How to Select Optimal Gateway in Multi-Domain Wireless Networks: Alternative Solutions without Learning

Sheng Zhong and Yuan Zhang

***Abstract*—Gateways are a crucial part of wireless networks. In multi-domain wireless networks, the existing solution to the problem of optimal gateway selection is based on distributed learning. While such a solution is interesting and useful, it has a fundamental limitation: the learning algorithm may stay for long time in a Nash Equilibrium that does not correspond to the optimal gateway selection. This stay can be so long that people can hardly wait for the convergence of the learning algorithm to the optimal gateway selection.**

In this paper, we present a systematic study of the gateway selection problem. We distinguish three cases and present an alternative solution to the problem in each of these cases: for public link costs, an algorithm to compute the optimal gateway selection directly; for two domains with private link costs, a cryptographic protocol; for three or more domains with private link costs, a perturbation-based protocol. In all the three cases, our solutions accurately compute the optimal gateway selection within reasonable amounts of time, although we emphasize that our solutions are NOT improvements to the gateway selection solutions based on distributed learning (mainly because our solutions are centralized instead of distributed).

***Index Terms*—Gateway selection, multi-domain networks, optimization.**

I. Introduction

GATEWAYS are a crucial part of wireless networks that can affect the network performance in various aspects. Hence, they have received significant attention from the research community [1]–[4].

In multi-domain wireless networks, the problem of gateway selection is particularly important. In this paper, we consider a coalition network [4] with a wireless communication backbone serving for information exchange. The backbone is established by all domains inside this coalition network. Domains could be heterogeneous in the sense that the communication protocols in the domains could be different. In order to facilitate the inter-domain communication, each domain designates a gateway node which works as a translator. To communicate with a node in another domain A, a node would send her message to the gateway node of her domain B. Through the wireless channel between the gateway nodes of B and A, the message is sent to domain A and finally to the destination. The paradigm above has been proposed and adopted in areas of mobile ad-hoc networks [5], wireless mesh networks [6], [7], and military communication [8]. Since all domains' gateway selections would determine the efficiency of the entire inter-domain backbone, there is a problem of *optimal gateway selection*: which nodes should be selected as gateways for these domains so that the total cost of all domains is minimized?

Song, Wong, and Lee [4] have studied the problem of optimal gateway selection and developed a nice solution based on distributed learning. If domains execute this distributed learning algorithm, the learning algorithm will finally converge to a Nash Equilibrium (NE) of the gateway selection game, which corresponds to the optimal gateway selection when there are only two domains or the triangle inequality holds for the link cost metric [4]. Furthermore, it is shown that their learning algorithm is the fastest in a class of learning algorithms.

While the existing solution to the optimal gateway selection problem is very nice and interesting, it has a fundamental limitation: In order to guarantee the distributed learning algorithm converges to the optimal NE with probability close to 1, a parameter, τ, used by the algorithm must be set to a very small value, close to 0. However, when τ goes to 0, we find that the learning algorithm may stay in another NE for long time, and that NE may *not* correspond to the optimal gateway selection. Theoretically, the learning algorithm can still leave the suboptimal NE and move to the optimal NE finally. But the entire process of moving from the suboptimal NE to the optimal NE may take so long that, in practice, people can hardly wait for the convergence of the learning algorithm to the optimal gateway selection.

To further illustrate the above limitation, we present both theoretical analysis and experimental data. Specifically, in Section IV, we prove a lower bound for the expected number of iterations that the distributed learning algorithm will spend on a suboptimal NE. In Section VIII-A, we conduct experiments to demonstrate that, the learning algorithm can stay in a suboptimal NE for extremely long time, so that one can hardly wait for the algorithm to reach the optimal NE.

Given the limitation of the existing solution based on distributed learning, clearly it is useful to find alternative solutions to the problem of gateway selection that do not depend on distributed learning. In this paper, we distinguish

Manuscript received November 19, 2012; revised April 3 and August 11, 2013; accepted September 5, 2013. The associate editor coordinating the review of this paper and approving it for publication was D. Tarchi.

This work was supported by RPGE, and NSFC-61021062.

The authors are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China; Computer Science and Technology Department, Nanjing University, Nanjing 210046, China (e-mail: {sheng.zhong, zhangyuan2014}@gmail.com). Y. Zhang is the corresponding author.

Digital Object Identifier 10.1109/TWC.2013.100113.121828

three cases and present an alternative solution to the problem in each of these cases. We emphasize that our solutions on the optimal gateway selection problem is complementary, *not* contradictory, to the existing work based on distributed learning. Moreover, our solutions are NOT improvements of Song, Wong and Lee's paper [4]. (Please see more details about this in the related work section.) Song et al. provide a really elegant and interesting solution to the optimal gateway selection problem from the game theory's prospective. The main objective of this paper is to present a complete picture of this problem, and suggest efficient alternative solutions.

The first case, also the simplest case, is that the involved domains are willing to reveal the link costs to each other, i.e., all link costs are public information. In this case, we propose an algorithm for computing the optimal gateway selection directly from the link costs. Since all link costs are public information in this case, each involved domain can execute this algorithm to compute the optimal gateway selection for itself. This algorithm always terminates within reasonable amounts of time. (Our experimental results in Section VIII-B empirically demonstrate the efficiency of our algorithm. Theoretically, the time complexity is not polynomial in the number of nodes, because our algorithm uses a standard solver that is very efficient but theoretically not polynomial time bounded.)

The second case, which is slightly more challenging than the first case, is that the involved domains are unwilling to reveal the link costs to each other, and there are two domains only. In this case, we propose a cryptographic protocol that computes the optimal gateway selection in a privacy preserving manner. During the entire computation, this protocol does not reveal any knowledge about the private link costs except the result of the computation—the optimal gateway selection itself. Compared with distributed learning, it has the advantages similar to our proposed algorithm in the first case.

The third case, which is the most challenging case we study in this paper, is that the involved domains are unwilling to reveal the link costs to each other, and there are three or more domains involved. In this case, theoretically it is also possible to design a cryptographic protocol to compute the optimal gateway selection. Nevertheless, from a practical point of view, such protocols normally require high computational overheads. Consequently, we propose to use a perturbation-based approach [9]–[13] to protect the link costs while computing the optimal gateway selection. Specifically, we let domains add noises to their private link costs before revealing these costs. The noises are added in a smart way such that the computed result, i.e., the optimal gateway selection, remains accurate. On the other hand, these added noises provide protection of private link costs.

In summary, we make the following contributions in this paper:

- We formally prove the existing learning algorithms' limitation.
- We systematically study the problem of optimal gateway selection in multi-domain wireless networks and present solutions to this problem in all the three cases: public link costs, private link costs for two domains, private link costs for three or more domains.
- In all the three cases, our solutions always terminate within reasonable amounts of time. Hence, our solutions are very useful in the situations where suboptimal NEs exist.

The rest of this paper is organized as follows. We discuss the related work in Section II. Section III introduces the optimal gateway selection problem and its corresponding gateway selection game. After analyzing the limitation of the existing solution based on distributed learning in Section IV, we present three alternative solutions for the three cases in Section V, VI, and VII, respectively. Evaluation results are given in Section VIII and the discussion on adapting our solutions to handle link cost changes and enable load balance is provided in Section IX. We conclude this paper in Section X.

II. Related Work

Although a lot of work has been done on inter-domain routing protocols or architectures for multi-domain networks with gateway nodes [8], [14]–[16], the selection and deployment of gateway nodes have been investigated only in a few existing studies [1]–[4].

In [1], Li, Wang, and Li study gateway placement in wireless mesh networks. Unlike our work, which considers multiple domains and targets at total cost minimization, their work considers a single domain and targets at throughput maximization.

Xu, et al. [2] design algorithms for placing gateways to minimize costs in wireless mesh networks. Unlike our work, they consider a single domain and allow multiple gateway nodes to be selected from this domain.

He, et al. [3] present an algorithm for utility-based gateway deployment. However, their algorithm is designed for Delay Tolerant Networks, which completely differ from the multi-domain wireless networks we study in this paper.

