Scaling Construction of Low Fan-out Overlays for Topic-based Publish/Subscribe Systems

Version 1.1. Updated Dec, 2010.

Chen Chen
Department of Electrical and Computer Engineering
University of Toronto
chen@msrg.utoronto.ca

Roman Vitenberg
Department of Informatics
University of Oslo, Norway
romanvi@ifi.uio.no

Hans-Arno Jacobsen
Department of Electrical and Computer Engineering
Department of Computer Science
University of Toronto
jacobsen@eecg.toronto.edu

Abstract

It is a key challenge and fundamental problem in the design of distributed publish/subscribe systems to construct the underlying dissemination overlay. In this paper, we focus on effective practical solution for the MinMax-TCO problem: Create a topic-connected pub/sub overlay in which all nodes interested in the same topic are organized in a directly connected dissemination sub-overlay while keeping the maximum node degree to the minimum.

Previously known solutions provided an extensive analysis of the problem and an algorithm that achieves a logarithmic approximation for MinMax-TCO. Yet, they did not focus on efficiency of the solution or feasibility of decentralized operation that would not require full knowledge of the system. Compared to these solutions, our proposed algorithm produces an overlay with marginally higher degrees. At the same time, it has drastically reduced runtime cost, which is corroborated by both theoretical analysis and empirical evaluation. The latter shows a speedup by a factor of more than 25 on average for typical pub/sub workloads.

I. Introduction

We are witnessing an increasingly widespread use of the publish/subscribe (pub/sub) communication paradigm in the design of large-scale distributed systems. Pub/Sub is regarded as a technology enabler for a loosely coupled form of interaction among many publishing data sources and many subscribing data sinks. Many applications report benefits from using this form of interaction, such as application integration [1], financial data dissemination [2], RSS feed distribution and filtering [3], [4], and business process management [5]. As a result, many industry standards have adopted pub/sub as part of their interfaces. Examples of such standards include WS Notifications, WS Eventing, the OMG's Real-time Data Dissemination Service, and the Active Message Queuing Protocol.

In pub/sub, subscribers convey their interests in receiving messages and publishers disseminate publication messages. The language and data model for subscriptions and publications vary across systems. In this paper, we focus on the topic-based pub/sub model. In a topic-based system, publication messages are associated with topics and subscribers register their interests in receiving all messages published on topics of interest. Many commercial systems follow this design. For example, TIBCO RV [2] has been used extensively for market data feed dissemination and Google's GooPS [1] and Yahoo's YMB [6] constitute the distributed message exchange for Web-based applications operating worldwide.

In a distributed pub/sub system, so called pub/sub brokers, often connected in a federated manner as an application-level overlay network, efficiently route publication messages from data sources to sinks. The distributed design was introduced to address pub/sub system scalability. The overlay of a pub/sub system directly impacts the system's performance and the message routing cost. Constructing a high-quality broker overlay is a key challenge and fundamental problem for distributed pub/sub systems that has received attention both in industry [1], [6] and academia [7], [8], [9], [10], [11], [12].

In [7], the authors defined the notion of topic connectivity, which informally speaking means that all nodes (i.e., pub/sub brokers) interested in the same topic are organized in a connected dissemination sub-overlay. This property ensures that nodes not interested in a topic would never need to contribute to disseminating information on that topic. Publication routing atop such overlays saves bandwidth and computational resources otherwise wasted on forwarding messages of no interest to the node. It also results in smaller routing tables.

An additional desirable property for a pub/sub overlay is to have a low node degree. High node degrees increase the probability of hotspots and aggravate the impact of node failures on the system. A node with a high number of adjacent links has to maintain those links (i.e., monitor the links and the neighbors [7], [9]). While overlay designs for different applications might be principally different, they all share the strive for maintaining bounded node degrees, whether in DHTs [13], wireless networks [14], or for survivable network design [15].

Unfortunately, the properties of topic-connectivity and low node degree are at odds with each other. Intuitively, a sparse overlay is unlikely to be topic-connected while a dense overlay is suboptimal with respect to the node degree. In light of this dichotomy, Onus and Richa [8] introduced the fundamental MinMax-TCO publish/subscribe problem: *Build a topic-connected* *overlay (TCO) such that the overlay degree (the maximal degree of any node in the overlay) is minimal.* Onus and Richa proved that there exists no efficient solution for overlay construction that guarantees constant-factor approximation for MinMax-TCO (unless P=NP). In face of this impossibility result, the authors propose the MinMax-ODA algorithm and a neat proof that it achieves logarithmic approximation [8].

While the results of [8] establish a fundamental baseline for any MinMax-TCO algorithm and logarithmic approximation works sufficiently well in most cases, it is vital for a practical solution to consider a number of additional design issues. The running time of MinMax-ODA is $O(|V|^4|T|)$ wherein $|V|$ is the number of nodes and $|T|$ is the number of topics in the system. This makes the overlay construction prohibitively expensive. Furthermore, MinMax-ODA is centralized and requires complete knowledge of all the nodes in the system and their interests.

The main contribution of this paper is the design of a MinMax-TCO solution that focuses on efficiency and alleviates the above limitations. In its core lies a divide-and-conquer approach to the problem: We partition the set of nodes into subsets, build a TCO for each subset, and combine all TCOs into a global overlay. The appeal of this scheme is in the substantially faster TCO construction for each subset of nodes that requires only partial knowledge about the nodes within the partition. Since the creation of TCOs for different partitions is independent, the process can be parallelized and decentralized. Yet, in order to apply this approach we need to overcome a number of obstacles.

The first challenge is comprised in the impact of partitioning on the quality of the solution. We show that the minimal overlay degree is very sensitive to partitioning and it may increase by up to a factor of $\Theta(|T|)$ in the worst case. Our solution is based on the study of workloads in existing pub/sub systems and the observation that in practical pub/sub deployments, only a relatively small number of nodes might be interested in a large number of topics [3]. We formalize this as an assumption and optimize our solution for this case. This assumption does not simplify the MinMax-TCO problem: the number of bulk subscribers is still too large to make any brute force solution around the impossibility result effective. Furthermore, it does not reduce the running time of MinMax-ODA sufficiently for practical applications. Yet, it allows us to come up with a partitioning scheme for which the divide-and-conquer approach retains the logarithmic upper bound on the overlay degree provided by MinMax-ODA.

Next, we devise an algorithm for the combine phase. Our first solution is an adaptation of the MinMax-ODA algorithm along with the proof that the adapted algorithm preserves the approximation ratio. This solution serves as a baseline for analyzing performance, identifying bottlenecks and weaknesses, and devising more advanced algorithms. Unfortunately, it does not improve the running time of MinMax-ODA. Furthermore, it still requires global knowledge about the interests of each node.

To address this issue, we observe that not all nodes need to participate in the combine phase. In each partition, we can select a number of representative nodes so that their combined interest covers the interest of all nodes in the partition. We show that if the combine phase is only performed on the representative nodes (one representative set from each partition), then the resulting overlay will still be topic-connected. Running the combine phase only on representative nodes drastically improves the running time and eliminates the need for a central point of control that possesses complete knowledge about the system. At the same time, it may have a profound impact on the overlay degree unless we select representative nodes in a careful and controlled way. We show how to perform this selection so as to tread the balance between overlay degree and running time.

We evaluated our solution through a series of simulations on characteristic pub/sub workloads with up to $8\,000$ nodes and $1\,000$ topics. The results indicate that on average, our solution requires less than 4.0% of the running time of known state-of-the-art algorithms while yielding an insignificant increase in the maximum node degree of 2.0. While we did not analyze space complexity or measure the program footprint, the improvements in this respect are also noteworthy: For the same pub/sub workload distribution with $8\,000$ nodes and under the same environmental settings, our solution was taking less than a minute to construct the overlay whereas known state-of-the-art algorithms would experience memory-related problems (with $14GB$ of RAM allocated).

II. Related Work

Traditionally, research in the area of distributed pub/sub systems has been focusing on the efficiency and scalability of message dissemination from numerous publishers to a large number of subscribers [16], [17], [18], [19]. A more recent research direction is to consider the fundamental properties of the underlying overlays for pub/sub [7], [8], [9], [10], [11], [12], [20]. This is the direction we pursue in this paper.

Topic-connectivity is explicitly stated as a desirable property for pub/sub overlays in [7], [8], [9], [12], [21]. A number of additional topic-based pub/sub systems (e.g., [17], [22]) construct an overlay per-topic thereby attaining topic connectivity without explicitly discussing this property. A related concept of creating overlay links according to node interests has been explored in [11], [20], [23], [24], [25].

The authors of [7] introduced the Minimum Topic Connected Overlay (MinAvg-TCO) problem, which aims at constructing a topic-connected overlay with minimum number of edges, i.e., minimizing the average node degree. They proved the problem is NP-complete and presented a greedy algorithm which achieves a logarithmic approximation ratio for the average node degree.

In our previous work [12], a divide-and-conquer algorithm is developed for MinAvg-TCO, which dramatically improves the running time at the expense of a minor increase in the average node degree.

The approach of this paper is different from [12] in several significant ways. The underlying MinMax-ODA algorithm exhibits principally different behavior compared to the greedy MinAvg-TCO solution of [7], which leads to different analytical results with respect to both node degree and running time. The MinMax-TCO problem itself is also different from MinAvg-TCO. In particular, we show that the maximum node degree is much more sensitive to partitioning and other elements employed in the divide-and-conquer approach compared to the average node degree. Key elements of our solution for MinAvg-TCO such as the coverage set, produce inadequate results for MinMax-TCO. In order to address this challenge, we developed new techniques, such as the division of nodes into bulk and lightweight subscribers and a representative set with a coverage factor.

The motivation for defining MinMax-TCO in [8] is that existing algorithms for MinAvg-TCO may produce an overlay in which edges are unevenly distributed across nodes. The authors point out that the maximum node degree could be as bad as $\Theta(|V|)$ compared to that of the optimal solution. More recently, Onus and Richa [9] introduced another problem, Low-TCO, which simultaneously considers average and maximum node degree. The solution for Low-TCO proposed in [9] achieves a sub-linear approximation on both maximum and average node degrees.

III. Background

In this section we present some definitions and background information essential for the understanding of the algorithms developed in this paper.

Let V be the set of nodes and T be the set of topics. The interest function Int is defined as $Int : V \times T \rightarrow \{true, false\}$. Since the domain of the interest function is a Cartesian product, we also refer to this function as an interest matrix. Given an interest function Int, we say that a node v is interested in some topic t if and only if $Int(v,t) = true$. We then also say that v subscribes to t. The topic set which the node v subscribes to is denoted as T_v, and we call $|T_v|$ the *subscription size* of node v.

An overlay network $G(V,E)$ is an undirected graph over the node set V with the edge set $E \subseteq V \times V$. Given an overlay network $G(V,E)$, an interest function Int, and a topic $t \in T$, we say that a subgraph $G_t(V_t,E_t)$ of G is *induced* by t if $V_t = \{v \in V | Int(v,t)\}$ and $E_t = \{(v,w) \in E | v \in V_t \wedge w \in V_t\}$. An overlay G is called *topic-connected* if for each topic $t \in T$, the subgraph G_t of G induced by t contains at most one connected component.

Beside topic-connectivity, it is important to keep the degrees of the nodes in the overlay low. Onus and Richa [8] introduced the MinMax-TCO problem for minimizing the maximum degree in a topic-connected overlay. The formal definition of the problem is as follows.

Definition 1. *MinMax-TCO(V,T,Int): Given a set of nodes V, a set of topics T, and the interest function Int, construct a topic-connected overlay network $TCO(V,T,Int,E)$ with the smallest possible maximum node degree.*

MinMax-TCO was proven to be NP-complete, and it can not be approximated by a polynomial time algorithm within a constant factor unless P=NP [8]. Onus *et al.* proposed the MinMax-ODA algorithm, which always delivers a TCO that has a maximum node degree within at most $\log(|V||T|)$ times the minimum possible maximum node degree for any TCO. Here, we refer to the MinMax-ODA algorithm by the shorter name, GM-M (Greedy algorithm for MinMax-TCO), for consistency with our notation. GM-M is specified in Algorithm 1 and operates in a greedy manner as follows: It starts with an empty set of edges and iteratively adds carefully selected edges one by one until topic-connectivity is attained. The edge selection criterion is as follows: If there exist edges whose addition to the overlay does not increase the maximum node degree, the algorithms picks an edge with the largest contribution from the set of all such edges. Otherwise, an edge with the largest contribution among all edges is selected. The contribution of an edge is defined as the number of topic-connected components reduced by adding the edge to the current overlay.

