Computer Networks 73 (2014) 112–127

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Multi-AS cooperative incoming traffic engineering in a transit-edge separate internet

Yaodong Zhang, Yue Wang, Dan Pei *, Jian Yuan

Tsinghua University, Beijing, China

ARTICLE INFO

Article history:
Received 24 January 2014
Received in revised form 28 July 2014
Accepted 31 July 2014
Available online 19 August 2014

Keywords:
Incoming traffic engineering
Edge ASes
Cooperative game
LISP

ABSTRACT

Internet is composed of independent autonomous systems (ASes) which are selfish by nature. In the current inter-domain routing system, for an AS, incoming traffic is harder to control than outgoing traffic because its incoming traffic routing is ultimately determined by other ASes. This problem stays the same in the leading new routing architecture proposals (such as transit edge separation protocols) when addressing scalability challenges. In this paper, we use LISP as an example to study cooperative incoming traffic engineering in a selfish multi-AS context in such proposals. First, we analyze conflicted egress tunnel routers (ETRs) selections of edge ASes, which choose ETRs that are best for their individual delay performance, but neglect incoming traffic engineering performance of other edge ASes. Then, we propose the cooperative incoming traffic engineering, where edge ASes sacrifice limited delay performance for optimizing incoming traffic engineering performance between each other. Furthermore, we measure the delay variations between different RLOCs with experiments on PlanetLab to verify the rationality of considering the impact of ETR selections on delay performance. Finally we evaluate the performance of cooperative incoming traffic engineering. The proposed method achieves Pareto optimality and can greatly improve the robustness and resiliency of ASes based on simulations.

1. Introduction

Internet is composed of more than 45,000 autonomous systems (ASes). About 84% are edge ASes [1], i.e., they appear only as source ASes or destination ASes, first or last in routing table's AS paths. The other ASes are transit ASes. Edge ASes provide network connections for clients, content providers and data centers while transit ASes transit packets between different edge ASes. Most of the edge ASes are multi-homed [1], that is, typically, each edge AS has more than one path connecting it to the rest of the Internet.

Standard inter-domain traffic engineering (TE) consists of outgoing traffic engineering and incoming traffic engineering. For a specific edge AS, outgoing traffic engineering is relatively easy to optimize because routing outgoing traffic is decided by the AS itself. On the contrary, incoming traffic engineering is difficult to optimize because routing incoming traffic is decided by other ASes in the Internet. For an edge AS, the ideal incoming traffic engineering is that the incoming traffic is uniformly distributed over its entrance paths (edge routers) so that its link bandwidth resources can be best utilized and the probability of congestion caused by traffic fluctuations can be reduced to the minimum. This is especially important for edge ASes, because incoming traffic is uncontrollable and is highly correlated with Quality of Service (QoS) of content consumers [2,3] inside the edge ASes.

In the current network architecture, the above goal is expected to be accomplished by inter-domain routing

* Corresponding author. Tel.: +86 (010) 62792837.
E-mail address: peidan@tsinghua.edu.cn (D. Pei).

http://dx.doi.org/10.1016/j.comnet.2014.07.015
1389-1286/

protocol BGP. In BGP, an AS can selectively advertise its prefixes, which is a coarse-grained method to control its incoming traffic. Furthermore, an AS also can modify relevant parameters to show its preferred path of incoming traffic to its neighbors, such as artificially increasing the minimum AS path or changing the minimum MED value [4]. In practice, however, ASes can choose to neglect their neighbors' preference declarations and use their own routing policies to maximize their own optimization objectives. For example, an AS may route traffic to its neighbor AS through the closest path (hot-potato policy) regardless of the impact on the incoming traffic engineering performance of its neighbor AS. Such selfish behaviors may lead to many undesirable consequences, such as low link utilization and reduced network robustness [5]. Although previous works have shown that cooperative methods can help achieve better overall performance in BGP [6–10], however, the cooperation typically works between neighboring ASes, not between two remote ASes.