A. Differences between our work and [4]

Song, Wong, and Lee [4] have a very nice piece of pioneering work on the problem of optimal gateway selection in multi-domain wireless networks. They have proved the optimal gateway selection is an NE when the triangle inequality holds or there are only two domains which inspires our work. There are several differences between Song et al.'s work and ours.

First, our work focuses on studying a problem model which is different from the model of Song, Wong and Lee's work. Specifically, they focus on studying the gateway selection as a game and propose a really elegant solution for each domain to reach the NE, while we focus on finding the optimal gateway selection (although the two goals can become one in a few cases as proved in [4].).

Second, their learning algorithm is distributed while our solution is centralized (which is also the reason we have studied the privacy issues in this paper.).

Our work adopts much of their model and provides a set of additional results. It is difficult to have a fair comparison between their paper and ours. Thus our work is a complementary solution to Song et al.s, rather than an improvement to Song et al.s work. If our results are combined with theirs, hopefully a more complete picture of the gateway selection problem can be obtained.

III. Preliminaries

In this section, we present the necessary technical preliminaries. First, we describe the network model, define notations and the problem formally. Second, we review the game theoretic prospective of the problem and the results that have been discovered so far.

A. Network Model and Optimal Gateway Selection Problem

We use (a slightly generalized version of) the network model from the existing work [4]. Consider a multi-domain network where the domains communicate with each other using wireless links. Each domain can choose one of its nodes as its gateway node. Nodes within one domain communicate with each other directly, while nodes from different domains have to rely on the gateway nodes of their domains to forward messages, in order to communicate with each other.

Denote by $\mathcal{D} = \{D_i\}$ $(i \in \mathbb{N}, 1 \leq i \leq |\mathcal{D}|\})$ the set of all domains in the network, by d_i ($d_i \geq 2$) the total number of nodes that domain D_i has, and by $d = \sum_{i=1}^{|\mathcal{D}|} d_i$ the total number of nodes in the network. To simplify the presentation, we assign universal indices from 1 to d to all the nodes. Without loss of generality, we assume that nodes 1 through d_1 are in domain D_1, nodes $d_1 + 1$ through $d_1 + d_2$ are in domain D_2, ..., and nodes $\sum_{i=1}^{|\mathcal{D}|-1} d_i + 1$ through $\sum_{i=1}^{|\mathcal{D}|} d_i$ are in domain $D_{|\mathcal{D}|}$. To simplify notations, we let $d'_0 = 0$ and $d'_i = \sum_{j=1}^{i-1} d_j$. Then, the indices of nodes in domain D_i are $d'_{i-1} + 1, d'_{i-1} + 2, \ldots, d'_i$.

Denote by $c(j, j')$ the cost of one-hop communication from node j to node j' ($c(j, j') \in [0, \mu]$ and $c(j, j) = 0$). We assume all links are symmetric, i.e., $c(j, j') = c(j', j)$. All these link costs together form a cost matrix $C \in R^{d \times d}$, where the element at the jth row and j'th column is $c(j, j')$.

Now suppose that gateway nodes have been selected. For each domain, the cost from one of its own nodes to its gateway node is called an intra-domain cost. In contrast, the link cost from the gateway of it to the gateway of another domain is called an inter-domain cost. Consequently, the objective of *optimal gateway selection* is to choose gateway nodes for all domains such that the total cost of the entire network, including all the intra- and inter-domain costs, is minimized.

Formally, denote by $\widehat{j_i}$ the universal index of the gateway node selected by domain i. We say the gateway selection is optimal if and only if the total cost of the network, defined as $\sum_{1 \leq i \leq |\mathcal{D}|} \sum_{j_i \neq \widehat{j_i}}^{d'_{i-1} < j_i \leq d'_i} c(\widehat{j_i}, j_i) + \sum_{i \neq i'}^{1 \leq i, i' \leq |\mathcal{D}|} c(\widehat{j_i}, \widehat{j_{i'}})/2$, is minimized. The first part of this total cost is the total intra-domain cost, while the second part is the total inter-domain cost.

B. Game Theoretic Concepts and Result

For each domain, different gateway choices may yield different intra-domain costs and also different inter-domain costs. Similar to the existing work [4], we assume that each domain selects its strategy to minimize the sum of its total intra-domain cost and all inter-domain costs that involve itself. Hence, gateway selection is actually a game.

Formally, the gateway selection game is a strategic game. (For definitions of strategic game and its Nash Equilibrium, please refer to [17].) In this strategic game, $\mathcal{D}$ is the player set. For each player $D_i \in \mathcal{D}$, its strategy in the game is $s_i = \widehat{j_i}$, its own choice of gateway. The strategy space of D_i is its set of nodes: $\{d'_{i-1} + 1, d'_{i-1} + 2, \ldots, d'_i\}$. Now denote by s_{-i} the strategy profile of all players except D_i. (It is a convention of game theory to use subscript $-i$ to represent "all players except the ith", we follow this convention throughout this paper.) Then, the utility of D_i in the game is its total intra-domain cost plus all inter-domain costs that involve D_i:

$$\mathcal{U}_i(s_i, s_{-i}) = \sum_{j_i \neq \widehat{j_i}}^{d'_{i-1} < j_i \leq d'_i} c(\widehat{j_i}, j_i) + \sum_{i' \neq i}^{D_{i'} \in \mathcal{D}} c(\widehat{j_i}, \widehat{j_{i'}}). \quad (1)$$

Note that we are considering a *utility-minimizing* game, in which each player tries to minimize its own utility, because a player domain's utility is its cost. This is different from most of the game theoretic literature, in which utility-maximizing games are studied.

In this strategic game, a Nash Equilibrium (NE) is a strategy profile $s^\star = \{s_1^\star, \ldots, s_{|\mathcal{D}|}^\star\}$ such that every player would prefer choosing her strategy specified in $s^\star$ if all other players choose to stick on their strategies specified in $s^\star$, i.e., $\forall i (D_i \in \mathcal{D})$, $\forall s_i \in \{d'_{i-1} + 1, d'_{i-1} + 2, \ldots, d'_i\}$,

$$\mathcal{U}_i(s_i^\star, s_{-i}^\star) \leq \mathcal{U}_i(s_i, s_{-i}^\star). \quad (2)$$

Note that, just like the utilities of players, the total cost of the entire network is also decided by the profile of all players' strategies. Hence, we denote by $\mathcal{F}(s)$ the total cost of the entire network when the players take strategy profile s. As mentioned earlier, this total cost is

$$\mathcal{F}(s) = \sum_{1 \leq i \leq |\mathcal{D}|} \sum_{j_i \neq \widehat{j_i}}^{d'_{i-1} < j_i \leq d'_i} c(\widehat{j_i}, j_i) + \sum_{i \neq i'}^{1 \leq i, i' \leq |\mathcal{D}|} c(\widehat{j_i}, \widehat{j_{i'}})/2. \quad (3)$$

Song, et al. have shown a nice connection between the minimization of total cost and the minimization of individual players' utilities:

Theorem 1: (Optimal Gateway Selection vs. NE [4]) In the strategic game of gateway selection, the profile of all players' strategies that minimize the total cost of the entire network is an NE when the link cost metric $c(j, j')$ satisfies the triangle inequality or there are only two domains.

The NE in Theorem 1 is called the *optimal NE* because it minimizes the total cost of the entire network and thus gives the domains the optimal gateway selection. It is worth noting that, besides the optimal NE(s), there may exist other NEs in the game. We call those NEs the *suboptimal NEs* because they do not minimize the total cost and thus the corresponding gateway selections are suboptimal. Also, if two or more NEs have the same total cost, we call these NEs *equivalent NEs*.

In this paper, we study how domains can find the optimal gateway selection or the optimal NE. Based on the privacy requirement of link costs and the number of domains involved, we can distinguish three cases for the optimal gateway selection problem. We provide a solution in each case, in Sections V, VI and VII, respectively.

