On the Construction of the Minimum Cost Content-Based Publish/Subscribe Overlays

Yaxiong Zhao and Jie Wu
Department of Computer and Information Sciences
Temple University
{yaxiong.zhao, jiewu}@temple.edu

***Abstract*—Content-based publish/subscribe overlay is gaining popularity in large-scale content distribution applications for its flexibility and anonymity. Since such overlays are built on top of diverse infrastructures, minimizing the cost of using network resources in the overlays is challenging. In this paper, we tackle the problem using a *combinatorial optimization* approach. We assume that each user can simultaneously be *publisher* and *subscriber*, and *brokers* are dedicated servers. The problem is proved to be NP-hard and APX-complete. Therefore, we formulate an integer programming (ILP) problem, and design a *two-stage optimization* algorithm, which captures the cost of routing traffic in different parts of the overlay. This algorithm separately approximates the sub-problem of each stage and then combines the results to obtain the final results. We further propose a novel formulation based on *sub-channeling* and *Steiner tree/star problem*, which simplifies the analysis of content-based routing and facilitates an intuitive approximation algorithm. The approximation algorithm is then translated into a distributed algorithm that dynamically adjusts the connections between brokers and clients when the network undergoes dynamisms. Simulation studies are conducted to verify the performance of our proposed algorithms.**

***Index Terms*—Content-based publish/subscribe overlay, combinatorial optimization, integer programming, Steiner tree/star, NP-hard, APX-complete, approximation algorithm.**

I. Introduction

Content-based publish/subscribe (CBPS) is a powerful communication paradigm in which data content is treated as a first class entity. CBPS uses multiple attribute constraints [8] to identify contents and clients' interests. Clients register their interests on brokers through *subscriptions*. Messages are routed by brokers according to the matching results between their contents and clients' subscriptions, which substantially deviates from the end-to-end model used in Internet [16]. Due to its inherent scalability, flexibility, and anonymity [8], it is gaining popularity in various content distribution applications, including stock exchange system [1], enterprise information platform [3], and online news dissemination system [14].

In this paper, we consider the problem of constructing content-based pub/sub overlay networks (we use "overlay", "pub/sub overlay", or "overlay network" in this paper to represent the same concept hereafter for brevity). As shown in Fig. 1, a pub/sub overlay organizes tens of thousands of users and brokers into a single connected network. A *user/client* can be a *publisher*, which produces messages; and/or a *subscriber*, which consumes messages from publishers. *Brokers* are responsible for collecting subscriptions and messages, which are dedicated servers that do not publish or consume messages.

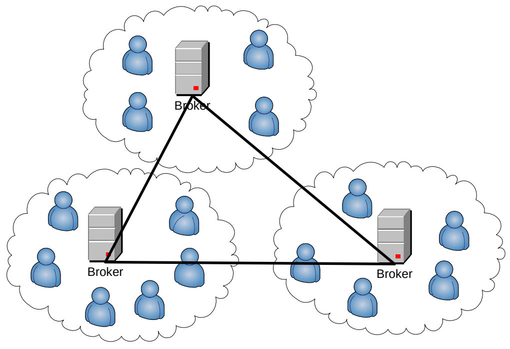


Fig. 1. A content-based publish/subscribe overlay.

Since a pub/sub overlay is formed by brokers and clients situated on top of a underlying network, nodes (users and brokers) rely on underlying network resources to communicate. Since using network resources, especially bandwidth, on overlays requires paying costs to the network owners, how the overlay is constructed and the traffic is routed determine the total cost. As a result, constructing such overlays gives rise to a *combinatorial optimization* problem that minimizes the cost of network resource use. Previous work on optimal construction of pub/sub overlays do not distinguish between clients and brokers [7], [6] and focuses on *topic-/channel-based pub/sub* that classifies the content space into multiple categories. In this model, clients are allowed to directly connect with each other; which is problematic, since clients are able to collect information from peers, which compromises the anonymity of the pub/sub system. Additionally, their goal is to minimize the number of overlay links, which is much simpler than ours.

In this paper, we assume that all overlay links have linear cost functions. That is, the price of transmitting w units of data on a link with a price of p incurs a cost of $p \times w$, where p in this paper. Here, p can be used to model various performance metrics. For example, p can be *delay*, and $p \times w$ represents the weighted sum of the delays of all network traffic. We choose linear cost function because it is sufficient in current Internet and is more tractable.

The first step is solving this optimization problem is to estimate the traffic load between users, namely, the *traffic matrix*. On the publisher side, we need to obtain the *message generating rate density function*. Given this and a subscriber's subscription, its integration on the content space of the subscription gives the rate of traffic needs to be forwarded from the publisher to the subscriber. This information can be obtained from publishers' advertisements. If no advertisement is available, the information is summarized from the publisher's message delivery history.

After obtaining the traffic matrix in the network, we try to connect all the users through brokers. The restriction is that users are not allowed to directly connect with each other, and they must connect with one and only one broker. Brokers form a complete graph on which traffic can be placed, and the link costs between brokers satisfy the *triangular inequality*. Our objective is then to find a connection between users and brokers and a traffic placement between brokers, so that the overall cost of the overlay is minimized.

We attack the problem first by separating the entire network cost into two parts: *access*, which deals with the cost of forwarding traffic between brokers and users; *core*, for the cost between brokers. It turns out that if the cost of overlay links comply with the triangular inequality [2], the optimal solutions to both problems can be found in polynomial time. Our first approximation algorithm is based on optimizing each sub-problem sequentially in two stages, and then combining them together to get the final overlay. We then look at another scheme that takes advantage of the property of content-space, which optimizes the cost of both access and core parts together. The corresponding solution is based on *Steiner tree/star* and has a better performance.