In the past decade, as new network architectures are designed and promoted (e.g. LISP [11], SDN [12], NDN [13]), it is expected that network performance can be greatly improved. At the same time, these new architectures provide new methods and challenges for incoming traffic engineering. Among all the new proposals, one leading direction lies in a family of protocols called Transit-Edge Separate Internet, such as [11,14–16], some of which, such as LISP (Locator Identifier Separation Protocol) [11], have recently become RFC [17] and Internet standards. These new protocols can greatly alleviate the scalability problem of BGP (e.g. the global routing table size could be reduced by 43–90% in LISP [18]). At present, it is implemented in some new routers (e.g., in some Cisco routers) and is under testing in the (http://www.lisp4.nettestbed). Some companies such as Facebook have deployed their testing serving systems based on LISP for exchanging traffic [19].

We take LISP as a case study to investigate what changes new architectures bring to incoming traffic engineering. Our work may not be adapted to all the new architectures. We focus on theses changes, which we consider necessary for better understanding traffic engineering in new environments and future Internet protocol designs.

In LISP, edge ASes can exchange their routing information with each other for incoming traffic engineering, which is quite different from that in BGP. On the one hand, it provides a fine-grained and more direct way for edge ASes to control their incoming traffic (specified in Section 2). On the other hand, however, an AS has to decide how to optimize its incoming traffic engineering when it can exchange routing information with many other ASes. Although LISP RFC provides an interface for traffic engineering negotiation between edge ASes [11], it does not explicitly indicate how negotiation works in a multi-AS scenario. Due to the above reasons, we propose cooperative incoming traffic engineering in a selfish multi-AS scenario based on the cooperative game theory and make the following contributions.

- Motivated by the potential conflict of incoming traffic engineering performance and delay performance of independent edge ASes under LISP architecture, this paper proposes multi-AS incoming traffic engineering, formulates it with a heuristic cooperative game model, and solves it in a distributed manner.
- This paper evaluates delay differences between different RLOCs with experiments on PlanetLab for verifying the rationality of considering delay performance when optimizing incoming traffic engineering performance. Furthermore, extensive simulations driven by PlanetLab measurement data are performed to show the performance of cooperative incoming traffic engineering.

The remainder of the paper is organized as follows. Section 2 presents the LISP mechanism and analyzes motivations of cooperative incoming traffic engineering between edge ASes. Section 3 presents the heuristic cooperative game model and the distributed solution. Section 4 evaluates delay differences between different RLOCs with experiments on PlanetLab. Section 5 evaluates the performance of cooperative incoming traffic engineering. Section 6 reviews related works, and Section 7 concludes the paper.

2. LISP and motivation

In this section, we first explain how a packet is transmitted between edge ASes in LISP. Then we analyze the conflicted ETR selections between edge ASes under LISP, followed by our design motivation.

2.1. Locator identifier separation protocol

In LISP, transit core and end hosts are in two separate address spaces divided by edge routers: Routing Locators (RLOCs) and Endpoint Identifiers (EIDs). EIDs are the addresses used to identify end hosts inside an edge AS while RLOCs are the addresses used to identify edge routers in the transit network. Generally, an edge router serves as both an ITR (Ingress Tunnel Router) and an ETR (Egress Tunnel Router) with functions for exchanging traffic with other edge ASes. Therefore, an edge router is also called an xTR (x Tunnel Router).

An ITR can map EIDs to RLOCs. When receiving a packet from inside the source edge AS, the ITR treats the source address (in the source address field of the packet) and the destination address (in the destination address field of the packet) as two EIDs and performs the EID-to-RLOC mapping lookup. First, the ITR prepends an "outer" IP header, in which its globally-routable RLOC is added to the source address field for the source EID. Then, one globally-routable RLOC of the destination AS is added to the destination address field for the destination EID.

From EID-to-RLOC mapping lookup, the RLOC (the ITR of the source AS) is added to the source address field of the "outer" IP header while one of RLOCs (the ETRs of the destination AS) is added to the destination address field of the "outer" IP header. In the transit network, only RLOCs are used as addresses for packet transition. An ETR can map RLOCs to EIDs. When receiving a packet from the transit network, the ETR strips the "outer" IP header and forwards this packet to the end host inside the destination AS.

We illustrate the packet transmission process by an example as depicted in Fig. 1. Assuming that an end host with the address EID_x in AS_i sends traffic to another end host with the address EID_y in AS_j. For the sake of simplicity, we use the address to stand for its entity. For example, EID_x denotes the end host with the address EID_x.