Algorithm 1 Greedy algorithm for MinMax-TCO

```
GM-M($I(V,T,Int)$)
Input: $I(V,T,Int)$
Output: A topic-connected overlay $TCO(V,T,Int,E_{\mathsf{GMM}})$
1: $E_{pot} \leftarrow \emptyset$
2: for all $e = (v,w)$ s.t. $(w,v) \notin E_{pot}$ do
3:     add $e$ to $E_{pot}$
4: $E_{new} \leftarrow$ buildMMEdges($V,T,Int,\emptyset,E_{pot}$)
5: return $TCO(V,T,Int,E_{new})$
```

Algorithm 2 Overlay Construction for GM-M Algorithm

```
buildMMEdges($V, T, Int, E_{cur}, E_{pot}$)
Input: $V$, $T$, $Int$, $E_{cur}$, $E_{pot}$
    // $E_{cur}$: Set of current edges that exist in the overlay
    // $E_{pot}$: Set of potential edges that can be added
Output: Edge set $E_{new}$ that combined with $E_{cur}$, forms a TCO
 1: $E_{new} \leftarrow \emptyset$
 2: for all $e = (v, w) \in E_{pot}$ do
 3:     $contrib(e) \leftarrow |\{t \in T | Int(v,t) \wedge Int(w,t) \wedge$ $v$, $w$ belong to different connected components for $t$ in $G(V, E_{cur})\}|$
 4: while $G(V, E_{new} \cup E_{cur})$ is not topic-connected do
 5:     $e \leftarrow$ find edge $e$ s.t. $contrib(e)$ is maximal and $e$ increases the maximum degree of $G(V, E_{new} \cup E_{cur})$ minimally
 6:     $E_{new} \leftarrow E_{new} \cup \{e\}$
 7:     $E_{pot} \leftarrow E_{pot} - \{e\}$
 8:     for all $e = (v, w) \in E_{pot}$ do
 9:         $contrib(e) \leftarrow$ update the contribution of a potential edge $e$ as the reduction on the number of topic-connected components which would result
            from the addition of $e$ to $G(V, E_{new} \cup E_{cur})$
10: return $E_{new}$
```

The following results about GM-M were proven in an elegant fashion in [8]:

Lemma 1 (GM-M Approximation Ratio & Running Time)**.** *The overlay network output by Algorithm 1 has a maximum node degree within a factor of* $\log(|V||T|)$ *of the maximum node degree of the optimal solution for* MinMax-TCO(V, T, Int)*. Algorithm 1 has a running time of* $O(|E_{pot}|^2|T|) = O(|V|^4|T|)$*.*

IV. Divide-and-Conquer for MinMax-TCO

Taking into account the GM-M running time of $O(|V|^4|T|)$, the number of nodes is the most significant factor determening the performance and scalability of the solution for MinMax-TCO. In view of this, we devise a divide-and-conquer strategy to solve the problem: (1) *divide* the MinMax-TCO problem into several sub-overlay construction problems that are similar to the original overlay but with a smaller node set, (2) *conquer* the sub-MinMax-TCO problems independently and build sub-overlays into sub-TCOs, and then (3) *combine* these sub-TCOs to one TCO as a solution to the original problem.

In this section we present the design steps and key decisions of the divide-and-conquer approach for MinMax-TCO. We show analytically that the resulting algorithm leads to a significantly improved running time cost and reduced knowledge requirement as compared to the GM-M algorithm. In Section VI, we quantify these improvements empirically.

A. *GM-M as a Building Block for Divide-and-Conquer*

In this section, we analyze the GM-M algorithm in greater depth and derive several new results about its running time. GM-M is employed as a building block in our divide-and-conquer algorithms and it serves as the baseline for our experimentation.

In essence, Algorithm 1 and MinMax-ODA follows the same greedy strategy, which is captured by the generic description in Algorithm 2, and Algorithm 3 offers a detailed implementation how to apply this approach. The algorithm works in phases, where in the i-th phase a collection of edges is added so that along with edges in previous phases $(1, 2, ..., i-1)$ forms i *matchings* of the nodes. Note a matching in a graph is defined as a set of edges without common vertices[?]. There is a slight difference between the GM-M algorithm presented in Algorithm 1 and the MinMax-ODA in [8]: as line 4 of Algorithm 3, GM-M never adds 0-contribution edges to the overlay, whereas MinMax-ODA would firstly add 0-contribution edges to construct a maximal matching, and then discard them at the end. This tiny difference will not affect the maximum node degree in the output overlay, but allow us to develop a more efficient implementation.

Denote the maximum degree of the optimal solution for MinMax-TCO(V, T, Int) by $D_{OPT}(V, T, Int)$. Let TCO_{GMM} be the overlay network produced by Algorithm 1, D_{GMM} be the maximum node degree in TCO_{GMM}, and d_{GMM} be the average node degree in TCO_{GMM}. TCO_{ODA}, D_{ODA} and d_{ODA} follow the same notations for MinMax-ODA algorithm.

We can divide the edge set E_{GMM} into $(D_{\mathsf{GMM}} + 1)$ matchings, and let M_i be the edge set of the i-th matching added to E_{GMM} by Algorithm 1, $1 \leq i \leq (D_{\mathsf{GMM}} + 1)$. Similarly, let S_j be the edge set of the j-th matching added to E_{ODA} by MinMax-ODA algorithm, $1 \leq j \leq (D_{\mathsf{ODA}} + 1)$.

Lemma 2 (Maximum degree approximation ratio)**.** *The overlay network* TCO_{GMM} *output by Algorithm 1 has maximum node degree:* $D_{\mathsf{GMM}} = D_{OPT}(V, T, Int) \cdot \log(|V||T|)$.

Proof: As pointed out before, the only difference between GM-M and MinMax-ODA is that M_i in GM-M does not have to be a maximal matching which S_j in MinMax-ODA always guarantees. However, at the end of each matching adding phase, we can always extend M_i to M'_i by adding 0-contribution edges (so that $(\cup_{j=1}^{i-1} M_j) \cup M'_i$ can be decomposed of i maximal matchings), and then discard edges $M'_i - M_i$ before proceeding to the next phase. Obviously, these ”redundant” actions will not impact the output overlay TCO_{GMM}. However, following exactly the same reasoning for Lemma 5 in [8], we can obtain the following statement:

The edge set M_i' reduces the total number of topic-connected components of $G(V, \bigcup_{j=1}^{i-1} M_j)$ by at least $\frac{1}{3}$ of any optimal matching which reduces by maximum amount.

Further, $M_i' - M_i$ does not contribute to the reduction of the number of topic-connected components, so this following lemma is established:

Lemma 3. *The edge set M_i reduces the total number of topic-connected components of $G(V, \bigcup_{j=1}^{i-1} M_j)$ by at least $\frac{1}{3}$ of any optimal matching which reduces by maximum amount.*

With the assistance of Lemma 3, same techniques for the approximation ratio for MinMax-ODA (Theorem 1 in [8]) could be applied here, and therefor accomplish the complete proof of Lemma 2.

■

Then, we show that the maximum node degree of the overlay produced by GM-M is bounded by the maximum subscription size in the input. Let $|T_V|_{\max} = \max_{v \in V} |T_v|$, d_{GMM} denote the average node degree in TCO_{GMM}, and D_{GMM} denote the maximum node degree in TCO_{GMM}, then

Lemma 4 (Bound on the average degree)**.** *The average node degree of the TCO produced by GM-M is $O(|T_V|_{\max})$.*

Proof: Let C_{start} denote the total number of topic-connected components at the beginning of the GM-M algorithm, C_{end} denote the number at the end.

$$C_{start} = \sum_{v \in V} |\{t \in T | Int(v,t) = \text{true}\}| = \sum_{v \in V} |T_v| \leq |V| \cdot |T_V|_{\max} \tag{1}$$

$$C_{end} = |\{t \in T | \exists v \in V \text{ s.t. } Int(v,t) = \text{true}\}| \geq |T_V|_{\max} \tag{2}$$

GM-M always selects an non-zero-contribution edge e, i.e. $contrib(e) \geq 1, \forall e \in E_{\mathsf{GMM}}$, so

$$|E_{\mathsf{GMM}}| \leq \sum_{e \in E_{\mathsf{GMM}}} contrib(e) = C_{start} - C_{end} \leq (|V| - 1) \cdot |T_V|_{\max}$$

$$\Rightarrow d_{\mathsf{GMM}} = \frac{2 \cdot |E_{\mathsf{GMM}}|}{|V|} \leq 2 \cdot \frac{|V| - 1}{|V|} \cdot |T_V|_{\max} \tag{3}$$

■

Lemma 5 (Bound on the maximum degree)**.** *The maximum node degree of the TCO produced by GM-M is $O(|T_V|_{\max})$.*

In [8], Algorithm 1 is the only entry point for Algorithm 2. This means that E_{cur} is always equal to $\emptyset$ and E_{pot} to $V \times V$ upon the invocation of Algorithm 2. When we adapt GM-M for the combine phase of the divide-and-conquer approach, we need to apply GM-M on a collection of TCOs already produced in the conquer phase. Therefore, we have to extend the analysis of GM-M for the case when E_{cur} is non-empty and E_{pot} is equal to $(V \times V) \backslash E_{cur}$. Let E_{new} be the set of edges returned by Algorithm 2. Denote the maximal degree of $G(V,E)$ by $D(V,E)$, then, the following result holds:

Lemma 6. *If invoked on V, T, Int, E_{cur}, and E_{pot} such that $E_{cur} \cup E_{pot} = V \times V$, Algorithm 2 outputs E_{new} such that*
(a) $G(V, E_{cur} \cup E_{new})$ is topic-connected and
(b) the maximum node degree $D(V, E_{cur} \cup E_{new})$ is bounded by $O(D(V, E_{cur}) + D_{OPT}(V, T, Int) \cdot \log(|V||T|))$.

Lemma 7. *The running time of Algorithm 1 is $O(|V|^2|T|)$.*

Proof: The cost of GM-M algorithm is determined by line 4 of Algorithm 1, which is the cost of Algorithm 3 with $E_{cur} = \emptyset$ and $E_{pot} = V^2$.

Since $E_{cur} = \emptyset$, the of initialization GM-M(Algorithm 4) is dominated by the calculation of each edge in E_{pot} in the for loop of lines 12-15 in Algorithm 4. The complexity of this computation for all potential edges will be $O(\sum_{e=(v,w) \in E_{pot}} |\{t \in T | Int(v,t) \wedge Int(w,t)\}|) = O(|V|^2|T|)$.

The cost of edge selection is determined by line 4 in Algorithm 3. All potential edges are stored in the 2-dimensional array $EdgeContrib$ with $|V||T|$ entries; and each entry, positioned as $EdgeContrib[contrib][deg]$, is pointed to a list of potential edges. As shown in Algorithm 5, it first looks for a non-empty entry in $EdgeContrib$, and then pick an edge from that entry. The runtime cost can be divided into two parts:

1) the cost of finding a non-empty entry in $EdgeContrib$: Using aggregate analysis, finding non-empty entries for a sequence of edges in M_i in each phase, can cost at most $O(|V||T|)$. This is because: in the duration of each matching adding phase, the edge pointer only moves in one-way direction: as lines 1- 6 in Algorithm 5 shows, it travels from one entry

to the immediate one, and never traverse backwards. There are at most $O(|V|)$ matchings, so the total cost of moving pointers to find non-empty entries in $EdgeContrib$ is $O(|V|) \cdot O(|V||T|)$.
2) the cost of picking an edge from that non-empty entry: It takes $O(1)$ to get an edge, and totally there are at most $O(\min\{|V||T|, |V|^2\})$ edges to be added, so the running time is $O(\min\{|V||T|, |V|^2\})$.

All together, the total cost for edge selection is $O(|V|) \cdot O(|V||T|) + O(\min\{|V||T|, |V|^2\}) = O(|V|^2|T|)$.

The cost of updates in Algorithm 6, invoked as a result of adding an edge has two separate parts: (1) the updates of edge contributions in lines 1-11; (2) the updates of node degrees in lines 14-19.