IV. Limitation of Solution based on Learning

As pointed out in [4], the existing solution based on distributed learning guarantees that a learning algorithm (in the class of γ-Logit) definitely converges to the optimal NE in the gateway selection game when an important parameter τ (called the *smoothing factor*) goes to 0. While this is obviously a nice result, we notice a fundamental limitation of such learning algorithms in the situations where a suboptimal NE exists: In these situations, if a learning algorithm reaches a suboptimal NE at *any* point, the learning algorithms may stay in this suboptimal NE (and its equivalent NEs if it has equivalent NEs) for really long time. More precisely, we have the following theorem.

Theorem 2: If a distributed learning algorithm in the class of γ-Logit reaches a (suboptimal) NE, the expected number of iterations needed by the algorithm before leaving this NE and its equivalent NEs is at least $2|\mathcal{D}|e^{\Delta(\mathcal{F})/\tau}$, where $\Delta(\mathcal{F})$ is the minimum non-zero difference of the entire network's total cost between two strategy profiles.

Proof: Let $I_t \in \mathcal{S}$ be the strategy profile that the learning algorithm chooses at iteration t ($t = 0, 1, \ldots$), where $\mathcal{S}$ is the set of all possible different strategy profiles. Suppose at an iteration t_0, the strategy profile I_{t_0} chosen by the algorithm is an NE. Let $S^\star$ be the set of all NEs equivalent to I_{t_0}.

Denote by $\Delta(t)$ the number of iterations that the algorithm would stay in $S^\star$ after iteration t_0. Since the algorithm is not deterministic, we calculate the mathematical expectation of $\Delta(t)$. By definition we have:

$$\begin{aligned} \mathrm{E}[\Delta t] = &\ \textstyle\sum_1^\infty k \times \mathrm{P}(\Delta t = k) \\ = &\ \textstyle\sum_1^\infty k \times \mathrm{P}(I_{t_0}, \ldots, I_{t_0+k-1} \in S^\star, I_{t_0+k} \notin S^\star). \end{aligned} \tag{4}$$

Let $p_k \triangleq \mathrm{P}(I_{t_0+k+1} \notin S^\star | I_{t_0+k} \in S^\star)$. Since the learning algorithm is a Markov chain such that I_{t+1} is decided only by I_t for all $t = 0, 1, \ldots$ and $\mathrm{P}(I_{t_0} \in S^\star) = 1$ here, we have

$$\begin{aligned} &\mathrm{P}(I_{t_0}, \ldots, I_{t_0+k-1} \in S^\star, I_{t_0+k} \notin S^\star) \\ = &\ 1 \times (1-p_0) \times \ldots \times (1-p_{k-2}) \times p_{k-1} \end{aligned} \tag{5}$$

and (4) can be written as

$$\mathrm{E}[\Delta t] = \ p_0 + 2(1-p_0)p_1 + 3(1-p_0)(1-p_1)p_2 + \ldots \tag{6}$$

According to the law of total probability, we have:

$$\begin{aligned} &p_k \\ = &\ \mathrm{Pr}(I_{t_0+k+1} \notin S^\star | I_{t_0+k} \in S^\star) \\ = &\ \textstyle\sum_{s \in S^\star, s' \notin S^\star} \mathrm{Pr}(I_{t_0+k} = s) \mathrm{Pr}(I_{t_0+k+1} = s' | I_{t_0+k} = s) \end{aligned} \tag{7}$$

where $\mathrm{Pr}(I_{t_0+k+1} = s' | I_{t_0+k} = s)$, the probability that the algorithm chooses s' at iteration $t_0 + k + 1$ if it chooses s at iteration $t_0 + k$, can be calculated according to the following rules [4]:

If s and s' differ at only one domain say D_i, i.e. only the gateway selections of domain D_i are different,

$$\mathrm{Pr}(I_{t_0+k+1} = s' | I_{t_0+k} = s) = \frac{1}{|\mathcal{D}|} \frac{1}{c_i} \frac{e^{-\mathcal{U}_i(s')/\tau}}{\gamma(s, s')}$$

where $\gamma(s, s') \geq \max(e^{-\mathcal{U}_i(s)/\tau}, e^{-\mathcal{U}_i(s')/\tau})$. Otherwise,

$$\mathrm{Pr}(I_{t_0+k+1} = s' | I_{t_0+k} = s) = 0.$$

In the first case, since s is an NE and s' is not an equivalent NE, we know $\mathcal{U}_i(s') > \mathcal{U}_i(s)$, thus,

$$\gamma(s, s') \geq \max(e^{-\mathcal{U}_i(s)/\tau}, e^{-\mathcal{U}_i(s')/\tau}) = e^{-\mathcal{U}_i(s)/\tau},$$

and

$$\begin{aligned} \mathrm{Pr}(I_{t_0+k+1} = s' | I_{t_0+k} = s) \leq &\ \tfrac{1}{|\mathcal{D}|} \tfrac{1}{c_i} e^{(\mathcal{U}_i(s) - \mathcal{U}_i(s'))/\tau} \\ = &\ \tfrac{1}{|\mathcal{D}|} \tfrac{1}{c_i} e^{(\mathcal{F}(s) - \mathcal{F}(s'))/\tau}, \end{aligned} \tag{8}$$

where the identity holds because s and s' have the same strategies for all domains except D_i, and thus $F(s) - \mathcal{U}_i(s) = F(s') - \mathcal{U}_i(s')$.

Denote by $\Delta(\mathcal{F})$ the minimum non-zero difference of the entire network's total cost for every two profiles. It is straightforward to see in either case,

$$\mathrm{Pr}(I_{t_0+k+1} = s' | I_{t_0+k} = s) \leq \frac{1}{|\mathcal{D}|} \frac{1}{2} e^{-\Delta(\mathcal{F})/\tau} \tag{9}$$

Let $p(\tau) = \frac{1}{|\mathcal{D}|} \frac{1}{2} e^{-\Delta(\mathcal{F})/\tau}$, it is easy to see $p(\tau) < \frac{1}{2}$. In addition, for all k, s, s' we have

$$p_k \leq \sum_{s \in S^\star, s' \notin S^\star} \mathrm{Pr}(I_{t_0+k} = s) p(\tau) \leq p(\tau). \tag{10}$$

By checking the coefficients of p_k in (6), it is easy to prove that $E[\Delta(t)]$ is a non-increasing function of p_k given $p_k < \frac{1}{2}$ for $k = 0, 1, \ldots$. Thus,

$$\begin{aligned} E[\Delta(t)] \geq &\ p(\tau) + 2(1-p(\tau))p(\tau) + \ldots \\ = &\ 1/p(\tau) \\ = &\ 2|\mathcal{D}|e^{\Delta(\mathcal{F})/\tau}. \end{aligned} \tag{11}$$

■

Remark: According to Theorem 2, if the distributed learning algorithm chooses an NE at some iteration, it would stay in that NE(or the equivalent NEs) for at least $2|\mathcal{D}|e^{\Delta(\mathcal{F})/\tau}$ iterations even if the NE is not optimal. Given that τ must be close to 0 for the convergence of learning, $e^{\Delta(\mathcal{F})/\tau}$ and $E[\Delta(t)]$ could be very large numbers. In other words, the distributed learning algorithm may stay for long time in a suboptimal NE. This stay can be so long that people can hardly wait for the convergence of the learning algorithm to the optimal gateway selection. One may suggest to use the following quick fix: The domains terminate the learning algorithm once it reaches a suboptimal NE, and re-execute it in hope that it will reach an optimal NE. We emphasize that this quick fix does not work, because the domains cannot tell that the learning algorithm is staying in a suboptimal NE. All they can see is that the learning algorithm has stayed in a state for a number of iterations.

V. Optimal Gateway Selection Algorithm for Public Link Cost Case

When the domains have no privacy concern about the link costs, we can develop an algorithm that directly computes the optimal gateway selection from the link costs. The main idea of our algorithm is to transform the gateway selection problem into a Binary Quadratic Programming (BQP) problem and solve it. Note that BQP is NP-hard in general, but standard solvers for BQP exist and are very efficient in practice, although these solvers are not polynomial-time.

A. BQP Problem and Its Solution

In general, a BQP problem is of the form

$$\min\{f(x) = \frac{1}{2}x^{\mathrm{T}}Qx + c^{\mathrm{T}}x : Ax = b, A'x \le b', x \in \{0,1\}^n\}, \tag{12}$$

where $Q \in R^{n\times n}$, $c \in R^{n\times 1}$, $A \in R^{m\times n}$, $b \in R^{m\times 1}$, $A' \in R^{l\times n}$, and $b' \in R^{l\times 1}$.