EID_x sends traffic flows to one of its ITR, such as ITR ($RLOC_{i1}$) according to the intra-domain routing protocol such as the shortest path in OSPF. ITR ($RLOC_{i1}$), then performs EID-to-RLOC mapping process and prepends an"outer" IP header for each packet of the traffic. In the source address field of the "outer" IP header, EID_x is mapped to $RLOC_{i1}$. In the destination address field of the "outer" IP header, EID_y may be mapped to ETR ($RLOC_{j1}$) or ETR ($RLOC_{j2}$), which is suggested by AS_j, e.g., ETR ($RLOC_{j1}$). After that, the packet is forwarded to the transit core with the destination address $RLOC_{j1}$. The transit core works in a usual way and routes traffic to $RLOC_{j1}$. Upon receiving the packet, ETR ($RLOC_{j1}$) strips the "outer" IP header and forwards this packet to EID_y. EID_y sends traffic to EID_x in the same way.

In summary, The ETR selection is suggested by the destination AS but decided by the source AS. Actually, the destination AS first sends available ETRs with additional preferences, which suggests that the source AS better sends traffic flows based on announced preferences. However, the source AS has the final routing choice of the ETR for the destination edge AS. In an interest-dependent context (e.g. the source AS and the destination AS belong to the same ISP), the source AS will follow the announced preferences of the destination AS. However, in the scenario that independent ISPs manage ASes for their own independent interests, there may exist conflict over ETR selections.

2.2. Conflict of ETR selections

The conflict of ETR selections is basically caused by interest-independent edge ASes pursuing different aspects of performance. Under LISP architecture, ETR selections affect two aspects of performance: delay performance and incoming traffic engineering performance in different ways.

First, ETR selections affect delay performance of traffic flows. For two remote edge ASes (such as AS_i and AS_j in Fig. 1) with bidirectional traffic, they are both the source ASes and the destination ASes for the other AS. As the source ASes, they can choose their ETRs for their outgoing traffic to the other edge AS. Which ETRs to choose affects the one-way delay of their outgoing traffic because different ETR selections lead to different routing paths of traffic in the transit network (the second phase of packet transmissions from EID_x to EID_y in Fig. 1).

Formally, let $RLOC_{is}$ denote the sth $RLOC$ of AS_i (s = 1,2 in Fig. 1), $q_{ij}(s,t)$ denote the traffic volume from $RLOC_{is}$ to $RLOC_{jt}$, and $d_{ij}(s,t)$ denote the one-way delay between $RLOC_{is}$ to $RLOC_{jt}$. Then the total delay d_{ij} (d_{ji}) of traffic between AS_i and AS_j can be expressed in Eq. (1). The first item is total delay of outgoing traffic from AS_i to AS_j, and

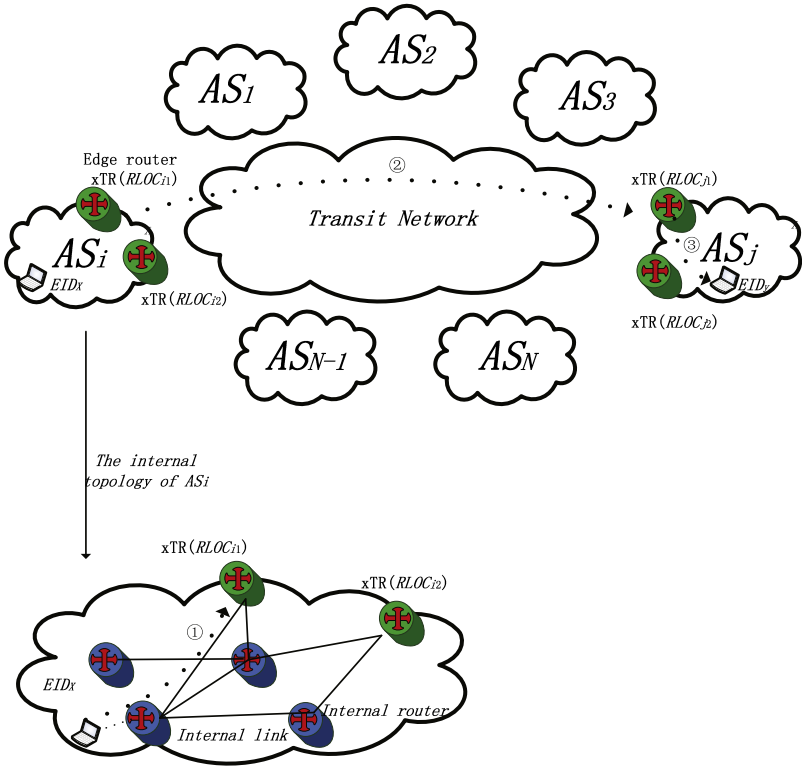

Fig. 1. LISP network architecture.

