RC24935 (W1001-071) January 25, 2010
Computer Science

IBM Research Report

Constructing Minimal Overlay to Support Policy-Based Access Control Groups

Bong Jun Ko, Starsky H. Y. Wong, Kang-Won Lee
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA

Chi-Kin Chau
Computer Laboratory
University of Cambridge
Cambridge, UK

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Constructing Minimal Overlay to Support Policy-Based Access Control Groups

Bong Jun Ko, Starsky H. Y. Wong, Kang-Won Lee
IBM T. J. Watson Research Center
Hawthorne, NY, USA
Email: {bongjun_ko,hwong,kangwon}@us.ibm.com

Chi-Kin Chau
Computer Laboratory, University of Cambridge
Cambridge, UK
Email: Chi-Kin.Chau@cl.cam.ac.uk

***Abstract*—Overlay networks have been studied extensively in recent years as a flexible means to improving the reliability, resiliency, and performance of many networking applications. In this paper we present a novel use of overlay networks and distributed mechanisms to construct them for handling information assurance issues in networking systems. The problem is explored in the context of constructing an overlay that satisfies a given access control policies in decentralized information sharing systems. We formulate a new graph-theoretic optimization problem of constructing a minimum *policy-compatible* graph, which we show is NP-complete. We provide efficient centralized and fully-distributed heuristics, and prove the convergence property of the distributed proces. Our simulation study with synthetic and empirical data set shows that our methods result in the performance (in terms of total number of links) very close to the optimal case (within 3%) for small input, and that they can reduce the number by up to 30% compared to a method based on minimum spanning tree algorithm for larger data set.**

I. Introduction

Overlay networks have been studied extensively in recent years for their practical importance in many applications. Various types of overlay networks and mechanisms to construct them have been proposed to achieve various objectives: reliable and failure-resistant communications among nodes [5], [12], performance optimizations for bandwidth [13] or data transfer delay [4], [11], efficient P2P search [10], [14], reliable content delivery [1], and multicast/group communication [6], etc. However, one of the important considerations for building an overlay has been missing in the literature, namely *policies*.

In the networking community, cyber security and information assurance issues (i.e., making sure that information access is granted to the users that have credential) are becoming increasingly more critical. Policies are quite often employed as a tool to address these issues, and also useful addressing inter-operability issues in information federation (e.g.., compatibility in data format, protocol, and even data availability in time). Broadly speaking, policies are the rules that specify high level intent of a person or an organization, and are often described in the form of constraints [2]. In this paper, we consider policies that define constraints as to which nodes can *access* which types of sensitive data. Such policies will be useful in distributing and controlling sensitive information, such as classified and secret information in the military and intelligence community, sensitive business information (e.g., HR data such as salary information), and private information of an organization (e.g., tentative budget numbers) or of an individual (e.g., health record of a patient).

Of particular interest to this research is controlling the information access in distributed publisher-subscriber (pub-sub) networking environments, *where the information in the network is accessed and shared by multiple networking nodes in a distributed manner*. In such systems, since a large number of information flows (from the publishers to subscribers) can be established dynamically, controlling the information access by conventional methods (e.g., access control list based on node ID) does not scale to the system size. Instead, the policies can be effectively used to limit the information access privilege based on higher-level criteria like the attributes of the nodes. For example, in a sensor network formed by a large number of various types of sensor nodes and data collecting nodes from multiple organizations, policies can be used to limit the access based on the nodes' organizational affiliations, security clearance levels, sensitivity level of the data, etc.

We note that the access control policies in such information sharing systems in particular specify multiple node groups such that all nodes in each group are given access privileges to some common pieces of information that can be shared by the group members. We call such node groups specified by the policies as "access-control groups". Note also that each node can belong to many access-control groups since multiple policies are specified based on different criteria (e.g., based on different attributes of nodes).

Enforcing the access control can then be achieved by having each access-control group perform a secure group communication for distributing the information permitted for the group members. While there are many mechanisms to enable secure group communications proposed in the literature (see, for example, a survey article [9] on key management issues in group communication), they mostly concern how to securely distribute the secret keys to the group members. This

Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

research is focused on an orthogonal issue: *what are the pairs of nodes between which a secure connection is established to satisfy the policy constraints?*

We explore the problem of enforcing the access control policies in the context of building an overlay network, in which the links represent the pairs of nodes with secure pairwise connections and are used by the access-control groups for their secure group communications. Note that such an overlay is required to satisfy the policy constraints: nodes in each group should be able to form a connected network between themselves using only the links in the overlay. While more overlay links would mean that the overlay network is to satisfy the policy constraints. it also results in higher overhead associated with establishing secure connections (e.g., key distribution and management). It is therfore of practical importance to build a minimum overlay network that satisfies given set of access control policies.

The challenge in building such a minimum overlay is that the systems considered here are of potentially very large, in terms of not only the number of nodes, but also that of access-control groups resulted from the policies; since each attribute of a node and information source can be used to specify policies and there can be numerous attributes per object (e.g., gender, race, occupation, etc. for a patient object), there could be potentially a huge number of groups.[1]

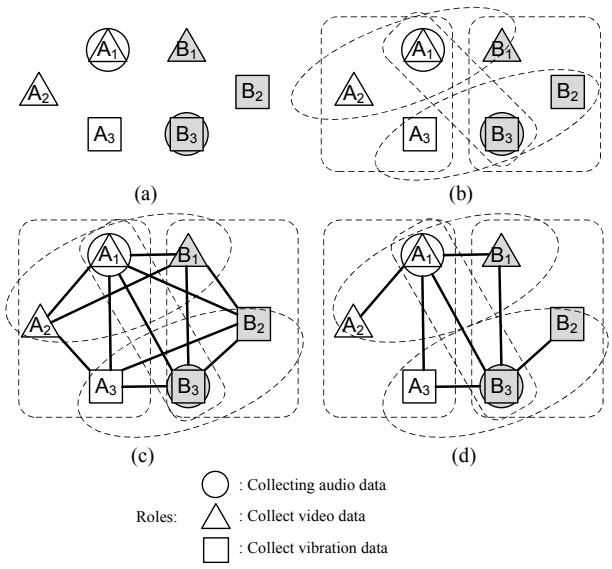

Fig. 1. Example sensor data sharing scenario. (a) Three nodes (A_1, A_2, A_3) from organization A, and three (B_1, B_2, B_3) from organization B are deployed. Nodes' shaped indicate their roles as a data collecting node. (b) Groupings of nodes according to the example policy constraints. Nodes within each group have access privilege to the sensor data of the type specified in corresponding policy. (c) All pairs of nodes in each node group are connected each other. (d) A minimum overlay configuration that satisfies the policy constraints.

A. Motivating example

For a motivating example, consider a scenario in Fig. 1(a), where nodes from two organizations, A and B, are deployed in a sensor network so that each node takes the role(s) of collecting data from one or more types of sensors, namely, audio, video, and vibration. In the figure, the nodes are depicted as ◯, △, □, or the their combinations to show their respective roles as a data collection node (sensors are not shown in the figure for brevity as they are not relevant to the discussion). Suppose the following access control policies are specified to permit a node n's access to the data from a sensor s based on the organizations and the roles of the nodes:

- If organization(n) == organization(s), then permit n's access to data from s.
- If roles(n) includes dataType(s), permit n's access to data from s..

First of all, from the above policies, five node groups are readily identifiable according to their roles and organization attributes (see Fig. 1(b)). Specifically, according to the first policy, two node groups $\{A_1, A_2, B_3\}$ and $\{B_1, B_2, B_3\}$ can be formed for those having access privileges to data from sensors in organization A and B respectively. Similarly, from the second policy, three groups can be identified based on their respective roles as the collecting nodes for audio, video, and vibration data, respectively (In Section II-A, we will discuss in details how these node groups can be derived from given set of policies).

One way of building secure connections is to connect pairs of nodes as long as they are permitted to share some information, and the resulting graph is shown in Fig. 1(c). This would clearly satisfy the policy constraints, but can lead to an inefficient solution in terms of the number of secure connections to be made. One can in fact reduce the number of links further and still satisfy the policy constraints as the one shown in Fig. 1(d). Note that the graph in Fig. 1(d) is *minimal* in the sense that removing any link from it results in a graph that violates the policy constraints: for instance, if the link between nodes A_1 and B_1 was removed, the node group $\{A_1, A_2, B_1\}$ is not connected on its own, hence causing B_1 to be unable to share the video data (△) data with the other two nodes, which violates the second policy above (In fact it is an optimal one in the sense that it has as small number of links as graph satisfying the policy constraints).

