Weighted Overlay Design for Topic-based Publish/Subscribe Systems on Geo-Distributed Data Centers

Chen Chen
IBM Research - Haifa
chenc@il.ibm.com

Yoav Tock
IBM Research - Haifa
tock@il.ibm.com

Hans-Arno Jacobsen
University of Toronto
jacobsen@eecg.toronto.edu

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

***Abstract*—We incorporate underlay information into overlay design for topic-based publish/subscribe (pub/sub) systems on geo-distributed data centers. We propose the MinAvg-WTCO problem that optimizes the weighted average node degree while constructing a topic-connected overlay (TCO), i.e., each topic induces a connected sub-overlay among all nodes interested in this topic. Most existing TCO designs are oblivious to the low-level network infrastructure and assume edge equivalence.**

We prove that MinAvg-WTCO is NP-complete and difficult to approximate within a logarithmic factor with regard to the number of nodes. We devise several approximation algorithms for MinAvg-WTCO using different design techniques. Both theoretical analysis and empirical evaluation show that our designed algorithms tread the balance between overlay quality and runtime cost. Our algorithms significantly outperform the state of the art for TCO design that ignores edge differences.

***Keywords*-weighted graph; topic-connected overlay; pub/sub**

I. Introduction

Publish/subscribe (pub/sub) has become a popular communication paradigm that provides a loosely coupled form of interaction among many publishing data sources and many subscribing data sinks [1], [19], [26], [29]. This work focuses on *topic-based* pub/sub: publishers associate each publication message with one or more specific topics, and subscribers register their interests in a subset of all topics.

A distributed pub/sub system often organizes nodes (i.e., brokers or servers) as an application-level overlay in a federated or peer-to-peer manner. The overlay infrastructure forms the foundation for distributed pub/sub and directly impacts the system's performance and scalability, e.g., message latency and routing cost. Constructing a high-quality overlay for distributed pub/sub is a key challenge and fundamental problem that has received attention in both industry [13], [26] and academia [5], [9], [10], [11], [12], [21], [27].

Topic-connected overlay (TCO) was introduced for pub/sub [11]. Informally speaking, TCO organizes all nodes interested in the same topic in a directly connected dissemination sub-overlay. First, TCO ensures that publications on each topic can be transmitted to all subscribers of this topic without using non-interested nodes as intermediate relays. Publication routing atop TCOs saves bandwidth and computational resources otherwise wasted on forwarding and filtering out irrelevant messages. Second, TCO leads to simpler matching engine and smaller routing tables. Third, from the perspective of security, TCO is preferable in the scenario where users within a *trusted group* want to share sensitive data among themselves without having it travel outside the group [3]. Fourth, TCO generally improves graph connectivity and helps reduce the diameter of the overlay [7], [8], [11]. A low diameter is desirable for pub/sub overlays [11], [28]: the overlay diameter impacts many performance factors of pub/sub, e.g., message latency. With lower diameters, message delays tend to be smaller, because fewer hops are needed for message delivery. For example, the initial GooPS design required that each source-sink pair would be no more than three overlay hops apart [26].

Many existing approaches build a separate dissemination sub-overlay per topic [4], [6], [22], [27], thereby attaining TCO. While these construction methods are efficient and easy to implement, the resulting overlays often exhibit prohibitively high node degrees. This does not meet the imperative requirement for a pub/sub overlay to have a low link fan-out per node. While overlay designs for different applications might be principally different, they all strive to minimize node degrees, e.g., DHTs [6], [25] and small-world networks [23]. First, it costs a lot of resources to maintain adjacent links for a high-degree node (e.g., monitoring links and neighbors [11], [21]). For a typical pub/sub system, each link would also have to carry a number of protocols, service components, message queues, etc. Second, in low-degree overlays, the set of nodes that participate in coordination of distributed tasks tends to be smaller. Particularly for pub/sub routing, the node degree directly influences the size of the routing tables, the complexity of matching, and the efficiency of message delivery. While the benefits for systems where nodes have a limited amount of resources are obvious (e.g., in the context of peer-to-peer and mobile), it should be noted that the original motivation for the TCO [11], [12] stem from cloud environments. Namely, it was observed that in an IBM production data center full-mesh overlays for connecting applications do not scale and that bounded-degree overlays are required.

To address these issues, [11] posed the problem of constructing a TCO with the minimum average node degree, proved the NP-hardness of the problem, and developed a greedy approximation algorithm. Other problems for TCO construction have also been considered [3], [8], [10], [21].

However, the TCO model and related approaches are oblivious to the underlying network infrastructure. First, incorporating topology awareness is a performance optimization but not required for correctness. Second, the topology plays a less important role in their targeted environments [4], [7], [8], [9], [10], [11], [12], [21], [22], where all nodes reside in one data center. As a result, the TCO model does not consider locality in the underlying network and implicitly assumes that edges between any two nodes are equivalent. The assumption of edge equivalence is reasonable for a pub/sub system in a data center but holds no longer for Internet-scale applications that are deployed across multiple, diverse data centers [1], [13], [26].

Most *Internet of Things* (IoT) systems rely on MQTT [2] or other pub/sub protocols to offer connectivity and communication between enterprise applications and IoT devices. A cloud-hosted IoT platform (e.g., IBM IoT Foundation [1]) is expected to support tens or hundreds of millions of IoT devices. To accommodate such large-scale workloads, we should deploy thousands of MQTT servers on geographically distributed data centers. This motivates a more advanced pub/sub overlay design. Unfortunately, it has not been widely studied (or hardly at all considered) what effects the underlying (physical) network topology may induce on the pub/sub overlay.

The advent of *software defined networking* (SDN) [20] exposes the underlying infrastructure to applications and services. With scalable centralized controllers, we can use centralized designs to realize large-scale pub/sub overlays across multiple data centers. This practical potential further promotes an algorithmic exploration of pub/sub overlays.

We aim to capture underlying network structure and physical locality for the pub/sub overlay design. We introduce edge weights into the TCO model, which measure node distances based on some network-level metrics [5], [6], [22], [23], [25], [28]. More specifically, the edge weights may reflect many practical concerns: (1) network latencies; (2) bandwidths; (3) costs of traffic, i.e., transmitting data on local links is more economical and energy-efficient than that on long-range ones; and (4) social distances. For example, PNuts [13] showed that the latencies of remote operations (e.g., reads and writes across data centers) cost 2 to 10 times more than the local ones. Many distributed systems model locality via weighted edges [18], [23], [24], [25], [28]. Intuitively, we want the distributed pub/sub system to cluster nodes with similar interests while preserving locality in terms of small overlay edge weights.

We summarize our main contributions as follows:

1. We propose the MINAVG-WTCO problem to formally study the fundamental trade-offs between attaining the TCO property while preserving low *weighted* node degrees (Problem 1). We prove that MINAVG-WTCO is NP-complete and derive an $\Omega(\log |V|)$ hardness of approximation result where $|V|$ is the number of nodes in the system (Theorem 1).

2. We model MINAVG-WTCO with *submodular set functions* (see §III) and devise several approximation algorithms with different design techniques, including the greedy GrMAW in §VI and primal-dual algorithms in §VII.

3. In §IX, we conduct extensive evaluation under a variety of characteristic pub/sub workloads of up to 10 000 nodes and 1 000 topics. We empirically show the pros and cons of each algorithm for MINAVG-WTCO. Besides, our algorithms display substantial merits as compared to the best known TCO design algorithms that make no distinction between edges.

II. Related Work

Pub/sub overlay: A significant body of research has been considering the construction of an overlay topology underlying pub/sub systems such that network traffic is minimized (e.g., [9], [10], [11], [15], [21], [22], [23]). The TCO property is explicitly enforced in [4], [12], [22], [27] and implicitly manifest in [6], [15], [23], and they all strive to reduce intermediate overlay hops for message delivery.

The theoretical formulation of pub/sub overlay design originated in [11], which proposed the problem of constructing a TCO with minimum edges. Following this direction, a number of problems were formulated in constructing TCO while optimizing node degrees and other criteria [9], [10], [21]. Unfortunately, this body of work simply assumes edge equivalence and ignores locality or underlying topology.

The core ideas of [9], [10], [11], [21] are rooted in the greedy algorithm for the classical *minimum set cover* problem. These techniques are no longer effective when we introduce edge weights into the TCO model. In contrast, we apply the design and analysis for the *submodular set function* [14], [30] to the pub/sub overlay construction.

Topology-aware overlay: Previous research (e.g., DHT [6], [25]) has shown that the knowledge of the underlying network topology is critical to achieve low node and link stress and high efficiency for data dissemination.

A few pub/sub systems [6], [23], [28] try to address locality in the overlay design. However, they merely offer best-effort heuristics and have to sacrifice topic-connectedness for other conditions (e.g., small-world property or node degree constraints). One root cause for this reluctant compromise is the lack of a thorough understanding about the fundamental trade-offs in the pub/sub overlay design. To obtain this insight, this work initiates an attempt for a more comprehensive theoretical framework that captures overlay quality, resource constraints, locality, and the underlying topology.