A solution to the BQP problem above can be obtained through 0-1 linear reformulations or 0-1 convex quadratic reformations. In particular, a general BQP problem can be reformulated to an equivalent BQP problem with a convex objective function (see [18] for an example), and then the problem can be solved accurately using a Mixed Integer Quadratic Programming (MIQP) solver such as CPLEX [19] or TOMLAB [20].

B. Our Proposed Algorithm

In order to transform the problem of optimal gateway selection to a BQP problem, we use a binary variable x_j to indicate whether node j is chosen as a gateway: $x_j = 1$ if node j is, and $x_j = 0$ otherwise. In this way, every joint strategy profile is mapped to a unique binary vector $x = (x_1, \ldots, x_d)^{\mathrm{T}}$. We also add constraints to x to make sure only one node is chosen for each domain:

$$\sum_{j=d'_{i-1}+1}^{d'_i} x_j = 1 \qquad (i = 1, \ldots, |\mathcal{D}|) \tag{13}$$

It is easy to verify that the mapping above is a bijection from all possible joint strategy profiles to all binary vectors satisfying the constraints. In other words, every possible strategy profile has an equivalent binary vector satisfying the constraints, and vice versa. Hence, hereafter we do not distinguish a strategy from its corresponding binary vector.

Denote by $\sigma(j)$ the sum of link costs from node j to all other nodes in the same domain, i.e.

$$\sigma(j) = \sum_{\exists i, d'_{i-1} < j, j' \le d'_i} c(j, j'). \tag{14}$$

Clearly, $\sigma(j)$ is node j's domain's total intra-domain cost when node j is selected as the gateway node.

We construct a cost matrix $B = \{b_{jj'}\}_{d\times d}$ where the value of each entry is defined as follows: The two entries $b_{jj'} = b_{j'j}$ should both cover all the costs between the domains of these two nodes when these two nodes are both selected as gateways. Specifically, we have the following intuitive rules for determining their values.

- If nodes j and j' belong to two different domains, then $b_{jj'} = b_{j'j}$ represents the inter-domain cost between these two domains if both nodes are selected as gateways. It is clear that $b_{jj'} = b_{j'j}$ is just the link cost between these two nodes.
- If nodes j and j' belong to the same domain, then $b_{jj'} = b_{j'j}$ should both cover the intra-domain cost of this domain if both nodes are selected as gateways. However, since only one node can be selected as the gateway for each domain, in this case, $b_{jj'} = b_{j'j}$ has to be 0 if $j \ne j'$. If $j = j'$, then b_{jj} is twice the total intra-domain cost of this domain when node j is selected as the gateway. (It is twice the total intra-domain cost, not just the total intra-domain cost, because a single element b_{jj} represents both $b_{jj'}$ and $b_{j'j}$.)

Formally, we have

$$b_{jj'} = \begin{cases} 2\sigma(j) & \text{if } j' = j \\ 0 & \text{if } j' \ne j \text{ and } \exists i, d'_{i-1} < j, j' \le d'_i \\ c(j,j') & \text{otherwises.} \end{cases} \tag{15}$$

Given a binary vector x satisfying constraints (13), let its equivalent strategy profile be $s^x = \{\widehat{j_1^x}, \ldots, \widehat{j_{|\mathcal{D}|}^x}\}$. We can show (see Theorem 3) that the quadratic function $f(x)$ computes the network's total cost on gateway selection profile s^x, i.e.,

$$f(x) = \frac{1}{2}x^{\mathrm{T}}Bx = \mathcal{F}(s^x). \tag{16}$$

Therefore, the optimal NE can be found by solving the following BQP problem.

$$\begin{array}{lll} \textbf{Min} & : & f(x) = \frac{1}{2}x^{\mathrm{T}}Bx \\ \textbf{subject to} & : & \sum_{j=d'_{i-1}+1}^{d'_i} x_j = 1 \quad (1 \le i \le |\mathcal{D}|) \\ & & x_j \in \{0,1\} \qquad\quad (1 \le j \le d) \end{array} \tag{17}$$

Hereafter, we use Algorithm I to refer to this algorithm.

Theorem 3: Algorithm I is correct, i.e., the solution to the BQP problem (17) provides the optimal gateway selection.

Proof: Algorithm I's correctness follows from the equation (16)'s correctness. For every binary vector x satisfying the constraints, we have

$$\begin{aligned} f(x) =& \frac{1}{2}x^{\mathrm{T}}B\ \ x \\ =& \frac{1}{2}\sum_{1\le j=j'\le d} x_j b_{jj'} x_{j'} + \frac{1}{2}\sum_{1\le j\ne j'\le d} x_j b_{jj'} x_{j'} \\ =& \frac{1}{2}\sum_{x_j=1}^{1\le j=j'\le d} x_j b_{jj'} x_{j'} + \frac{1}{2}\sum_{x_j=1, x_{j'}=1}^{1\le j\ne j'\le d} x_j b_{jj'} x_{j'} \\ =& \frac{1}{2}\sum_{1\le i\le |\mathcal{D}|} 2\sigma(\widehat{j_i^x}) + \frac{1}{2}\sum_{i\ne i'}^{1\le i,i'\le |\mathcal{D}|} c(\widehat{j_i^x}, \widehat{j_{i'}^x}) \\ =& \sum_{1\le i\le |\mathcal{D}|} \sum_{j_i\ne \widehat{j_i^x}}^{d'_{i-1}<j_i\le d'_i} c(\widehat{j_i^x}, j_i) + \sum_{i\ne i'}^{1\le i,i'\le |\mathcal{D}|} c(\widehat{j_i^x}, \widehat{j_{i'}^x})/2 \\ =& \mathcal{F}(s^x). \end{aligned}$$

Given the mapping relation between all possible strategy profiles and all binary vectors satisfying the constraints, $f(x)$'s minimizer x also provides the optimal gateway profile s^x. ■

VI. Cryptographic Protocol for Two Domains with Private Link Costs

Algorithm I can be used to find the optimal gateway selection when the domains have no privacy concern about their link costs. However, when domains are unwilling to reveal their link costs, a different solution is needed. In this section, we assume there are two domains that are unwilling to reveal their link costs. We design a cryptographic protocol that allows these two domains to find the optimal gateway selection while protecting the privacy of each domain.

Specifically, suppose the numbers of nodes in these two domains are m and n, respectively. It is easy to see that, in total there are a limited number (mn) of possible gateway selections only. Consequently, we can first compute the total cost for each possible pair of gateways in a privacy preserving way, and then compare these costs in a privacy preserving way, so that we can find the pair that minimizes the total cost.

A. Cryptographic Tools

In order to compute and compare the costs in a privacy preserving manner, we use a cryptosystem that supports *rerandomization* and has the *additively homomorphic property*. Here rerandomization refers to an operation that takes a ciphertext as input and outputs a new ciphertext of the same plaintext. The additively homomorphic property allows one to compute the ciphertext of two numbers' sum from the ciphertexts of these two numbers, without decrypting either number. Two cryptosystems that support rerandomization and have the additively homomorphic property are a variant of ElGamal cryptosystem [21] and the Paillier cryptosystem [22].

In order to compare the costs for possible gateway pairs in a privacy preserving way, we use a *secure comparison* protocol. Here a secure comparison protocol allows two parties to decide which of them has a greater input without revealing their inputs to each other. In the literature, a number of secure comparison protocols have been proposed (see, for example, [23]–[25]). These protocols have been proved to provide cryptographically strong guarantee of privacy.

In our cryptographic protocol, what we use is actually a slightly modified version of secure comparison, in which each party has an *encrypted input* for secure comparison. Each party cannot decrypt by itself. Hence, while the cleartexts are actually compared, both parties see only ciphertexts, not cleartexts.