B. Our contribution

Our objective is to build a minimum overlay network that is *compatible* with given set of access control policies. The followings summarize our contribution toward this goal:

- We first formulate the overlay construction problem as a graph-theoretic optimization for minimum number of links, for which we formally define the *policy-compatibility* of a graph and prove the NP-completeness. The input to our problem instance is a collection of node groups, and as such we also discuss this pre-processing

[1]In the current IT systems that use policies, the number of policies could be in the range of O(10) - O(1000). If we assume the similar scale for the emerging distributed pub-sub applications, the number of groups that may need to be maintained could be also large because a single policy may result in multiple groups.

step of deriving these groups from given policies (Section II).
- We develop both the centralized and distributed heuristic algorithms for constructing policy-compatible graph. The centralized algorithm in Section III is presented as a baseline mechanism for our fully-distributed one (Sectionsec:distributed) that is suitable in large-scale environments. Several properties of our algorithms such as correctness, convergence, etc., are also provided.
- We extend our algorithms to support a secondary optimization goal of minimizing the total link cost as well. A mechanism based on a single, tunable parameter that can control the behavior of the algorithms to gracefully transition between the two objectives (Section V) is provided.
- Via simulation study on synthetic and empirical data set, we show that the proposed algorithm can achieve results that are very close (within 3%) to optimal for a small input, and can reduce the number of links by up to 30% compared to the solution based on Minimum-Spanning-Tree algorithm for larger input data sets (Section VI).

II. Problem Statement

In this section, we formally state our overlay construction problem. The input to our problem instance is a collection of node subsets, each of which defines a group of nodes that can share a certain set of information contents according to access control policy. The output is a graph that satisfy the connectivity constraints in all node groups (and hence satisfy the access control policies). We begin by describing how node subsets can be derived from a given set of access control policies.

A. Access control policies and node groups

An access control policy (or *policy* in short hereafter) specifies which nodes are permitted to access what contents in the network. Here, we use the term "content" to mean a piece of information in any form (e.g., file, URL, feed, etc.) accessed by a set of user nodes.

We assume the policies are specified in the following canonical format:

> "Permit node n's access to a content c if $f(n)$ = true AND $g(c)$ = true",

where f and g are boolean functions that determine the conditions for node n and content c, respectively, under which n should be given the permission to access c. For instance, the example policies in Section I-A can be easily transformed into this form by enumerating all values of the attributes. We assume the access is denied by default if none of the statements allows it.

This form of policy statements (i.e., "permit-only-when-specified") is widely employed in many access control mechanisms such as file systems, firewall, network access control in routers and servers, etc.. Also, decoupling the conditions for access permission into the node parts and content parts (i.e., f and g, resp.) is appropriate in the pub-sub type of networking environments due to the inherent separation of information sources and consumers under such networking paradigm.

Suppose now we are given K policy statements, each of which specifies the condition of permitting node n's access to content c by two boolean functions $f_k(n)$ and $g_k(c)$ ($k = 1, \cdots, K$).

Then, given a set of nodes N, let us denote by N_k a subset of nodes that satisfy the node-condition f_k of k-th policy, N_k = $\{n \in N : f_k(n)$ = true $\}$ ($k = 1, \cdots, K$). It is then straightforward to see all nodes in N_k commonly have access permission to any content c that satisfies the condition $g_k(c)$ of k-th policy. In other words, N_k ($k = 1, \cdots, K$) is a group of nodes among which contents satisfying $g_k(c)$ can be shared.

We notice that it is possible to further reduce the number of node groups by, for instance, identifying identical conditions across policies and merging those policies, or after analyzing the policies to check their consistency and coverage [3]. Optimizing the number of groups in this pre-processing steps, however, is beyond the scope of this paper.

B. Problem definition

Suppose we have a group of nodes $N = \{1, 2, \cdots, |N|\}$, and a collection of K subsets (or "*node groups*") of N, $\Omega = \{N_1, ..., N_K \subseteq N\}$ derived from a set of access control policies as described in Section II-A. We assume that, for each group N_k, all nodes in N_k are "trusted" by one another, so that the node exercises properly the access control polcies and will not pass the information shared in N_k to other nodes not in N_k. We also assume the same policies are applied to all nodes.

Our goal is to construct an "efficient" overlay network of the nodes N such that the overlay is "compatible" with the information access control policies imposed by Ω, where the compatibility of an overlay network w.r.t Ω can be defined in a graph-theoretic terms as follows.

Definition 1: (*Policy-compatible Graph*) Given a set of nodes N, and a collection of subsets, $\Omega = \{N_1, ..., N_K \subseteq N\}$. An undirected graph $G = (N, E)$ is called *compatible* w.r.t. Ω, if for each $N_k \in \Omega$, the induced subgraph $G_k = (N_k, E_k \subseteq E)$, where $E_k = \{(u, v) | u \in G_k, v \in G_k, (u, v) \in E\}$, is a connected graph.

In a policy-compatible graph, each subgraph of node groups is a connected one on its own. This means that each group of nodes N_k can securely share the contents specified by k-th policy on a connected network G_k of secure links, where the connectivity of G_k ensures that any other nodes outside group N_k need not be part of the information distribution within N_k. In other words, a policy-compatible graph *realizes* the given access control policies.

Note that there always exists some compatible graph w.r.t. any Ω, with the complete graph being one such graph. As it is typically expensive to set up and maintain secure links in the overlay networks, however, it is of practical importance to build a graph compatible w.r.t. given Ω with as small a number of edges as possible (thus "efficiency" of the overlay

network). This is formally defined as an optimization problem on graph in the following definition.

Definition 2: (*Policy-compatible Overlay (PoCO) Optimization Problem*) Given a set of nodes N, and a collection of subsets, $\Omega = \{N_1, ..., N_K \subseteq N\}$, find a compatible $G = (N, E)$ w.r.t. Ω such that $|E|$ is the smallest among all compatible graphs.

As the PoCO problem is NP-complete as formally shown in the following sub-section, the optimization problem is unlikely to be solvable in polynomial time, and we seek for efficient heuristic mechanisms in subsequent sections.

C. Complexity of PoCO problem

We prove the NP-completeness of the Policy-compatible Overlay (PoCO) optimization problem. The decision version of the problem is as follows.

Definition 3: (*Policy-compatible Overlay (PoCO) Decision Problem*) Given a set of nodes N, and a collection of subsets, $\Omega = \{N_1, ..., N_K \subseteq N\}$. A graph $G = (N, E)$ is called *compatible* w.r.t. Ω, if for each $N_k \in \Omega$, the induced subgraph $G_k = (N_k, E_k \subseteq E)$ is a connected graph. Decide if there exists a compatible $G = (N, E)$, such that $|E| \leq \Delta$.

Theorem 1: Policy-compatible overlay decision problem PoCO (N, Ω, Δ) is NP-complete.

Proof: It is easy to show that PoCO is in NP. We examine each induced subgraph G_k of G to see if it is connected or not. Checking the connectivity in a graph is polynomial in time.

To show PoCO is NP-hard, we rely on a polynomial time reduction from 3SAT problem.

Definition 4: (*3SAT Problem*) Consider a 3-CNF formula F consisting m clauses and h variables, i.e. $F = c_1 \wedge c_2 \wedge \cdots \wedge c_m$, where each $c_i = y_{j_1} \vee y_{j_2} \vee y_{j_3}$ and $y_{j_1}, y_{j_2}, y_{j_3} \in \{x_1, \bar{x}_1, ..., x_h, \bar{x}_h\}$. F is said to be satisfiable, if there exists a truth assignment to F, such that every clause has at least one true literal. 3SAT is well-known to be NP-complete.

Given a 3-CNF formula F, we assume each clause does not contain a literal and its complement (as this is trivially satisfiable). We construct a corresponding PoCO (N, Ω, Δ), such that F is satisfiable, if and only if (N, Ω, Δ) is satisfiable.