III. Background

A. Topic-connected overlay (TCO)

Let (V, T, I) be an instance: V is the set of nodes, T is the set of topics, and I is the interest function $I : V \times$

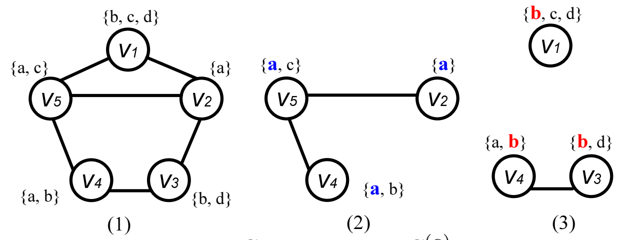


Figure 1: (1) An overlay G. (2) Subgraph $G^{(a)}$ is topic-connected. (3) Subgraph $G^{(b)}$ is not topic-connected.

$T \rightarrow \{\textit{true, false}\}$. A node v is interested in some topic t iff $I(v,t) = \textit{true}$. We also say node v subscribes to topic t.

We denote a *topic-based pub/sub overlay network* (TPSO) as $TPSO(V,T,I,E)$. A $TPSO(V,T,I,E)$ can be illustrated as an undirected graph $G = (V,E)$ over the node set V with the edge set $E \subseteq V \times V$. Given $TPSO(V,T,I,E)$, the sub-overlay *induced* by $t \in T$ is a subgraph $G^{(t)} = (V^{(t)}, E^{(t)})$ such that $V^{(t)} = \{v \in V | I(v,t) = \textit{true}\}$ and $E^{(t)} = \{(u,v) \in E | u \in V^{(t)} \wedge v \in V^{(t)}\}$. A *topic-connected component* (TCC) on topic $t \in T$, is a maximal connected subgraph in $G^{(t)}$. A $TPSO$ is called *topic-connected overlay* if each topic $t \in T$ induces at most one TCC in $G^{(t)}$, which we denote as $TCO(V,T,I,E)$, TCO for short. See Fig. 1.

We denote by $T(v)$ the topic set to which node v subscribes, and by $|T(v)|$ the *subscription size* of node v.

B. Submodular set function

Let a ground set K be finite and nonempty. We consider an integral valued nonnegative function

$$\mu : 2^K \rightarrow \mathbb{N}^0 = \{0, 1, 2, \ldots\}$$

Definition 1: A set function $\mu : 2^K \rightarrow \mathbb{N}^0$ is

(a) *nondecreasing* if $\forall E \subseteq E' \subseteq K$,

$$\mu(E) \leq \mu(E') \tag{1}$$

(b) *submodular* if $\forall E \subseteq E' \subseteq K, \forall e \in K$,

$$\mu(E + e) - \mu(E) \geq \mu(E' + e) - \mu(E') \tag{2}$$

We use e and $\{e\}$ interchangeably if there is no ambiguity.

Given a submodular function $\mu : 2^K \rightarrow \mathbb{N}^0$ and $E \subseteq K$, the *contraction* of μ is a function $\mu_E : 2^{\overline{E}} \rightarrow \mathbb{N}^0$ s.t.

$$\mu_E(S) = \mu(E + S) - \mu(E), \forall S \subseteq \overline{E} \tag{3}$$

where $\overline{E}$ is the *complement* of E (w.r.t. the ground set K), i.e., $\overline{E} = K \backslash E$. We can rewrite Eq. (2) as

$$\mu_E(e) \geq \mu_{E'}(e), \forall E \subseteq E' \tag{4}$$

IV. Weighted TCO Model

We extend the TCO structure by assigning a weight to each edge, which can be regarded as locality information, the network latency, the cost to establish and to maintain the connection, the charge rate for data transmission over the link, or the social distance in a social network. We represent an instance with a four-tuple (V,T,I,ω), and ω is the edge weight function that maps an edge to a positive real value, i.e., $\omega : K \rightarrow \mathbb{R}^+ = (0, +\infty)$, where K is the ground set of all possible edges among V, i.e., $K = V \times V$. We denote by $\omega(e)$ the weight of an edge $e \in K$. For any edge set $E \subseteq K$, $\omega(E)$ stands for the sum of weights for all edges in E, i.e., $\omega(E) = \sum_{e \in E} \omega(e)$.

We denote by $WTPSO(V,T,I,E,\omega)$ a *weighted topic-based pub/sub overlay network* (WTPSO) for the instance (V,T,I,ω) with edge set E. If $WTPSO(V,T,I,E,\omega)$ is a TCO, we call it *weighted topic-connected overlay* (WTCO), denoted by $WTCO(V,T,I,E,\omega)$, $WTCO$ for short.

Given a $WTPSO(V,T,I,E,\omega)$, we define the *weighted degree of a node* $v \in V$ as the sum of edge weights that are incident to node v, i.e.,

$$\omega(v; E) = \sum_{(u,v) \in E} \omega(u,v) \tag{5}$$

Further, the average weighted node degree is:

$$\overline{\omega}(E) = \frac{\sum_{v \in V} \omega(v; E)}{|V|} \tag{6}$$

We consider the optimization problem of constructing a WTCO with the minimum average weighted node degree.

Problem 1: MinAvg-WTCO(V,T,I,ω): Given a node set V, a topic set T, an interest function I, and a weight function ω, construct an edge set E that forms a $WTCO$ for (V,T,I,ω) with the smallest weighted average node degree, i.e., the sum of all edge weights is minimum.

$$\min_{E \subseteq K} \left\{ \sum_{e \in E} \omega(e) : E \text{ forms a } WTCO \text{ for } (V,T,I,\omega) \right\} \tag{7}$$

Theorem 1: MinAvg-WTCO(V,T,I,ω) is NP-complete. The approximation ratio of any polynomial-time algorithm for MinAvg-WTCO is $\Omega(\log |V|)$, if $P \neq NP$.

Proof sketch: We construct a polynomial-time reduction from MinAvg-TCO to MinAvg-WTCO. Therefore, MinAvg-WTCO inherits all complexity results of MinAvg-TCO [11], [3]. ■

V. Tree-per-topic algorithm for MinAvg-WTCO

Many topic-based pub/sub systems adopt the most straightforward approach of constructing an independent sub-overlay for each topic [4], [6], [22], [27]. Alg. 1 mimics this common practice and constructs a *minimum spanning tree* (MST) for each topic independently.

Lemma 1: Alg. 1's approximation ratio is at most $|T|$.

Proof: Given an MinAvg-WTCO instance (V,T,I,ω), let O be an optimal edge set with the minimum sum of edge weights. Since $E^{(t)}$ forms an MST among $(V^{(t)}, \omega)$, $\omega(E^{(t)}) \leq \omega(O^{(t)})$, $\forall t \in T$. Therefore,

$$\omega(E) \leq \sum_{t \in T} \omega(E^{(t)}) \leq \sum_{t \in T} \omega(O^{(t)}) \leq \sum_{t \in T} \omega(O) = |T| \cdot \omega(O) \quad ■$$

Alg. 1 Tree-per-topic algorithm for MinAvg-WTCO

```
TrMAW(V, T, I, ω)
Input: $(V, T, I, \omega)$
Output: $E$, which forms a WTCO for $(V, T, I, \omega)$
1: for $t \in T$ do
2:     $E^{(t)} \leftarrow$ build an MST independently for $(V^{(t)}, \omega)$
3: return $E \leftarrow \bigcup_{t \in T} E^{(t)}$
```

Lemma 2: Alg. 1 outputs a *WTCO* in time $\mathcal{O}(|V|^2|T|)$.

Proof: We can find an MST $E^{(t)}$ for $(V^{(t)}, \omega)$ in time $\mathcal{O}(|V^{(t)}|^2)$. Alg. 1's complexity of is asymptotically the sum of building an MST for each topic, $\mathcal{O}(\sum_{t\in T}|V^{(t)}|^2)$. ■

VI. Greedy Algorithm for MinAvg-WTCO

Alg. 2 specifies GrMAW, the greedy algorithm for MinAvg-WTCO. Alg. 2 starts from an empty edge set $E = \emptyset$ and greedily adds to E the most *cost-effective* edge that *contributes* to the WTCO construction at each iteration (see Line 3), until attaining a WTCO. Before introducing the cost-effectiveness and contribution of an edge, we define the progress measure that Alg. 2 employs.

Given the instance (V, T, I, ω), $E \subseteq K$ stands for the *current* edge set in the overlay throughout the execution of Alg. 2. We denote by $\overline{E}$ its *complement* (w.r.t. the complete edge set K), i.e., $\overline{E} = K \backslash E$. We can regard $\overline{E}$ as the *potential* edge set from which we can select a new edge for the WTCO construction at each iteration.