the second item is total delay of outgoing traffic from AS_j to AS_i. Because the outgoing traffic $q_{ij}(s,t)$ of AS_i ($q_{ji}(t,s)$ of AS_j) is also the incoming traffic of AS_j (AS_i), then $d_{ij} = d_{ji}$.

$$d_{ij} = d_{ji} = \sum_{s=1,2}\sum_{t=1,2} q_{ij}(s,t)d_{ij}(s,t) + \sum_{t=1,2}\sum_{s=1,2} q_{ji}(t,s)d_{ji}(t,s). \tag{1}$$

$q_{ij}(s,t)$ and $q_{ji}(t,s)$ are controlled by ETR selections of AS_i and AS_j respectively. AS_i and AS_j can adjust $q_{ij}(s,t)$ and $q_{ji}(t,s)$ to partially affect d_{ij}. That is, AS_i controls $q_{ij}(s,t)$ to change the first item in Eq. (1) while AS_j controls $q_{ji}(t,s)$ to change the second item in Eq. (1). d_{ij} (d_{ji}) only depends on ETR selections of AS_i and AS_j. The smaller d_{ij}, the higher throughput between AS_i and AS_j. Hence, if AS_i and AS_j both want to maximize the throughput between them, they choose ETRs to best minimize d_{ij}.

d_{ij} only reflects delay performance of traffic between AS_i and AS_j. Generally, an edge AS AS_i have bidirectional traffic with many other edge ASes. the delay (d_i) of all the traffic of AS_i can be expressed in Eq. (2). N_i is the set of edge ASes which exchange traffic with AS_i. According to Eqs. (1) and (2), the delay performance of AS_i depends on both ETR selections of AS_i and ETR selections of other edge ASes exchanging traffic with AS_i.

$$d_i = \sum_{j\in N_i} d_{ij}. \tag{2}$$

Besides, ETR selections also affect incoming traffic engineering of each edge AS. Different from its impact on delay performance, the incoming traffic engineering performance of one edge AS only depends on ETR selections of the other edge ASes sending traffic to it but not ETR selections of itself. For an edge AS (AS_i in Fig. 1), it receives traffic from the outside Internet via RLOCs ($RLOC_{i1}$ and $RLOC_{i2}$). When other edge ASes send traffic to AS_i, they can independently select ETRs of AS_i, which results in different incoming traffic ($r_i(1)$ and $r_i(2)$) over $RLOC_{i1}$ and $RLOC_{i2}$. Incoming traffic $r_i(1)$ and $r_i(2)$ further affect the load distribution of links inside AS_i, because the incoming traffic should be transferred to clients inside AS_i based on the intra-domain routing policy such as OSPF. Hence, when $r_i(1)$ and $r_i(2)$ change, the load distributions of links inside AS_i change along with it.

2.3. Motivation of cooperative incoming traffic engineering

From the above analysis, the incoming traffic of an edge AS (AS_i) is actually decided by many other edge ASes sending traffic to it under LISP architecture. In an interest-independent context, other edge ASes may choose ETRs only for their individual delay performance goals, but neglect incoming traffic engineering performance of AS_i, which could result in unbalanced incoming traffic of AS_i and degrade its robustness and resiliency of network management.

Given the fact that one edge AS cannot optimize its own incoming traffic engineering performance, our motivation is whether a group of edge ASes can cooperatively change their ETR selections for optimizing incoming traffic engineering performance between each other. On the one hand, an edge AS changes its ETR selections for optimizing incoming traffic engineering performance of other edge ASes. On the other hand, the incoming traffic engineering performance of this edge AS is optimized by other edge ASes.

3. Cooperative incoming traffic engineering

In this section, we first model inter-domain traffic engineering between edge ASes under LISP, then formulate constraints of cooperative incoming traffic engineering of edge ASes, followed by the heuristic optimization objective. Finally, we solve the problem in a distributed manner. The notations in this section are summarized in Table 1.