First, set $N = \varnothing$. For each $x_j, \bar{x}_j$, we add three nodes $\mathsf{a}_j, \mathsf{b}_j, \mathsf{c}_j \in N$, and create two subsets in Ω: $N_{x_j} = \{\mathsf{a}_j, \mathsf{b}_j, \mathsf{c}_j\}$ and $N_{x_j}^{\mathsf{a}_j, \mathsf{b}_j} = \{\mathsf{a}_j, \mathsf{b}_j\}$.

Then, for each $c_i = y_{j_1} \vee y_{j_2} \vee y_{j_3}$, we add three nodes $\mathsf{o}_i, \mathsf{p}_i, \mathsf{q}_i \in N$, and create eight subsets in Ω:

$$N_{c_i}, N_{c_i}^{\mathsf{p}_i, x_{j_1}}, N_{c_i}^{\mathsf{p}_i x_{j_3}}, N_{c_i}^{\mathsf{q}_i, x_{j_2}}, N_{c_i}^{\mathsf{q}_i x_{j_3}}, N_{c_i}^{\mathsf{o}_i x_{j_1}}, N_{c_i}^{\mathsf{o}_i x_{j_2}}, N_{c_i}^{\mathsf{o}_i x_{j_3}}$$

We next describe the construction of these subsets:

1) First, we set $N_{c_i} = \varnothing$ and add $\mathsf{o}_i, \mathsf{p}_i, \mathsf{q}_i \in N_{c_i}$.
2) Next, if $y_i^{j_1} = x_{j_1}$, then
   i) Add $\mathsf{a}_{j_1}, \mathsf{c}_{j_1} \in N_{c_i}$,
   ii) Create $N_{c_i}^{\mathsf{p}_i, x_{j_1}} = \{\mathsf{a}_{j_1}, \mathsf{p}_i\}, N_{c_i}^{\mathsf{o}_i, x_{j_1}} = \{\mathsf{c}_{j_1}, \mathsf{o}_i\}$.

   elseif $y_i^{j_1} = \bar{x}_{j_1}$, then
   i) Add $\mathsf{b}_{j_1}, \mathsf{c}_{j_1} \in N_{c_i}$,
   ii) Create $N_{c_i}^{\mathsf{p}_i, x_{j_1}} = \{\mathsf{b}_{j_1}, \mathsf{p}_i\}, N_{c_i}^{\mathsf{o}_i, x_{j_1}} = \{\mathsf{c}_{j_1}, \mathsf{o}_i\}$.
3) Repeat for y_{j_2}, but we use q_i instead of p_i.
4) Repeat for y_{j_3}, but we use both q_i and p_i.

It is easy to see that the construction of (N, Ω) is polynomial in time. See Fig. 2 for an illustration of the construction of PoCO (N, Ω, Δ) for a given F.

There are some remarks. If a subset $N_k \in \Omega$ has two nodes only (i.e. $N_k = \{\mathsf{a}, \mathsf{b}\}$), then $(\mathsf{a}, \mathsf{b}) \in E$ for any compatible $G = (N, E)$. Thus, $N_{x_j}^{\mathsf{a}_j, \mathsf{b}_j}$, $N_{c_i}^{\mathsf{p}_i, x_{j_1}}$, $N_{c_i}^{\mathsf{p}_i x_{j_3}}$, $N_{c_i}^{\mathsf{q}_i, x_{j_2}}$, $N_{c_i}^{\mathsf{q}_i x_{j_3}}$, $N_{c_i}^{\mathsf{o}_i x_{j_1}}$, $N_{c_i}^{\mathsf{o}_i x_{j_2}}$, $N_{c_i}^{\mathsf{o}_i x_{j_3}}$ are two-node subsets, which must be included as edges in any compatible G.

Finally, we set $\Delta = 2h + 7m$.

(If Part): We show that if F is satisfiable, then (N, Ω, Δ) is satisfiable by constructing compatible $G = (N, E)$. First, we set $E = \varnothing$, and we include the two-node subsets $N_{x_j}^{\mathsf{a}_j, \mathsf{b}_j}$, $N_{c_i}^{\mathsf{p}_i, x_{j_1}}$, $N_{c_i}^{\mathsf{p}_i x_{j_3}}$, $N_{c_i}^{\mathsf{q}_i, x_{j_2}}$, $N_{c_i}^{\mathsf{q}_i x_{j_3}}$, $N_{c_i}^{\mathsf{o}_i x_{j_1}}$, $N_{c_i}^{\mathsf{o}_i x_{j_2}}$, $N_{c_i}^{\mathsf{o}_i x_{j_3}}$ as edges in E, for all variable j and clause i. Thus, the induced subgraphs of these subsets are connected.

Then, for each variable x_j, either one of x_j or $\bar{x}_j$ is true. If x_j is true, then we add $(\mathsf{a}_j, \mathsf{c}_j) \in E$. Otherwise, if $\bar{x}_j$ is true, then we add $(\mathsf{b}_j, \mathsf{c}_j) \in E$. Thus, the induced subgraph of N_{x_j} is connected.

Next, for each clause c_i, one literal must be true. Hence, the corresponding $(\mathsf{x}_j, \mathsf{c}_j)$, where $\mathsf{x}_j \in \{\mathsf{a}_j, \mathsf{b}_j\}$, has been already included in E. Thus, the induced subgraph of N_{c_i} is connected.

Therefore, $G = (V, E)$ is compatible. Moreover, $|E| = 2h + 7m = \Delta$, and (N, Ω, Δ) is satisfiable.

(Only-if Part): We show that if (N, Ω, Δ) is satisfiable, then F is satisfiable. Suppose $G = (V, E)$ is compatible and $|E| = \Delta = 2h + 7m$. Since the induced subgraphs of two-node subsets $N_{x_j}^{\mathsf{a}_j, \mathsf{b}_j}$, $N_{c_i}^{\mathsf{p}_i, x_{j_1}}$, $N_{c_i}^{\mathsf{p}_i x_{j_3}}$, $N_{c_i}^{\mathsf{q}_i, x_{j_2}}$, $N_{c_i}^{\mathsf{q}_i x_{j_3}}$, $N_{c_i}^{\mathsf{o}_i x_{j_1}}$, $N_{c_i}^{\mathsf{o}_i x_{j_2}}$, $N_{c_i}^{\mathsf{o}_i x_{j_3}}$ must be connected in G and their edges are distinct, they take up $h + 7m$ edges from E. Hence, there remain h edges.

Next, we know that the induced subgraphs of h number of N_{x_j} subsets are also connected in G using distinct edges. That totally take up all remaining h edges. Hence, each N_{x_j} takes up exactly one remaining edge. Therefore, we set x_j as true, if $(\mathsf{a}_j, \mathsf{c}_j) \in E$. Otherwise, set $\bar{x}_j$ as true, if $(\mathsf{b}_j, \mathsf{c}_j) \in E$. This is a consistent assignment for each variable.

Finally, we also know that the induced subgraphs of all N_{c_i} are connected in G. This implies that at least one $(\mathsf{x}_j, \mathsf{c}_j)$ in N_{x_j}, where $\mathsf{x}_j \in \{\mathsf{a}_j, \mathsf{b}_j\}$, is also present in N_{c_i}. Otherwise, other edge in N_{c_i} will require to take up an extra edge in E. Hence, every clause is satisfiable.