At the end of our cryptographic protocol, decryption is needed in a privacy preserving manner. For decryption, the two domains have to jointly execute a protocol of $(2, 2)$-*threshold decryption* [22], [26]. This $(2, 2)$-threshold decryption protocol requires $(2, 2)$-secret sharing of the private key. Decryption of any ciphertext can be made only when both domains agree.

B. Protocol Design

Using the above cryptographic tools, we can now design our cryptographic protocol to compute the optimal gateway selection.

First, one of the two domains, D_1, encrypts its total intra-domain cost for its each possible gateway. D_1 sends all these encrypted intra-domain costs to the other domain D_2.

Then, for each possible gateway pair, D_2 computes the ciphertext of the network's total cost using the additively homomorphic property. For example, for possible gateway pair (j, j') $(1 \le j \le m, m+1 \le j' \le m+n)$, D_2 computes $2\mathcal{F}(j, j') = \sigma(j) + \sigma(j') + 2c(j, j')$ using the additively homomorphic property. To be precise, this is actually a ciphertext of *twice* the total cost $2\mathcal{F}(j, j')$, not a ciphertext of the total cost $\mathcal{F}(j, j')$. However, comparing two times the total costs is equivalent to comparing the total costs. Hence, it suffices to compute a ciphertext of $2\mathcal{F}(j, j')$ in this step.

Next, the two domains jointly and randomly permute the ciphertexts of all these possible (twice) total costs for the entire network. In order to guarantee that neither domain learns the permutation of these ciphertexts, rerandomization of these ciphertexts is needed.

Finally, the two domains use a secure comparison protocol to compare the network's (twice) total costs and find the gateway pair that minimizes the total cost.

We summarize our cryptographic protocol in Algorithm II. In this protocol, $E(x)$ denotes the ciphertext of integer x; the secure comparison protocol takes the ciphertexts of two numbers x and y as inputs, outputting 1 if $x > y$, and 0 otherwise.

Algorithm II

Setup: A public key pk and two shares of the corresponding private key, pr_1 and pr_2, are generated using Shamir's secret sharing for $(2, 2)$-threshold decryption such that domain D_1 (D_2, resp.) knows pk and pr_1 (pr_2, resp.) but not the other share of private key. (This can be achieved using either a trusted authority or a secure protocol like [22].)

Stage 1:

D_1 uses pk to encrypt its intra-domain costs and then sends the ciphertexts $E(\sigma(1))$, $E(\sigma(2))$,..., $E(\sigma(m))$ to D_2.

for: $j = 1, \ldots, m$

for: $j' = m+1, \ldots, m+n$

D_2 computes ciphertext pair $(E(2\mathcal{F}(j, j')), E(j(m+n)+j'))$. The first element is used to do comparisons, while the second is used to keep track of the corresponding indices.

end for

end for

D_2 randomly permutes these ciphertext pairs and sends them to D_1. D_1 rerandomizes every pair, randomly permutes them, and sends them back to D_2.

Stage 2:

D_2 and D_1 jointly execute secure comparison protocol for $nm-1$ times to find the ciphertext of the minimum total cost.

Let $E(j(m+n)+j')$ be the corresponding tracking ciphertext. D_1 and D_2 jointly execute secure threshold decryption protocol to decrypt it and get $j(m+n)+j'$, thus j and j'.

C. Correctness and security analysis

It is straightforward to get the following result on the correctness of our proposed cryptographic protocol.

Theorem 4: Algorithm II is correct, i.e., if both domains follow the protocol, the output is the optimal gateway selection.

In addition to being correct, our cryptographic protocol is also privacy preserving in a standard security model called *semi-honest model* (see [27] for more about this model).

Theorem 5: In the semi-honest model, Algorithm II is privacy preserving, i.e. it reveals nothing more than the final gateway selection result to both two domains.

Proof: To prove this theory, we construct a simulator for each party such that, given this party's every possible input and the algorithm's final output, it can simulate a view that is *computationally indistinguishable* [27] from this party's view when participating in the algorithm.

In particular, we construct $\mathcal{S}_1$ for D_1 as follows. The simulator for D_2 can be constructed in a similar way.

Denote by w_1 and w_2 the private input from D_1 and D_2 respectively. Denote by $gs(w_1, w_2)$ Algorithm II's output. On input $(w_1, gs(w_1, w_2))$, $\mathcal{S}_1$ simulates the coin flips of D_1 as described in the algorithm.

$\mathcal{S}_1$ simulates every permutated ciphertext pair received from D_2 using two random encryptions of two random cleartexts. Then $\mathcal{S}_1$ simulates the sequence of secure comparison results by randomly picking $mn-1$ numbers in $\{0,1\}$ and the random numbers or ciphertexts received during the secure comparisons using the same amount of random numbers or random encryptions of random cleartexts. Finally, $\mathcal{S}_1$ simulates the random numbers or ciphertexts during the threshold decryption using the same amount of random numbers or random encryptions of random cleartexts.

The computational indistinguishability follows the semantic security of the cryptosystem, the randomness of the permutation operations, and the security of the secure comparison protocol and the threshold decryption algorithm. ■

Note that Algorithm II requires both parties to exchange encrypted messages for several times, this introduces additional inter-domain communication overhead to the negotiation phase of the gateway selection. To be complete, below we examine the communication overhead of our algorithm.

Let q be the bit length of the ciphertext, and p be the bit length of the plaintext. In Stage 1 of Algorithm II, both parties jointly compute the ciphertexts of the mn possible total costs. This requires $m + 4mn$ ciphertexts to be sent between two parties. In Stage 2, both parties need to jointly run a secure comparison protocol for $mn-1$ times and then jointly execute a secure threshold decryption protocol once. In this paper, we implement our algorithm II based on the secure comparison protocol in [25] and the threshold cipher in [22]. By carefully examining these protocols, we know it requires $15p + 25$ ciphertexts and 1 bit (the comparison result) to be sent in one run of the secure comparison protocol and 1 ciphertext and 1 plaintext to be sent in one run of the threshold decryption. Thus, the Stage 2 requires $(mn-1)(15p+25)+1$ ciphertexts, 1 plaintext and $mn-1$ bits to be sent. In total, our Algorithm II requires $(mn-1)(15p+25)+1+m+4mn$ ciphertexts, 1 plaintext and $mn-1$ bits to be sent. The communication overhead is $15mnp$ ciphertexts approximately which is equal to $15mnpq$ bits. If we use a Paillier cryptosystem where $p = 256$ and $q = 512$ as the underlying cipher, the average communication overhead would be less than 256 KB per possible gateway node pair.

VII. Protocol for Three or More Domains with Private Link Costs

If there are more than two domains in the network, using a cryptographic protocol to compute the optimal gateway selection may require high computational overheads. Hence, we propose a perturbation-based protocol for this case, which has low computational overheads.

A. Protocol Design

Our perturbation-based protocol computes the optimal gateway selection in a way similar to Algorithm I. However, unlike Algorithm I which assumes the link costs are all public information, this protocol requires the involved domains to exchange information about their private link costs. To protect the privacy of domains, this protocol lets domains add noises to their private link costs before revealing these link costs to other domains. Therefore, during the entire protocol, domains can see only the perturbed values of other domains' private link costs, not the true values. On the other hand, we make sure that the computed optimal gateway selection is not affected by the perturbation, remaining correct.

Without loss of generality, suppose that domain $D_{|\mathcal{D}|}$ is selected to compute the optimal gateway selection. Hence, other domains need to reveal (perturbed values of) their private link costs to domain $D_{|\mathcal{D}|}$. We can view these link costs as parts of the matrix B defined in Section V-B.

Consider the sub-matrix consisting of the $(d'_{i-1}+1)$st, $(d'_{i-1}+2)$nd, ..., d'_ith rows of B. Generally this sub-matrix can be divided into three parts: (P1) one $d_i \times d_i$ matrix whose diagonal elements are domain D_i's intra-domain cost $\sigma(j_i)$ and other elements are zero; (P2) some $d_i \times d_{i'}$ matrices $(i' \neq i)$ that contain the one-hop link costs between every node in D_i and every node in $D_{i'}$; (P3) one $d_i \times d_{|\mathcal{D}|}$ matrix that contains the one-hop link cost between every node in D_i and every node in $D_{|\mathcal{D}|}$. Among these three parts, part (P3) is known by domain $D_{|\mathcal{D}|}$ itself, but (perturbed values of) parts (P1) and (P2) need to be revealed by other domains to domain $D_{|\mathcal{D}|}$. To protect the private values in these two parts, the protocol adds random noises to these values. These noises are generated in such a way that the noises added to part (P1) are cancelled with the noises added to part (P2) when we compute the optimal gateway selection.