Given $E \subseteq K$, we define $TCC(E)$ to be the total number of TCCs in $WTPSO(V, T, I, E, \omega)$ over all topics T.

$$TCC(E) = \sum_{t\in T}\Big(\#\text{TCCs in } G^{(t)} = (V^{(t)}, E^{(t)})\Big) \tag{8}$$

By definition,

$$TCC(\emptyset) = \sum_{t\in T}|V^{(t)}|,\ TCC(K) = \Big|\{t \in T : V^{(t)} \neq \emptyset\}\Big|$$

Alg. 2 uses $TCC(E)$ to measure its *progress* towards WTCO: as Alg. 2 proceeds, $TCC(E)$ starts from $TCC(\emptyset)$ and *strictly decreases* with every edge addition up to an absolute limit, i.e., $TCC(K)$.

With $TCC(E)$, we define an energy function $\mu : 2^K \rightarrow \mathbb{N}^0$ in Line 3 of Alg. 2:

$$\mu(E) = TCC(\emptyset) - TCC(E),\ \forall E \subseteq K \tag{9}$$

Function $\mu(E)$ linearly depends on $TCC(E)$, since $TCC(\emptyset)$ is a constant for a given instance. As a result, $\mu(E)$ is equivalent to $TCC(E)$ in the sense of measuring the progress of Alg. 2: as Alg. 2 proceeds, $\mu(E)$ starts from $\mu(\emptyset) = 0$ and *strictly increases* with every edge addition up to an absolute limit, i.e., $\mu(K) = TCC(\emptyset) - TCC(K)$.

With Eq. (9) and Eq. (3), we have

$$\begin{aligned}\mu_E(e) &= \mu(E+e) - \mu(E)\\ &= TCC(E) - TCC(E+e),\ \forall e \in \overline{E}, \forall E \subseteq K\end{aligned} \tag{10}$$

The value $\mu_E(e)$ defines the *contribution* (towards WTCO) of edge e w.r.t. the current edge set E, which is the number of TCCs that would be reduced by adding e upon E.

Further, the *contribution* of an edge set $S \subseteq \overline{E}$ is

$$\mu_E(S) = TCC(E) - TCC(S+E), \forall S \subseteq \overline{E}, \forall E \subseteq K \tag{11}$$

Going back to Line 3 of Alg. 2, we define the *cost-effectiveness* of a potential edge $e \in \overline{E}$ to be the weight averaged over its contribution, i.e., $\frac{\omega(e)}{\mu_E(e)}$.

Greediness leads to the nice properties of Alg. 2.

Lemma 3: Alg. 2's approximation ratio is at most $\log|T|$.

To prove Lemma 3, we dive deep into MinAvg-WTCO. First, we show that we can cast MinAvg-WTCO into the form of *Submodular Set Cover* [3], [30]. Then, we analyze Alg. 2 based on an existing theorem.

By definition of the set function μ in Eq. (9), $\forall E \subseteq K$

$$\mu(E) = \mu(K) \Leftrightarrow E \text{ forms a WTCO for } (V, T, I, \omega) \tag{12}$$

According to Eq. (1) and (2), function μ is

(a) *nondecreasing*, because

$$\mu(E) \leq \mu(E'),\ \forall E \subseteq E' \subseteq K \tag{13}$$

(b) *submodular*. If $E \subseteq E' \subseteq K$, then

$$\mu(E+e) - \mu(E) \geq \mu(E'+e) - \mu(E'), \forall e \in K\ , \tag{14}$$

because E may induce additional TCCs for e as compared to its superset E'. In other words, adding $(E' - E)$ can only reduce the contribution of e towards WTCO. See Fig. 2.

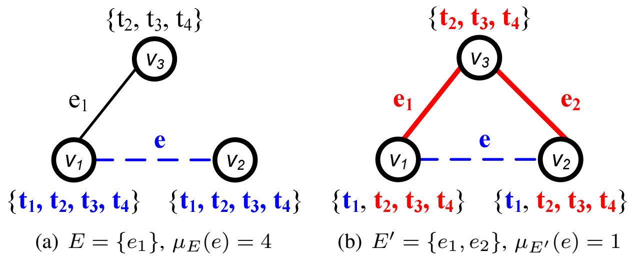


Figure 2: Function μ is submodular: if $E \subseteq E'$, then $\mu_E(e) \geq \mu_{E'}(e)$, i.e., the contribution of e decreases, as the current edge set progressively extends from E to E'.

With Eq. (12), (13), and (14), we can formulate Problem 1 as an instance of the *Submodular Set Cover* (SSC) problem:

Problem 2: SSC(K, ω, μ): Given a finite set K, a weight function $\omega : K \rightarrow \mathbb{R}^+$, and an integral-valued non-decreasing modular set function $\mu : 2^K \rightarrow \mathbb{N}$, find a subset $E \subseteq K$ s.t. $\sum_{e\in E}\omega(e)$ is minimum and $\mu(E) = \mu(K)$, i.e.,

$$\min_{E\subseteq K}\Big\{\sum_{e\in E}\omega(e) : \mu(E) = \mu(K)\Big\} \tag{15}$$

Alg. 2 Greedy algorithm for MinAvg-WTCO

```
GrMAW$(V,T,I,\omega)$
Input: $(V,T,I,\omega)$
Output: $E$, which forms a WTCO for $(V,T,I,\omega)$
1: $E \leftarrow \emptyset$
2: while $E$ does not form WTCO for $(V,T,I,\omega)$ do
3:     $e \leftarrow \arg\min_{e\in\overline{E}} \frac{\omega(e)}{\mu_E(e)}$
4:     $E \leftarrow E + e$
5: return $E$
```

Alg. 3 [30] Greedy algorithm for SSC

```
Input: $(K,\omega,\mu)$
Output: $E$ such that $\mu(E)=\mu(K)$
1: $E \leftarrow \emptyset$
2: while $\mu(E) \neq \mu(K)$ do
3:     $e \leftarrow \arg\min_{e\in\overline{E}} \frac{\omega(e)}{\mu(E+e)-\mu(E)}$
4:     $E \leftarrow E + e$
5: return $E$
```

Further, Alg. 2 for MinAvg-WTCO follows the same greedy heuristic as Alg. 3 for SSC.

Theorem 2 provides an approximation guarantee of Alg. 3 for SSC, which has guided us to derive Lemma 3.

Theorem 2: [30] Given a SSC instance (K,ω,μ), the output of Alg. 3 never exceeds the optimal by over a factor of $H\left(\max_{e\in K}\left(\mu(e)-\mu(\emptyset)\right)\right)$ where $H(m)=\sum_i^m\left(\frac{1}{i}\right)$.

Proof of Lemma 3: With Eq. (9), we have $\mu(\emptyset)=0$ and $\max_{e\in K}\mu(e) \leq |T|$, since a single edge can reduce the number of TCCs at most $|T|$. According to Theorem 2, the approximation ratio of Alg. 2 is bounded by $\log|T|$. ■

We can regard GrMAW for MinAvg-WTCO as a generalization of GM [11] for MinAvg-TCO: If all edges are of a uniform weight, then GrMAW is equivalent to GM. Lemma 3 improves the approximation bound of GM from $\mathcal{O}(\log(|V||T|))$ [11] to $\mathcal{O}(\log|T|)$ by using the analysis of submodular set functions. Lemma 3 and Theorem 1 provide two logarithmic bounds: Alg. 2 achieves $\mathcal{O}(\log|T|)$, while $\Omega(\log|V|)$ is a lower bound of polynomial-time approximation unless $P=NP$. There is no definite relationship between these two asymptotic bounds, because Problem 1 imposes none correlation between $|T|$ and $|V|$. In practice, the approximation ratio of Alg. 2 (i.e., $\mathcal{O}(\log|T|)$) can be better than $\Omega(\log|V|)$, if $|T|<|V|$. Note that [3] independently proves that the approximation ratio of $\mathcal{O}(\log|T|)$ is attainable in polynomial time in the context of graph construction with subgraph constraints.

The approximation ratio in Lemma 3 is tight for Alg. 2, because [11] constructs a MinAvg-TCO instance on which GM reaches the approximation bound $\mathcal{O}(\log|T|)$.

Lemma 4: There exists an instance (V,T,I,ω) on which GrMAW achieves an approximation ratio of $\mathcal{O}(\log|T|)$.

Lemma 5: Alg. 2 outputs a *WTCO* in time

$$\mathcal{O}(\max\left\{|V|^2|T|,|V|^4\right\})$$

Proof: Alg. 2 outputs a *WTCO*, because the algorithm only terminates when its progress measure $\mu(E)$ increases up to its limit $\mu(E)=\mu(K)$, i.e., $TCC(E)$ decreases $TCC(E)=TCC(K)$.

The while loop in Lines 2-4 dominates the runtime cost of Alg. 2, which contains two major parts: (a) the computation of edge contribution $\mu_E(e)$ and (b) the search for the most cost-effective edge.