Therefore, we show that PoCO is NP-hard, because 3SAT problem is NP-complete. ■

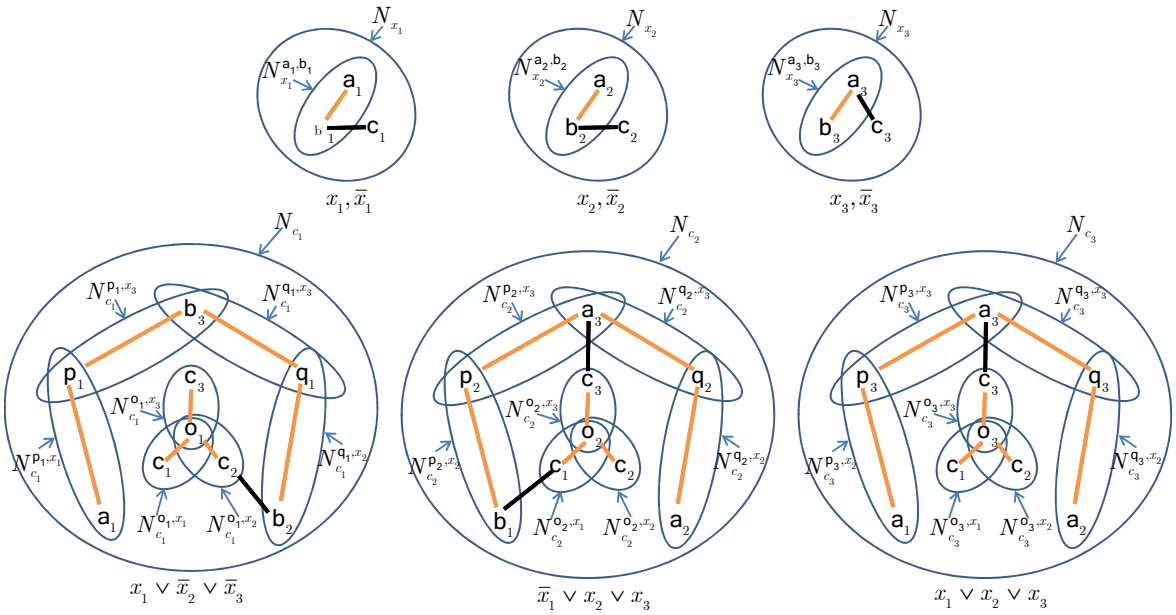

Fig. 2. An illustration of construction of PoCO (N, Ω, Δ) for a given $F = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \vee (x_1 \vee x_2 \vee x_3)$. The truth assignment is $x_1 = 0, x_2 = 0, x_3 = 1$, which is depicted as the selected black edges in the figures.

III. Centralized overlay construction algorithms

We begin by defining a few terms and notations that will be used in the description of the algorithms throughout this and subsequent sections.

Suppose we are given a set of nodes N and a collection of its subsets $\Omega = \{N_1, \cdots, N_K\}$. We denote by $\Omega_v \subset \Omega$ the groups that a node $v \in N$ belongs to, i.e., $\Omega_v = \{N_k \in \Omega | v \in N_k\}$. Similarly, $\Omega_{u,v}$ denotes the groups that nodes u and v both belong to, i.e., $\Omega_{u,v} = \Omega_u \cap \Omega_v$.

We say a pair of nodes u and v both in some group N_k are *group-connected in* N_k (N_k*-group-connected* in short) in a graph $G = (N, E)$ if there exists a connected path between them in the subgraph $G_k = (N_k, E_k)$. We denote by $C_{E_k}(N_k) \subset N_k \times N_k$ the set of N_k-group connected node pairs in $G_k = (N_k, E_k)$.

Given a subgraph $G_k = (N_k, E_k)$, the following function returns the number of *new* group-connected pairs in N_k when an edge (u, v) is added to E_k:

$$\text{NEW-CONNS}(u, v, N_k, E_k) = |C_{E_k \cup \{(u,v)\}}(N_k) - C_{E_k}(N_k)|.$$

The value returned by NEW-CONNS function is used in our algorithms hereafter as the "utility" of an edge for a graph G_k, i.e., as a measure of how many new node pairs will be connected due to the addition of an edge to a graph. Note that it is easy to see NEW-CONNS returns in a polynomial time since there exist efficient algorithms to verify the connectivity of all node pairs in a graph (e.g., breadth-first-search, depth-first-search, etc.).

A. Centralized algorithm

Our first algorithm, GREEDY-CONN (see Algorithm III-A), is based on a greedy decision such that, at each step, an edge is inserted whose addition maximizes the number of new group-connected node pairs in the graph constructed thus far. Let E^U be the set of all pairs of nodes in N, i.e., $E^U = \{(u, v) | u \in N, v \in N, u \neq v\}$.

Algorithm 1 *GREEDY-CONN*: Input (N, Ω), Output $G = (N, E)$

1: $E \leftarrow \emptyset$
2: $F \leftarrow E^U$
3: **while** $F \neq \emptyset$ **do**
4: **for** each $(u, v) \in F$ **do**
5: $m_{(u,v)} \leftarrow \sum_{k=1}^{K} \text{NEW-CONNS}(u, v, N_k, E_k)$
6: **end for**
7: **if** $\max m_{(u,v)} = 0$ **then**
8: break
9: **end if**
10: $(u^*, v^*) \leftarrow \arg\max\{m_{(u,v)} : (u, v) \in F\}$
11: $F \leftarrow F - \{(u^*, v^*)\}$
12: $E \leftarrow E \cup \{(u^*, v^*)\}$
13: **end while**
14: **return** $G = (N, E)$

At each step of the WHILE loop (line 3- 13), the GREEDY-CONN selects an edge whose insertion to the graph can get the most pair of nodes group-connected (lines 5, 10), and adds it one by one to the graph (lines 11, 12), until no edge can add any new group-connected node pair (line 7). When there is a tie, one of the edges will be selected arbitrarily.

It is easy to see GREEDY-CONN completes in a polynomial time w.r.t. $|N|$ and $|\Omega|$ because the NEW-CONNS routine returns in a polynomial time which is repeated for each of K groups (line 5), and, at each step of the while loop, one edge is taken from all potential edge sets of size $|N|(|N| - 1)/2$ (line 11) and added to the graph G (line 12).

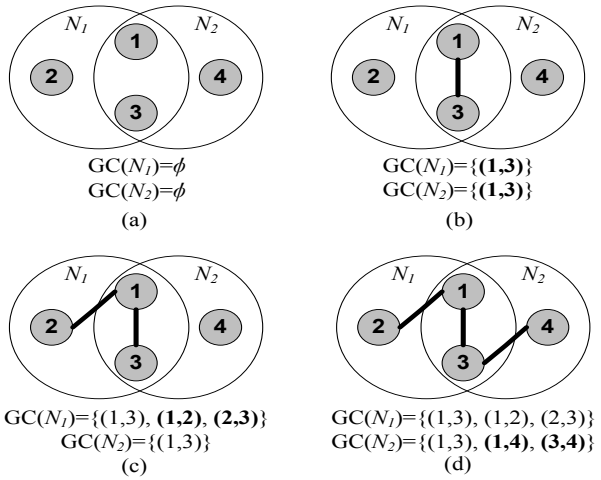

Fig. 3. Example edge assignment by GREEDY-CONN algorithm for two groups $N_1 = \{1, 2, 3\}$ and $N_2 = \{1, 3, 4\}$. The new node pairs that become group-connected in each group is indicated in bold letters.

Fig. 3 illustrates an example of the execution of GREEDY-CONN algorithm for two node groups $N_1 = \{1, 2, 3\}$ and $N_2 = \{1, 3, 4\}$, where, at each stage of the execution, the set of group-connected node pairs for each group is shown in GC(N_k). In the first step, the edge $(1, 3)$ is added since it will make two group-connections (one in each group), as shown in Figure 3(b)–all others make only one group-connection. Then one of the edges $(1, 2)$, $(2, 3)$, $(1, 4)$, and $(3, 4)$ can be added at the second step as any of them would make two pairs of nodes group-connected. In this example, edge $(1, 2)$ is added with the tie broken arbitrarily (Fig. 3(c)). Note that the addition of $(1, 2)$ makes nodes 2 and 3 connected in addition to the node pair 1 and 2. This makes the addition of edge (2,3) unnecessary thereafter. Finally, edge (3,4) is added from group N_2 (Figure 3(d)), resulting in a graph compatible with the grouping. Note that, the greedy algorithm achieves the optimal assignment in this particular example.

The following lemman shows the correctness of the GREEDY-CONN.

Lemma 1: Graph $G = (N, E)$ returned by GREEDY-CONN(N, Ω) is compatible w.r.t. Ω.

Proof: The proof is by contradiction. Suppose there is a subgraph $G_k = (N_k, E_k)$ that is induced by a set $N_k \in \Omega$ and is not connected when the algorithm returns. By definition, G_k has at least one pair of nodes, u and v, between which there is no connected path in G_k. The edge (u, v) is not E_k (hence not in E either) because it would make a connected path between u and v. Therefore, upon the completion of the algorithm, F is not empty and contains (u, v), for which $m_{(u,v)} \geq 1$. This means none of the two terminating conditions of the algorithm (lines 4 and 7) is satisfied, contradicting to the assumption that the algorithm has returned. ■

B. Further reducing the number of links

The greedy algorithm tries to avoid adding unnecessary edges to the graph by adding only the edges whose utility is positive. However, the choice of edges is only based on the graph topology built in the previous steps, not on the edges that will be added in the later stage. Thus, the final output graph may contain *redundant* edges, which can be removed from the graph without rendering it incompatible.