The update of contribution for each individual edge can be performed in $O(1)$, e.g., if the elements stored in entries of $EdgeContrib$ are implemented as a doubly-linked list. In order to calculate the total count of individual edge updates at all iterations, it is sufficient to notice that every update decrements the contribution of the edge by one (lines 3-8). Algorithm 1 starts when the total contribution of all edges is $O(\sum_{e=(v,w)\in E_{pot}} |\{t \in T | Int(v,t) \wedge Int(w,t)\}|) = O(|V|^2|T|)$ and terminates when the contribution of all the edges is reduced to zero. So the total updates of edge contribution takes $O(|V|^2|T|)$.

The update of node degrees when adding an edge is bounded by $O(|V|)$, and there are at most $O(\min\{|V||T|, |V|^2\})$ in the output edge set, so the overall cost of updates of node degrees is $O(\min\{|V|^2|T|, |V|^3\})$.

All together, the runtime cost of Algorithm 1 is $O(|V|^2|T|)$.

■

Algorithm 3 Overlay Construction for GM-M Algorithm

```
buildMMEdges($V, T, Int, E_{cur}, E_{pot}$)
Input: $V$, $T$, $Int$, $E_{cur}$, $E_{pot}$
Output: Edge set $E_{new}$ that combined with $E_{cur}$, forms a TCO
    /* Global data structures:
    * $NodeDegree$: an array with length $|V|$ such that $NodeDegree[v]$ is the degree of node $v$ in $G(V, E_{new} \cup E_{cur})$.
    * $TC\text{-}Nodes$: a 2-dimensional array over $V \times T$ whose elements are subsets of $V$ such that for each $v \in V$, $t \in T$: (1) $Int(v,t) = true$, and (2) for each $w \in TC\text{-}Nodes[v][t]$: $Int(w,t)$ and both $w$ and $v$ belong to the same topic-connected component for $t$.
    * $EdgeContrib$: an 2-dimensional array over $|T| \times |V|$ with elements being sets of edges chosen from $V \times V$. If edge $e(v,w) \in EdgeContrib[c][d]$, then: (1) $e \notin E_{new}$; (2) $c = contrib(e)$, i.e. adding $e$ to the overlay at the current iteration will reduce the number of topic-connected components of $G(V, E_{new} \cup E_{cur})$ by $c(1 \leq c \leq |T|)$; (3) $d = \max\{deg(v), deg(w)\}$, where $deg(v)$ is the degree of node $v$ in $G(V, E_{new} \cup E_{cur})$.
    * $highestContrib$: the highest edge contribution in $E_{pot}$
    * $maximumDegree$: the maximum node degree in $G(V, E_{new} \cup E_{cur})$.
    * $currentContrib$: the edge contribution of currently selected edge.
    * $currentDegree$: the larger node degree of currently selected edge.
    */
 1: initDataStructures()
 2: $E_{new} \leftarrow \emptyset$
 3: while $highestContrib > 0$ /* $G(V, E_{new} \cup E_{cur})$ is not topic-connected */ do
 4:     $e(v,w) \leftarrow$ findMMEdge()
 5:     $E_{new} \leftarrow E_{new} \cup \{e\}$
 6:     $E_{pot} \leftarrow E_{pot} - \{e\}$
 7:     updateDataStructures($e(v,w)$)
 8:     while $highestContrib > 0$ do
 9:         $highestContrib - -$
10: return $E_{new}$
```

Algorithm 4 Data Structure Initialization

initDataStructures()

```
1: for all $v \in V$ do
2:     $NodeDegree[v] \leftarrow 0$
3: $maximumDegree \leftarrow \max\{NodeDegree[v], v \in V\}$
4: for all $v \in V \wedge t \in T$ such that $Int(v,t)$ do
5:     $TC\text{-}Nodes[v][t] \leftarrow \{v\}$
6: for all $e = (v,w) \in E_{cur}$ do
7:     $deg(v)++, deg(w)++$
8:     for all $t \in T$ such that $Int(v,t) \wedge Int(w,t) \wedge TC\text{-}Nodes[v][t] \neq TC\text{-}Nodes[w][t]$ do
9:         $new\_tc\_component\_list \leftarrow TC\text{-}Nodes[v][t] \cup TC\text{-}Nodes[w][t]$
10:        for all $u \in new\_tc\_component\_list$ do
11:            $TC\text{-}Nodes[u][t] \leftarrow new\_tc\_component\_list$
12: for all $e = (v,w) \in E_{pot}$ do
13:     $c \leftarrow |\{t \in T | Int(v,t) \wedge Int(w,t) \wedge$ $v$, $w$ belong to different connected components for $t$ in $G(V, E_{cur})\}|$
14:     $d \leftarrow \max\{deg(v), deg(w)\}$
15:     add $e$ to $EdgeContrib[c][d]$
16: $highestContrib \leftarrow \max\{c | \exists d$ such that $EdgeContrib[c][d] \neq \emptyset\}$
17: $currentContrib \leftarrow highestContrib$, $currentDegree \leftarrow 0$
```

Algorithm 5 Find a MinMax Edge

findMMEdge()

Output: an edge e to be added to E_{new}

```
1: for $contrib \leftarrow currentContrib$ to 1 do
2:     for $deg \leftarrow currentDegree$ to $maximumDegree$ do
3:         if $EdgeContrib[contrib][deg] \neq \emptyset$ then
4:             $e \leftarrow$ some edge from $EdgeContrib[contrib][deg]$
5:             $currentContrib \leftarrow contrib$, $currentDegree \leftarrow deg$
6:             return $e$
7: $e \leftarrow$ some edge from $EdgeContrib[highestContrib][maximumDegree]$
8: $currentContrib \leftarrow highestContrib$, $currentDegree \leftarrow maximumDegree$
9: return $e$
```

Algorithm 6 Date Structures Update

updateDataStructures($e(v,w)$)

```
1: for all $t \in T$ such that $Int(v,t) \wedge Int(w,t) \wedge TC\text{-}Nodes[v][t] \neq TC\text{-}Nodes[w][t]$ do
2:     for all $v' \in TC\text{-}Nodes[v][t] \wedge w' \in TC\text{-}Nodes[w][t] \wedge (v',w') \neq (v,w)$ do
3:         locate $c$ and $d$ such that $(v',w') \in EdgeContrib[c][d]$
4:         delete $(v',w')$ from $EdgeContrib[c][d]$
5:         if $c > 1$ then
6:             add $(v',w')$ to $EdgeContrib[c-1][d]$
7:         else
8:             delete $(v',w')$ from $E_{pot}$
9:     $new\_tc\_component\_list \leftarrow TC\text{-}Nodes[v][t] \cup TC\text{-}Nodes[w][t]$
10:    for all $u \in new\_tc\_component\_list$ do
11:        $TC\text{-}Nodes[u][t] \leftarrow new\_tc\_component\_list$
12: $NodeDegree[v]$++, $NodeDegree[w]$++
13: $maximumDegree \leftarrow \max\{maximumDegree, NodeDegree[v], NodeDegree[w]\}$
14: for all $(x,y) \in E_{pot}$ that is incident on either $v$ or $w$ do
15:     $d_{new} \leftarrow \max\{NodeDegree[x], NodeDegree[y]\}$
16:     if $d < d_{new}$ then
17:         locate $c$ and $d$ such that $(x,y) \in EdgeContrib[c][d]$
18:         delete $(x,y)$ from $EdgeContrib[c][d]$
19:         add $(x,y)$ to $EdgeContrib[c][d_{new}]$
```

B. Divide and Conquer Phases of the Solution

There exist two principal methods to *divide* the nodes: (1) node clustering and (2) random partitioning. Node clustering is oganizing the original node set into groups so that nodes with similar interests are placed in the same group while nodes with diverging interests belong to different groups. Random partitioning assigns each node in the given node set to one of the partitions based on a uniformly random distribution. Once the partitions are determined, the existing GM-M algorithm can be employed to *conquer* the sub-MinMax-TCO problems by determining inner edges used for the construction of the sub-overlays.

The idea of node clustering seems attractive because well-clustered nodes with strongly correlated interests would result in lower maximum node degrees in the sub-TCOs produced by GM-M. The problem with this approach is the high runtime cost of clustering algorithms taking into account the large number of nodes and varying subscription size. Additionally, they require the computation of a "distance" metric among nodes. In our case this translates to calculating pairwise correlations among node interests with significant run time cost implications. It is challenging to fit node clustering into the divide-and-conquer approach so that the latter is still superior to the GM-M algorithm in terms of running time cost. Furthermore, it is difficult to devise an effective decentralized algorithm for node clustering that would not require complete knowledge of V and Int. Finally, node clustering by interests may yield clusters that vary in size depending on the clustering algorithm used. On the other hand, the divide-and-conquer approach performs optimally when partitions are equal-sized and there are no large clusters that stand out.

Algorithm 7 Naive algorithm for *divide* and *conquer* phases

```
Input: $V, T, Int, p$
    // $p$: the number of partitions, $1 \leqslant p \leqslant |V|$.
Output: A list of TCOs $List_{TCO}$, one TCO for each partition
1: $List_{TCO} \leftarrow \emptyset$
2: Randomly divide $V$ into $p$ partitions $V_d$, $d$=$1, 2, ..., p$
3: for $d = 1$ to $p$ do
4:     $Int_d \leftarrow Int|_{V_d}$
5:     $TCO_d(V_d, T, Int_d, E_d) \leftarrow$GM-M($V_d$,$T$,$Int_d$)
6:     add $TCO_d$ to $List_{TCO}$
7: return $List_{TCO}$
```

We choose random partitioning for the divide-and-conquer approach because it is extremely fast, more robust than node clustering, easier to tune, and it can be realized in a decentralized manner. Furthermore, the construction of inner edges for each overlay only requires knowledge of node interests within the overlay. Hence, random partitioning can be oblivious to the composition of nodes and their interests. The number of partitions p is given as an input parameter and each sub-overlay has $k = |V_d| = \frac{|V|}{p}$ nodes, where $d = 1, ..., p$. This equal-sized division is optimal with respect to the running time. The resulting algorithm for the *divide* and *conquer* phases is presented in Algorithm 7.

Unfortunately, random partitioning may place nodes with diverging interest into the same partition thereby reducing the amount of correlation that is present in the original node set. As Lemma 8 shows, this may have a profound effect on the maximum node degree. Consider the overlay $G(V, E)$ for the *conquer* phase produced by this algorithm where $List_{TCO}[d] = TCO_d(V_d, T, Int_d, E_d), E = \cup_{d=1}^{p} E_d$. Then,

Lemma 8. *There is an instance $I(V, T, Int)$ of MinMax-TCO on which the maximum degree $D(V, E)$ of the overlay output by Algorithm 7 is greater by a factor of $\Theta(|T|)$ than the maximum node degree $D_{OPT}(V, T, Int)$ of the optimal solution for $I(V, T, Int)$.*

Proof: Construct an instance $I(V, T, Int)$ of MinMax-TCO as follows: Consider the topic set $T = \{t_1, t_2, ..., t_m\}$ of size $m = 2^h$. Node set V consists of $h + 1 = \log m + 1$ subsets, denoted as $A_i, 0 \le i \le h$; each node subset A_i contains 2^i nodes, i.e., $A_i = \{v_{(i,1)}, ..., v_{(i,2^i)}\}$. Each node $v_{(i,j)}, 1 \le j \le 2^i$ subscribes to a topic set $T_{(i,j)}$ of size $\frac{m}{2^i}$, defined as follows:

$$V = \bigcup_{i=0}^{h} A_i \qquad\qquad T = \{t_1, ..., t_m\}$$

$$A_0 = \{v_{(0,1)}\} \qquad\qquad T_{(0,1)} = \{t_1, ..., t_m\}$$

$$A_1 = \{v_{(1,1)}, v_{(1,2)}\} \qquad\qquad T_{(1,1)} = \{t_1, ..., t_{\frac{m}{2}}\}, T_{(1,2)} = \{t_{\frac{m}{2}+1}, ..., t_m\}$$

$$\vdots \qquad\qquad \vdots$$

$$A_i = \{v_{(i,1)}, ..., v_{(i,2^i)}\} \qquad\qquad T_{(i,j)} = \{t_{\frac{jm}{2^i}+1}, ..., t_{\frac{(j+1)m}{2^i}}\}, 1 \le j \le 2^i$$

$$\vdots \qquad\qquad \vdots$$

$$A_h = \{v_{(h,1)}, ..., v_{(h,m)}\} \qquad\qquad T_{(h,1)} = \{t_1\}, ..., T_{(h,m)} = \{t_m\}$$

The optimal overlay TCO_{OPT} for $I(V, T, Int)$ is shown in Fig.1(a). Its maximum node degree is a constant: $D_{OPT} = 3$. Consider the output of Algorithm 7. Due to the random partitioning in Line 2, there is a chance for generating a partition that consists of nodes in $A_0 = \{v_{(0,1)}\}$ and $A_h = \{v_{(h,1)}, ..., v_{(h,m)}\}$ (see Fig.1(b)). To attain topic-connectivity for this partition, $v_{(0,1)}$ has to be linked to all m nodes in A_h, which makes the node degree of $v_{(0,1)}$ to be $m = |T|$.