When the link costs are perturbed as described above, domain $D_{|\mathcal{D}|}$ actually gets a perturbed matrix $\widetilde{B}$ as opposed to B in Algorithm I. Although $\widetilde{B} \neq B$, it is guaranteed (see Theorem 6) that $x^{\mathrm{T}} B\ x = x^{\mathrm{T}} \widetilde{B}\ x$ always holds for all valid gateway selection x. Consequently, the protocol computes the same optimal gateway selection as Algorithm I.

Our perturbation-based protocol is summarized in Algorithm III.

Algorithm III:

Stage 1:

for $i = 1, \ldots, |\mathcal{D}| - 1$ ($|\mathcal{D}| \geq 3$), domain D_i does the following steps:

for every node j in domain D_i and every $i' \in \{1, \ldots, |\mathcal{D}|\} - \{i\}$, domain D_i first generates a random number $r^i_{ji'} \in [-\mu, \mu]$ (Recall μ is the upper bound of the one-hop link cost value). Set $r^i_{j|\mathcal{D}|} = 0$.

for every node j' in domain $D_{i'}$, domain D_i calculates $\widetilde{c}(j, j') = c(j, j') + r^i_{ji'}$ as the perturbed inter-domain link cost from node j to node j'.

end for

Let $r^i_j = \sum_{1 \leq i' \neq i \leq |\mathcal{D}|} r^i_{ji'}$. D_i calculates $\widetilde{\sigma}(j) = \sigma(j) - r^i_j$.

Domain D_i computes $\widetilde{B}_j$, the jth row of the perturbed cost matrix $\widetilde{B}$, where the j'th element is

$$\widetilde{b}_{jj'} = \begin{cases} \widetilde{\sigma}(j) & \text{if } j' = j \\ 0 & \text{nodes } j' \neq j \text{ are in the same domain} \\ \widetilde{c}(j, j') & \text{otherwises.} \end{cases}$$

Domain D_i sends the row to domain $D_{|\mathcal{D}|}$.

end for

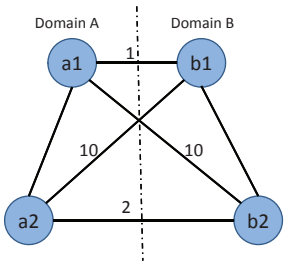

Fig. 1. An example network from [4].

end for

After receiving all $\widetilde{B}_j$ $(1 \leq j \leq d'_{|\mathcal{D}|-1})$, $D_{|\mathcal{D}|}$ sets $\widetilde{B}_k = B_k$ $(d'_{|\mathcal{D}|-1} < k \leq d)$.

Stage 2:

Domain $D_{|\mathcal{D}|}$ uses a BQP solver to find x that minimizes $\widetilde{f}(x) = x^{\mathrm{T}}\widetilde{B}\ x$ under the same constraints as problem (17).

Remark: In the protocol above, to cancel the noise that a domain adds in its intra-domain link cost, it needs to add proper noise to its inter-domain link cost to other domains except the domain who is responsible for collecting noised data and performing the computation. Thus, there has to be at least three domains involved. In other words, Algorithm III only works when there are three or more domains.

B. Correctness and security analysis

Given the above perturbation-based protocol, we first show its correctness.

Theorem 6: Algorithm III is correct, i.e., if all domains follow the protocol, the output is the optimal gateway selection.

Proof: Similar to the proof for Theorem 3, we only need to prove that function $\widetilde{f}(x)$ computes the total cost of the entire network on x's corresponding gateway selection profile s^x. Notice that the change of the inter-domain cost's sum is cancelled out by the change of the intra-domain cost's sum for all domains when the cost matrix is changed from B to $\widetilde{B}$. Thus $\widetilde{f}(x)$ computes the total cost as $f(x)$ does. ■

Next, we analyze the privacy protection of our perturbation-based protocol.

For each pair (i, i') $(1 \leq i \neq i' < |\mathcal{D}|)$, D_i and $D_{i'}$'s inter-domain cost $c_{jj'}$ is disguised by a random noise $r^i_{ji'}$ to prevent domain $D_{|\mathcal{D}|}$ from learning it. For each i $(1 \leq i < |\mathcal{D}|)$, every intra-domain cost $c_{jj'}$ in D_i is firstly aggregated into $\delta(j)$. Then $\delta(j)$ is disguised by the sum of $|\mathcal{D}| - 2$ random noises $r^i_{ji'}$ $(1 \leq i' \neq i < |\mathcal{D}|)$ to prevent domain $D_{|\mathcal{D}|}$ from learning it.

Among all domains, $D_{|\mathcal{D}|}$ clearly knows the most. So it suffices to analyze the possible attacks by $D_{|\mathcal{D}|}$. If $D_{|\mathcal{D}|}$ chooses to guess an inter-domain cost, say $c_{jj'}$, all she knows is that $c_{jj'}$ belongs to an interval $[0, \mu] \cap [\widetilde{c}_{jj'} - \mu, \widetilde{c}_{jj'} + \mu]$. Moreover, it can be proved that the mathematical expectation of the interval's length is no less than $\frac{2}{3}\mu$. If $D_{|\mathcal{D}|}$ chooses to guess an intra-domain cost of a domain, say $\sigma(j)$ where $d'_{i-1} < j \leq d'_i$, similarly all she knows is that $\sigma(j)$ belongs to an interval $[0, (d_i - 1)\mu] \cap [\widetilde{\sigma}(j) - \mu(|\mathcal{D}|-2), \widetilde{\sigma}(j) + \mu(|\mathcal{D}|-2)]$. Furthermore, the mathematical expectation of the interval's length is no less than $\frac{2(|\mathcal{D}|-2)}{3}\mu$ assuming $d_i \geq |\mathcal{D}| - 1$.

TABLE I
500 RUNS OF MAX-LOGIT ON THE EXAMPLE NETWORK.

First NE Reached	Number of Runs	Percentage
(a_2, b_2)	187	37.4%
(a_1, b_1)	313	62.6%

VIII. EVALUATIONS

We conduct experiments to empirically study the problem of optimal gateway selection and its solutions. Our experiments can be divided into two parts.

A. Limitation of Solution based on Distributed Learning

To illustrate the limitation of the existing solution based on distributed learning, we conduct experiments using an example network from [4]. This example network is shown in Figure 1. Clearly, the gateway selection game has two NEs: (a_1, b_1) (the optimal NE) and (a_2, b_2) (the suboptimal NE).

The limitation of the existing solution based on distributed learning is that it may stay at a suboptimal NE for too long. To demonstrate this limitation, we execute the Max-logit algorithm from [4] on the example network and measure:

- the probability that Max-logit reaches a suboptimal NE;
- the time spent on a suboptimal NE by Max-logit.

To make our simulation accurate, we use the NTL library [28], a library that supports computations on arbitrary length integers and arbitrary precision float point numbers, to implement the exponentiation calculations and sampling operations in Max-logit algorithm. We set the smoothing factor τ to 0.0001 as in the experiments of [4].

Table I shows the results on the probability that Max-logit reaches a suboptimal NE. Among 500 runs of Max-logit, 187 runs (37.4%) reach the suboptimal NE before converging to the optimal NE. Consequently, there is a *significant* chance for Max-logit to reach a suboptimal NE.

Once Max-logit reaches a suboptimal NE, how long will it stay there? In our experiment, we let Max-logit reach a suboptimal NE and then wait for 172800 seconds (2 days). By the end of the experiment, Max-logit still stays in the suboptimal NE. Consequently, once Max-logit reaches a suboptimal NE, it can stay there for *extremely long*.