3.1. Modeling multi-AS inter-domain traffic engineering

We consider the multi-AS scenario in LISP, where N interest-independent edge ASes ($AS_i,\ i = 1,2,\ldots,N$) exchange a relevant amount of traffic in a stable manner between each other. AS_i exchanges traffic with other edge ASes via its RLOCs falling into the edge of AS_i, as shown in Fig. 1. Specifically, AS_i has K_i RLOCs. Its sth RLOC ($RLOC_{is}$) sends traffic $q_{ij}(s,t)$ to tth RLOC ($RLOC_{jt}$) of AS_j and receives traffic $q_{ji}(t,s)$ from $RLOC_{jt}$. The total sending traffic from $RLOC_{is}$ to AS_j is the sum of traffic from $RLOC_{is}$ to $RLOC_{jt},\ t = 1,2,\ldots,K_j$, as shown in Eq. (3). In reality, $Q_{ij}(s)$ is the sending traffic demand from clients inside AS_i to clients inside AS_j, which is fixed and is decided by the intra-domain routing policy of AS_i.

$$Q_{ij}(s) = \sum_{t=1}^{K_j} q_{ij}(s,t), \forall i,\ j; \quad s = 1,2,\ldots,K_i. \tag{3}$$

The incoming traffic $r_i(s), s = 1,2,\ldots,K_i$ of $RLOC_{is}$ depends on ETR selections of other edge ASes. When other edge ASes send traffic to AS_i, they can independently select ETRs of AS_i and choose whether to send their traffic to AS_i via $RLOC_{is}$. Hence, $r_i(s)$ is the result of ETR selections of the other $(N-1)$ edge ASes, as shown in Eq. (4).[1]

$$r_i(s) = \sum_{j\neq i}\sum_{t=1}^{K_j} q_{ji}(t,s), \forall i; \quad s = 1,2,\ldots,K_i. \tag{4}$$

$r_i(s),\ s = 1,2,\ldots,K_i$ decides the load distributions of links inside AS_i, as shown in Fig. 1. $r_i(s)$ includes many traffic flows which should be transmitted to internal routers of AS_i. When $RLOC_{is}$ (such as $RLOC_{i1}$ in Fig. 1) receives traffic $r_i(s)$, $RLOC_{is}$ will further route the traffic flows to the corresponding internal routers of AS_i through internal links of AS_i. Let E_i denote the internal link set of AS_i and $\beta_{il}(s) \in [0,1]$ denote the proportion of $r_i(s)$ flowing through link l. Then the load of $l \in E_i$ is the sum of traffic which is received from all the RLOCs of AS_i and goes through link l, as shown in Eq. (5).[2] The normalized traffic load of link l

[1] $RLOC_{is}$ also receives traffic from ASes which do not belong to N edge ASes in this model, which is fixed and can be regarded as the background traffic. We omit it in Eq. (4).

[2] We ignore the intra-domain traffic because AS_i is the edge AS and has fewer intra-domain traffic compared with the inter-domain traffic.

Table 1
Notations.

AS_i	The ith edge AS
$RLOC_{is}$	The sth $RLOC$ of AS_i
K_i	The number of RLOCs of AS_i
$q_{ij}(s,t)$	Traffic sent from $RLOC_{is}$ to $RLOC_{jt}$
$d_{ij}(s,t)$	One way delay from $RLOC_{is}$ to $RLOC_{jt}$
$Q_{ij}(s)$	Traffic demand from $RLOC_{is}$ to AS_j
$r_i(s)$	The incoming traffic to AS_i via $RLOC_{is}$
E_i	The link set inside the edge AS_i
l	One link inside some specific edge AS
$\beta_{il}(s)$	The proportion of $r_i(s)$ flowing through link l
C_l	The link capacity of link l
u_l	The traffic (load) of link l
$\hat{u}_l$	The normalized load of link l: u_l divided by C_l

is u_l divided by the link capacity c_l, as shown in Eq. (6). When intra-domain routing policy is fixed, $\beta_{il}(s)$ is fixed, and incoming traffic distribution over RLOCs uniquely decides the traffic load of links inside AS_i.