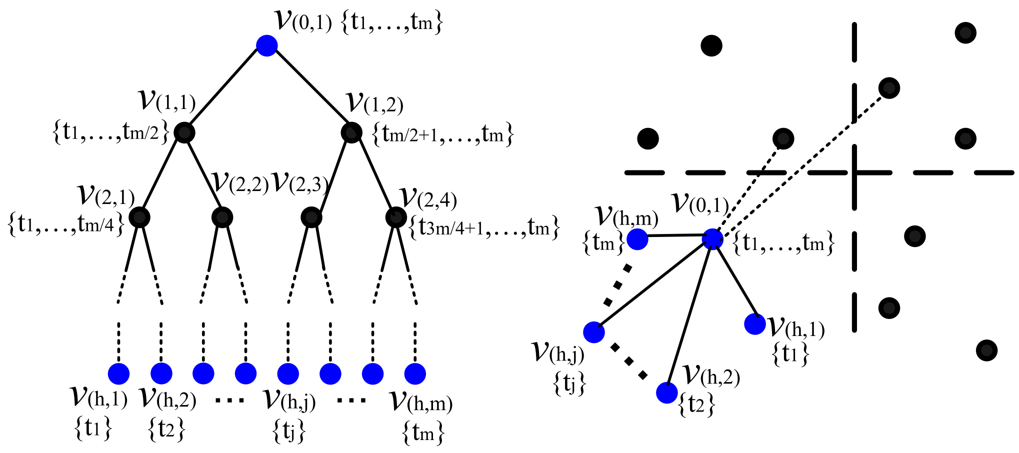


Fig. 1. (a) TCO_{OPT} with D_{OPT} = 3. (b) TCO for random partitioning.

Consequently, the maximum degree $D(V, E)$ of the overlay output by Algorithm 7 for $I(V, T, Int)$ is greater by a factor of $\Theta(\frac{m}{3}) = \Theta(|T|)$ than the maximum node degree $D_{OPT}(V, T, Int)$ of the optimal solution for $I(V, T, Int)$. ■

Essentially, Lemma 8 shows that if we use random partitioning for the divide-and-conquer approach, then the overlay degree for the *conquer* phase alone may exceed the overlay degree for the complete optimal solution by a factor of $\Theta(|T|)$. Furthermore, our empirical validation indicates that not only for a manually constructed worst case but also for the typical pub/sub workloads, random partitioning causes significant increase in the maximum node degree. Fig. 2 illustrates this effect for the default workload defined and motivated in Section VI. It compares the maximum degree for the *conquer* phase produced by Algorithm 7 with the total degree of the overlay produced by GM-M.

This effect of increased maximum degree occurs when a node subscribed to a large number of topics (i.e., a *bulk subscriber*) is placed into the same partition with nodes whose subscriptions are not correlated. Then, such nodes do not benefit from creating a link to each other and need to connect to the bulk subscriber. Our solution to this problem is based on the study of representative pub/sub workloads used in actual applications that are described and characterized in [21]. According to these characterizations, the "Pareto 80–20" rule works for pub/sub workloads: Most nodes subscribe to a relatively small number of topics. The phenomenon of increased maximum degree due to partitioning still exists in such workloads as the example of Lemma 8 indicates. Yet, this observation allows us to devise an effective solution: Following it we provide an algorithm that applies random partitioning only to such *lightweight* subscribers, performs the *conquer* phase for lightweight partitions, and merges them with bulk subscribers at the *combine* phase.

Formally, given an instance $I(V, T, Int)$ for MinMax-TCO, we introduce an additional parameter called bulk subscription threshold η, $\eta \in (0, |T|]$. η determines the division of V into the set of bulk and lightweight subscribers B and L, respectively: $B = \{v : |T_v| > \eta\}$ and $L = \{v : |T_v| \leq \eta\}$. Algorithm 8 applies random partitioning to L, creates a TCO for each of the partitions, and returns a list of these TCOs.

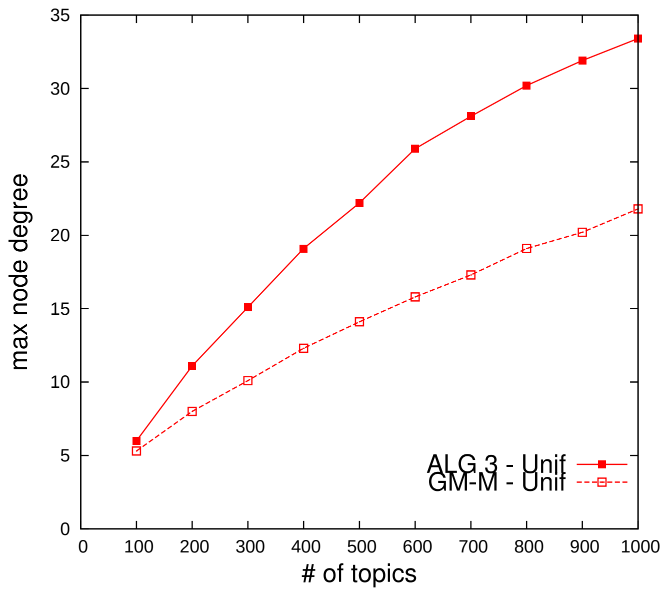


Fig. 2. Poor performance of naive divide-and-conquer

Algorithm 8 *Divide* and *conquer* phases for lightweight nodes

```
conquerLightweight($I(V,T,Int),\eta,p$)
Input: $I(V,T,Int),\eta$
    // $\eta$: the bulk node threshold;
    // $p$: the number of partitions for lightweight nodes.
Output: A list of TCOs $List_{TCO}$, one TCO for each partition
1: $B \leftarrow \{v \in V : |T_v| > \eta\}$, $Int_B \leftarrow Int|_B$
2: $List_{TCO} \leftarrow \emptyset$
3: Randomly divide $L = V - B$ into $p$ partitions $L_d$, $d = 1, ..., p$
4: for $d = 1$ to $p$ do
5:     $Int_d \leftarrow Int|_{V_d}$
6:     $TCO_d(V_d, T, Int_d, E_d) \leftarrow$GM-M($L_d$,$T$,$Int_d$)
7:     add $TCO_d$ to $List_{TCO}$
8: return $List_{TCO}$
```

Lemma **??** can be directly applied to the overlay output by Algorithm 8 to produce a bound on its maximum degree.

Lemma 9 (Bound on the maximum degree for the *conquer* phase)**.** *The maximum node degree of an overlay $G(V,E)$ produced by Algorithm 8 is bounded by the bulk subscription threshold: $D(V,E) = O(\eta)$.*

We now analyze the running time of Algorithm 8.

Lemma 10 (Running time for the *conquer* phase)**.** *The running time cost of Algorithm 8 is $O(\frac{|L|^4|T|}{p^3})$.*

Proof: The loop in Lines 4–7 of Algorithm 9 uses GM-M to build a sub-TCO for each sub-overlay. Each sub-overlay has at most $\frac{|L|}{p}$ nodes so that by Lemma 7, the running time for constructing each sub-TCO is $O(\frac{|L|^4|T|}{p^4})$. Thus, the running time for constructing all p overlays is $O(\frac{|L|^4|T|}{p^3})$. ■

This algorithm is used as a building block for the complete divide-and-conquer solution to MinMax-TCO presented in Section IV-C.

C. Combine Phase of the Solution

Algorithm 9 Divide-and-Conquer with Bulk Nodes and Lightweight Nodes for MinMax

```
DCB-M($I(V, T, Int), \eta, p$)
Input: $I(V, T, Int), \eta$
    // $\eta$: the bulk node threshold;
    // $p$: the number of partitions for lightweight nodes.
Output: A topic-connected overlay $TCO(V, T, Int, E_{\mathsf{DCB}})$
1: $List_{TCO} \leftarrow$ conquerLightweight($I(V, T, Int), \eta, p$)
2: $TCO(V, T, Int, E_{\mathsf{DCB}}) \leftarrow$ combineB&L($V, T, Int, List_{TCO}$)
3: return $TCO(V, T, Int, E_{\mathsf{DCB}})$
```

Algorithm 10 *Combine* B nodes and L nodes greedily

```
combineB&L($V, T, Int, List_{TCO}$)
Input: $V, T, Int, List_{TCO}$
    /* $List_{TCO}$: a list of $p$ node-disjoint TCOs for lightweight nodes: $List_{TCO}[d] = TCO_d(L_d, T, Int_d, E_d)$, $d$=1, ..., $p$. */
Output: A topic-connected overlay $TCO(V, T, Int, E_{DCB})$
1: $B \leftarrow V - \bigcup_{d=1}^{p} L_d$ // $L_d$ is short for $List_{TCO}[d].L_d$
2: $E_{inDCB} \leftarrow \bigcup_{d=1}^{p} E_d$ // $E_d$ is short for $List_{TCO}[d].E_d$
3: $E_{pot\mathsf{DCB}} \leftarrow \{e = (v, w) | (v \in B, w \in V) \wedge (w, v) \notin E_{pot\mathsf{DCB}}\}$
4: $E_{pot\mathsf{DCB}} \leftarrow E_{pot\mathsf{DCB}} \bigcup \{e = (v, w) | v \in L_i, w \in L_j, i < j)\}$
5: $E_{out\mathsf{DCB}} \leftarrow$ buildMMEdges($V, T, Int, E_{in\mathsf{DCB}}, E_{pot\mathsf{DCB}}$)
6: $E_{\mathsf{DCB}} \leftarrow E_{in\mathsf{DCB}} \cup E_{out\mathsf{DCB}}$
7: return $TCO(V, T, Int, E_{\mathsf{DCB}})$
```

In the core of our design for the *combine* phase solution lies the observation that Algorithm 2 can be used to merge the sub-overlays for different partitions and the set of bulk subscribers into a single TCO. To implement this idea, we devise Algorithm 10 that applies Algorithm 2 on a union of the sub-overlays produced at the *conquer* phase by Algorithm 8. Algorithm 9 called DCB-M, presents a complete divide-and-conquer solution for MinMax-TCO.

Lemma 11. (Correctness) *Algorithm 9 is correct: it yields an overlay such that for every topic t, all nodes interested in t are organized in a single connected component.*

Given an instance $I(V, T, Int)$ for MinMax-TCO, let TCO_{DCB} be the TCO produced by Algorithm 9. Denote its maximum node degree as D_{DCB}. There are two types of edges that form the TCO_{DCB}: (1) $E_{in\mathsf{DCB}}$, the inner edges constructed by Algorithm 8, $E_{in\mathsf{DCB}} = \bigcup_{d=1}^{p} E_d$ and (2) $E_{out\mathsf{DCB}}$, the outer edges conjoining bulk subscribers and lightweight node sub-TCOs, which are created in Line 5 of Algorithm 10. The maximum node degree induced by $E_{in\mathsf{DCB}}$ and $E_{out\mathsf{DCB}}$ are denoted as $D_{in\mathsf{DCB}}$ and $D_{out\mathsf{DCB}}$, respectively. It holds that

$$D_{\mathsf{DCB}} \leq D_{in\mathsf{DCB}} + D_{out\mathsf{DCB}}. \tag{4}$$

Equation 4 along with Lemma 9 and Lemma 6 allow us to establish an upper bound on the degree of the overlay produced by DCB-M.