B. Efficiency Measurement and Comparison

To measure the efficiency of our solutions and compare them with the existing solution based on distributed learning, we use two randomly generated networks with suboptimal NEs: network A (shown in Figure 2) with three domains, and network B (shown in Figure 3) with two domains. In these two networks, the centers of domains are picked from a 1000×1000 m^2 square area. The nodes of each domain are deployed in a round area within $125\ m$ from the center. We use the Euclidean distance as the link cost and set η, the maximum link cost between any two nodes, to $500\ m$.

Table II shows the CPU time used by our Algorithm I and Max-logit algorithm to find the optimal gateway selection

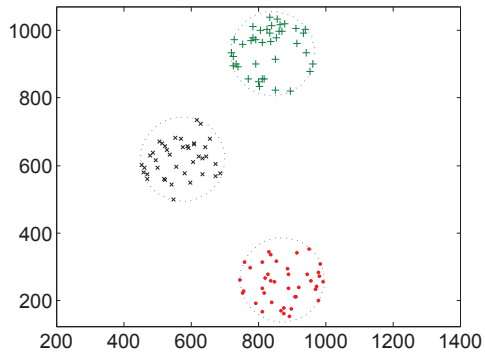

Fig. 2. Network A (40 nodes × 3 domains).

TABLE II
CPU TIME TO FIND THE OPTIMAL NE IN NETWORK A: ALGORITHM I VS MAX-LOGIT

Algorithm	CPU time (sec)				
Algorithm I	24	24	24	24	24
Max-logit	>3600	>3600	2+T_c	>3600	22+T_c
Algorithm I	24	24	24	24	24
Max-logit	5+T_c	24+T_c	11+T_c	8+T_c	>3600

TABLE III
CPU TIME TO FIND THE OPTIMAL NE IN NETWORK B: ALGORITHM II VS MAX-LOGIT

Algorithm	CPU time (sec)				
Algorithm II	322	324	319	320	320
Max-logit	>3600	2+T_c	>3600	>3600	>3600
Algorithm II	321	320	322	322	320
Max-logit	7+T_c	>3600	>3600	>3600	>3600

in network A during 10 runs. From the table, we can see that Algorithm I finds the optimal gateway selection within 24 seconds in all the 10 runs. In comparison, Max-logit algorithm's efficiency varies. In 4 out of 10 runs, Max-logit has not found the optimal NE within 3600 seconds. In the other 6 runs, Max-logit uses $2 \sim 24$ seconds to *reach* the optimal NE. After that, Max-logit requires time T_c to decide that it has indeed converged to the optimal NE and thus should terminate. Here the value of T_c depends on how many iterations with identical states are considered "convergence" by Max-logit. The more iterations are required by Max-logit (i.e., the higher standard Max-logit has for convergence judgement), the larger T_c is. We emphasize that this number of iterations should *not* be small, because otherwise, Max-logit may mistakenly recognize another state (such as a suboptimal NE) as optimal NE and decide to terminate in that state. Consequently, even in these 6 runs, Max-logit can still be slower than Algorithm I.

Table III shows the CPU time used by our Algorithm II and Max-logit algorithm to find the optimal gateway selection in network B (see Fig 3) during 10 runs. (The communications time costs of our Algorithms II and III are much less than the corresponding computation costs. So we focus on CPU time here.) In all these 10 runs, our Algorithm II finds the optimal gateway selection within 324 seconds. In contrast, in 8 out of these 10 runs, Max-logit algorithm has not found the optimal gateway selection within 3600 seconds.

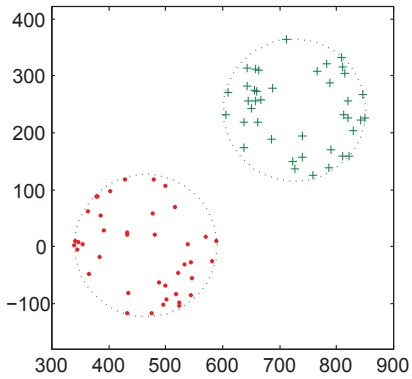

Fig. 3. Network B (40 nodes × 2 domains).

TABLE IV
CPU TIME TO FIND THE OPTIMAL NE IN NETWORK A: ALGORITHM III VS MAX-LOGIT

Algorithm	CPU time (sec)				
Algorithm III	24	24	24	24	24
Max-logit	>3600	>3600	2+T_c	>3600	22+T_c
Algorithm III	24	24	24	24	24
Max-logit	5+T_c	24+T_c	11+T_c	8+T_c	>3600

Table IV shows the CPU time used by our Algorithm III and Max-logit algorithm to find the optimal gateway selection in network A. Algorithm III successfully find the optimal gateway selection in all 10 runs, with efficiency similar to Algorithm I.

From the results of the above experiments, we can see that our proposed solutions are superior to the existing solution for the situations in which suboptimal NEs exist. In these situations, the existing solution based on distributed learning may stay in suboptimal NEs for very long time. Furthermore, even in the runs that the existing solution reaches the optimal NE quickly, the judgment of convergence still takes time and so the existing solution can still be slower than our proposed solutions.

In comparison, all of our solutions have very stable time costs across all runs: Our data shows that Algorithms I and III both have good efficiency. Our proposed Algorithm II, which is a cryptographic protocol, needs more time than Algorithms I and III. This additional time can be understood as the cost of having cryptographically strong privacy guarantee. Since gateway selection does not have very tight time constraints, the efficiency of Algorithm II is also acceptable.

Figure 4 shows the minimum total cost found in network A by Max-logit, our Algorithm I and Algorithm III given a limit amount of time. For Max-logit, we randomly pick one run in which the algorithm finds the optimal NE and one run in which it enters a sub-optimal NE and stays there (for a long time). For Algorithm I and Algorithm III, we randomly pick one run. Since our Algorithm I and Algorithm III behave exactly the same in this comparison, we use one line to represent them both in the figure. From the figure, we can see Max-logit converges to a NE in less than 7 seconds in both runs. Different from Max-logit which needs several seconds to converge, both our Algorithm I and Algorithm III find the optimal total cost in less than 1 second, thus are much faster. Although, the

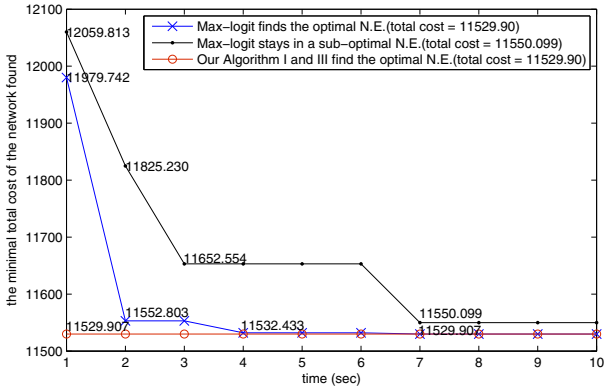

Fig. 4. A comparison on the minimum total cost found by Max-logit, our Algorithm I and Algorithm III given a limit amount of time.

difference between the optimal NE and sub-optimal NE is 20 which is not much here (the differences is determined by the network topology.).

Our Algorithm II is different from Max-logit, Algorithm I and III. Algorithm II traverses the gateway solutions in a random order and compares the solutions one by one. Accordingly, the minimal total cost found given a limited amount of time is determined by the random order, and is also random. Thus, we skip the comparison between our algorithm II and the Max-logit in the network B.

IX. Enhancements for Dynamic Link Costs and Load Balance

In this section, we discuss how to adapt our solutions to deal with dynamic link cost (the link quality changes.), and how to enable load balance functionality to avoid overloading a single gateway when the traffic is heavy.

A. To deal with dynamic link costs

Sometimes, the link cost matrix is not static, but rather dynamic (i.e. the link cost value changes over time.). To deal with link cost changes, an intuitive idea would be to re-run our solutions when some domain notices the change or after a certain time interval. However, starting to run our solutions from the scratch every time is not very efficient. Fortunately, the CPLEX solver used in our solutions for cases I and III allows users to modify the original problem and compute results to the new problem based on the running data of the original problem (please see the Chapter of Modifying a model in [19] for more details), which can help improve the efficiency significantly. To improve the efficiency of our solution based on Secure Comparison protocol for case II, the two domains can simply re-compute the overall cost of gateway node pairs containing nodes incident to links with changed costs, and then use the overall cost's ciphertexts previously computed for other gateway node pairs in Stage 1. In addition, the comparison job in Stage 2 can also be divided into two halves and distributed to the two domains. Each domain will be responsible for searching a local minimal in half of all possible gateway node pairs (note that a domain still needs the other domain to help it to run the secure comparison protocol, but the helper's workload is much less.). In the end the two domains can compare their local minimal overall cost with each other to find the global minimal overall cost.