For a given graph $G = (N, E)$, we say an edge $(u, v) \in E$ is *redundant* in G if, for each $N_k \in \Omega_{u,v}$, there exists a connected path between u and v other than the edge (u, v) itself in the induced subgraph (N_k, E_k). Note that an edge (u, v)'s redundancy in a given graph G can be easily verified by comparing the number of connected node pairs in (N_k, E_k) and that in $(N_k, E_k - \{(u, v)\})$ for all k.

The following lemma shows a redundant edge can be safely removed without affecting the group-connectivity of the nodes.

Lemma 2: Suppose there is a redundant edge (u, v) in a graph $G = (N, E)$. If a pair of nodes u' and v' are N_k-group-connected in G for any group N_k, u' and v' are also N_k-group-connected in $G^- = (N, E - \{(u, v)\}$.

Proof: Consider a redundant edge (u, v) in $G = (N, E)$. Let $G_k = (N_k, E_k)$ be the subgraph induced by any arbitrary N_k in $G = (N, E)$. Suppose an arbitrary pair of nodes u' and v' are N_k-group-connected for some N_k, and let a sequence of nodes $(u' = u_0, u_1, \cdots, u_l = v')$ denote a path in between u' and v' that go through nodes $u_1, \cdots, u_{l-1}$ ($u_i \in N_k$). If the path does not include a subsequence (u, v), then the removal of edge (u, v) does not affect this path, hence u' and v' are still connected in $(N_k, E_k - \{(u, v)\}$. If the path does include the subsequence (u, v), because there is a path between u and v in N_k other than the edge (u, v) by definition of the redundant edges, after the removal of (u, v), the pair u' and v' must be connected by another path which is represented by replacing the subsequence (u, v) in $(u' = u_0, u_1, \cdots, u_l = v')$ with the node sequence representing the alternate path between u and v. ■

Therefore, since removing a redundant edge does not hurt the group-connectivity of a graph, once we have a graph G compatible with Ω, we can check the edges in G one by one (in any order) to see if there are redundant edges to remove from G as a final step after the completion of GREEDY-CONN.

IV. Distributed overlay construction algorithm

A. Distributed algorithm

In the distributed algorithm, each node continually makes a local decision as to whether to add or delete edges that are incident to itself based on the local connectivity information of the access control groups that it belongs to until the algorithm converge.

Given a subgraph $G_k = (N_k, E_k)$ for some $N_k \in \Omega_u$, let us denote by $\mathcal{N}_u(G_k)$ the set of neighbor nodes of u in G_k, i.e., $\mathcal{N}_u(G_k) = \{v \mid (u, v) \in E_k\}$. Similarly, $\mathcal{E}_u(G_k)$ represents the set of local edges incident to u in G_k. Also $\mathcal{E}_u$ denotes

all edges incident to u in the entire graph $G = (N, E)$, where $E = \bigcup_{k=1}^{K} E_k$.

For the correctness of the algorithm execution, we assume for now the following two conditions hold before the distributed algorithm is invoked by any node (we describe in Section IV-C how a distributed protocol of message exchange among nodes can ensure these conditions):

- (C1) Before executing the algorithm, u has the correct, up-to-date topology information of $G_k = (N_k, E_k)$ for all $N_k \in \Omega_u$.
- (C2) When u adds or deletes an edge (u, v) for some $v \in N_k$ after executing the algorithm, no other node in N_k adds or deletes an edge at the same time.

Now, in our distributed construction of policy-compatible overlay, each node invokes DISTRIBUTED-GREEDY algorithm (see Algorithm 2) whenever it obtains a new group topology $G_k = (N_k, E_k)$, including the initial stage with empty topology information.

Given $G_k = (N_k, E_k)$ for all $N_k \in \Omega_u$, DISTRIBUTED-GREEDY outputs an updated set of a node u's local edges $\tilde{\mathcal{E}}_u$ to be included in the graph.

Algorithm 2 *DISTRIBUTED-GREEDY*: Input $(u, \mathcal{N}_u, N_k, E_k)\ \forall\ N_k \in \Omega_u$, Output $\tilde{\mathcal{E}}_u$

1: $\tilde{\mathcal{E}}_u \leftarrow \mathcal{E}_u$
2: $V_u \leftarrow \bigcup_{N_k \in \Omega_u} N_k$
3: $F_u \leftarrow V_u - \mathcal{N}_u$
4: **for all** $v \in F_u$ **do**
5: $\quad m_v \leftarrow \sum_{N_k \in \Omega_u}$NEW-CONNS$(u, v, N_k, E_k)$
6: **end for**
7: **if** $\max_v > 0$ **then**
8: $\quad v^* \leftarrow \arg\max_{v \in F_u} m_v$
9: $\quad \tilde{\mathcal{E}}_u \leftarrow \tilde{\mathcal{E}}_u \cup \{(u, v^*)\}$
10: $\quad$ **for all** $N_k \in \Omega_u$ **do**
11: $\quad\quad E_k \leftarrow E_k \cup \{(u, v^*)\}$
12: $\quad$ **end for**
13: **end if**
14: $A_u[\cdot] \leftarrow$ Random ordering of edges in $\tilde{\mathcal{E}}_u$
15: **for** $i \leftarrow 1$ to $|\tilde{\mathcal{E}}_u|$ **do**
16: $\quad (u, v) \leftarrow A[i]$
17: $\quad$ **if** (u, v) is redundant in any $G_k = (N_k, E_k)$ for $N_k \in \Omega_u$ **then**
18: $\quad\quad \tilde{\mathcal{E}}_u \leftarrow \tilde{\mathcal{E}}_u - \{(u, v)\}$
19: $\quad$ **end if**
20: **end for**
21: **return** $\tilde{\mathcal{E}}_u$

Like the centralized GREEDY-CONN algorithm, DISTRIBUTED-GREEDY chooses an edge such that it can maximize the number of new group-connected node pairs added to the current graph (lines 4-13). However, there are a few differences between DISTRIBUTED-GREEDY and its centralized counterpart:

- Each node only adds its own edges (the algorithm returns new set of edges for node u only: line 21), by selecting one from those that are in at least one of its groups but not yet in its own edges $\tilde{\mathcal{E}}_u$.
- Each node only considers the topology and the utility of its edges within the groups it belongs to (the input includes only the groups the node belongs to).
- At each invocation of the algorithm, the node also removes redundant edges among its own existing edges (lines 14-20).

Note that, after each execution of the algorithm, at most one edge is added, but multiple redundant edges can be removed, and sometimes no edge is added or deleted (when there is no edge with positive utility or redundant edge). Through the repeated execution of DISTRIBUTED-GREEDY algorithm by all nodes, the collective edge addition and deletion results in a policy-compatible graph as a whole. We formally show the convergence of this process in Section IV-B.

Note also that, while each node adds and deletes only its own edges, its decision is based on the utility and redundancy of the edges not only for the node itself but for other pairs of nodes in the groups that the node belongs to. This is possible because each node keeps getting updated with the graph topology of its groups through the distributed protocol described in the Section IV-C.

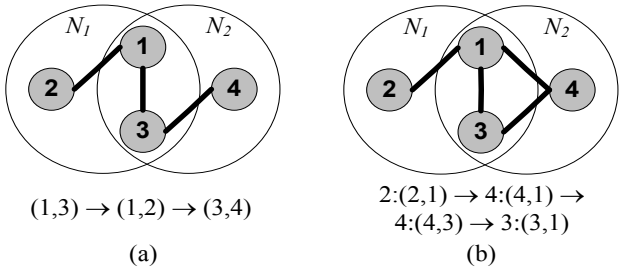

Fig. 4. Distributed greedy assignment can assign more edges than its centralized counterpart. (a) The same assignment by the centralized algorithm as in Figure 3 (b) The edges added by a run of distributed assignments by nodes in the order of 2, 4, 4, and 3. The number before each edge in the sequence indicates the node that adds that edge.