Lemma 12 (Degree bound for DCB-M)**.** *The overlay network output by Algorithm 9 has maximum node degree $D_{\mathsf{DCB}} = O(\eta + D_{OPT}(V, T, Int) \cdot \log(|V||T|))$.*

Corollary 1 (Approximation ratio for DCB-M)**.** *If we regard the bulk node threshold as a constant factor or if the maximum degree $E_{in\mathsf{DCB}}$ of the sub-overlays constructed at the conquer phase is smaller than the maximum degree $D_{OPT}(V, T, Int)$ of the optimal overlay for MinMax-TCO(V, T, Int), then*

$$D_{\mathsf{DCB}} = D_{OPT}(V, T, Int) \cdot O(\log(|V||T|)). \tag{5}$$

If the conditions in Corollary 1 hold, then the DCB-M algorithm achieves the same logarithmic approximation ratio as the GM-M algorithm (Algorithm 1).

Next, we consider the running time of the DCB-M algorithm. Let $\mathbb{T}_{in\mathsf{DCB}}$ and $\mathbb{T}_{out\mathsf{DCB}}$ denote the running time to build $E_{in\mathsf{DCB}}$ and $E_{out\mathsf{DCB}}$, respectively. Let $\mathbb{T}_{\mathsf{DCB}}$ be the total running time cost of Algorithm 9. Then, we obtain the running time of DCB-M with:

Lemma 13 (Running time for DCB-M)**.** *The running time of Algorithm 9 is* $\mathbb{T}_{\mathsf{DCB}} = O(\mathbb{T}_{in\mathsf{DCB}}+\mathbb{T}_{out\mathsf{DCB}}) = O(|T|\cdot(|B||V|+|L|^2)^2)$.

Proof: Following Lemma 10, $\mathbb{T}_{inDCB} = O(\frac{|L|^4|T|}{p^3})$.

$\mathbb{T}_{outDCB}$ is determined by Algorithm 10, whose running time is dominated by the invocation of Algorithm 2 in Line 5. Based on Lemma 7, we have:

$$\begin{aligned}\mathbb{T}_{out\mathsf{DCB}} &= O(|T|\cdot|E_{pot\mathsf{DCB}}|^2)\\ &= O(|T|\cdot(|B||V|+|L|^2)^2)\end{aligned} \tag{6}$$

$$\begin{aligned}\mathbb{T}_{\mathsf{DCB}} &= O(\mathbb{T}_{in\mathsf{DCB}}+\mathbb{T}_{out\mathsf{DCB}})\\ &= O(\mathbb{T}_{out\mathsf{DCB}}) = O(|T|\cdot(|B||V|+|L|^2)^2)\end{aligned} \tag{7}$$

■

In summary, Algorithm 9 has asymptotic performance very similar to that of Algorithm 1, both with respect to the maximum degree and running time. As both the above analysis and experimental evaluations in Section VI indicate, it produces a marginally higher overlay degree at marginally better runtime cost. Furthermore, the *combine* phase still requires complete knowledge of V and Int, which makes decentralization infeasible. These shortcomings motivate the development of an improved solution for the *combine* phase. Our improvement is based on the notion of *Representative Set*, which we explain next.

Given $I(V,T,Int)$ for MinMax-TCO and a topic $t, t\in T$, we denote the set of subscribers to t by $subs(t)$, $subs(t) = \{v|v\in V \wedge Int(v,t)\}$. Then, the notion of a *representative set (rep-set)* is defined as follows:

Definition 2 (Representative set)**.** *Given $I(V,T,Int)$, a* rep-set *with the* coverage factor *λ, denoted as $R(\lambda)$ (or R), is a subset of V such that*

$$|subs(t)|_R| \geq \min\{\lambda, |subs(t)|\}, \forall t\in T \tag{8}$$

A node $r\in R(\lambda)$ is referred to as a representative node (rep-node).

As illustrated in Figure 3, a rep-set is a subset of overlay nodes that represents the interests of all the nodes in the overlay. Each topic of interest is covered by at least λ subscribers in R unless the total number of subscribers to this topic is smaller than λ. The complete node set V is always a rep-set, but there might exist many other rep-sets with much fewer rep-nodes. In essence, these nodes can function as bridges for the purpose of determining cross-TCO connections. Observe that it is possible to attain full topic-connectivity only by using cross-TCO links among rep-nodes for different partitions. Suppose we have a number of TCOs, and each TCO is represented by a rep-set (of a smaller size). To achieve topic-connectivity for a topic $t\in T$, we can just connect nodes from different rep-sets which are interested in t.

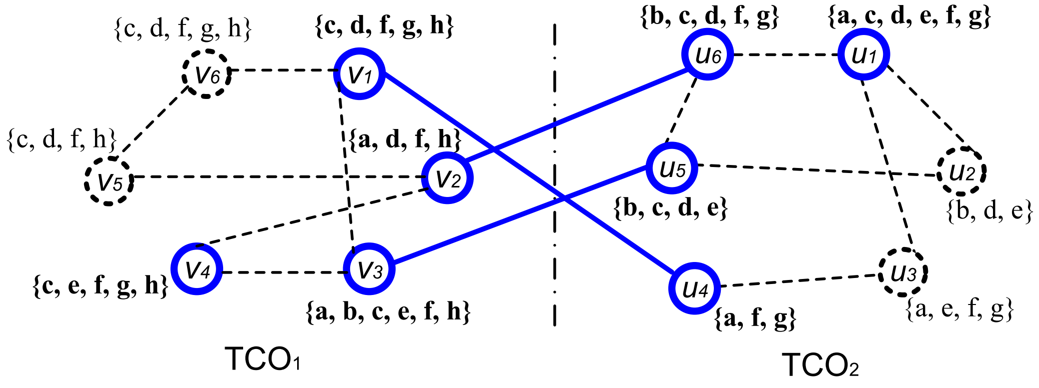


Fig. 3. $R_1 = \{v_1, v_2, v_3, v_4\}$ and $R_2 = \{u_1, u_4, u_5, u_6\}$ are rep-sets with $\lambda = 2$ for TCO_1 and TCO_2 respectively; a complete TCO for all nodes is obtained by adding cross-TCO links between R_1 and R_2.

For typical pub/sub workloads and sufficiently large partitions, minimal rep-sets are several times smaller than the total number of nodes. This leads to significant benefits if we consider only rep-nodes as candidates for cross-TCO links. One, the running time of the overlay construction algorithms discussed in this paper is roughly proportional to the number of nodes up to the fourth degree, therefore our algorithm that only considers rep-nodes runs much faster. Two, calculation of cross-TCO links no longer requires complete knowledge of V and Int, and only a partial view of rep-nodes from rep-sets and their interests is needed. Three, rep-sets of different TCOs can be computed in parallel in a fully decentralized fashion.

At the same time, minimality of rep-sets also has an adverse effect on the maximum overlay degree due to reduced correlation across rep-sets for different TCOs. Revisit the instance $I(V, T, Int)$ of MinMax-TCO described in the example of Lemma 8. Suppose V is divided into two partitions: a partition of bulk subscribers that includes node subsets $A_i, 0 \leq i \leq h-1$ and a partition of lightweight subscribers that includes node subset A_h. The minimal rep-set for the first partition contains a single node $v_{(0,1)}$ whereas the minimal rep-set for the second partition contains all nodes in A_h. If we merge the rep-sets at the combine phase, the degree of $v_{(0,1)}$ will be $|A_h| = \Theta(|T|)$. On the other hand, if we merge the whole partitions, the optimal overlay will be the one depicted in Fig. 1(a) with constant maximum degree.

The difference arises due to the fact that in the former case, $v_{(0,1)}$ serves the focal point for all cross-overlay links while in the latter case, these links are evenly distributed across all the nodes of the first partition. We employ two techniques to prevent the above effect. First, we only use rep-sets for the partitions of lightweight nodes and not for bulk subscribers. This is because the degree of lightweight nodes is bounded by $O(\eta)$ as we later show in Lemma 15 so that the effect is not as significant for lightweight nodes compared to bulk subscribers. Second, we use a coverage factor greater than one to ensure that there are always multiple choices when connecting the nodes for any topic.

We still need to consider, how to efficiently determine a minimal rep-set given V, T, Int, and λ. The problem of computing a minimal rep-set set is equivalent (through a linear reduction) to a variation of the classic NP-complete *Set Cover* problem, in which each item has to be covered by at least λ sets. Algorithm 13 provides a greedy implementation that attains a provable logarithmic approximation [26]. The algorithm starts with an empty rep-set and continues adding nodes to the rep-set one by one until all topics of interest are λ-covered, i.e., covered by at least λ nodes. At each iteration, the algorithm selects a node that is interested in the largest number of topics that are not yet λ-covered.

Algorithm 12 presents the resulting implementation for combining sub-TCOs. The algorithm operates in two phases. First, it determines a rep-set for each sub-TCO. Note that the rep-set for TCO_d does not need to cover all of T_d. It suffices to cover $T_{out\_d} = T_d \bigcap (\bigcup_{i \neq d} T_i)$. Second, the algorithm connects all the nodes in the rep-sets as well as bulk subscribers into a TCO in a greedy manner by using Algorithm 2. Algorithm 11 called DCBR-M, presents our complete solution for MinMax-TCO.

Below, we establish correctness, approximation ratio and running time properties for the DCBR-M algorithm.

Lemma 14 (Correctness)**.** *Algorithm 11 is correct: it yields an overlay such that for every topic t, all nodes interested in t are organized in a single connected component.*

Following the notations for DCB-M algorithm, we denote the TCO produced by Algorithm 11 as TCO_{DCBR} and its maximum node degree as D_{DCBR}. Observe that by operating on a reduced set of nodes at the *combine* phase, the invocation of Algorithm 2 in line 12 of Algorithm 12 solves an instance of MinMax-TCO$(BR, T, Int|_{BR})$ where BR is a union of B and all rep-sets $\bigcup_{d=1}^{p} R_d$. The fact that $D_{\mathsf{DCBR}} \leq D_{in\mathsf{DCBR}} + D_{out\mathsf{DCBR}}$ along with Lemma 9 and Lemma 6 allow us to establish an upper bound on the degree of the overlay produced by DCBR-M.

Lemma 15 (Degree bound for DCBR-M)**.** *The overlay network TCO_{DCBR} output by Algorithm 11 has maximum node degree: $D_{\mathsf{DCBR}} = O(\eta + D_{OPT}(BR, T, Int|_{BR}) \cdot \log(|BR||T|))$.*

According to Lemma 15, if we choose a sufficiently large coverage factor λ so that $D_{OPT}(BR, T, Int|_{BR}) \approx D_{OPT}(V, T, Int)$, then Algorithm 11 will generate a TCO whose maximum node degree is asymptotically the same as that of the TCO output by Algorithm 9.

Algorithm 11 Divide-and-Conquer with Bulk Nodes and Lightweight Rep-nodes for MinMax

```
DCBR-M($I(V,T,Int), \eta, p, \lambda$)
Input: $I(V,T,Int), \eta, p, \lambda$
    // $\lambda$: the coverage factor.
Output: A topic-connected overlay $TCO(V,T,Int,E_{\mathsf{DCBR}})$
 1: $L_{TCO} \leftarrow$ conquerLightweight($I(V,T,Int), \eta, p$)
 2: $TCO(V,T,Int,E_{\mathsf{DCBR}}) \leftarrow$ combineB&LReps($V,T,Int,L_{TCO},\lambda$)
 3: return $TCO(V,T,Int,E_{\mathsf{DCBR}})$
```

Algorithm 12 *Combine* B nodes and L rep-nodes greedily

```
combineB&LReps($V, T, Int, List_{TCO}, \lambda$)
Input: $V, T, Int, List_{TCO}, \lambda$
Output: A topic-connected overlay $TCO(V, T, Int, E_{\mathsf{DCBR}})$
 1: $B \leftarrow V - \bigcup_{d=1}^{p} L_d$ // $L_d$ is short for $List_{TCO}[d].L_d$
 2: $E_{in\mathsf{DCBR}} \leftarrow \bigcup_{d=1}^{p} E_d$ // $E_d$ is short for $List_{TCO}[d].E_d$
 3: for $d = 1$ to $p$ do
 4:     $T_d \leftarrow \bigcup_{v \in L_d} T_v$
 5: for $d = 1$ to $p$ do
 6:     $T_{out\_d} \leftarrow T_d \bigcap (\bigcup_{i \neq d} T_i)$
 7:     $R_d \leftarrow$ getRepSetFromNodes($L_d, T_{out\_d}, Int, \lambda$)
 8: $R \leftarrow \bigcup_{d=1}^{p} R_d$
 9: $BR \leftarrow B \cup R$
10: $E_{pot\mathsf{DCBR}} \leftarrow \{e = (v, w) | (v \in B, w \in BR) \wedge (w, v) \notin E_{pot\mathsf{DCBR}}\}$
11: $E_{pot\mathsf{DCBR}} \leftarrow E_{pot\mathsf{DCBR}} \bigcup \{e = (v, w) | v \in R_i, w \in R_j, i < j)\}$
12: $E_{out\mathsf{DCBR}} \leftarrow$ buildMMEdges($BR, T, Int|_{BR}, E_{in\mathsf{DCBR}}|_{BR}, E_{pot\mathsf{DCBR}}$)
13: $E_{\mathsf{DCBR}} \leftarrow E_{in\mathsf{DCBR}} \cup E_{out\mathsf{DCBR}}$
14: return $TCO(V, T, Int, E_{\mathsf{DCBR}})$
```