Part (a): the cumulative running time of computing edge contribution during the entire execution of Alg. 2 can be implemented in $\mathcal{O}(|V|^2|T|)$ (see Lemma 2 and 3 in [9]).

Part (b): the cumulative running time of edge search of all iterations of Alg. 2 is $\mathcal{O}(|V|^4)$. First, it takes at most $|V|^2$ comparisons to select the most cost-efficient edge at each iteration. Second, the number of all possible edges $|K|=\Theta(|V|^2)$ is an upper bound for the total number of iterations.

Combining Part (a) and (b), the time complexity of Alg. 2 is $\mathcal{O}(|V|^2|T|+|V|^4)=\mathcal{O}(\max\left\{|V|^2|T|,|V|^4\right\})$. ■

VII. Primal-dual algorithms for MinAvg-WTCO

A. *Basic primal-dual algorithm for* MinAvg-WTCO

We design an approximation algorithm for MinAvg-WTCO using the *primal-dual* scheme [16].

We first formulate MinAvg-WTCO as a linear integer program, which Wolsey used to represent SSC [30]:

$$\begin{aligned} \text{minimize} \quad & \sum_{e\in K}\omega(e)x_e \\ \text{subject to} \quad & \sum_{e\in\overline{S}}\mu_S(e)x_e \geq \mu_S(\overline{S}), S\subseteq K \\ & x_e\in\{0,1\}, e\in K \end{aligned} \tag{16}$$

We assign an $0/1$ variable x_e for each edge $e\in K$. Each edge set $E\subseteq K$ corresponds to a *characteristic* vector $\mathbf{x}$ in Eq. (16): $e\in E$ iff $x_e=1, \forall e\in K$.

Lemma 6: [30] $\forall E\subseteq K$, E's characteristic vector $\mathbf{x}$ is *feasible* for Eq. (16) iff $\mu(E)=\mu(K)$, i.e., E forms a *WTCO*.

We relax the integrality constraint $x_e\in\{0,1\}$ to $x_e\geq 0$.

$$\begin{aligned} \text{minimize} \quad & \sum_{e\in K}\omega(e)x_e \\ \text{subject to} \quad & \sum_{e\in\overline{S}}\mu_S(e)x_e \geq \mu_S(\overline{S}), S\subseteq K \\ & x_e\geq 0, e\in K \end{aligned} \tag{17}$$

Note $x_e\leq 1$ is redundant, because any optimal solution $\mathbf{x}^*$ to this linear program satisfies $x_e^*\leq 1, \forall e\in K$.

Alg. 4 Primal-dual algorithm for MinAvg-WTCO

```
PdMAW(V, T, I, ω)
Input: $(V, T, I, \omega)$
Output: $E$, which forms a WTCO for $(V, T, I, \omega)$
1: $i \leftarrow 1, E_i \leftarrow \emptyset, \mathbf{y} \leftarrow 0$
2: while $WTPSO(V, T, I, E_i, \omega)$ is not a WTCO do
3:     $e_i \leftarrow \arg\min_{e \in \overline{E_i}} \left\{ \frac{\omega(e) - \sum_{S: e \notin S, S \neq E_i} \mu_S(e) y_S}{\mu_{E_i}(e)} \right\}$
4:     $y_{E_i} \leftarrow \frac{\omega(e_i) - \sum_{S: e_i \notin S, S \neq E_i} \mu_S(e_i) y_S}{\mu_{E_i}(e_i)}$
5:     $E_{i+1} \leftarrow E_i + e_i$
6:     $i \leftarrow i + 1$
7: $m \leftarrow i, E \leftarrow E_m$
8: return $E$
```

Eq. (18) is the dual linear program for Eq. (17), where we introduce a variable y_S for each subset of edges $S \subseteq K$:

$$\begin{aligned} \text{maximize} \quad & \sum_{S: S \subseteq K} \mu_S(\overline{S}) y_S \\ \text{subject to} \quad & \sum_{S: e \notin S} \mu_S(e) y_S \leq \omega(e), e \in K \\ & y_S \geq 0, S \subseteq K \end{aligned} \tag{18}$$

Based on Eq. (16), (17) and (18), we design PdMAW in Alg. 4, the primal-dual algorithm for MinAvg-WTCO. Alg. 4 follows the basic primal-dual method for approximation algorithm design [16]. Line 1 starts with a primal infeasible $\mathbf{x} = \mathbf{0}$ (i.e., $E_i = \emptyset$) and a dual feasible $\mathbf{y} = \mathbf{0}$. Lines 2-6 iteratively improve the feasibility of the primal solution and the optimality of the dual solution, ensuring a feasible primal solution in the end. The improvements to the primal and the dual progress hand-in-hand: The current dual is used to determine the improvement to the primal, and vice versa.

In Alg. 4, E_i is the edge set at the beginning of the i-th iteration, e_i is the edge added in the i-th iteration, and E is the final edge set returned. Let $E = \{e_1, \ldots, e_m\}$, then $E_1 = \emptyset$ and $E_i = \{e_1, ..., e_{i-1}\}, 1 < i \leq m$.

Suppose E_i does not constitutes a feasible primal solution $\mathbf{x}$ for Eq. (16), then at the start of the i-th iteration, $y_{E_i} = 0$ and E_i violates the primal constraint, because $\sum_{e \in \overline{E_i}} \mu_{E_i}(e) x_e = 0$ and $\mu_{E_i}(\overline{E_i}) > 0$. At the i-th iteration of Alg. 4, Lines 2-6 (a) find an edge e_i to be added to the current overlay E_i and (b) increase one dual variable y_{E_i} so that the dual constraint for e_i becomes tight, i.e.,

$$\sum_{S: e_i \notin S} \mu_S(e_i) \cdot y_S = \sum_{1 \leq l \leq i} \mu_{E_l}(e_i) \cdot y_{E_l} = \omega(e_i) \tag{19}$$

The first equality holds because we only set $y_{E_i} > 0$ at the i-th iteration. Hence, at the end of the i-th iteration, $y_S = 0$ if $S \notin \{E_1, ..., E_i\}$.

To find the edge e_i, we increase the dual variable y_{E_i} as much as possible from zero without violating any dual constraint, until some edge $e_i \in \overline{E_i}$ becomes tight, as Eq. (19) shows. This amounts to how Line 3 selects $e_i \in \overline{E_i}$ and how Line 4 sets y_{E_i}.

Lemma 7: Alg. 4 outputs a *WTCO* in time

$$\mathcal{O}(\max\{|V|^2|T|, |V|^4\})$$

Proof sketch: it follows along the same lines as the proof of Lemma 5 for GrMAW. Similar to GrMAW, we can think of PdMAW as an *adaptive greedy* algorithm that selects the most *cost-effective* edge at each iteration. It is adaptive in the sense that both algorithms update cost-effectiveness throughout their executions. The difference lies in how each algorithm defines the cost-effectiveness of an edge (cf. Line 3 of Alg. 2 and Line 3 of Alg. 4). Fortunately, all the arguments for GrMAW apply to PdMAW in terms of correctness and complexity. ■

Now we analyze the output edge set of Alg. 4. We introduce *contributive augmentation* in Def. 2.

Definition 2: Given an instance (V, T, I, ω) of MinAvg-WTCO, suppose $E_i = \{e_1, ..., e_{i-1}\}$ does not form a WTCO for (V, T, I, ω). Any edge set $A \subseteq \overline{E_i}$ is an *augmentation* of E_i, if $A \cup E_i$ forms a WTCO. Further, E_i's augmentation A is a *contributive augmentation*, if we can place an indexing order on $A = \{e_i, \ldots, e_m\}$ such that $\mu_{E_j}(e_j) > 0$, where $E_j = \{e_1, \ldots, e_{i-1}, \ldots, e_{j-1}\}, \forall i \leq j \leq m$. In other words, if we add $e_j \in A$ on top of E_i one by one according to the indices, then each edge e_j has a positive contribution towards *WTCO* with regard to E_j, the set of edges added before e_j.

By definition, in Alg. 4, $A = E \backslash E_i$ is a contributive augmentation of $E_i, \forall 1 \leq i \leq m$, since Line 3 always selects an edge with positive contribution.