Even an optimal static topology cannot guarantee the efficiency when the traffic matrix of the network changes. In a overlay network, there are churns, and changes of the advertisements and subscriptions of users happening constantly. The message generation rates of publishers also change. In such situations, the system should be able to reconfigure the topology using distributed operations. We show in this paper that our Steiner-tree/star-based centralized algorithm can be directly implemented as a distributed manner. To summarize, our contributions in this paper are:

- We study the problem of constructing optimal content-based pub/sub overlay. The problem is formulated as a *combinatorial optimization* problem with the objective of minimizing the cost of bandwidth use in the entire network. We prove its NP-hardness and give *integer programming* (ILP) formulations.
- We present novel formulations of the problem based on the separation of network cost and sub-channeling of the content space. We present two-stage approximation algorithms that run fast and produce fairly good results. A novel Steiner-tree-based approximation algorithm is then presented, which produces better results. We also show that the steiner-tree-based algorithm allows a straightforward distributed implementation.
- We conduct extensive simulations studies based on realistic application settings. Our simulation results show that our solution helps reduce system cost significantly.

The rest of the paper is organized as follows: Section II defines the problem; Section III presents two two-stage approximation algorithms. Section IV discusses another formulation of the problem and proposes another approximation scheme; Section V introduces our system APIs; Section VI presents the simulation results; Section VII summarizes related work; Section VIII concludes this paper.

II. The minimum cost pub/sub overlay construction problem

In this section, we formally define the minimum cost pub/sub overlay construction problem and give its complexity. We then present an integer programming formalism.

A. Problem statement

Given a set of brokers $\mathbb{B}$ and a large number of users $\mathbb{U}$, for each user $i \in \mathbb{U}$, a generating rate function is given to summarize the distribution of messages generated at this user; it also has a subscription indicating its interests. We then are able to find expected bandwidth use for any user or broker. We are requested to wire the brokers and nodes into a connected overlay that has the following constraints:

- Users are not allowed to connect with each other. Each user must connect with one and only one broker;
- Brokers can connect to users and brokers;
- All links between brokers and users are bidirectional, which have the same cost function for traffic directed through both directions;
- The maximum traffic that can pass through a broker is a constant.

Our goal is to wire the nodes into a connected overlay and distribute traffic on the overlay links, so that the overall cost of using the overlay links are minimized.

In a traditional distributed pub/sub system, brokers are connected in an acyclic graph. In overlay networks, a tree topology can substantially reduce the routing complexity, which is desirable in many situations where the complexity of the system is of great importance. Therefore, another variation of the problem is to rewire all the brokers into a connected acyclic graph, where the objective is the same as we discussed before. We are not concerned with the wiring of users and brokers since their wiring process guarantees that the entire network is a tree once brokers are organized into a tree.

A similar problem is the topic-connected-pub-sub-overlay construction [7] problem, which is defined as *topic connectivity* problem. The problem's objective, however, is to find the minimum number of edges to connect users interested in multiple subjects. Our problem is considerably harder than the optimal topic connectivity problem. The differences are:

- Our problem aims to provide optimal connectivity and traffic scheduling on content-based pub-sub networks. The difference of the requirements of these two types of pub/sub networks results in completely different problem structures. The solutions of our problem have more extensive use than the the topic-/channel-based networks;
- The objective of our problem is to find the wiring with the minimum cost. The optimal topic connectivity problem, however, is to find one with minimum edges, which does not consider the bandwidth use;

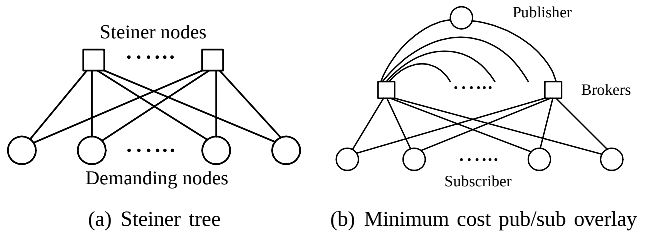


Fig. 2. The conversion from a Steiner tree problem to a minimum cost content-based pub/sub overlay problem.

- Our problem treats brokers and users differently. But in optimal topic connectivity problem these two roles are treated as the same;

A more closely related problem is the Steiner tree problem [17]. The major difference is that our problem aims to find the configuration that has the minimum delay instead of minimum edge weight in the steiner tree problem. We obtain the complexity of this problem in the following theorem:

Theorem 1. *The minimum cost CBPS overlay problem is NP-hard and APX-complete.*

Proof: This problem is NP-hard and APX-complete. We can show that the edge weighted Steiner tree/star problem can be reduced to a special version of the problem, which is a special case of the general Steiner tree/star problem [13] obtained by omitting the opening cost of steiner nodes. We show below that any Steiner tree problem can be expressed as an optimal overlay construction problem. To see this, consider an instance of the Steiner tree problem, as shown in Fig. 2. An identical optimal overlay construction problem is obtained as follows:

- All Steiner nodes become brokers;
- All demand nodes become users;
- Select a user as the only publisher. Set all other users' generating function to none, so that they do not generate messages;
- All subscribers have the identical subscription that matches all the messages generated by the sole publisher;
- The publisher generates a message with a unit bandwidth demand;
- All edges become overlay links;
- The unit bandwidth cost of each link is set to the edge weight;

Note that since all subscribers are interested in the same message, any message will not traverse a link more than once. Additionally, since the publisher generates messages that consume a unit of bandwidth, the cost of the optimal overlay will be identical to the weight of the optimal Steiner tree. Therefore the Steiner tree problem is reduced to the minimum cost pub/sub overlay construction problem, which means that it is at least as hard as the Steiner tree problem and is NP-hard. Similarly, since the metric Steiner-tree problem is APX-complete [17], our problem also is APX-complete. The theorem is proved. ■

B. An integer programming formulation

Integer linear programming (ILP) is a convenient tool of solving large-scale combinatorial problems because there are plenty of available solvers. We provide an ILP formulation of the minimum cost CBPS overlay problem, which is based on the classic Steiner tree formulation.