Algorithm 13 Determine a representative set for a partition

```
getRepSetFromNodes($V_d, T_{out\_d}, Int, \lambda$)
Input: $V_d$, $T_{out\_d}$, $Int$, $\lambda$
Output: $R_d$: A representative set for $V_d$
 1: Start with $T_{toCover} = T_{out\_d}$ and $R_d = \emptyset$
 2: for all $t \in T_{toCover}$ do
 3:     $N_{toCover}[t] = \lambda$
 4: while $T_{toCover} \neq \emptyset$ do
 5:     $r \leftarrow \arg\min_{v \in V_d - R_d} (\frac{1}{|\{t | t \in T_{toCover} \wedge Int(v,t)\}|})$
 6:     $R_d \leftarrow R_d \cup \{r\}$
 7:     for all $t \in T_{toCover} \wedge Int(r, t)$ do
 8:         $N_{toCover}[t] \leftarrow N_{toCover}[t] - 1$
 9:         if $N_{toCover}[t] = 0$ then
10:             $T_{toCover} \leftarrow T_{toCover} - \{t\}$
11: Return $R_d$
```

Using the same reasoning as in Lemma 10, we can derive the running time cost of DCBR-M.

Lemma 16 (Running time for DCBR-M)**.** *The running time of Algorithm 11 is* $\mathbb{T}_{\mathsf{DCBR}} = O(|T| \cdot ((|B| + |R|)^4 + \frac{|L|^4}{p^3}))$.

Proof:

$$\begin{aligned}\mathbb{T}_{out\mathsf{DCBR}} &= O(|T| \cdot |E_{pot\mathsf{DCBR}}|^2) \\ &= O(|T| \cdot ((|B| \cdot (|B| + |R|) + |R|^2)^2)) \\ &= O(|T| \cdot ((|B| + |R|)^4)) \end{aligned} \tag{9}$$

$$\begin{aligned}\mathbb{T}_{\mathsf{DCBR}} &= O(\mathbb{T}_{out\mathsf{DCBR}} + \mathbb{T}_{in\mathsf{DCBR}}) \\ &= O(|T| \cdot ((|B| + |R|)^4 + \frac{|L|^4}{p^3})) \end{aligned} \tag{10}$$

■

Lemma 16 shows that if representative sets are significantly smaller than partitions, the bulk subscribers threshold is selected so that there are few bulk subscribers, and the number of partitions is sufficiently large, then Algorithm 11 achieves significant speedup compared to Algorithm 9. This is also corroborated by our experiments in Section VI.

D. Decentralizing the DCBR-M algorithm

Note that the DCBR-M algorithm as presented above is fully centralized. It is possible to decentralize it in the following way: (1) each lightweight node autonomously decides which random partition it belongs to and registers itself under the partition name (it is possible, e.g., to use a DHT for that purpose), (2) nodes from the same partition learn about each other and establish a communication channel, (3) different partitions construct sub-TCOs in parallel, i.e., the nodes within each partition exchange their interests and execute the GM-M algorithm, (4) different partitions compute rep-sets in parallel, (5) bulk subscribers and rep-nodes from different rep-sets communicate their interests and compute outer edges. Note that the original GM-M algorithm does not lend itself to such decentralization.

This decentralization scheme has several important benefits: reducing the total runtime cost, optimizing distributed resource utilization, and balancing the computational load. The time $\mathbb{T}_{in\text{DCBR}}$ for computing inner edges becomes $O(\frac{|L|^4}{p^4})$ and the total time $\mathbb{T}_{\text{DCBR}}$ becomes

$$\mathbb{T}_{\text{DCBR}} = O(|T| \cdot ((|B| + |R|)^4 + \frac{|L|^4}{p^4})). \tag{11}$$

Furthermore, decentralization eliminates the need for a central entity that must maintain a global knowledge of all nodes and their interests. To quantify this benefit, we introduce additional performance characteristic of the algorithm called *potential neighbor set*, which is the set of other nodes a node has to learn about in the course of the algorithm. This characteristic is important because gathering nodes' interests in a scalable and robust decentralized manner is a problem in its own right. Additionally, the fan-out of node v in the overlay produced by *any* algorithm cannot exceed the size of the potential neighbor set of v. Therefore, minimizing the potential neighbor set has an additional desirable effect from this point of view.

To formalize this argument, we define the *potential neighbor ratio* for a node v, denoted as $pn\text{-}ratio(v)$. Potential neighbor ratio is the size of potential neighbor set for v (including v itself) normalized by the total number of nodes $|V|$. For any centralized algorithm, this ratio is equal to 1. For DCBR-M, the potential neighbor set for v consists of three subsets: (1) nodes in the same partition as v: V_d such that $v \in V_d$ (if v is a lightweight node); (2) bulk subscribers B (if v is a bulk subscriber itself or it belongs to some rep-set); and (3) all rep-nodes from other partitions: $\{u | u \in R_i \text{ s.t. } v \in B \vee v \in R_d \wedge R_i \neq R_d\}$ (if v is a bulk subscriber or it belongs to the rep-set R_d). Consequently, the potential neighbor ratio is always the biggest for lightweight nodes selected as rep-nodes. For such nodes, $pn\text{-}ratio(v) = \frac{|L|}{|V| \cdot p} + \frac{|B|}{|V|} + \frac{|R|}{|V|} \cdot \frac{p-1}{p}$.

We extend the definition of a potential neighbor ratio to apply to the entire node set:

$$pn\text{-}ratio(V) = \max_{v \in V} pn\text{-}ratio(v) = \frac{|L|}{|V| \cdot p} + \frac{|B|}{|V|} + \frac{|R|}{|V|} \cdot \frac{p-1}{p} \tag{12}$$

Equation 12 shows that DCBR-M has improved $pn\text{-}ratio$ compared to any centralized algorithm (such as GM-M). This is further confirmed by our experiments in Section VI.

V. Selecting Parameters

This section discusses how to choose optimal parameter values for our DCBR-M algorithm. Algorithm 11 is parametrized with (1) the bulk subscriber threshold η, (2) the coverage factor λ, and (3) the number of partitions for lightweight nodes p. The choice of values for these parameters substantially affects the algorithm's behavior. It is therefore essential to identify a combination of them that leads to satisfactory performance. First, we pick reasonable values for η and λ for typical pub/sub workloads; then, we provide a numerical method to determine the optimal value of p.

The selection of the bulk subscriber threshold η exhibits the tradeoff between maximal overlay degree and running time: Small threshold values cause all nodes to be treated as bulk subscribers thereby favoring the degree over running time and making the overall performance very similar to that of GM-M. On the other hand, large threshold values favor the running time and $pn\text{-}ratio$ at the expense of increased overlay degree. Fortunately, even relatively small threshold values result in small bulk subscriber sets for typical pub/sub workloads that follow the "Pareto 80–20" rule, as discussed in Section IV-B. In our implementation, we sort the subscribers by subscription size and choose η that causes $\leq 20\%$ of the nodes to be considered bulk subscribers.

The coverage factor selection exhibits a similar tradeoff: If we choose the coverage factor to be as large as the size $|L|/p$ of the partitions, then the behavior of Algorithm 11 becomes identical to that of Algorithm 9. On the other hand, $\lambda = 1$ minimizes the size of rep-sets, $pn\text{-}ratio$, and running time but leads to a severe impact on the node degree. According to our experiments in Section VI, an increase in λ beyond 3 only marginally improves the node degree, even for large partitions. The rep-sets for $\lambda = 3$ are significantly smaller for such large partitions than partitions themselves so that we choose 3 as the default value for λ.

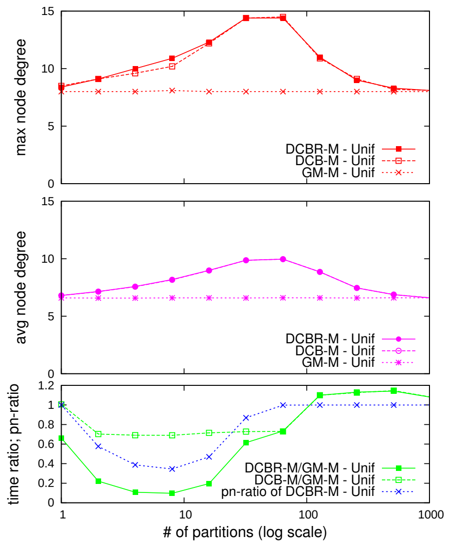
The tradeoff in the selection of the number p of partitions is more complex. When p is as large as $|L|$, the performance is dominated by the invocations of the `combineB&LReps()` function in the *combine* phase. As p decreases, the effect of executing the GM-M algorithm at the *conquer* phase becomes more and more pronounced, both with respect to the degree and the running time. When we use a very small number of partitions, it starts to dominate the running time assuming $|B|$ is relatively small. Therefore, we need to find intermediate values of p that minimize the running time and $pn\text{-}ratio$.

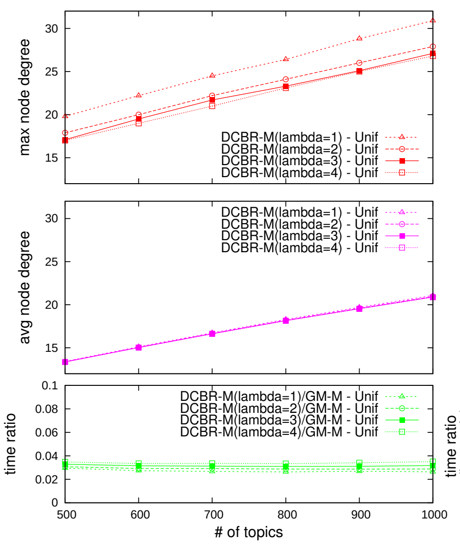
The bound on the running time $\mathbb{T}_{\text{DCBR}}$ is established by Lemma 16 and Equation 12 for centralized and distributed implementations, respectively. Since the bound on $\mathbb{T}_{\text{DCBR}}$ depends on $|R|$, which is difficult to assess analytically, we use an adaptive way for selecting p. Since partitioning the nodes and computing the rep-sets is relatively cheap, we try partitioning for different p values. Each time, we only compute the rep-sets and thus obtain $|R|$ without running the expensive calculation

TABLE I
ALGORITHMS FOR SOLVING THE MINMAX-TCO PROBLEM

GM-M	Greedy Merge algorithm for MinMax
DCB-M	Divide-and-Conquer with Bulk and Lightweight Nodes
DCBR-M	Divide-and-Conquer with Bulk and Lightweight Rep-nodes
RingPT	Ring-per-topic algorithm
TCO_{ALG} *	The TCO produced by ALG
D_{ALG}	Maximum node degree in TCO_{ALG}
d_{ALG}	Average node degree in TCO_{ALG}
$\mathbb{T}_{\mathsf{ALG}}$	Running time of ALG

* ALG is GMM, DCB, DCBR or RingPT for the GM-M, DCB-M, DCBR-M or RingPT algorithms, respectively.





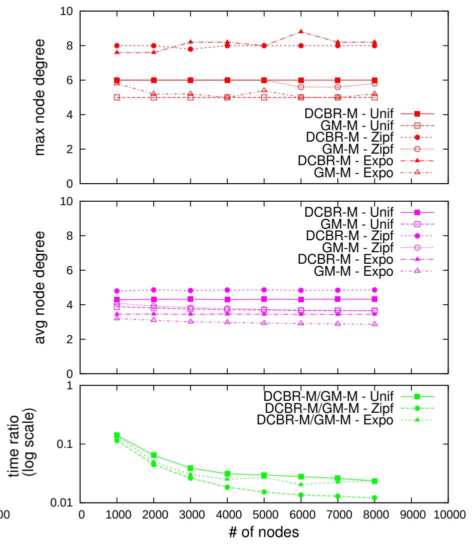


Fig. 4. DCBR-M parameterized with different p Fig. 5. DCBR-M parameterized with different λ Fig. 6. DCBR-M under different distributions

of inner and outer edges. Then, we use fast numerical methods to approximately determine the value of p that minimizes $\mathbb{T}_{\mathsf{DCBR}}$. We also apply the same technique to Equation 12 in order to determine the value of p that is optimal for $pn\text{-}ratio(V)$.