Lemma 8: Alg. 4's approximation ratio is at most

$$\max_{\left\{\substack{X: X \subseteq K \\ \text{infeasible}}\right\}} \max_{\left\{\substack{A\text{: contributive} \\ \text{augmentation of } X}\right\}} \left\{ \frac{\sum_{e \in A} \mu_X(e)}{\mu_X(A)} \right\}$$

Proof: Alg. 4 produces an edge set E that forms a WTCO and a feasible solution $\mathbf{y}$ for Eq. (18). The algorithm selects $e \in K$ into E only when the corresponding dual constraint for e becomes tight, so we have:

$$\sum_{e \in E} \omega(e) = \sum_{e \in E} \sum_{S: e \notin S} \mu_S(e) y_S \tag{20}$$

$$= \sum_{S: S \subseteq K} \left(\sum_{e \in E \backslash S} \mu_S(e) \right) y_S \tag{21}$$

$$= \sum_{1 \leq i \leq m} \left(\sum_{e \in E \backslash E_i} \mu_{E_i}(e) \right) y_{E_i} \tag{22}$$

$$= \sum_{1 \leq i \leq m} \frac{\left(\sum_{e \in E \backslash E_i} \mu_{E_i}(e) \right)}{\mu_{E_i}(\overline{E_i})} \mu_{E_i}(\overline{E_i}) y_{E_i} \tag{23}$$

$$\leq \max_{1\leq i\leq m} \left\{ \frac{\left(\sum_{e\in E\setminus E_i} \mu_{E_i}(e)\right)}{\mu_{E_i}(\overline{E_i})} \right\} \sum_{1\leq i\leq m} \mu_{E_i}(\overline{E_i}) y_{E_i} \tag{24}$$

$$\leq \max_{\left\{\substack{X:X\subset K\\ \text{infeasible}}\right\}} \max_{\left\{\substack{A:\text{ contrib.}\\ \text{aug. of } X}\right\}} \left\{ \frac{\sum_{e\in A} \mu_X(e)}{\mu_X(A)} \right\} \sum_{S:S\subseteq K} \mu_S(\overline{S}) y_S \tag{25}$$

Eq. (25) compares the primal objective function with the dual objective function (cf. Eq. (18)), which serves as a lower bound for the optimal solution. ■

B. Primal-dual with reverse delete for MinAvg-WTCO

Now we turn to a modification of Alg. 4, the basic primal-dual algorithm. The idea is relatively simple: once a feasible edge set E has been obtained, we should examine the edges of E and delete any that are not needed for a feasible solution. In particular, [16], [14] showed that it is useful for the algorithm analysis to check for the possible removal of the elements of E in a certain order – in the reverse order in which the elements of E were added. This step is often referred to as *reverse delete*. We present the modified PdrdMAW in Alg. 5, the primal-dual with reverse delete for MinAvg-WTCO.

Definition 3: Given an instance of MinAvg-WTCO, (V, T, I, ω), suppose X is not a feasible solution and A is an augmentation of X. We call A a *minimal augmentation* of X, if $X \cup A - e$ is not feasible for any $e \in A$.

By Def. 2 and Def. 3, a minimal augmentation must be a contributive augmentation.

Lemma 9: Alg. 5's approximation ratio is at most

$$\max_{\left\{\substack{X:X\subset K\\ \text{infeasible}}\right\}} \max_{\left\{\substack{A:\text{ minimal}\\ \text{augmentation of } X}\right\}} \left\{ \frac{\sum_{e\in A} \mu_X(e)}{\mu_X(A)} \right\}$$

Proof: The proof follows the same lines of that for Lemma 8, except that in Eq.(25), we take A as a minimal augmentation instead of a contributive augmentation. ■

Together with Lemma 8, Lemma 9 shows that reverse delete helps guarantee the approximation factor of the primal-dual algorithm from a contributive augmentation to a minimal augmentation. We construct an example to further analyze this improvement. Denote by E_{Pdrd} and E_{Pd} the output edge set of PdrdMAW and PdMAW, respectively, then we have Lemma 10.

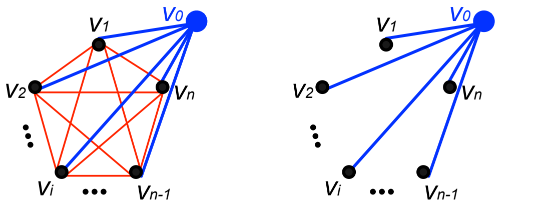
Lemma 10: The gap between PdrdMAW and PdMAW is at least $\frac{3}{2}$ in the worst-case scenario, i.e.,

$$\max \left\{ \frac{\omega(E_{\mathsf{Pd}})}{\omega(E_{\mathsf{Pdrd}})} \right\} \geq \frac{3}{2}$$

Proof: Consider an $(n+1)$-node set $V = \{v_0, v_1, \ldots, v_n\}$ and a n^2-topic set $T = \{t_{pq} | p, q = 1, 2, \ldots, n\}$.

Alg. 5 Primal-dual with reverse delete for MinAvg-WTCO

```
PdrdMAW($V, T, I, \omega$)
Input: $(V, T, I, \omega)$
Output: $E$, which forms a WTCO for $(V, T, I, \omega)$
1: Run PdMAW (i.e., Alg. 4) and get $E = \{e_1, ..., e_m\}$
2: for  $i = m$ down to 1 do
3:     if  $E - e_i$ forms a WTCO for $(V, T, I, \omega)$ then
4:         $E \leftarrow E - e_i$
5: return  $E$
```



(a) E_{Pd} (b) E_{Pdrd}

Figure 3: PdMAW vs. PdrdMAW

We assign edge weights as follows:

$$\omega(v_0, v_p) = \frac{2n-1}{2} + \epsilon,\ \epsilon > 0 \tag{26}$$
$$\omega(v_p, v_q) = 1 \tag{27}$$

where $1 \leq p, q \leq n$ and $p \neq q$.

Let $T(v)$ be the subscribed topics of node v, then:

$$T(v_0) = T \tag{28}$$
$$T(v_p) = \{t | t = t_{pq} \vee t = t_{qp}, 1 \leq q \leq n\}, 1 \leq p \leq n \tag{29}$$

In other words, if we divide T into n topic groups where

$$T_p = \{t_{p1}, t_{p2}, ..., t_{pn}\}, 1 \leq p \leq n,$$

then $T(v_p)$ contains two parts: (a) all topics in T_p and (b) the p-th topic t_{qp} from another topic group $T_q (\neq T_p)$. The interest matrix restricted to $V \times T_p, 1 \leq p \leq n$ is:

$$\begin{array}{c|cccccc} & t_{p1} & t_{p2} & \ldots & t_{pp} & \ldots & t_{pn} \\ \hline v_0 & 1 & 1 & \ldots & 1 & 1 & 1 \\ v_1 & 1 & 0 & \ldots & 0 & \ldots & 0 \\ v_2 & 0 & 1 & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v_p & 1 & 1 & \ldots & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v_n & 0 & 0 & \ldots & 0 & \ldots & 1 \end{array}$$

Therefore,

$$\mu_\emptyset(v_0, v_p) = |T(v_0) \cap T(v_p)| = 2n - 1 \tag{30}$$
$$\mu_\emptyset(v_p, v_q) = |T(v_p) \cap T(v_q)| = 2 \tag{31}$$

where $1 \leq p, q \leq n$ and $p \neq q$.

On the one hand, PdMAW produces a clique for this instance, i.e., $E_{\mathsf{Pd}} = K$. At the start of the first iteration,

$E_1 = \emptyset$; when we increase $y_\emptyset$ up to $1/2$, we have

$$\mu_\emptyset(v_0, v_p) \cdot y_\emptyset = \frac{2n-1}{2} < \frac{2n-1}{2} + \epsilon = \omega(v_0, v_p) \quad (32)$$

$$\mu_\emptyset(v_p, v_q) \cdot y_\emptyset = 1 = \omega(v_p, v_q) \quad (33)$$

where $1 \leq p, q \leq n$ and $p \neq q$. In other words, all edges in $\{(v_p, v_q), 1 \leq p, q \leq n \text{ and } p \neq q\}$ become tight, and Alg. 4 would select them all in $\frac{n(n-1)}{2}$ iterations. After that, Alg. 4 selects all n edges in $\{(v_0, v_p), 1 \leq p \leq n\}$ to attain WTCO.

On the other hand, PdrdMAW outputs $E_{\mathsf{Pdrd}} = \{(v_0, v_p) | 1 \leq p \leq n\}$, because the reverse delete step removes all edges in $\{(v_p, v_q), 1 \leq p, q \leq n \text{ and } p \neq q\}$.

As a result,

$$\frac{\omega(E_{\mathsf{Pd}})}{\omega(E_{\mathsf{Pdrd}})} = \frac{1 \cdot \frac{n(n-1)}{2} + (\frac{2n-1}{2} + \epsilon) \cdot n}{(\frac{2n-1}{2} + \epsilon) \cdot n}$$
$$= \frac{n^2 - n}{2n^2 + 2\epsilon n - 1} + 1 = \frac{3}{2}, \text{ as } n \to \infty$$

■

Lemma 9 and Lemma 10 show that reverse delete can be essential in eliminating redundant edges and bounding the approximation factors. However, this improvement is insignificant for typical pub/sub workloads in practice (see §IX). Moreover, this marginal gain comes at the cost of reduced time efficiency, as Lemma 11 shows.

Lemma 11: Alg. 5 outputs a *WTCO* in time $\mathcal{O}(|V|^4|T|)$.