The problem is quite complex since content-based routing makes messages *non-replicative*. Non-replicative means that a message will be consumed by any broker that has the matched subscriptions, so that it will never be transmitted more than once on any overlay link. An example is that the actual amount of traffic of two flows traversing the same overlay links in the overlay might be less than the sum of the traffic of the two flows. As a result, we cannot distribute traffic among brokers using the traditional flow model.

For each user $i \in \mathbb{C}$, there is a message generating rate function $G_i(\cdot)$. This function is defined on the entire content space $\mathbb{S}$. The message generated at user i is given by integrating $G_i(\cdot)$ over $\mathbb{S}$, assuming that the union of the subscriptions of all users encompass the entire content space. The outbound traffic of user i is:

$$b_i(out) = \int_{\mathbb{S}} G_i(\cdot) d\cdot$$

On the contrary, the incoming traffic for user i is all the messages that matches its subscriptions S_i. It is obtained by integrating all the generating function of users other than i over S_i and then summing them up:

$$b_i(in) = \sum_{j \neq i} \int_{S_i} G_j(\cdot)) d\cdot$$

We use a binary variable $x_{ij} \in \{0, 1\}$ to indicate whether user i is connected to broker j, and a real-valued variable c_{ij} to represent the price of the overlay link between i and j. It is straightforward to compute the cost of forwarding traffic between brokers and users with the following equation:

$$C_1 = \sum_i \sum_j [x_{ij}(b_i(out) + b_i(in))c_{ij}]$$

Then, we need to formulate the cost of carrying traffic between brokers. We use a binary variable y_{ij} to indicate whether brokers i and j are connected via an overlay link, and z_{ijk} to indicate whether the link is chosen to forward flow k that needs bandwidth f_k. The cost of the overlay link between brokers i and j is denoted as c'_{ij}. The cost is thus obtained in the following equation:

$$C_2 = \sum_i \sum_j \sum_k (y_{ij} z_{ijk} f_k c'_{ij}) \qquad (1)$$

In the above equation, the *content conservation* must be held, i.e. the content of the incoming and outgoing traffic must be the same, which is different from the flow conservation of the traditional network flow model.

Algorithm 1 Two-stage greedy packing: The first stage

```
$\mathbb{B} := \{User\ IDs\}$
$\mathbb{U} := \{Broker\ IDs\}$
$conns := \emptyset$                                    //The set of connections
for $i \in \mathbb{U}$ do
    $min := +\infty$
    $min\_index := -1$
    for $j \in \mathbb{B}$ do
        if $load(j) + b_i(in) + b_i(out) \leq \mathbb{C}_j$ && $c_{ij} < min$ then
            $min := c_{ij}$
            $min\_index := j$
        end if
    end for
    $conns := conns \cup <i, min\_index>$
end for
RETURN $conns$
```

Note that the above equation is not applicable in content-based networking. As we discussed before, the content conservation in a content-based pub/sub network is different from the simple fluid model of traditional networks, where flows traversing the same links are summed up. In a pub/sub network, if two flows are originated from the same publisher, the amount of traffic when they pass trough a link is determined by the subscriptions of the destinations of those two flows.

Let us take a closer look at Eq. 1. The flow from brokers i to j is determined by the messages collected in i and the subscriptions registered in j. Note that the flow here is unidirectional. The sum of all the generating rate functions for all the users connected with i is $\sum_k x_{ki} G_k(\cdot)$, which is denoted as $\mathbb{G}_i$. The union of all the subscriptions registered on j is $\bigcup_k x_{kj} S_k$, which is denoted as $\mathbb{S}_j$. Since each flow is corresponding to an ordered pair $<i, j>$ for all brokers, we can get the flows that path through the overlay link $<i, j>$ as $\mathbb{F}_{ij} = \{k \mid x_{ij} \neq 0, z_{ijk} \neq 0\}$, where there is one-to-one correspondence between flow id k and two end-points of the flow $<i, j>$. Therefore, we can get the traffic that traverses the overlay link from broker i to j as follows:

$$\mathbb{T}_{ij} = \sum_{<i,j> \in \mathbb{F}_{ij}} \bigcup_{<i,j> \in \mathbb{F}_{ij}} \int_{\bigcup \mathbb{S}_j} \mathbb{G}_i(\cdot) d\cdot$$