The distributed algorithm is more scalable than its centralized counterpart as it reduces the overhead for each node and it utilizes only local information. This however comes at a cost: Due to each node's myopic view of local network topology, the number of links collectively added by all nodes can be greater than the result computed by the centralized algorithm. Figure 4 illustrates such an example, where the edges are added by each individual nodes in a particular order of nodes 2, 4, 4 (again), and 3. This inefficiency results from the fully distributed execution of the algorithm by each individual node, which only controls its own edges and can not force other nodes to choose edges to add or delete in their set. However, our simulation study in Section VI shows that the performance of DISTRIBUTED-GREEDY is generally within a small margin of what the centralized counterpart achieves.

B. Convergence of distributed algorithm

In this section, we show that the distributed process of adding and deleting edges by each node converges to a global

graph compatible w.r.t. Ω under the two conditions C1 and C2 in Section IV-A.

For a given graph $G = (V, E)$ and node grouping $\Omega = \{N_1, \cdots, N_K\}$, let us denote $Q_\Omega(G)$ be the total number of group-connected node pairs under G in all $N_k \in \Omega$, i.e.,

$$Q_\Omega(G) = \sum_{k=1}^{K} \sum_{u,v \in N_k} \mathbb{1}(u \text{ and } v \text{ are } N_k\text{-group-connected}).$$

Every time a node changes its edge, the global graph $G = (N, E)$ changes: when an edge (u, v) is added or deleted by either u or v, E will change to $E \cup \{(u, v)\}$ or $E - \{(u, v)\}$, respectively. Since only one node can add or delete its edges at a time (due to the condition C2), without loss of generality, we denote the evolving sequence of graphs by $(G_0, G_1, G_2, \cdots)$.[2]

Theorem 2: Starting from an arbitrary graph G_0, the evolving sequence of the graphs $(G_0, G_1, G_2, \cdots)$ generated by the distributed process DISTRIBUTED-GREEDY converges to a stable graph $\tilde{G} = (N, \tilde{E})$ which is compatible w.r.t. Ω in a final number of steps.

Proof: We first prove the convergence of the process. For any $i = 0, 1, \cdots$, the transition from G_i to G_{i+1} occurs when some edge (u, v) is either added or deleted. Since an edge (u, v) is added only when it can create some new group-connected node pair in some $N_k \in \Omega_{u,v}$ (line 7), $Q_\Omega(G_i) < Q_\Omega(G_{i+1})$ if the transition is caused by an edge addition. If the transition occurs due to an edge deletion, $Q_\Omega(G_i) = Q_\Omega(G_{i+1})$ because only redundant edges can be deleted, and, by Lemma 2, the number of connected pairs does not decrease for any group due to the removal of redundant edges ($Q_\Omega(G_i)$ does not increase either).

Now since $Q_\Omega(G_i)$ is bounded from above by some constant $\Delta = \sum_{k=1}^{K} \frac{|N_k|(|N_k|-1)}{2}$, the number of times that $Q_\Omega(G_i)$ increases (hence the number of edge additions) is also bounded by Δ. Also, for each interval between two consecutive edge additions, there can be only up to $\frac{|N|(|N|+1)}{2}$ edge deletions. Therefore, the distributed process should terminate within a number of steps less than $\frac{\Delta|N|(|N|+1)}{2}$.

Next we show the converged graph is a compatible graph by contradiction. Suppose the distributed process results in a graph $G = (V, E)$ not compatible w.r.t. Ω. Then there must be some pair of nodes (u, v) in some group N_k that are not group-connected in $G_k = (N_k, E_k)$. For each such pairs, however, this condition must be recognized by all nodes in N_k as all nodes in N_k have the correct topology information of $G_k = (N_k, E_k)$. If no other node in N_k adds an edge to improve the connectivity of u and v in G_k, either u or v will add an edge between them since adding such an edge will increase the number of group-connected pairs in N_k. This contradicts to the assumption that the process has terminated. ■

The proof of Theorem 2 reveals the following property.

[2]Even if a node deletes multiple edges after a single execution of the algorithm, one can break the multi-edge deletions into multiple, consecutive single-edge deletions in arbitrary order.

Corollary 3: The distributed process of DISTRIBUTED-GREEDY terminates in a polynomial number of steps w.r.t $|N|$ and $|\Omega|$.

Proof: In the proof of Theorem 2, we have shown the total number of times DISTRIBUTED-GREEDY is executed by all nodes is bounded by $\sum_{k=1}^{K} \frac{|N_k|(|N_k|-1)}{2} \times \frac{|N|(|N|+1)}{2}$ which is in $O(|\Omega||N|^4)$. Also since each execution of DISTRIBUTED-GREEDY is also polynomial in $|N|$ and $|\Omega|$, the whole process finishes in a polynomial time steps. ■

C. Obtaining group topology

Nodes use a distributed protocol to ensure the two conditions C1 and C2 presented in Section IV-A. Our mechanism is similar in spirit to the distributed protocol proposed in [8], which is designed for distributed resource replication. Here we provide only a high-level sketch of the protocol, with particular emphasis on how it is used in our context of distributed overlay construction. We assume existence of some separate control-plane communication methods for the following protocol, i.e., nodes can exchange messages with each other without the existence of the secure overlay link being built by the distributed process.

Before a node changes its edges in a group, it initiates a three-way handshake of messages with all other nodes in the same group to get consent of other nodes in the group for the change. The handshake is composed of three stages: *REQUEST-ACCEPT/REJECT-UPDATE/ABORT*: (i) First, when node u decides to change an edge (u, v), u sends REQUEST messages to other nodes in the group u and v belong to, indicating u's intention to change its edge. Other nodes respond to this message either by accepting the change or rejecting it. (ii) Any node w that receives the request from u REJECTs it if the responding node has itself initiated a handshake process for its own edge's change, to prevent a simultaneous edge change by u in the same group. Otherwise, w ACCEPTs u's request, after which it refrains from initiating its own change process until it receives the further outcome (UPDATE or ABORT) of the change process from u. (iii) Finally, if node u receives ACCEPT messages from all nodes in the group, it changes the edge (u, v), and sends UPDATE messages to group nodes. If it receives at least one REJECT, it does not change the edge, and sends ABORT messages.

The handshake process serves two purposes: (i) A node u changes its edge in group N_k only when all other nodes in N_k have accepted the change. Since, if all of them have accepted the changes, they are prevented from changing their own edges until the requesting node completes its change, the edge change by u must be the only one at the time of its change. This ensures condition C2. (ii) If a node $v \in N_k$ receives an *UPDATE* message from some other node $u \in N_k$, the edge change contained in the message is used to update v's view of the topology of N_k. Since every node will send the *UPDATE* message within N_k for every edge change, and since a node v would not attempt to change its edge in N_k if some other node in N_k is in the changing process, the group

topology $G_k = (N_k, E_k)$ is always accurate by the time any node v in N_k changes its edge. This ensures condition C1.

V. Taking the link cost into account

While PoCO optimization is our primary objective in this paper, we acknowledge that it is also important to consider the "cost" of adding a link. For example, the link can be assigned a cost proportional to the distance in the underlying substrate network, indicating how far apart the two nodes of an overlay link in the underlying substrate network. Another example is to have a link cost inversely proportional to the mutual trust-level between two nodes (i.e., what security level needs to be maintained for a given node with certain trust-level?). In such cases, it is advantageous to minimize the overall link cost as well as the total number of links.

To account for such link cost, we assume a cost function $c : N \times N \rightarrow \Re^+$ is available to map each pair of nodes u and v in N to a positive real number.[3] Then we can formulate another optimization problem of constructing the graph $G = (V, E)$ compatible w.r.t. Ω, such that the total cost $c(E) = \sum_{(u,v)\in E} c(u, v)$ is minimized.

Achieving both objectives of reducing $|E|$ and $C(E)$ together is, however, basically a multi-objective optimization problem, for which a solution that achieves better performance for one objective will suffer for other objectives. In our context, these two objectives indeed often conflict, especially when the link costs are widely varied: minimizing only $|E|$ can result in a graph whose $C(E)$ is far worse from what would have been resulted from an effort of minimizing $C(E)$ only.

Our approach is to design a solution that can address both objectives in a *systematic* manner, such that the outcome of the solution can be easily adjusted and optimized toward either objective. This is done by Algorithm 3, a modification of GREEDY-CONN, in which a single parameter α can be used to systematically balance the algorithm's tendency toward either objectives.