Proof: The reverse delete step costs $\mathcal{O}(|V|^4|T|)$, since it takes $\mathcal{O}(|V|^2|T|)$ to check the feasibility of $E - e_i, \forall e_i \in E$. With Lemma 7, we obtain the time complexity of Alg. 5. ■

In principle, we can add reverse delete at the end of other algorithms (e.g., Alg. 1 and Alg. 2). However, this step is not critical to improve the approximation ratios of these algorithms. Besides, the additional time complexity imposed by the reverse delete step is substantial (see Lemma 11 and empirical evaluation in §IX).

VIII. Divide-and-conquer for MinAvg-WTCO

Inspired by [7], [10], we extend a divide-and-conquer framework for MinAvg-WTCO to reduce time complexity.

According to §V, VI and VII, the number of nodes is a dominant factor for the running time of the algorithms. We can improve the running time by downsizing the node set in the computation. This advises a divide-and-conquer strategy for MinAvg-WTCO (see Alg. 6):

I. *Divide* the given MinAvg-WTCO problem into several subproblems that are similar to the original problem but smaller than the size of the original node set (Line 1).

II. *Conquer* sub-MinAvg-WTCO problems independently and build a WTCO for each partition (Lines 2-4). We refer to $E_d, d = 1, \ldots, p$, as *inner edges*.

III. *Combine* these sub-WTCOs to one WTCO for the original problem by adding cross-partition edges, i.e., *outer edges* (Line 5).

Alg. 6 Divide-and-conquer framework for MinAvg-WTCO

```
DcMAW($V, T, I, \omega$)
Input: $(V, T, I, \omega)$
Output: $E$, which forms a WTCO for $(V, T, I, \omega)$
1: Divide $V$ into $p$ partitions: $V_d, d = 1, \ldots, p$
2: for $d = 1$ to $p$ do
3:     $I_d \leftarrow I|_{V_d}, \omega_d \leftarrow \omega|_{V_d}$
4:     $E_d \leftarrow$ Conquer the subproblem $(V_d, T, I_d, \omega_d)$ such
       that $E_d$ forms a WTCO for each partition
5: $E_{out} \leftarrow$ Combine by adding cross-partition edges
6: return $E \leftarrow (\bigcup_{d=1}^{p} E_d) \cup E_{out}$
```

Alg. 6 specifies a divide-and-conquer framework, which embraces a broad design space at each phase. We provide a general discussion about possible approaches at each phase and show the practicality of divide-and-conquer for pub/sub workloads deployed across multiple data centers in §IX-G. It is of interest to dive deep into different design options in this divide-and-conquer framework, which will be our future work.

For the *divide* phase, [10] adopts random partitioning for MinAvg-TCO, because it is fast and robust, preserves interest correlation, and supports decentralization. For MinAvg-WTCO, however, we need more intelligent mechanisms with respect to topology, e.g., the binning scheme with predefined landmark nodes [25], or clustering of nodes with closer identifiers that encode network structure and geographical positions [6], [15]. In practice, we can also divide the node set according to the prior knowledge about the data centers – each data center naturally forms a partition.

For the *conquer* phase, we can employ existing algorithms (e.g., Alg. 2) to tackle the sub-MinAvg-WTCO problems.

For the *combine* phase, we can follow the methods in [10]: selecting representative nodes (rep nodes) from each partition and adding edges across rep nodes in different partitions. Note that it is an open problem about how to select rep nodes and how to add cross-partition edges.

IX. Evaluation

We implement GrMAW, PdMAW, PdrdMAW, and other auxiliary algorithms in Java. We denote by $WTCO_{\mathsf{Alg}}$ the WTCO produced by Alg, by $\overline{\omega}_{\mathsf{Alg}}$ the average weighted node degree in $WTCO_{\mathsf{Alg}}$, by $\overline{d}_{\mathsf{Alg}}$ the average unweighed node degree in $WTCO_{\mathsf{Alg}}$, and by $\mathbb{T}_{\mathsf{Alg}}$ the running time of Alg, where Alg stands for any of the discussed algorithms. [1]

A. Experiment workloads

1. Pub/Sub interest matrix: Our inputs have the following ranges: $|V| \in [1\,000, 10\,000]$, $|T| \in [100, 1\,000]$, and $|T(v)|_{\text{mid}} \in [50, 150]$, where we define the middle

[1] When there is no ambiguity, we often omit MAW in the subscript: for example, $\overline{\omega}_{\mathsf{Gr}}$ is short for $\overline{\omega}_{\mathsf{GrMAW}}$.

node subscription size as $|T(v)|_{\mathrm{mid}} = \frac{|T(v)|_{\min}+|T(v)|_{\max}}{2}$. Each topic $t \in T$ is associated with probability $q(t)$ such that $\sum_t q(t) = 1$ and each node subscribes to t with a probability $q(t)$. The value of $q(t)$ is distributed according to either a uniform, a Zipf (with $\alpha = 2.0$), or an exponential distribution, which we call Unif, Zipf, or Expo, respectively. These distributions are representative of actual workloads used in industrial pub/sub systems today [12]. Stock market monitoring engines use Expo for the study of stock popularity in the New York Stock Exchange [29]. Zipf faithfully describes the feed popularity distribution in RSS feeds [19].

2. Edge weight function:

We assign edge weights based on the Internet latency data sets: (1) King [17] measures of the latencies between a set of DNS servers. (2) Meridian [31] provides node-to-node latency measurements between 2500×2500 nodes between May 5-13 2004. We normalize both data sets with their average latencies and initialize weights for all edges.

B. Impact of $|V|$

Fig. 4 depicts the comparison among GrMAW, PdMAW, and TrMAW as $|V|$ increases from 100 to 1 000, where we fix $|T| = 200$, $|T(v)|_{\min} = 20$, and $|T(v)|_{\max} = 100$.

We look at weighted average node degree. First, GrMAW and PdMAW significantly outperform TrMAW: $\overline{\omega}_{\mathsf{Tr}} \approx 7.28 \cdot \overline{\omega}_{\mathsf{Gr}}$ on average, and when $|V| = 1\,000$ under Unif, $\overline{\omega}_{\mathsf{Tr}} = 224$, while $\overline{\omega}_{\mathsf{Gr}} = 18.76$. These results conform to the design of GrMAW and PdMAW: both exploit the subscription correlation for constructing the WTCOs and achieve better approximation ratios than TrMAW. This design becomes increasingly important as the number of nodes scales up: $\overline{\omega}_{\mathsf{Tr}}$ tends to increase, while both $\overline{\omega}_{\mathsf{Gr}}$ and $\overline{\omega}_{\mathsf{Pd}}$ decrease, because larger node sets lead to higher opportunities for both algorithms to find more cost-effective edges. Second, GrMAW is slightly better than PdMAW and consistently produces the lowest weighted average node degrees: $\overline{\omega}_{\mathsf{Gr}} \approx \overline{\omega}_{\mathsf{Pd}} - 3.35$, on average.

We look at the running time. First, GrMAW runs significantly faster than PdMAW: the ratio of $\mathbb{T}_{\mathsf{Gr}}/\mathbb{T}_{\mathsf{Pd}}$ is 15.7% under Unif, 26.6% under Zipf, and 28.2% under Expo. The runtime cost gap between GrMAW and PdMAW enlarges as the number of nodes increases: $\mathbb{T}_{\mathsf{Pd}} = 60.7 \cdot \mathbb{T}_{\mathsf{Gr}}$ at $|V| = 1000$ under Unif. As we mentioned in §VII, we can regard both GrMAW and PdMAW as greedy algorithms. However, PdMAW defines a more complicated *cost-effectiveness* for edge selection, which requires more runtime expense. Second, TrMAW has slightly better time efficiency than GrMAW at the cost of overlay quality.

C. Impact of $|T|$

Fig. 5 shows how GrMAW, PdMAW, and TrMAW perform when $|T|$ varies. We increase $|T|$ from 100 to 1000 and fix $|V| = 800$, $|T(v)|_{\min} = 20$, and $|T(v)|_{\max} = 100$.

Referring to weighted average node degrees, first, they increase with the number of topics for all algorithms. The reason is that increasing the number of topics leads to reduced correlation among the nodes. GrMAW and PdMAW output much lower weighted average node degrees than TrMAW, since TrMAW does not leverage correlation: $\overline{\omega}_{\mathsf{Tr}} - \overline{\omega}_{\mathsf{Gr}} = 221$, under Unif on average. Second, GrMAW obtains a marginally better weighted average node degree as compared to PdMAW.

Referring to the time efficiency, GrMAW is superior to PdMAW. As the correlation increases, GrMAW gains more time efficiency as compared to PdMAW: $\mathbb{T}_{\mathsf{Gr}}/\mathbb{T}_{\mathsf{Pd}}$ is on average 19.5%, 9.6%, and 4.5%, under Unif, Zipf, and Expo, respectively. This runtime gap comes from the fact that the cost-effectiveness of GrMAW is relatively easy to compute: GrMAW simply updates edge contribution $\mu_E(e)$ at each iteration (Line 3 of Alg. 2), while PdMAW also has to recalculate the dual variables accordingly (Line 3 of Alg. 4).

