0.79
50	50	0.248	0.433	0.248	0.604	0.711	0.655	0.75
50	70	0.149	0.262	0.144	0.556	0.627	0.603	0.65
70	70	0.210	0.329	0.207	0.451	0.493	0.488	0.51

TABLE III

ABW WASTED ON PRE-CLUSTERED INPUTS WITH $N = 100,000$, $n = 64$, & $\log_2 M = 6$: COL I: $p_1 * 100$; COL II: $p_2 * 100$; COL III: EXPECTED VALUE FOR ABW; COL IV: GREEDY M -CLUSTERING; COL V: GREEDY $4M$ -CLUSTERING; COL VI: 2 COLUMN-CLUSTERS; COL VII: RPS; COL VIII: SIS; COL IX: ONE-CLUSTER ABW ($1 - p_1 * p_2$)

with probability p_2 —and see how greedy row clustering performs. The greedy algorithm in seeding the M clusters with a DS each will miss M/e of the clusters (its akin to dropping M balls into M bins). We therefore set the target number of clusters for the greedy algorithm at $4M$ and reduce the number of clusters from $4M$ to M by what we call the *two-greedy* algorithm. The two-greedy algorithm involves repeatedly picking the best pairwise merge until the number of clusters left is M , and has complexity $O(M^2 \log M)$. Since typically $N \gg M$, the overall algorithm still has complexity $O(NM)$. We are currently implementing the two-greedy phase of our overall algorithm and hence report the $4M$ -clustering cost in Table III. The observed results match our analytical predictions—setting more initial clusters makes a difference in the discovery of the desired clustering. 2-column clusters performs next best; the signature methods (RPS & SIS) yield some improvement over the maximum ABW (column IX).

VIII. SUMMARY & CONCLUSION

We studied the problem of multicast destination label assignment so that we minimally supercast. We formalized the problem as a clustering problem and proved that the corresponding combinatorial optimization problem is NP-complete and hard to approximate as well. The key to clustering with cost savings is columns of all zeros. We have presented a range of methods to cluster DSs while minimizing the amount of supercast. The methods with lesser run time can support a greater multicast connection establishment rate. The more time consuming methods can be invoked offline to compute better solutions. Our algorithms have been implemented and are being deployed in our product. In the context of the RPSs method, more experiments are necessary to refine the choice of min-index policy and number of bits to extract from each min-index to form the signature.

The problem has been previously considered by Marsan et al [1]. Our computational complexity characterization results are new and comprehensive. We have presented several effective and novel methods. All our methods perform a clustering and choose the cluster representative as the Boolean OR of the DSs in the cluster. Subset intersection and random permutation signatures are interesting constant time methods. The simple structured code of [1] can be viewed as a special case of subset

intersection signatures (random permutation signatures does not appear to be comparable to any of the methods of [1]). Our best performing method—the two-greedy algorithm—is more general than the UAS scheme—the best scheme of [1]—and performs better than the UAS scheme. Our column clustering scheme is a good quality for time trade-off versus greedy row clustering; and performs better than the constant time signature methods.

An extended version of this paper can be obtained either by emailing the authors or on the www [11].

REFERENCES

- [1] M. G. A. Marsan, F. M. Chiussi, A. Francini, G. Galante, and E. Leonardi, "Compression of Multicast Labels in Large IP Routers," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 4, pp. 630–641, 2003.
- [2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman and Company, 1979.
- [3] P. Brucker, "On the complexity of clustering problems," in *Optimization and operations research*, R. Henn and B. Korte and W. Oetli, Ed. Springer-Verlag, 1977, pp. 45–54.
- [4] N. Megiddo and K. J. Supowit, "On the complexity of common geometric location problems," *SIAM Journal of Computing*, vol. 13, pp. 182–196, 1984.
- [5] P. Berkhin, "Survey of clustering data mining techniques," Accrue Software, San Jose, CA, Tech. Rep., 2002. [Online]. Available: citeseer.nj.nec.com/berkhin02survey.html
- [6] X. Wu, "Color quantization by dynamic programming and principal analysis," *ACM Transactions on Graphics (TOG)*, vol. 11, no. 4, pp. 348–372, 1992.
- [7] R. Shamir and R. Sharan, "Algorithmic approaches to clustering gene expression data," in *Current Topics in Computational Biology*, T. Jiang, T. Smith, Y. Xu, and M. Q. Zhang, Eds. MIT press, 2001. [Online]. Available: citeseer.nj.nec.com/shamir01algorithmic.html
- [8] R. Peeters, "The maximum edge biclique problem is np-complete," *Discrete Appl. Math.*, vol. 131, no. 3, pp. 651–654, 2003.
- [9] N. McKeown, V. Anantharam, and J. C. Walrand, "Achieving 100% throughput in an input-queued switch," in *INFOCOM (1)*, 1996, pp. 296–302. [Online]. Available: citeseer.nj.nec.com/170185.html
- [10] U. Feige, "Relations between average case complexity and approximation complexity," in *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*. ACM Press, 2002, pp. 534–543.
- [11] P. Bhargava, S. C. Krishnan, and R. Panigrahy, "Efficient multicast on a terabit router," <http://alumni.eecs.berkeley.edu/~krishnan/hoti04.html>, Tech. Rep., 2004.

APPENDIX

Proof of Theorem 7.1: Let c_i be the number of zeros in the assigned set for i^{th} cluster and X_i be the number of DSs mapped to the i^{th} cluster. Then $X_i \leq 2^{n-c_i}$. The cost of placing all the DSs in the same cluster is $(n/2)2^n$. Let Γ be the cost savings over the one-clustering by using M clusters. $\Gamma = \sum_{i=1}^M c_i X_i$, where $\sum_{i=1}^M X_i = 2^n$.

Let $x_i = X_i/2^n$ be the fraction of destination sets in cluster i . Then,

- $\sum_{i=1}^M x_i = 1$,
- $x_i \leq 2^{-c_i}$, and
- $c_i \leq -\log x_i$.

Thus $\Gamma/2^n \leq \sum_{i=1}^M -x_i \log x_i$. Since the function $-x \log x$ is concave, Γ is maximized when all x_i are equal. Therefore, $\Gamma \leq M2^n(-1/M)(-\log M)$, i.e., $\Gamma \leq 2^n(\log M)$. Deducting the costs savings Γ from the one-cluster cost yields the desired lower bound on the cost of any M -clustering.