We need to rewrite Eq. 1 as follows:

$$C_2 = \sum_i \sum_j \mathbb{F}_{ij} c'_{ij}$$

The final ILP formulation is formulated as follows:

$$\text{minimize } C_1 + C_2 \tag{2}$$

for any broker $j \in \mathbb{B}$:

$$\sum y_{ij} z_{ijk} = \sum y_{jh} z_{jhk} = 1 \tag{3}$$

$$x_{ij}, y_{ij}, z_{ijk} \in \{0, 1\} \tag{4}$$

$$\sum_i x_{ij}(b_i(in) + b_i(out)) \leq \mathbb{C}_j \tag{5}$$

In practice, the performance of each broker will inevitably degrade with the increasing workload, which is shown in Eq. 5. Therefore, we need to limit the volume of traffic passing through each broker. The traffic handled by a broker includes two parts: 1) messages sending from and to all the users associated with it; 2) messages transmitting between other brokers that pass through it. The second part is omitted in the following analysis. The reason is that we assume brokers can collect information of all the brokers in the overlay, so that each message will be assigned a path before sending out. Therefore, brokers do not need to perform the content matching, which is the dominant part of the message processing overhead. So it can be removed in Eq. 5. Additionally, the problem becomes more tractable to solve in this way.

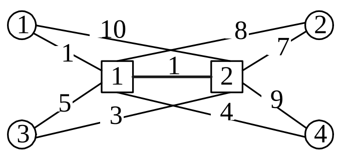
III. Two-stage approximation

We can relax the ILP in Section II to a linear programming (LP) problem, and get the result using rounding. In this paper, we do not use this approach for its complexity. Nonetheless, the optimization goal shown by Eq. 2 gives rise to an natural greedy approximation algorithm. As presented below, our greedy approximation is based on the idea that separating C_1 and C_2 in the optimization steps and find an approximation result for each of them, then combining the results to form a solution to the whole problem.

A. Two-stage greedy packing

We do not seek to find a bounded approximation to the problem formulated in Section II. A *two-stage greedy packing* algorithm is presented in this section. In the first stage, each user connects with the broker that minimizes the cost specified in Eq. 1. In the second stage, brokers are connected and traffic is routed using the shortest path algorithm. The combination of the results of the two stages is used as the approximation solution to the original problem. In the first stage, each user connects to the broker that has the lowest-cost overlay link. Users are packed together to each broker as long as the aggregated traffic does not exceeds the broker's capacity. The algorithm is shown in Algorithm 1.

We define *uniform subscriber group* (USG) as a set of users that have the same subscription. Given a publisher and a USG, we essentially need to find a minimum cost multicast tree to connect them, since each message does not traverse any link more than once. We use this property in the flow placement for brokers. For each broker, we compute the USGs for all other brokers on the entire content space. The content spaces of all USGs are non-overlapped. Since the link cost between brokers satisfies the triangular inequality, the minimum multicast tree connects the broker and a USG is a MST on their induced complete graph, which involves no nodes other than the broker and the USG. The proof is as follows: Assume that there is at least one such broker that is not the publisher and is not in the USG and is included in the multicast tree. By replacing the links between two brokers in the set by a directly connected link, we obtain a tree with less cost. This contradicts the assumption, which proves the claim.



	1	2	3	4	
1		10	4	3	17
2	3		6	6	15
3	9	2		1	12
4	4	9	5		18
	16	21	15	10	

Fig. 3. The overlay network used in the examples.

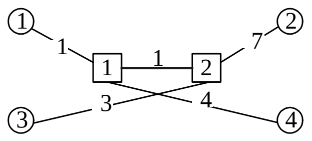


Fig. 4. The result of executing Algorithm 1.

An sample overlay network is given in Fig. 3. Circles represent users, and squares represent brokers. Their IDs are labeled. The edges between users and brokers are labeled with their costs. The corresponding traffic matrix is given in the table. The number in the ith row and jth column is the bandwidth demand from user i to j. The last column and row summaries the total outgoing traffic and incoming traffic, respectively, for each user.

Fig. 4 shows the obtained connections between users and brokers using Algorithm 1. Since the bandwidth demand is not used in the algorithm the traffic table is not shown. The resultant overlay is consisted of the minimum cost links. Assuming that flows between users have no overlapped interests, the overall cost of the whole overlay is 525.

B. Two stage clustering

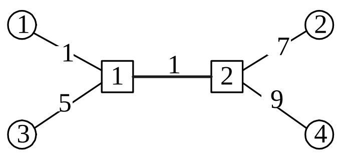
A better approximation scheme is based on clustering. The global traffic matrix for all users is known. We try to assign users that have a large amount of traffic to the same broker that has the minimum end-to-end cost. This is different from the greedy packing in which only the cost between brokers and users are considered. Here, the idea is to put as much traffic as possible into the two-hop path, which is provided by assigning those users to the same brokers.

The first stage of the clustering algorithm is to assign all users to brokers. The pseudo code is shown in Algorithm 2. We call the users connected to the same broker a cluster. In this stage, we try to collocate users on the same brokers to minimize the traffic inside clusters. The first step of this stage is to get an initial assignment for all the brokers. Each broker will be assigned a pair of users that have the maximum *bandwidth-to-cost ratio* (BCR). After then, each user is assigned to the broker that has the maximum BCR. The assigning process ends when all users are connected to brokers. The second stage is the same to the two-stage greedy packing algorithm.