D. Impact of subscription size

Fig. 6 depicts how the subscription size affects these algorithms. We set $|V| = 800$, $|T| = 200$, and $|T(v)|_{\mathrm{mid}} \in [50, 150]$.

We focus on the weighted average node degrees. First, as the subscription size increases, both $\overline{\omega}_{\mathsf{Gr}}$ and $\overline{\omega}_{\mathsf{Pd}}$ decrease. This decrease occurs because the growth of subscription size causes increased correlation across the nodes. Upon higher correlation, an edge addition to the overlay has on average a higher contribution towards the WTCO, because nodes share more common interests. Second, TrMAW suffers from much higher weighted average node degrees, because it ignores interest correlation.

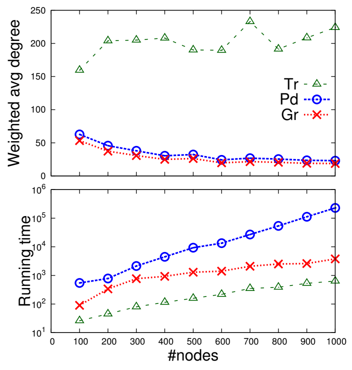
We focus on the running time. GrMAW runs significantly faster than PdMAW. $\mathbb{T}_{\mathsf{Gr}}/\mathbb{T}_{\mathsf{Pd}}$ is on average 3.0%, 4.6%, and 6.4%, under Unif, Zipf, and Expo, respectively.

In summary, for Fig. 4, Fig. 5 and Fig. 6, GrMAW achieves the best performance with regard to the trade-offs between weighted average node degree and runtime cost. The weighted average node degrees of TrMAW are a number of times higher than GrMAW (and PdMAW), which demonstrates the general importance of exploiting correlation for pub/sub WTCO design. Although PdMAW is comparable to GrMAW in terms of the weighted average node degrees ($\overline{\omega}_{\mathsf{Gr}}$ is slightly better than $\overline{\omega}_{\mathsf{Gr}}$), the running time of PdMAW is inferior, mainly due to its more expensive computation for edge cost-effectiveness at each iteration.

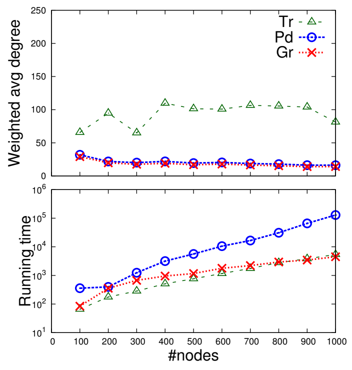
E. Reverse delete step

We only report on PdMAW in §IX-B, §IX-C and §IX-D. This subsection empirically explores the reverse delete step in the primal-dual algorithms for MINAVG-WTCO.

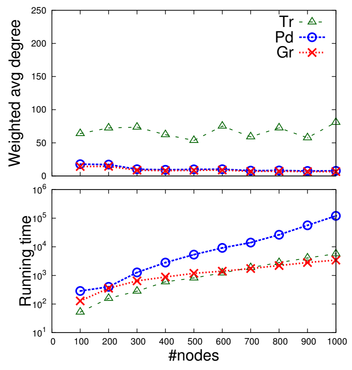
Fig. 7 compares PdMAW and PdrdMAW under Zipf. We fix $|T| = 200$, $|T(v)|_{\min} = 20$, $|T(v)|_{\max} = 100$, and $|V| \in [100, 1\,000]$. PdrdMAW marginally improves weighted and



(a) Meridian Unif

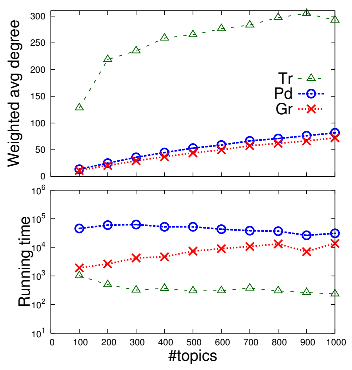


(b) Meridian Zipf

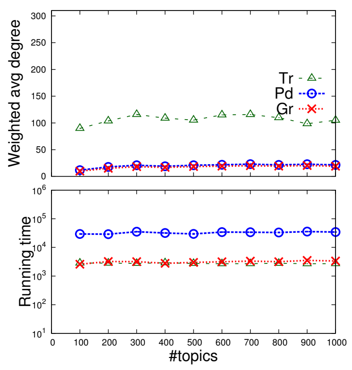


(c) Meridian Expo

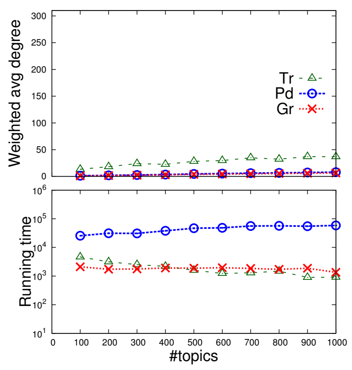
Figure 4: GrMAW vs. PdMAW vs. TrMAW wrt. $|V|$



(a) Meridian Unif

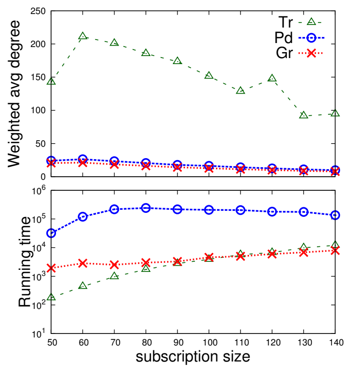


(b) Meridian Zipf

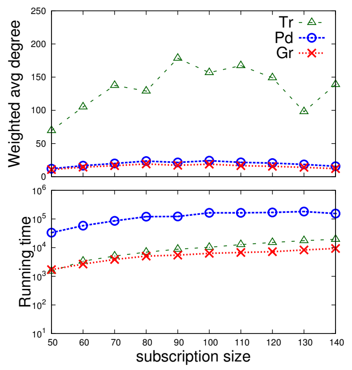


(c) Meridian Expo

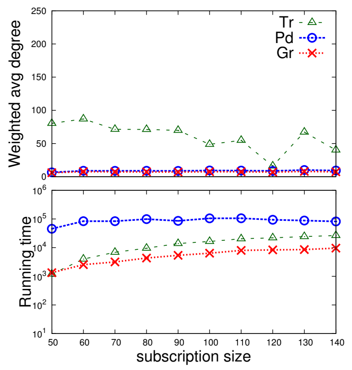
Figure 5: GrMAW vs. PdMAW vs. TrMAW wrt. $|T|$



(a) Meridian Unif



(b) Meridian Zipf



(c) Meridian Expo

Figure 6: GrMAW vs. PdMAW vs. TrMAW wrt. subscription size

unweighed average node degrees: $\overline{\omega}_{\mathsf{Pd}} - \overline{\omega}_{\mathsf{Pdrd}} = 5.10$ and $\overline{d}_{\mathsf{Pd}} - \overline{d}_{\mathsf{Pdrd}} = 3.87$, on average. However, with this insignificant reduction in the output node degrees, the reverse delete step imposes a 50-fold increase on the runtime cost: $\mathbb{T}_{\mathsf{Pdrd}} \approx 50 \cdot \mathbb{T}_{\mathsf{Pd}}$, on average. These results are in line with our analysis about primal-dual algorithms in §VII: the feasibility check for each possible edge deletion is costly.

F. *GrMAW versus GM*

In this subsection, we compare GrMAW and GM [11] to show the significance of incorporating edge weights into the pub/sub overlay design. Both GrMAW and GM adopt the greedy algorithm design. The only difference is that GrMAW considers edge weights while GM does not. GM guarantees the lowest unweighted average node degree in the TCO among all known polynomial-time algorithms.

We also look at *topic diameters* in the output overlays. Given $WTCO(V, T, I, E, \omega)$, we denote by $\omega diam(t)$ and $diam(t)$, the weighted and unweighted topic diameter for $t \in T$. More specifically, $\omega diam(t) = \omega diam(G^{(t)})$ where $\omega diam(G^{(t)})$ is the maximum shortest weighted distance between any two nodes in $G^{(t)} = (V^{(t)}, E^{(t)}, \omega)$, and $diam(t)$ follows the same definition in the unweighted version. We denote the weighted (and unwegihted) average topic diameter across all topics as $\overline{\omega diam}$ (and $\overline{diam}$).