More specifically, we say a non-negative integer m is in the α-*margin* of another integer n for $m \leq n$ if $\frac{n}{m} \geq \alpha$ for real-valued constant $\alpha \in [0, 1]$. Then, given a link cost function $c : N \times N \rightarrow \Re^+$ and a constant α, GREEDY-MARGIN-CONN selects an edge at each step in the following manner.

Like in GREEDY-CONN algorithm. GREEDY-MARGIN-CONN keeps adding edges until all node pairs in all groups are group-connected. The main difference is that, instead of selecting an edge that maximize the number of new group-connected node pairs, GREEDY-MARGIN-CONN chooses the one with the smallest cost among those whose utilities ($\sum_{k=1}^{K}$NEW-CONNS(u, v, G, N_k)) are within α-margin of the maximum utility (lines 11-17).

By taking into account the edges with smaller utilities, GREEDY-MARGIN-CONN can search and select an edge of smaller cost in a larger candidate set. The parameter α determines the size of the search space–the smaller the value of α, the larger the candidate set. $\alpha = 1$ is a special case whose candidate set is identical to GREEDY-CONN algorithm except that ties are broken by the link cost. On the other hand, when $\alpha = 0$, only the link cost will be considered in constructing the graph.

[3]There are existing methods available to obtain certain cost functions in a distributed way; for instance, the routing information in the underlying network can be used for the network distance measure.

Algorithm 3 *GREEDY-MARGIN-CONN*: Input (N, Ω, α, c), Output $G = (N, E)$

```
1:  E ← ∅
2:  F ← E^U
3:  while F ≠ ∅ do
4:     for each (u, v) ∈ F do
5:        m_(u,v) ← Σ_{k=1}^{K} NEW-CONNS(N_k, E_k)
6:     end for
7:     m_max ← max m_(u,v)
8:     if m_max = 0 then
9:        break
10:    end if
11:    H ← ∅
12:    for each (u, v) ∈ F do
13:       if m_(u,v) > 0 and m_(u,v) is in α-margin of m_max then
14:          H ← H ∪ {(u, v)}
15:       end if
16:    end for
17:    (u*, v*) ← arg min{c(u, v) : (u, v) ∈ H}
18:    F ← F − {(u*, v*)}
19:    E ← E ∪ {(u*, v*)}
20: end while
21: return G = (N, E)
```

The distributed counterpart of GREEDY-MARGIN-CONN is also possible and straightforward. In what would be called DISTRIBUTED-GREEDY-MARGIN, the selection of an edge to be added is made not only from the set of the edges with the highest utility, but also from the edges that have the utility within α-margin of the maximum utility, with the edge with the smallest cost selected from the extended set.

VI. Performance Evaluation

In this section we present the performance of the algorithms proposed in this paper by simulation. In our simulations, we first study the performance in a random setting, follow by a realistic social grouping extracted from a trace that contains conference Program Committee (PC) member information.

A. Random groupings

In the random setting, we create random grouping of nodes Ω, for which each node u is assigned to group N_k with some probability p. We conduct simulations by varying $|N|$ and $|\Omega|$, and also p. Due to space limitation, we only present a limited set of result, but report that we found essentially the same trend holds for other cases. We first present the result of $|E|$-performance for different algorithms, then we present the result of convergence- and $c(E)$-performance. We obtain the performance values by averaging the results of 10 runs.

1) $|E|$-performance: We first see the $|E|$-performance of GREEDY-CONN algorithm against the optimal value. Figure 5 shows the ratio of $\frac{|E_{greedy}|}{|E_{optimal}|}$, where $|E_{greedy}|$ and $|E_{optimal}|$

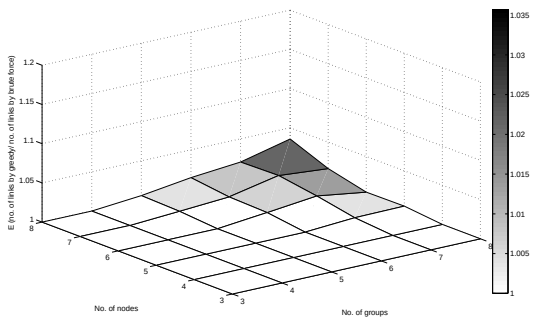

Fig. 5. Performance ratio of GREEDY-CONN versus optimal solution

are the numbers of edges assigned respectively by GREEDY-CONN and an exhaustive search. Since the complexity of the exhaustive search quickly explodes as $|N|$ and $|\Omega|$ increases, we are only able to show the results for small ranges of $|\Omega|$ and $|N|$ (i.e., both $|N|$ and $|\Omega|$ are less than 8). In this case, we find that GREEDY-CONN performs very well (i.e., GREEDY-CONN only performs around 3% worse than optimal), but the performance gap increases in general as $|N|$ and $|\Omega|$ increase.

For a comparison with a baseline approach, we include the results of an algorithm that, starts from complete graph for G, removes redundant edges one by one in the decreasing order of edge cost. We call this algorithm G-MST (Group-Minimum Spanning Tree) due to its similarity to Reverse-delete algorithm [7]. Reverse-delete algorithm solves optimal minimum spanning tree problem for a single tree, except that it removes only the redundant edge w.r.t. the grouping Ω. Since G-MST is designed only toward the goal of minimizing $c(E)$ but not $|E|$, we set $c(E) = 1$ for all links in this $|E|$-performance evaluation.

The performance of the distributed algorithms are quite close with their centralized counterparts. This can be verified more clearly in Figure 6 ($|\Omega|$=20), where $|E|$ resulted by DISTRIBUTED-GREEDY is about 5% to 10% higher than that from GREEDY-CONN. Figure 7 shows the impact of varying group size with p varied to the $|E|$ performance. Interestingly, there is a trend of increasing $|E|$ initially when p increases in small values (0.2 to 0.4) but it decreases as p further increases. This is because when p is small, increasing p means bigger sizes of individual groups, thus requiring more links for each group. However as p keeps increasing and approaching to 1.0, the number of overlapping nodes across groups also increases, and links assigned to a group can be shared and reused by other groups.

2) Convergence performance: In Figure 8, we plot the total number of times that the links are added or deleted by the collective process of DISTRIBUTED-GREEDY (i.e. how many steps the algorithm takes to terminate). The curves suggest that the total number of link changes increases sub linearly with the number of the groups and nodes. This in turn means the edge changes per node is kept within a constant level (less than 2 times on average in the figure) regardless of the size of input $|N|$ and $|\Omega|$, suggesting our algorithm will be highly scalable in large-scale networks.

3) $c(E)$-performance: In section V, we discuss a way to extend our algorithms to minimize $c(E)$. To evaluate the $c(E)$-performance of our extended algorithms, we assign the cost of link between each pair of nodes uniformly in $[1, |N|]$. We evaluate both the number of links $|E|$ and the total link cost $c(E)$ assigned by the algorithms. Figures 9 and 10 compare the $|E|$ and $c(E)$ performance of the distributed algorithms respectively.

Figures 9 show the average $|E|$, selected by our distributed algorithms with $|\Omega| = 10$. Our first observation (and rather obvious one) is that $|E|$ increases as $|N|$ increases. It is because with fixed p, average group size $|N_k|$ increases with $|N|$, for which more links are needed for group-connectivity.

We also notice in these figures is that, as intended by the design, the $|E|$ performance of MARGIN-version of the algorithms degrades gracefully as we decrease the marginal parameter α from 1 to 0; With $\alpha = 1$, the design of (DISTRIBUTED-)GREEDY-MARGIN-CONN algorithms are equivalent to simple greedy algorithms in terms of $|E|$ performance.

The benefit of increased α in terms of $c|E|$ performance is clear in Figures 10, where the total link cost gradually decreases as we decrease α. At $\alpha = 0$ for instance, GREEDY-MARGIN-CONN algorithm essentially results in the same performance by G-MST, whose goal is to solely optimize $c(E)$ performance. In summary, the results in figure 9 and 10 validate that our MARGIN-CONN algorithms effectively achieve our goal of balancing the weights given to $|E|$ and $c(E)$ performances easily by controlling a single parameter α.

B. Realistic social groupings

In the last evaluation, we evaluate our proposed algorithms in a realistic social group setting. We utilize the dataset from [15], which study the characteristics of Program Committee (PC) members more than 2900 conferences. From this dataset, we select the set of most recent high-quality conferences (i.e., conferences hold at 2006 and classified as reputable conferences by [15]), which results 32 conferences and 482 PC members. We create a group for each conference and assign the PC members to the corresponding groups to emulate a secure data distribution for each conference. Links in this scenario carry a same unit cost of 1, thus minimizing $c|E|$ equals to minimize $|E|$ in this case.