Use the same example in Fig. 3. The resultant overlay is shown in Fig. 5. The overall cost of the overlay is 706. This is considerably larger than the two-stage greedy packing. The results on large scale networks are more insightful. And in the

Algorithm 2 Two stage clustering: The clustering stage

```
$\mathbb{U}$ := $\{UserIDs\}$
$\mathbb{B}$ := $\{BrokerIDs\}$
**for** $broker \in \mathbb{B}$ **do**
    $\{i, j\} \leftarrow argmax(\frac{b_{ij}}{c_{ij}})$
    $\mathbb{U} \leftarrow \mathbb{U} \backslash \{i, j\}$
    Assign $i$ and $j$ to $broker$
**end for**
**for all** $i \in \mathbb{U}$ **do**
    $B \leftarrow \mathbb{B}$
    **for** $broker \in \mathbb{B}$ **do**
        **if** $\mathbb{L}_{(broker \bigcup i)} > \mathbb{C}_{broker}$ **then**
            Remove $broker$ from $B$
        **end if**
    **end for**
    $min \leftarrow argmin_{b \in B}(\frac{\mathbb{B}_{ib}}{c_i})$
    Assign $i$ to broker $min$
**end for**
```



	1	2	3	4	
1		10	4	3	17
2	3		6	6	15
3	9	2		1	12
4	4	9	5		18
	16	21	15	10	

Fig. 5. The result of executing Algorithm 2.

example, the connections between brokers are too simple to represent the distinctions of two algorithms.

IV. Another formulation based on sub-channeling and its solution

The integer programming formulation is too contrived to be solved efficiently. Besides, the greedy approximation cannot provide sufficiently good results, as indicated in the simulation results in Section VI. Therefore, in this section, we consider a much clearer formulation and present the corresponding solutions.

A. Sub-channeling

Note that what makes the ILP complicated is the integration of the message generating function on specific content space. Here, we use histograms to approximate the message generating function. In the following discussions, we use one-dimensional content space for simple presentation, which apply directly in multi-dimensional cases.

In Fig. 6, a histogram is used to approximate an arbitrary function. The value at each interval is the average value

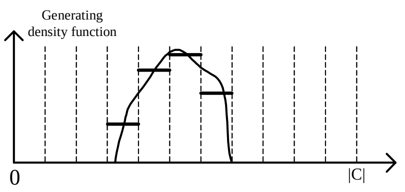


Fig. 6. The histogram used to approximate an arbitrary message generating function.

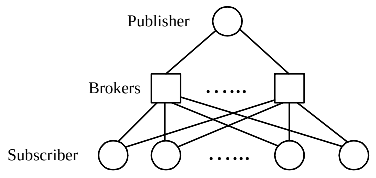


Fig. 7. For a single channel, the sub-network of a single publisher can be logically isolated from the rest of the network.

of the original function. A generating function now can be represented by a set of numbers $\{g_1, g_2, ..., g_T\}$, where $g_i \in \{0, 1\}$. The intervals are also used to approximate the subscriptions. A subscription is represented by a minimum number of intervals, and is then encoded as a set of numbers $\{s_1, s_2, ..., s_T\}$, where $s_i \in \{0, 1\}$. A number "1" indicates that a subscription includes this interval.

Now we can get the cost on each channel and then sum them together to obtain the overall cost. The reason of dividing content space into non-overlapping channels is that the traffic on different channels is non-overlapping so that their costs can be summed directly. The drawback is that false-positive will result in non-optimal results. Note that a false-positive will not happen in message forwarding since this sub-channeling is not used in message forwarding.

On each channel the overall cost is summed according to the connections between clients and brokers. For the cth channel, given a variable $x_{ij} \in \{0, 1\}$ that indicates the connection between client i and broker j (1 if connected, 0 if disconnected). Consider a single source s and channel c, the receipts of s's messages are determined by all the subscriptions of all users in the network. Messages are transmitted on links specified by x_{ij}.

Again, we first consider the cost for the traffic between clients and brokers, and do not take the traffic cost between brokers into consideration right now. The cost is given by $\sum_{b\in\mathbb{B}} x_{sb} g_{sc} + \sum_{b\in\mathbb{B};j\in\mathbb{U}} g_{sc} x_{bj} s_{jc}$. For all pairs of clients, say i and j, the cost in the access part is $\sum_{i\in\mathbb{U}} (\sum_{b\in\mathbb{B}} g_{ic} x_{ib} c_{ib} + \sum_{j\in\mathbb{U},j\neq i;b\in\mathbb{B}} g_{ic} s_{jc} x_{jb} c_{jb})$. We need to sum all costs on all channels to get the overall cost in the network. Let $cost^c$ denote the above equation, the objective function becomes:

$$\text{minimize} \quad cost = \sum_{c\in\mathbb{C}} cost^c \tag{6}$$

We then use minimum-cost spanning tree to connect brokers. We use $y_{ij} \in \{0, 1\}$ to indicate the connection between broker i and j. The cost of spanning tree is:

$$\sum_{i,j\in\mathbb{B}} c_{ij} y_{ij} \tag{7}$$

with the constraint that:

$$\begin{aligned} \text{subject to } & \sum_{i,j\in\mathbb{B}} \geq |\mathbb{B}| - 1 \\ & \sum_{i,j\in\mathbb{B}} \leq |B| - 1, \forall B \subseteq \mathbb{B} \end{aligned} \tag{8}$$

All other constraints are the same as Eq. 3, 4, 5. Now we have a linear objective function which has off-the-shelf tools to handle in practice. A primal-dual [10] approximation scheme can be applied directly. We aim to find intuitive heuristic algorithms that are simple to implement, so we do not seek the primal-dual solutions.

B. Solving the problem

The problem becomes an edge-weighted Steiner tree/star problem. However, since each user can connect to one and only one broker, when combining the solutions of the Steiner tree problem on each channel, assignments are likely conflicting. This issue also happens when combining trees obtained from different publishers. In other words, we obtain an approximated solution for each publisher on each channel, which are $|\mathbb{U}| \cdot T$ trees in total. The ultimate goal is to combine these trees to form a single overlay.