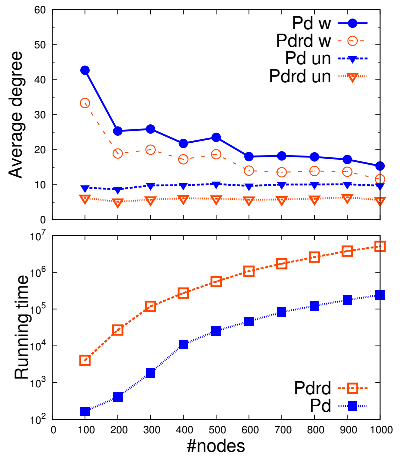


Figure 7: PdMAW versus PdrdMAW

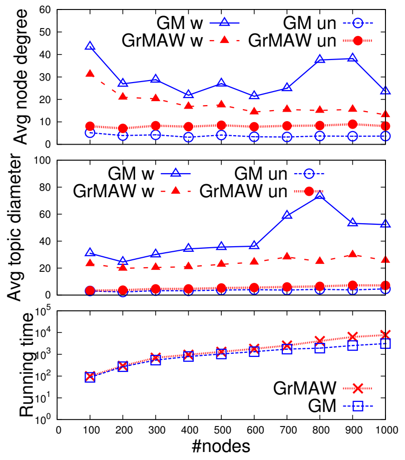


Figure 8: GrMAW versus GM

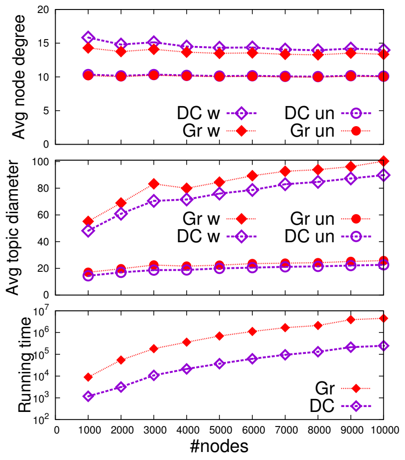


Figure 9: DcMAW versus GrMAW

In Fig. 8, $|V| \in [100, 1\,000]$, $|T| = 200$, $|T(v)|_{\min} = 20$, and $|T(v)|_{\max} = 100$, under Unif.

GrMAW considerably outperforms GM on the weighted metrics (i.e., average node degrees and topic diameters). GrMAW produces lower values than GM, and their gaps increase as the input instance scales up with the number of nodes: $\overline{\omega}_{\mathsf{GrMAW}} \approx 0.4 \cdot \overline{\omega}_{\mathsf{GM}}$, and the gap is about 30, on average; $\overline{\omega diam}_{\mathsf{GM}} - \overline{\omega diam}_{\mathsf{GrMAW}} = 124$ when $|V| = 1000$ under Unif. GM exhibits these disadvantages, because it ignores edge weights: GM always selects an edge with the highest contribution. Consequently, GM may add many edges with unnecessarily heavy weights. These results demonstrate that TCO design algorithms without edge weights are not powerful enough to address the pub/sub overlay with locality concern.

GrMAW and GM are close in unweighted criteria and running time: $\overline{d}_{\mathsf{GrMAW}} - \overline{d}_{\mathsf{GM}} \approx 4$, $\overline{diam}_{\mathsf{GrMAW}} - \overline{diam}_{\mathsf{GM}} \approx 2.7$, and $\mathbb{T}_{\mathsf{GrMAW}}/\mathbb{T}_{\mathsf{GM}} \approx 0.7$. GrMAW performs well in unweighted measurements, because it considers both edge contribution and weight for computing the cost-effectiveness (see Line 3 in Alg. 2). In other words, GrMAW implicitly optimizes unweighted $\overline{d}_{\mathsf{GrMAW}}$ and $\overline{diam}_{\mathsf{GrMAW}}$.

GrMAW and GM are of similar running time: $\mathbb{T}_{\mathsf{GrMAW}}/\mathbb{T}_{\mathsf{GM}} \approx 0.7$, on average. GrMAW is slightly slower than GM, because cost-effectiveness requires more computation than edge contribution at each iteration.

G. Divide-and-conquer

We demonstrate practicality of the divide-and-conquer framework for pub/sub overlay construction across multiple data centers. To simulate this application scenario, we generate edge weights with three parameters: the number of clusters C, the inner edge weight ω_{in}, and the outer edge weight ω_{out}. We divide the node set V into $C = |V|/100$ clusters, the weight for a local edge within a cluster is $\omega_{in} = 1$, and the weight for a long-range edge across clusters is $\omega_{out} = 10$.

We implement a simple divide-and-conquer algorithm under the framework of Alg. 6. In the *divide* phase, we employ the classic k-means clustering to divide all nodes into C partitions based on edge weights, where each partition corresponds to a cluster in the input instance. In the *conquer* phase, we use the GrMAW algorithm for each sub-problem. In the *combine* phase, we follow the representative set method [10]: we greedily select a subset of nodes from each partition and add cross-partition edges as GrMAW.

Fig. 9 depicts that DcMAW produces high-quality WTCOs in terms of both average node degrees and topic diameters. First, DcMAW has slightly higher average node degrees than GrMAW: $\overline{\omega}_{\mathsf{DC}} - \overline{\omega}_{\mathsf{Gr}} = 0.88$, $\overline{d}_{\mathsf{DC}} - \overline{d}_{\mathsf{Gr}} = 0.097$, on average. These gaps shrink as the input scales up with the number of nodes. Second, DcMAW outperforms GrMAW with respect to both weighted and unweighted average topic diameters, thanks to slightly higher node degrees.

Although DcMAW and GrMAW produce similar WTCOs, DcMAW runs considerably faster than GrMAW, and the runtime speedup of DcMAW is more profound as the instance scales up: $\mathbb{T}_{\mathsf{DC}}/\mathbb{T}_{\mathsf{Gr}} \approx 5.7\%$ on average, and the running time ratio decreases as the number of nodes increases.

X. Conclusions

We propose MinAvg-WTCO to study locality-aware overlays for topic-based pub/sub. We design several approximation algorithms for this NP-hard problem. The greedy algorithm GrMAW exhibits the best performance balance between overlay quality and time efficiency, both theoretically and empirically. We also show the practicality of the divide-and-conquer framework to build a pub/sub WTCO in cloud environments across multiple data centers.

Acknowledgment

This research has been partially supported by European Commission (EU) under the FP7 Programme Projects: iSocial (No. 316808) and COMPOSE (No. 317862).

References

[1] IBM IoT Foundation. http://internetofthings.ibmcloud.com/.

[2] Message Queue Telemetry Transport. http://mqtt.org/.

[3] D. Angluin, J. Aspnes, and L. Reyzin. Network construction with subgraph connectivity constraints. *J. of Combina. Optimization*, 2013.

[4] 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*.

[5] 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.*, 2007.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *JSAC*, 2002.

[7] C. Chen, H.-A. Jacobsen, and R. Vitenberg. Algorithms based on divide and conquer for topic-based publish/subscribe overlay design. *IEEE/ACM Transactions on Networking*, 2015.

[8] C. Chen, R. Vitenberg, and H.-A. Jacobsen. Brief announcement: Constructing fault-tolerant overlay networks for topic-based publish/subscribe. In *PODC' 13*.

[9] C. Chen, R. Vitenberg, and H.-A. Jacobsen. A generalized algorithm for publish/subscribe overlay design and its fast implementation. In *DISC'12*.

[10] C. Chen, R. Vitenberg, and H.-A. Jacobsen. Scaling construction of low fan-out overlays for topic-based publish/subscribe systems. In *ICDCS'11*.

[11] 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*.

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

[13] 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.

[14] T. Fujito. On approximation of the submodular set cover problem. *Operations Research Letters*, 1999.

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

[16] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In *Approximation Algorithms for NP-hard Problems*. 1997.

[17] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *IMW '02*.

[18] J. Han, D. Watson, and F. Jahanian. Topology aware overlay networks. In *Infocom' 05*.

[19] 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*.

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 2008.

[21] M. Onus and A. W. Richa. Parameterized maximum and average degree approximation in topic-based publish-subscribe overlay network design. In *ICDCS'10*.

[22] J. A. Patel, E. Rivière, I. Gupta, and A.-M. Kermarrec. Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. *Computer Networks*, 2009.

[23] F. Rahimian, T. Le Nguyen Huu, and S. Girdzijauskas. Locality-awareness in a peer-to-peer publish/subscribe network. In *DAIS'12*.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Sigcomm'01*.

[25] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Infocom'02*.

[26] J. Reumann. GooPS: Pub/Sub at Google. Lecture at CANOE Summer School, Oslo, Norway, August, 2009.

[27] V. Setty, M. van Steen, R. Vitenberg, and S. Voulgaris. Poldercast: Fast, robus, and scalable architecture for p2p topic-based pub/sub. In *Middleware'12*.

[28] M. A. Tariq, B. Koldehofe, and K. Rothermel. Efficient content-based routing with network topology inference. In *DEBS '13*.

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

[30] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 1982.

[31] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Sigcomm '05*.