VI. EVALUATION

We implemented all algorithms described in this paper in Java and compared them under various experimental conditions. Table I summarizes the algorithms evaluated. We use the GM-M algorithm as a baseline because it produces the lowest maximum node degree of all known algorithms that run in polynomial time. To be precise, we are using a faster implementation of the GM-M algorithm described in Section IV-A both for baseline GM-M and as a building block for DCB-M and DCBR-M. This faster implementation produces exactly the same overlay as the original one in [8] at a lower runtime cost by manipulating data structures more efficiently. Our divide-and-conquer design is orthogonal to the data structures used in the algorithm: Our faster version still has prohibitively high running time without divide-and-conquer. In fact, the speedup of DCBR-M compared to GM-M would have been even more significant for the slower, original GM-M implementation. However, using a faster implementation allows us to run comparative experiments on a larger scale.

In the experiments, we use the following ranges for the input instances: $|V| \in [1\,000, 8\,000]$ and $|T| \in [100, 1\,000]$. We define the average node subscription size, minimum subscription size, and the maximum subscription size as follows: $\overline{|T_v|} = \frac{\sum_{v \in V} |T_v|}{|V|}$, $|T_v|_{min} = \min_{v \in V}\{|T_v|\}$, $|T_v|_{max} = \max_{v \in V}\{|T_v|\}$. We used $|V| = 4\,000$, $|T| = 200$, and $\overline{|T_v|} = 50$ (with $|T_v|_{min} = 10, |T_v|_{max} = 90$) to generate the input workloads for most of the experiments unless specified otherwise. Each topic $t_i \in T$ is associated with probability q_i, $\sum_i q_i = 1$, so that each node subscribes to t_i with a probability q_i. The value of q_i is distributed according to either a uniform, a Zipf (with $\alpha = 2.0$), or an exponential distribution. According to [21], these distributions are representative of actual workloads used in industrial pub/sub systems today. The Zipf distribution is chosen because [3] shows it faithfully describes the feed popularity distribution in RSS. The exponential distribution is used by stock-market monitoring engines in [27] for the study of stock popularity in the New York Stock Exchange (NYSE). The η, p, and λ parameters are selected as described in Section V.

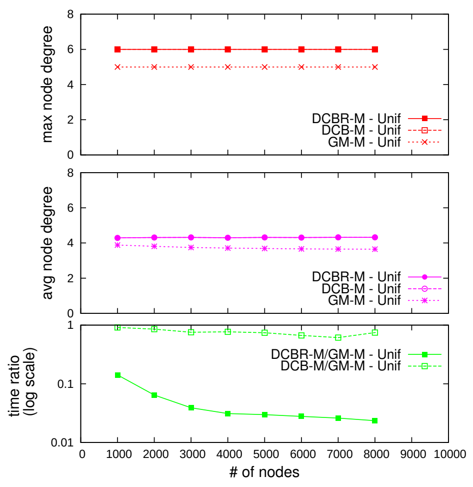


Fig. 7. DCBR-M as $|V|$ increases

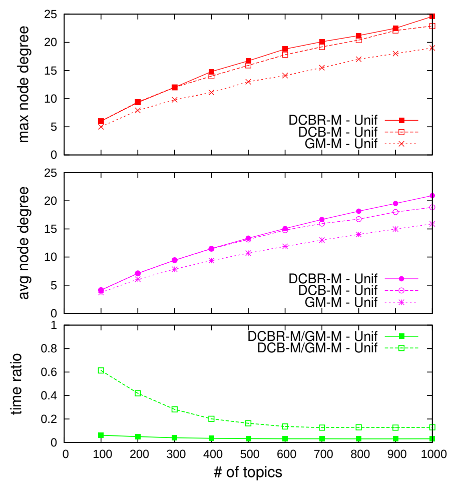


Fig. 8. DCBR-M as $|T|$ increases

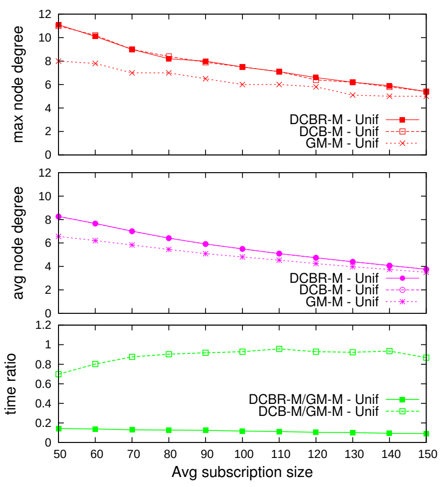


Fig. 9. DCBR-M as $\overline{|T_v|}$ increases

A. *Partitioning for Lightweight Nodes*

TABLE II
RANDOM PARTITIONING UNDER UNIFORM DISTRIBUTION

	mean	min	max	variance
D_{DCBR}	10.883	10	11	0.104
D_{DCB}	10.29	10	11	0.206
D_{GMM}	8.013	8	9	0.0124
d_{DCBR}	8.188	8.02	8.402	0.00387
d_{DCB}	8.170	8.018	8.388	0.00376
d_{GMM}	8.013	8	9	0.0124
$\mathbb{T}_{\text{DCBR}}/\mathbb{T}_{\text{GMM}}$	0.112	0.0983	0.132	0.0000283
$\mathbb{T}_{\text{DCB}}/\mathbb{T}_{\text{GMM}}$	0.655	0.615	0.690	0.0000820
$\lvert R\rvert/\lvert L\rvert$	0.142	0.105	0.189	0.000180

TABLE III
RANDOM PARTITIONING UNDER ZIPF DISTRIBUTION

	mean	min	max	variance
D_{DCBR}	16.3925	15	18	0.450
D_{DCB}	16.378	15	18	0.426
D_{GMM}	11.14	10	13	0.181
d_{DCBR}	8.065	7.878	8.318	0.00530
d_{DCB}	8.064	7.864	8.308	0.00530
d_{GMM}	6.487	6.356	6.66	0.00265
$\mathbb{T}_{\text{DCBR}}/\mathbb{T}_{\text{GMM}}$	0.248	0.207	0.287	0.000153
$\mathbb{T}_{\text{DCB}}/\mathbb{T}_{\text{GMM}}$	0.601	0.564	0.645	0.000135
$\lvert R\rvert/\lvert L\rvert$	0.0742	0.0504	0.110	0.0000862

a) Random partitioning for lightweight nodes: We first evaluate the effects of random partitioning of lightweight nodes for the DCB-M and DCBR-M algorithms. We run the algorithm 400 times for the same settings (namely, the default experimental settings discussed above except $|V| = 1\,000$, under three different distributions) so that the only difference between different runs is due to the random node interest generation according to the given distribution parameters and due to random node partitioning. The statistics pertaining to maximum node degree, average node degree, running time ratio, and the ratio of nodes selected as rep-nodes are reported in Table II, III, IV. As the table illustrates, under the exponential distribution, all the values are quite *stable* with negligible variance across different experiments; and the variance is even more insignificant under less skewed distributions. Besides, the results validate our assumption that $|R| \ll |L|$ (with $\lambda = 3$). We conclude that when the number p of partitions is *reasonable*, random partitioning of lightweight nodes is an efficient and robust way to implement the *divide* phase of DCBR-M. Furthermore, it results in small rep-sets, which is vital to the performance of the DCBR-M algorithm.

TABLE IV
RANDOM PARTITIONING UNDER EXPONENTIAL DISTRIBUTION

	mean	min	max	variance
D_{DCBR}	10.798	9	13	0.468
D_{DCB}	10.793	9	13	0.461
D_{GMM}	8.4425	7	12	0.348
d_{DCBR}	4.499	4.394	4.586	0.00122
d_{DCB}	4.499	4.392	4.59	0.00124
d_{GMM}	3.93	3.86	4.02	0.000747
$\mathbb{T}_{\mathsf{DCBR}}/\mathbb{T}_{\mathsf{GMM}}$	0.130	0.111	0.161	0.0000678
$\mathbb{T}_{\mathsf{DCB}}/\mathbb{T}_{\mathsf{GMM}}$	0.833	0.787	0.883	0.000225
$\lvert R\rvert/\lvert L\rvert$	0.0748	0.0438	0.0977	0.0000891

b) Impact of p: Given an instance $I(V, T, Int)$ where $|V| = 1\,000$, DCBR-M and DCB-M are executed with all possible p values ranging from 1 to $|L|$. Fig. 4 shows that under different values of p, TCO_{DCBR} and TCO_{DCB} have similar maximum and average node degrees, which are slightly higher than those of TCO_{GMM}. However, DCBR-M runs substantially faster than DCB-M when p is set appropriately.

We already know that DCBR-M and DCB-M exhibit identical behavior to GM-M when $p = |L|$. Fig. 4 shows that as the number of partitions p grows from 1 to $|L|$, $D_{\mathsf{DCBR}}(\approx D_{\mathsf{DCB}})$ first *increases* gradually and then it starts to *decrease* until it becomes equal to $D_{\mathsf{GM\text{-}M}}$. Note that the D_{DCBR} never moves far from the horizon line of D_{GMM}: It is always smaller than 6.5. It is less than 2.8 for the value of p that minimizes $\mathbb{T}_{\mathsf{DCBR}}$.

While $\mathbb{T}_{\mathsf{DCB}}$ stays close to $\mathbb{T}_{\mathsf{GMM}}$ for all values of p, $\mathbb{T}_{\mathsf{DCBR}}$ first slides *down* sharply and then climbs *up* as p increases. The range of p values for which this phenomenon occurs is small and relatively close to 0. (It touches the lowest point when p is around 10.) Furthermore, the $\mathbb{T}_{\mathsf{DCBR}}/\mathbb{T}_{\mathsf{GMM}}$ ratio follows the same trend as *pn-ratio* for DCBR-M, which is compatible with our discussion of choosing p in Section V.

B. Impact of Coverage Factor

Here, we explored the impact of λ on the output and performance of the DCBR-M algorithm. Given an input $I(V, T, Int)$ where $|V| = 2000$ and $|T| \in [500, 1000]$, the DCBR-M algorithm is evaluated for four different values of the coverage factor ($\lambda = 1, 2, 3, 4$.) As Fig. 5 shows, under uniform distribution, as λ *increases*, D_{DCBR} *decreases*. The differences in maximum node degrees also *decrease* with successive coverage factors, i.e., $D_{\mathsf{DCBR}}|_{\lambda=k-1} - D_{\mathsf{DCBR}}|_{\lambda=k} > D_{\mathsf{DCBR}}|_{\lambda=k} - D_{\mathsf{DCBR}}|_{\lambda=k+1}$. More specifically, compared to $D_{\mathsf{DCBR}}|_{\lambda=2} - D_{\mathsf{DCBR}}|_{\lambda=3} \approx 0.71$ on average, $D_{\mathsf{DCBR}}|_{\lambda=1} - D_{\mathsf{DCBR}}|_{\lambda=2}$ is noticeable, which could be as much as ≥ 3 for most cases. When $\lambda \geq 3$, the difference is insignificant (≤ 0.32 on average). Meanwhile, under all coverage factors that were tested, DCBR-M runs remarkably faster as compared to GM-M ($\mathbb{T}_{\mathsf{DCBR}} \leq 3\%\mathbb{T}_{\mathsf{GMM}}$ on average). Also, $\mathbb{T}_{\mathsf{DCBR}}$ slightly *increases* as λ *increases*, because *pn-ratio increases* as a result. However, differences among the running time cost for different coverage factors are insignificant: $\frac{\mathbb{T}_{\mathsf{DCBR}|\lambda=4}}{\mathbb{T}_{\mathsf{GMM}}} - \frac{\mathbb{T}_{\mathsf{DCBR}|\lambda=1}}{\mathbb{T}_{\mathsf{GMM}}} \leq 0.69\%$ on average.