B. To enable load balance functionality

When the traffic becomes heavy, a single gateway for each domain could be overloaded. In this case, domains might want to distribute the traffic among multiple gateway nodes. Note that our solutions take the cost matrix B defined in Section V-B as the input. To implement this functionality, domains could apply our solutions with an adapted cost matrix B_{adt}. The main idea of the adaption is to represent each possible multi-gateway combination as a single "combined node", use the mathematical expectation of link costs associated with the combined node as the inter-domain or intra-domain link costs in B_{adt}, and use B_{adt} to evaluate the overall cost. For instance, consider a multi-gateway combination of two nodes i and i' in a domain D, D uses one combined node i'' to represent this combination. The total intra-domain cost of i'' or gateway-nodes (i, i') can be computed as $\Sigma_{j\in D}(P_{ij}c_{ij} + P_{i'j}c_{i'j})$ where P_{ij} (and $P_{i'j}$ resp.) is the probability that node j uses node i (and i' resp.) as the gateway node to send/receive messages to/from other domains. The values of P_{ij} and $P_{i'j}$ can be retrieved from historical experience of the traffic or can be assigned by the domain owner based on some intra-domain gateway selection policy such as shortest-distance or random selection. Similarly, the inter-domain cost to a node j' in domain D' can be computed as $(P_{ij'}c_{ij'} + P_{i'j'}c_{i'j'})$ where $P_{ij'}$ (and $P_{i'j'}$ resp.) is the probability that domain D chooses node i (and i' resp.) as the gateway node to send/receive messages to a particular node j' in D'. With the adapted cost matrix, our solutions can evaluate the overall cost of a gateway selection that allows multi-gateway nodes to be selected by a domain and find the optimal gateway selection.

X. Conclusion

In this paper, we study the optimal gateway selection problem in multi-domain wireless networks. We notice an important limitation of the existing solution based on distributed learning and therefore propose three alternative solutions to the problem. Our alternative solutions always terminate within reasonable amounts of time even in situations with suboptimal NEs, although we emphasize that our solutions are NOT improvements to the gateway selection solutions based on distributed learning.

References

[1] F. Li, Y. Wang, and X.-Y. Li, "Gateway placement for throughput optimization in wireless mesh networks," in *Proc. 2007 IEEE ICC*, pp. 4955–4960.

[2] X. Xu, S. Tang, X. Mao, and X.-Y. Li, "Distributed gateway placement for cost minimization in wireless mesh networks," in *Proc. 2010 IEEE ICDCS*, pp. 507–515.

[3] T. He, K.-W. Lee, N. Sofra, and K. K. Leung, "Utility-based gateway deployment for supporting multi-domain dtns," in *Proc. 2010 IEEE SECON*, pp. 341–349.

[4] Y. Song, S. H. Y. Wong, and K.-W. Lee, "Optimal gateway selection in multi-domain wireless networks: a potential game perspective," in *Proc. 2011 ACM MOBICOM*, P. Ramanathan, T. Nandagopal, and B. N. Levine, Eds., pp. 325–336.

[5] C.-K. Chau, J. Crowcroft, K.-W. Lee, and S. H. Wong, "Inter-domain routing for mobile ad hoc networks," in *Proc. 2008 ACM International Workshop on Mobility in the Evolving Internet Architecture*, pp. 61–66. Available: http://doi.acm.org.gate.lib.buffalo.edu/10.1145/1403007.1403022
[6] U. Ashraf, S. Abdellatif, and G. Juanole, "Gateway selection in backbone wireless mesh networks," in *Proc. 2009 IEEE WCNC*, pp. 2548–2553.
[7] Y. Zhao, J. Wu, F. Li, and S. Lu, "VBS: maximum lifetime sleep scheduling for wireless sensor networks using virtual backbones," in *Proc. 2010 IEEE INFOCOM*, pp. 1–5.
[8] A. Durresi, M. Durresi, and L. Barolli, "Heterogeneous multi domain network architecture for military communications," in *Proc. 2009 IEEE CISIS*, L. Barolli, F. Xhafa, and H.-H. Hsu, Eds., pp. 382–387.
[9] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proc. 2000 ACM SIGMOD Conference*, W. Chen, J. F. Naughton, and P. A. Bernstein, Eds., pp. 439–450.
[10] W. Du and M. J. Atallah, "Privacy-preserving cooperative statistical analysis," in *Proc. 2001 IEEE ACSAC*, pp. 102–110.
[11] D. Agrawal and C. C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," in *Proc. 2001 ACM PODS*, P. Buneman, Ed.
[12] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "On the privacy preserving properties of random data perturbation techniques," in *Proc. 2003 IEEE ICDM*, pp. 99–106.
[13] V. Estivill-Castro and L. Brankovic, "Data swapping: balancing privacy against precision in mining for logic rules," in *Proc. 1999 Springer DaWaK*, ser. Lecture Notes in Computer Science, M. K. Mohania and A. M. Tjoa, Eds., vol. 1676, pp. 389–398.
[14] E. Royer and C. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Pers. Commun.*, vol. 6, no. 2, pp. 46–55, 1999.
[15] Q. Liu, M. A. Kok, N. Ghani, V. M. Muthalaly, and M. Wang, "Hierarchical inter-domain routing in optical dwdm networks," in *Proc. 2006 IEEE INFOCOM*..
[16] C. Chau, J. Crowcroft, K. Lee, and S. Wong, "Inter-domain routing for mobile ad hoc networks," in *Proc. 2008 ACM International Workshop on Mobility in the Evolving Internet Architecture*, pp. 61–66.
[17] M. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, 1994.
[18] A. Billionnet, S. Elloumi, and M. Plateau, "Convex quadratic programming for exact solution of 0-1 quadratic programs," *RAIRO-Operations Research*, 2005.
[19] I. IBM ILOG. (2009) IBM ILOG CPLEX v12.1 User's Manual for CPLEX. Available: ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps$_$usrmancplex.pdf
[20] K. Holmström, "The tomlab optimization environment in Matlab," 1999.
[21] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," in *Proc. 1997 Springer EUROCRYPT*, ser. Lecture Notes in Computer Science, W. Fumy, Ed., vol. 1233, pp. 103–118.
[22] I. Damgård and M. Jurik, "Efficient protocols based on probabilistic encryption using composite degree residue classes," *IACR Cryptology ePrint Archive*, vol. 2000, p. 8, 2000.
[23] A. C.-C. Yao, "Protocols for secure computations (extended abstract)," in *Proc. 1982 IEEE FOCS*, pp. 160–164.
[24] I. Damgård, M. Geisler, and M. Krøigaard, "Homomorphic encryption and secure comparison," *IJACT*, vol. 1, no. 1, pp. 22–31, 2008.
[25] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *Proc. 2007 Springer Public Key Cryptography*, ser. Lecture Notes in Computer Science, T. Okamoto and X. Wang, Eds., vol. 4450, pp. 343–360.
[26] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Proc. 1989 Springer CRYPTO*, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435, pp. 307–315.
[27] O. Goldreich, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
[28] V. Shoup, "NTL: a library for doing number theory, 2005." Available: http://www. shoup. net/ntl/index. html. Release, vol. 5.

Sheng Zhong received his BS (1996), MS (1999) from Nanjing University, and Ph.D. (2004) from Yale University, all in computer science. He is interested in security, privacy, and economic incentives.

Yuan Zhang received his BS (2005) in Automation from Tianjin University, MS (2009) in Software Engineering from Tsinghua University, and Ph.D. (2013) in Computer Science from State University of New York at Buffalo. He is interested in security, privacy, and economic incentives.