Figure 11 presents the result of average $|E|$ with different algorithms, and figure 12, 13 and 14 present the resulting graphs from different algorithms. We observe that in this realistic social grouping, the result is similar to those with random setting. The performance difference between Greedy-CONN and Distributed-Greedy is 5.89%. Comparing with G-MST, Greedy-CONN and Distributed-Greedy reduce the number of links used in the overlay by 29.8% and 25.7% respectively. From this result, we believe that our proposed algorithms work well for both homogeneous and heterogeneous settings.

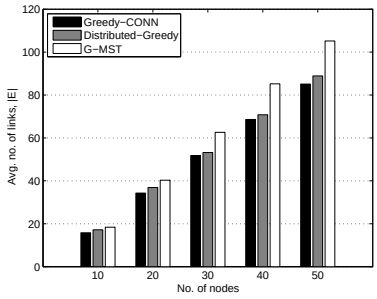

Fig. 6. Avg. no. of links with 20 groups

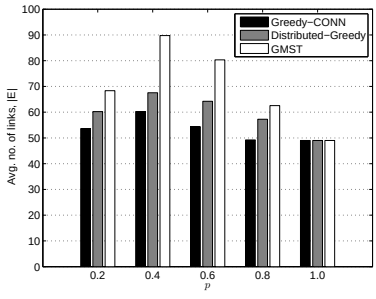

Fig. 7. Avg. no. of links with 10 groups and 50 nodes with different p

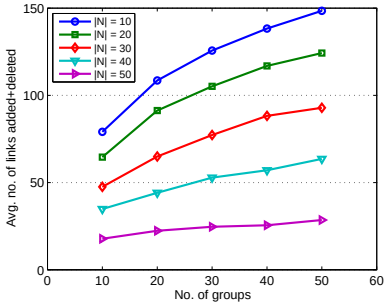

Fig. 8. Avg. no. of links added and deleted by DISTRIBUTED-GREEDY

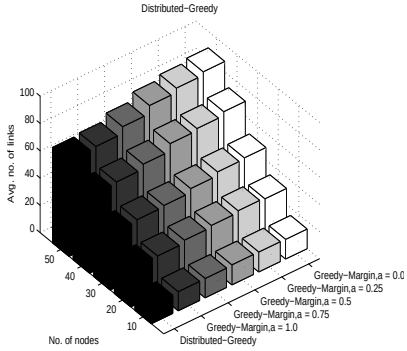

Fig. 9. Avg. $|E|$ with 10 groups (Distributed-Greedy)

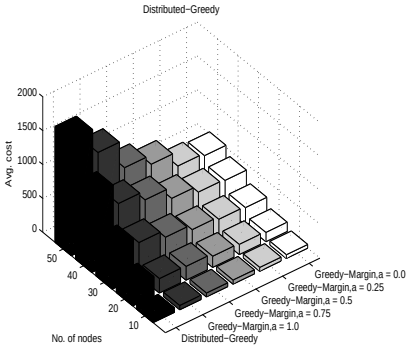

Fig. 10. Avg. $c(E)$ with 10 groups (Distributed-Greedy)

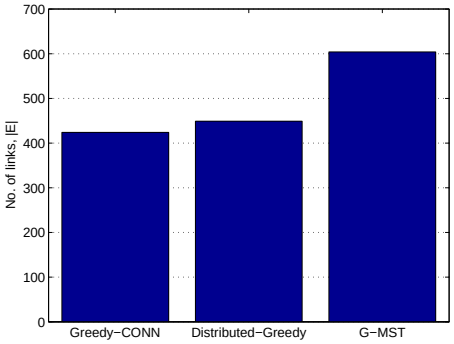

Fig. 11. Avg. $|E|$ with conference PC groupings

VII. Conclusion

In this paper, we studied the problem of how to construct an efficient overlay backbone that can be used to distribute data securely to multiple access groups, where the access groups are defined by a given set of policies. We formulated this problem as an optimization problem to construct a minimum overlay, and prove that it is a NP-complete problem. To address this problem, we designed a centralized algorithm that performs empirically closed to the optimal solutions (within 3%). We also extended the centralized algorithm to a distributed algorithm and showed that the distributed algorithm archives comparable performance to the centralized algorithm. Simulation results suggested that our proposed schemes can perform better than a simple algorithm up to 30%. Finally, with a simple extension, we showed that our proposed algorithms can also support the weighted version of the problem.

References

[1] Akamai. http://www.akamai.com.

[2] D. Agrawal, S. Calo, K.-W. Lee, J. Lobo, and D. Verma. Policy technologies for self-managing systems. 2008.

[3] D. Agrawal, J. Giles, K.-W. Lee, and J. Lobo. Policy ratification. In *Proceedings of IEEE Policy 2005*, June 2005.

[4] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang. Donnybrook: Enabling large-scale, high-speed, peer-to-peer games. In *In Proceedings of ACM SIGCOMM*, 2008.

[5] D. A. Hari, D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. pages 131–145, 2001.

[6] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. In *in Proceedings of ACM Sigmetrics*, 2000.

[7] J. Kleinberg and E. Tardos. *Algorithm Design*, 2006.

[8] B.-J. Ko and D. Rubenstein. Distributed self-stabilizing placement of replicated resources in emerging networks. *IEEE/ACM Transactions on Networking*, 13(3):476–487, 2005.

[9] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Comput. Surv.*, 35(3):309–329, 2003.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. pages 149–160, 2001.

[11] D. A. Tran, K. Hua, and T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *In Proc. of IEEE Infocom*, 2003.

[12] A. K. Vishal, V. Misra, and D. Rubenstein. Sos: Secure overlay services. In *In Proceedings of ACM SIGCOMM*, pages 61–72, 2002.

[13] D.-N. Yang and W. Liao. On bandwidth-efficient overlay multicast. *IEEE Trans. Parallel Distrib. Syst.*, 18(11):1503–1515, 2007.

[14] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22:41–53, 2004.

[15] Z. Zhuang, E. Elmacioglu, D. Lee, and C. L. Giles. Measuring conference quality by mining program committee characteristics. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 225–234, New York, NY, USA, 2007. ACM.

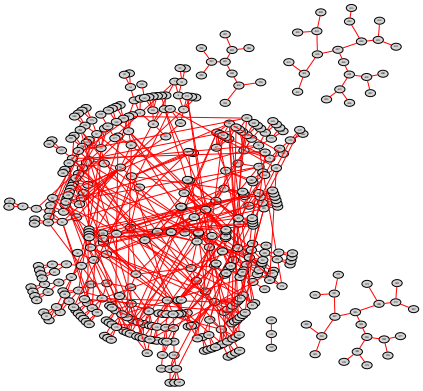

Fig. 12. Result of G-MST, $|E| = 604$

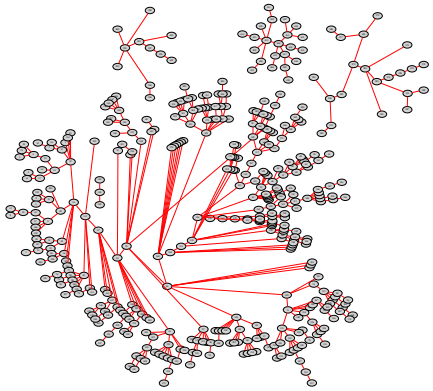

Fig. 13. Result of Greedy-CONN, $|E| = 424$

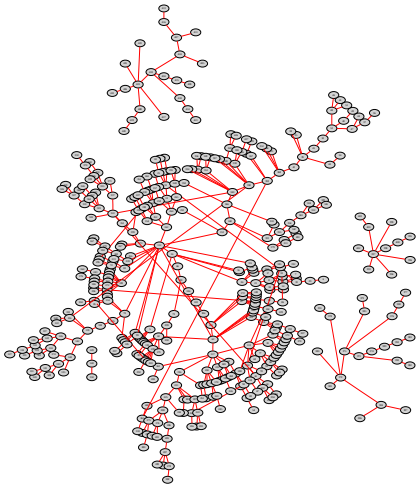

Fig. 14. Result of Distributed-Greedy, $|E| = 449$