This experiment confirms the validity of choosing a relatively small integer as coverage factor in Section V, both in terms of node degree and running time cost.

C. Effects Under Different Distributions

We now consider DCBR-M's behavior under different input instances. We first provide an overview of the overall DCBR-M performance under different typical distributions, and then we analyze how DCBR-M is affected by various aspects of the input.

Fig. 6 depicts that under different distributions, DCBR-M produces high-quality TCOs in terms of maximum and average node degrees, which are slightly higher than D_{GMM} and d_{GMM}, respectively. However, the differences are insignificant: $D_{\mathsf{DCBR}} - D_{\mathsf{GMM}} \leqslant 2.0, d_{\mathsf{DCBR}} - d_{\mathsf{GMM}} \leqslant 0.70$ on average.

Although DCBR-M and GM-M produce quite close maximum and average node degrees, DCBR-M runs considerably faster than GM-M: $\mathbb{T}_{\mathsf{DCBR}} \leq 4.0\% \cdot \mathbb{T}_{\mathsf{GMM}}$ on average. As the number of nodes *increases*, D_{DCBR} and d_{GMM} remain steadily *low* while the running time ratio $\mathbb{T}_{\mathsf{DCBR}}/\mathbb{T}_{\mathsf{GMM}}$ *decreases* considerably. This further attests to DCBR-M's scalability with respect to the number of nodes in the network.

The maximum node degrees tend to be a bit more *fluctuating* under Zipf and exponential distributions compared to those under the uniform distribution. This could be explained by the slightly higher variance under skewed distributions, as presented in Table IV. Although skewed distributions are more sensitive to the variations in the input, the maximum and average node degrees always stay low, even in the worst cases.

D. Impact of the number of nodes

We now demonstrate DCBR-M's scalability with respect to different input parameters. In the rest of the section, while we report on results and analysis for all distributions in the text, we only show figures for the uniform topic popularity distribution due to space limit.

Fig 7 depicts the comparison between DCBR-M, DCB-M and GM-M as the number of nodes increases where $|T| = 100$. The figure shows that DCBR-M and DCB-M output similar TCOs with regard to maximum and average node degrees, but DCBR-M runs considerably faster. Under the uniform distribution, for example, $\mathbb{T}_{\mathsf{DCBR}}$ is on average 4.79% of $\mathbb{T}_{\mathsf{GMM}}$ while $\mathbb{T}_{\mathsf{DCB}}$ is as much as 75.7% of $\mathbb{T}_{\mathsf{GMM}}$. Additionally, DCBR-M gains more speedup with the increase in the number of nodes compared to the other algorithm.

E. Impact of the number of topics

Fig. 8 depicts how DCBR-M and DCB-M perform compared to GM-M when we vary the number of topics. As the figure shows, under the uniform distribution, the maximum and average node degrees of all three algorithms *increase* for a higher number of topics. This is because *increasing* the number of topics leads to *reduced* correlation among subscriptions. However, the increase is slow paced and the difference $D_{\mathsf{DCBR}} - D_{\mathsf{GMM}}$ remains insignificant: 3.51 for the uniform, 4.46 for the Zipf, and 3.46 for the exponential distribution on average.

The running time ratio of DCBR-M to GM-M slightly *increases* as the number of topics *increases*, yet this effect is insignificant: $\mathbb{T}_{\mathsf{DCBR}}$ is less than 3.8% of $\mathbb{T}_{\mathsf{GMM}}$ on average.

F. Impact of the average subscription size

Fig. 9 depicts how the node subscription size affects the DCBR-M and DCB-M algorithms. We set $|V| = 1\,000$, $|T| = 400$, and $\overline{|T_v|}$ varies from 50 to 150.

The figure shows that under the uniform distribution, DCBR-M and DCB-M produces quite close TCOs in terms of both maximum and average node degrees. As the subscription size *increases*, D_{DCBR} and D_{GMM} *decrease*, and the difference of $(D_{\mathsf{DCBR}} - D_{\mathsf{GMM}})$ *shrinks*. d_{DCBR} follows the same trend.

This *decrease* occurs because the growth of $\overline{|T_v|}$ causes *increased* correlation across the subscriptions. Upon bigger correlation, an edge addition to the overlay reduces a higher number of topic-connected components on average because the nodes share more comment interests. Therefore, a smaller number of edge additions are required before the overlay becomes topic-connected.

The ratio of $\mathbb{T}_{\mathsf{DCBR}}$ to $\mathbb{T}_{\mathsf{GMM}}$ also *decreases* with the *increase* of $\overline{|T_v|}$. In both algorithms, an edge addition causes a higher number of updates to topic-connected components for bigger $|T_v|$ (Lines 8- 9 in Algorithm 2). Yet, this effect has less influence on $\mathbb{T}_{\mathsf{DCBR}}$ compared to $\mathbb{T}_{\mathsf{GMM}}$ since each update in DCBR-M affects a smaller portion of edges. Unlike $\mathbb{T}_{\mathsf{DCBR}}$, however, $\mathbb{T}_{\mathsf{DCB}}$ does not gain significant speedup.

G. Comparison with Ring-Per-Topic

Finally, we compare the maximum and average node degrees produced by DCBR-M GM-M and RingPT. RingPT is an algorithm that mimics the common practice of building a separate overlay for each topic (usually a tree but we use a ring that has the same average node degree). According to RingPT, all the nodes interested in the same topic form a ring for that topic, after which rings for different topics are merged into a single overlay. $|T|$ is set to 100 in this experiment. As Fig. 10 shows, D_{DCBR} and D_{GMM} are quite close (the average difference is 1.22), but the maximum node degree of RingPT exceeds D_{DCBR} by a factor of approximately 30. This demonstrates the general significance of overlay construction algorithms for pub/sub.

VII. Conclusions

This paper focuses on a number of design objectives for the MinMax-TCO problem that are central to creating a practical solution. We have designed the DCBR-M algorithm which is capable of constructing a low fan-out TCO, while being significantly more efficient than previously known solutions. Numerical techniques can be employed to effectively obtain a good combination of parameters which adapts to various inputs and guarantees the output and the performance of the algorithm. The algorithm is thoroughly examined via a comprehensive experimental analysis, which demonstrates the scalability of DCBR-M under different distributions as the number of nodes, the number of topics, and the subscription size increase.

References

[1] J. Reumann, "Pub/Sub at Google," lecture & Personal Communications at EuroSys & CANOE Summer School, Oslo, Norway, Aug'09.

[2] "Tibco rendezvous," http:\www.tibco.com.

[3] H. Liu, V. Ramasubramanian, and E. G. Sirer, "Client behavior and feed characteristics of RSS, a publish-subscribe system for web micronews," in *IMC'05*.

[4] M. Petrovic, H. Liu, and H.-A. Jacobsen, "G-ToPSS: fast filtering of graph-based metadata," in *WWW'05*.

[5] G. Li, V. Muthusamy, and H.-A. Jacobsen, "A distributed service oriented architecture for business process execution," *ACM TWEB*, 2010.

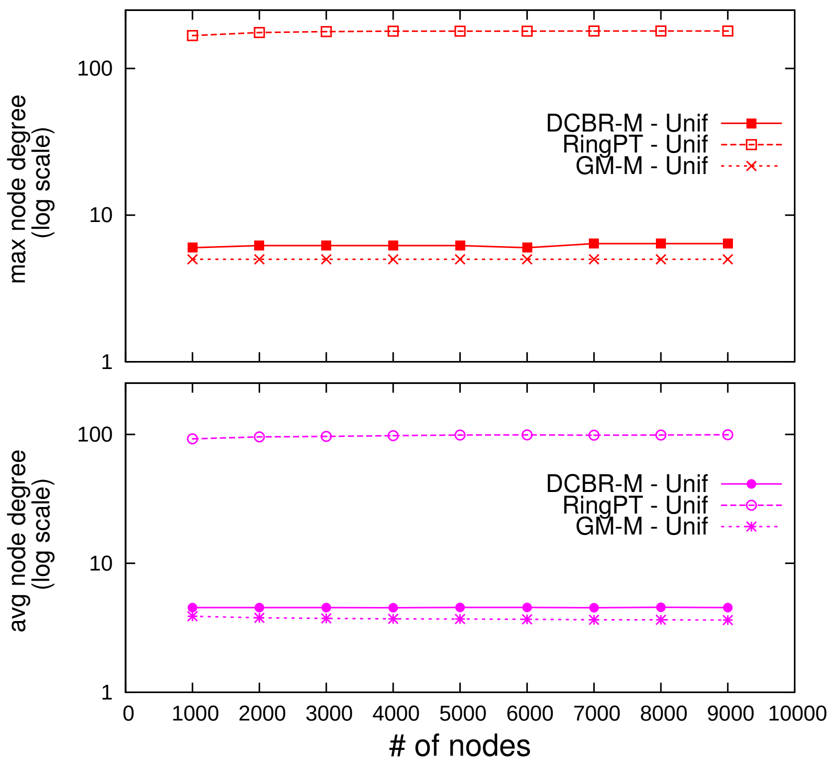


Fig. 10. DCBR-M vs. GM-M vs. RingPT

[6] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni, "Pnuts: Yahoo!'s hosted data serving platform," *Proc. VLDB Endow.*, 2008.

[7] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Constructing scalable overlays for pub-sub with many topics: Problems, algorithms, and evaluation," in *PODC'07*.

[8] M. Onus and A. W. Richa, "Minimum maximum degree publish-subscribe overlay network design," in *INFOCOM'09*.

[9] ——, "Parameterized maximum and average degree approximation in topic-based publish-subscribe overlay network design," in *ICDCS'10*.

[10] M. A. Jaeger, H. Parzyjegla, G. Mühl, and K. Herrmann, "Self-organizing broker topologies for publish/subscribe systems," in *SAC'07*.

[11] R. Baldoni, R. Beraldi, L. Querzoni, and A. Virgillito, "Efficient publish/subscribe through a self-organizing broker overlay and its application to SIENA," *Comput. J.*, vol. 50, no. 4, 2007.

[12] C. Chen, H.-A. Jacobsen, and R. Vitenberg, "Divide and conquer algorithms for publish/subscribe overlay design," in *ICDCS'10*.

[13] D. Liben-Nowell, H. Balakrishnan, and D. Karger, "Analysis of the evolution of peer-to-peer systems," in *PODC*, 2002.

[14] E. De Santis, F. Grandoni, and A. Panconesi, "Fast low degree connectivity of ad-hoc networks via percolation," in *ESA'07*.

[15] L. C. Lau, J. S. Naor, M. R. Salavatipour, and M. Singh, "Survivable network design with degree or order constraints," in *Proc. ACM STOC'07*.

[16] D. Tam, R. Azimi, and H.-A. Jacobsen, "Building content-based publish/subscribe systems with distributed hash tables," in *DBISP2P'03*.

[17] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *JSAC'02*.

[18] G. Li, V. Muthusamy, and H.-A. Jacobsen, "Adaptive content-based routing in general overlay topologies," in *Middleware'08*.

[19] F. Araujo, L. Rodrigues, and N. Carvalho, "Scalable QoS-based event routing in publish-subscribe systems," in *NCA'05*.

[20] S. Girdzijauskas, G. Chockler, Y. Vigfusson, Y. Tock, and R. Melamed, "Magnet: practical subscription clustering for internet-scale publish/subscribe," in *DEBS'10*.

[21] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Spidercast: A scalable interest-aware overlay for topic-based pub/sub communication," in *DEBS'07*.

[22] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni, "TERA: topic-based event routing for peer-to-peer architectures," in *DEBS'07*.

[23] E. Baehni, P. Eugster, and R. Guerraoui, "Data-aware multicast," in *DSN'04*.

[24] R. Chand and P. Felber, "Semantic peer-to-peer overlays for publish/subscribe networks," in *EUROPAR'05*.

[25] S. Voulgaris, E. Rivire, A.-M. Kermarrec, and M. V. Steen, "Sub-2-Sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks," in *IPTPS'06*.

[26] D. Peleg, G. Schechtman, and A. Wool, "Approximating bounded 0-1 integer linear programs," in *Theory of Computing and Systems*, 1993.

[27] Y. Tock, N. Naaman, A. Harpaz, and G. Gershinsky, "Hierarchical clustering of message flows in a multicast data dissemination system," in *IASTED PDCS*, 2005.