The following algorithm addresses this issue. We show in Section VI that this algorithm produces better results than the two-stage approximation schemes presented in Section III. The algorithm works in 3 phases:

- Phase 1. Find the optimal Steiner tree for each channel and each publisher. For an edge ij between user i and broker j, a weight is assigned on it, which is the traffic passing through it;
- Phase 2. For each user, the weight of the edges to all brokers is normalized. Arbitrarily chose a broker for the user using the normalized weight as the probability;
- Phase 3. According to the assigned connections, the MST between brokers is found;

In phase 1, we use MST approximation [17] to find the connection for each Steiner tree problem on each channel for each publisher. In the MST approximation algorithm shown in Algorithm 3, nodes (clients and brokers) are connected with shortest paths instead of edges. Note also that, after connecting a user to a broker, the user can no longer connect to other brokers in the successive steps. As a result, already connected users are substituted by their associated brokers in the following execution of the algorithm.

In phase 2, we rely on the spanning trees obtained in phase 1 to get the final results. Denote $\mathbb{B}_u$ the set of brokers associated with user u in all the spanning trees obtained in phase 1. Note the $|\mathbb{B}_u|$ is at most $\mathbb{T}$, which is the number of sub-channels. We randomly assign each user to one broker of the $\mathbb{B}_u$ using the normalized bandwidth as the probability.

In phase 3, we connect brokers and place flows based on the connections between clients and brokers obtained in phase 2. It is clear that the flows must be placed on a spanning tree of all the brokers. Since any message does not need to traverse any link more than once, the minimum cost flow placement

Algorithm 3 MST approximation for the Steiner tree/star problem

```
$\mathbb{U}$ := {*User IDs on sub-channel $i$*}
$\mathbb{T} := \emptyset$                    //*The set of edges of the spanning tree*
$dist[s] := 0$                 //*$s$ is the initial source*
**for** each user $u$ in $\mathbb{U}$ other than $s$ **do**
    $color[u]$ := WHITE
    $dist[u] := +\infty$
    $parent[u] := \emptyset$
    $path[u] := \emptyset$
**end for**
$ENQUEUE(PQ, s)$
**while** $PQ \neq \emptyset$ **do**
    $u := DEQUEUE(PQ)$
    **if** $parent[u] \neq NONE$ **then**
        **for** each $v \in \mathbb{U}$ **do**
            **if** $dist(parent[u], v) < dist[v]$ **then**
                $dist[v]$ := $dist(parent[u], v)$
                **if** $color[v]$ == WHITE **then**
                    $ENQUEUE(PQ, v)$
                    $color[v]$ := GRAY
                    $parent[v] := prev[v]$
                    $path[v] := path(u', v)$
                **else if** $color[v]$ == GRAY **then**
                    $UPDATE(PQ, v)$
                **end if**
            **end if**
        **end for**
    **else**
        $v_{min} := argmin_{v \in \mathbb{U}}(dist(u', v)$
        $parent[u] = next[u]$
        $parent[v_{min}] := prev[v_{min}]$
        $path[v_{min}] := path(u, v_{min})$
    **end if**
    color[u] = BLACK
    $INSERT(\mathbb{T}, path[u])$
**end while**
Return $\mathbb{T}$
```

must be on a tree. This tree connects all the brokers that are linked to the publisher and subscribers. Since traffic of different publishers cannot be mixed together, the spanning tree of different publishers can be calculated independently and obtain the optimal flow placement between brokers. Similarly, traffic of different channels can also be optimally routed on multiple spanning trees.

Another reminder is that we have not consider the case where messages generated by clients are not interested in by any users. In Eq. 6 and 2, all messages are treated as if they interested by some users. However, we can remove the part of non-interested messages before applying the optimization, and the results are the same.

C. Distributed implementation

The above centralized algorithm can be translated into a distributed implantation as follows: Each broker, knowing the subscriptions of all other brokers' associated clients, independently redirect its associated clients to another broker that can reduce its cost of routing messages to all other brokers. In this way, a broker is treated as a "bigger" user whose interests are the union of all the subscriptions of its associated clients.

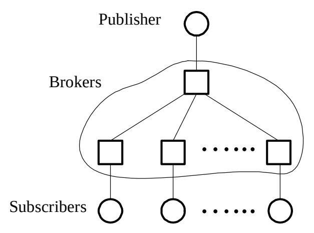


Fig. 8. Using minimum spanning tree to approximate the Steiner tree/star problem.

Each broker is able to find the best connection for a user with the following information:

- The user's subscription;
- The user's link cost to all brokers in the network;
- The cost of all the links between all brokers;

These information can be collected efficiently. The distributed algorithm works by periodically redirect the users that currently has the most traffic to another broker, if necessary. By enumerating the link from the user to all brokers, we can find the best connection for that user. The user is then redirected to that broker, using the APIs presented in Section V.

V. System design and APIs

Each user knows a subset of brokers that he/she can connect to. The list of brokers that is known to each user can be distributed by a central web server, like the download servers that are provided by the download website and can be selected by the user. The client side software is then responsible for interacting with the connected broker.

First, subscriptions are forwarded to the broker using the *register()* API. The generating function is obtained by summarizing the past message publishing history, which can be performed at client side or broker. For the sake of reducing client side burden and unnecessary message exchange, we let the broker perform this function using *summarize()* API.

Second, brokers run the user migration algorithm. A broker send request to the users that are selected to migrate to another broker using the *request_to_move()* API. The user can accept or decline the migration request, depending on the local policy employed by the user. For example, some user may stick to a broker that offers additional services or price reduction not shown in the protocol. If the client accept the migration request, if runs the *accept()* API. The acceptance notification is sent to its connected broker. The broker first replicate its data to the broker that the client is about to migrate to. This is done using the *replicate()* API.

The broker that the user connected to before is called *home broker*, whereas the broker it migrates to is called *foreign broker*, which follows the convention of Mobile IP [15]. The client then obtains the foreign broker's address from the home broker. It sets up the connection using *connect()* API. Since the foreign broker needs additional time to propagate the client's subscriptions, the home broker needs to forward the message on behalf of the client in the time TTL. This is done by the *redirect()* API.

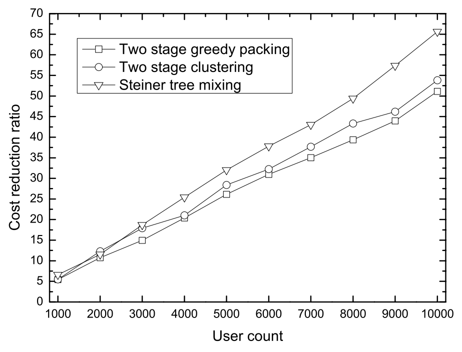


Fig. 9. Cost reduction ratio of three algorithms on networks with different scales.

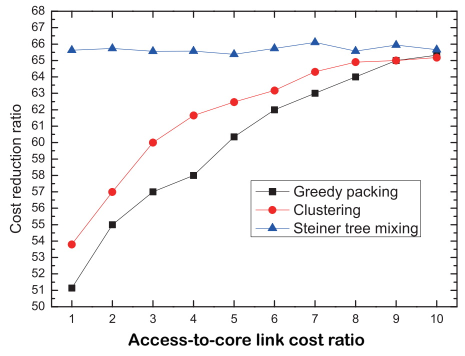


Fig. 10. Cost reduction ratio of networks that have different core-to-access link cost ratios.

VI. Performance evaluation

We use simulations to evaluate the performance of our algorithms. The results are analyzed in this section.

A. Simulation settings

We create a network comprised of 1000 to 10000 users. The number of brokers in the network is 1/10 to 1/100 of the users. We use one-dimensional content space for simplicity. Each user is assigned a quadratic generating function. We randomly generate quadratic functions and set their negative part as 0 to obtain the generating function. Subscriptions of each user is drawn randomly from the content space. All brokers' capacities are the same. It is set in such a way that the summed capacity of all brokers is twice of all users' traffic demands.

For a network of a given size, we generate 100 network samples and repeat the algorithm on each sample and get the average overall network cost. Currently, no similar work has been done. We compare with the random assignment where all brokers and users are connected randomly.

B. Cost reduction ratio

A *random assignment* method randomly pairs each user with a broker without considering any metrics. The resultant network of random assignment represents the worst case cost of any algorithm. For each assignment algorithm $\mathbb{A}$, we define its *cost reduction ratio* (CRR) as the ratio of the cost obtained from random assignment to its result on the same network. For each network, we can compute a CRR. The average CRR on numerous network instances indicates the algorithm's performance. Note that random assignment produces costs that grow linearly with the number of clients in the overlay.

We first look at the CRR of three proposed algorithms in Fig. 9. Each data point is the averaged value of 100 randomly generated network graph. The ratio between the numbers of users and brokers is 100. The average cost between brokers is the same as the average cost of the links between users and brokers. Since *greedy packing* only considers access cost in the first stage, it inevitably will render suboptimal connection in the second stage. But in *clustering*, a part of the inter-user traffic is considered in terms of clustering closely bound users on the same broker which has a lower cost. Whereas in *steiner-tree mixing* algorithm, the traffic on each channel is directly considered in the sterner tree connection phase, which aims to provide optimal connection.

Random	Greedy packing	Clustering	Mixing
0.9	91.445	100.235	179.654

TABLE I
The running time of each algorithm in second.

There is a subtle factor that affects the performance of each algorithm, that is, the difference between the cost of access and core links. Put it in an intuitive way, if the core links are so cheap that they almost can be treated as free, we can safely ignore them in calculating the optimal connections. Therefore, the performance difference between three algorithms becomes smaller when core links' cost is diminishing. We conduct another set of experiments to verify this assumption. As shown in Fig. 10, we vary the ratio of the average cost of access links to that of core links, which is denoted *access-to-core link cost ratio*. The ratio of the number of users to the brokers is 100, as in Fig. 9, and there are 10,000 users. We generate 100 instances for a given network scale and obtain the average results. Steiner-tree mixing explicitly optimizes the cost of routing messages through the core links. Its CRR curve almost does not change. The other two algorithms' changes conform with our analysis above, which gradually converge to Steiner-tree mixing. This indicates that if the cost of core links is sufficiently low, it can be safely ignored in the optimization without sacrificing the overall performance.

C. Computational complexity

We evaluate the running time of three algorithms in this section. We implement the algorithms in *C++*. The compiler used is *MingW G++ 3.4.5*. We use *-O0* flag to disable optimization. The input network has 10000 users and 100 brokers and a access-to-core link cost ratio of 1. The machine we use has an Intel® Q8600 quad-core processor with $4GB$ memory. We do not use parallelism in the code. The results are shown in Table I. The difference between three algorithms is dramatic, but is acceptable considering their performance. An optimized implementation can make the algorithm practical.

VII. Related work

For a general survey on the pub/sub system, please refer to [8]. Generally, pub/sub systems can be classified into two categories: content-based [4], where contents and subscriptions are specified by a content description language; and topic-/channel-based [5], where the content space is divided into multiple predefined channels. Minimum edge topic-/channel-based pub/sub overlay construction problem has been studied in [7]. The authors designed a greedy approximation algorithm that has a logarithmic approximation ratio. Later in [6], the authors proposed algorithms with similar approximation bounds but with much less complexity and running time. Due to the sophisticated content representation scheme of CBPS, these solutions are not applicable in our problem.

The problem studied in this paper is closely related to many hard problems in combinatorial optimization problem. The first stage of our problem is similar to the facility location problem [12]. The difference is that there is no broker opening cost in our problem, which makes it quite simple. The second stage of our problem generalizes the multi-commodity flow problem, where there is a cost reduction when forwarding multiple flows on the same links. Note that this is different from the non-linear multi-commodity flow [11]. A non-linear cost function is parameterized on the flow size where our cost function is parameterized on the combinatorial structure of flows. The complication of our problem lies in the complicated combinatorial nature of content space matching.

The Steiner tree/star problem [13] has the same network structure as ours, but has a different optimization goal. We have formulated a steiner-tree based combinatorial approximation problem based on sub-channeling. Steiner tree algorithms are then used as a component of our algorithm. Finding MST is a 2-approximation algorithm of the steiner-tree problem [17]. There are $(2-2/k)$-approximation algorithms for the Steiner tree problem [9]. The 3-phase scheme is related to tabu-search [18]. An important approximation scheme for combinatorial optimization problem is primal-dual method [10]. Although we presented an integer programming formulation of our problem, it is not used due to its high complexity.

VIII. Conclusion

In this paper, we formulated an optimization problem of constructing pub/sub overlay of the minimum cost of using underlying network resources. We present an integer programming formalism, which provide insightful knowledge about the problem structure. We have designed two simple and straightforward greedy approximation algorithms. In order to achieve better results, we proposed the idea of sub-channeling to facilitate a more efficient approximation algorithm called Steiner-tree-mixing. Simulation results verify the performance of our proposed algorithms. With the increasing adoption of the pub/sub system in overlay networks, the problem we studied in this paper will become more and more important. Our work sheds light on the complexity and approximation algorithms of the problem. Our future work will follows two directions: theory and system. We are going to study the approximation bound of the problem and implement a prototype system.

Acknowledgments

Thanks for the insightful comments from Dr. Abdallah Khreishah. This research was supported in part by NSF grants CCF 1028167, CNS 0948184 and CCF 0830289.

References

[1] "TIBCO Rendezvous." [Online]. Available: http://www.tibco.com/
[2] "Triangular inequlaity." [Online]. Available: http://en.wikipedia.org/wiki/Triangle_inequality
[3] "Pub/sub at google," *lecture at CANOE Summer School*, 2009.
[4] A. Carzaniga and A. L. Wolf, "Content-based networking: A new communication infrastructure," in *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, ser. Lecture Notes in Computer Science, no. 2538. Scottsdale, Arizona: Springer-Verlag, Oct. 2001, pp. 59–68.
[5] M. Castro, P. Druschel, A. M. Kermarrec, and A. I. T. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1489–1499, 2002. [Online]. Available: http://dx.doi.org/10.1109/JSAC.2002.803069
[6] C. Chen, H.-A. Jacobsen, and R. Vitenberg, "Divide and conquer algorithms for publish/subscribe overlay design," *Distributed Computing Systems, International Conference on*, vol. 0, pp. 622–633, 2010.
[7] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Constructing scalable overlays for pub-sub with many topics," in *PODC '07: Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*. ACM, 2007, pp. 109–118.
[8] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, 2003.
[9] P. W. F.K. Hwang, D.S. Richards, *The Steiner Tree Problem*. Elsevier, 1992.
[10] M. X. Goemans and D. P. Williamson, "The primal-dual method for approximation algorithms and its application to network design problems," pp. 144–191, 1997.
[11] J.-L. Goffin, J. Gondzio, R. Sarkissian, and J.-P. Vial, "Solving nonlinear multicommodity flow problems by the analytic center cutting plane method," *Math. Program.*, vol. 76, no. 1, pp. 131–154, 1997.
[12] K. Jain and V. V. Vazirani, "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation," *J. ACM*, vol. 48, no. 2, pp. 274–296, 2001.
[13] S. Khuller and A. Zhu, "The general steiner tree-star problem," *Inf. Process. Lett.*, vol. 84, no. 4, pp. 215–220, 2002.
[14] 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: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. Berkeley, CA, USA: USENIX Association, 2005, pp. 3–3.
[15] C. Perkins, "Ip mobility support for ipv4," no. 3344, 2002, rFC 3344 (Proposed Standard), interhash = e3c88fbb97920e9b73fbc4797054e502, intrahash = c36b8210f2ab2de46d7f0e1cad0e489e Updated by RFC 4721.
[16] J. Saltzer, D. Reed, and D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984.
[17] V. V. Vazirani, *Approximation Algorithms*. Springer, July 2001. [Online]. Available: http://www.worldcat.org/isbn/3540653678
[18] J. Xu, S. Y. Chiu, and F. Glover, "Using tabu search to solve the steiner tree-star problem in telecommunications network design," *Telecommunication Systems*, vol. 6, pp. 117–125.