$$u_l = \sum_{s=1}^{K_i} \beta_{il}(s) r_i(s), \forall i; \quad l \in E_i. \tag{5}$$

$$\hat{u}_l = \frac{u_l}{c_l}, \forall i; \quad l \in E_i. \tag{6}$$

The traffic engineering performance of one edge AS (AS_i) can be expressed by a function $\psi_i(\hat{u}_l), l \in E_i$ of normalized traffic loads of links $\hat{u}_l, l \in E_i$ in AS_i. Generally in the literature, there are two typical metrics for evaluating traffic engineering performance: maximum link utilization and network cost [20,21]. Maximum link utilization represents the utilization of the most congested link in an AS, as shown in Eq. (7) and the optimization objective of AS_i is to minimize the maximum link utility $\min \psi_i$.

$$\psi_i = \max_{l \in E_i} \hat{u}_l, \forall i. \tag{7}$$

Network cost represents the overall load balance performance of AS_i. Specifically, the normalized load of each link is mapped to a score $\phi(\hat{u}_l), l \in E_i$ and network cost is the sum of the scores in AS_i. Eq. (8) presents the general form of network cost, and the optimization objective of AS_i is to minimize the network cost $\min \psi_i$. Generally, the mapping function is an increasing convex function, where the link score $\phi(\hat{u})$ increases in a nonlinear way as the load over one link increases. The increasing convex mapping function represents that it is cheaper to add traffic to a link with light traffic, but it is more expensive to add traffic to a link with heavy traffic. This design is based on the fact that the heavier load one link has, the longer queueing delay the traffic on this link has. All the traffic flows across the link will suffer from this increasing delay. Therefore, longer delay in one link degrades the performance of all the traffic flows across this link. As a result, $\phi(\hat{u})$ can be used to measure the influence of the traffic load over a link on delay performance of traffic flows on this link.

$$\psi_i = \sum_{l \in E_i} \phi(\hat{u}_l), \forall i. \tag{8}$$

In some sense, "maximum link utility" is a kind of special form of "network cost". The metric network cost is equal to the metric maximum link utility when we map all the links except the most congested link to the score 0 while mapping the most congested link as it is (the mapping function is still an increasing convex function). Besides, the mapping function could have different forms, as long as $\phi(\hat{u})$ is an increasing convex function.

In our context, we use network cost as the metric, which can be easily extended to the maximum link utilization metric. The proposed cooperative game model and method in the rest of this section are applicable to a general network cost metric. One typical mapping function is proposed in [20], as shown in Eq. (9). We use it as an example to driven the simulations in Section 5, but the model is general and is also applied to other forms of ϕ. Overall, For the edge AS AS_i, the purpose of incoming traffic engineering is to adjust the distribution of $r_i(s), s = 1, 2, \ldots, K_i$ so that the traffic load over links inside AS_i could lead to the minimum network cost.

$$\phi(\hat{u}) = \begin{cases} \hat{u}, & 0 \leqslant \hat{u} < \frac{1}{3}; \\ 3\hat{u} - \frac{2}{3}, & \frac{1}{3} \leqslant \hat{u} < \frac{2}{3}; \\ 10\hat{u} - \frac{16}{3}, & \frac{2}{3} \leqslant \hat{u} < \frac{9}{10}; \\ 70\hat{u} - \frac{178}{3}, & \frac{9}{10} \leqslant \hat{u} < 1; \\ 500u - \frac{1468}{3}, & 1 \leqslant \hat{u} < \frac{11}{10}; \\ 5000\hat{u} - \frac{16318}{3}, & \frac{11}{10} \leqslant \hat{u} < \infty. \end{cases} \tag{9}$$

3.2. Constraints formulation

As presented in the previous subsection, incoming traffic engineering performance (ψ_i) of each edge AS (AS_i) is affected by its incoming traffic over RLOCs, which is finally decided by ETR selections of other edge ASes. However, ETR selections also impact on delay performance of both the source ASes and the destination AS. When another source edge AS (AS_j) sends traffic to AS_i, it chooses ETRs of the destination AS (AS_i) for either minimizing delay d_{ij} or minimizing network cost ψ_i.

In our context that N interest-independent edge ASes send traffic between each other, each AS is both the source AS and the destination AS. As the source AS, it wishes to select ETRs best for delay performance of its traffic flows. As the destination AS, it hopes that other source ASes can select ETRs best for its incoming traffic engineering performance. The conflict lies in the fact that one edge AS chooses ETRs to optimize its delay performance, but could have a negative effect on incoming traffic engineering performance of other edge ASes. From another point of view, in the scenario that N ASes wish to cooperatively optimize incoming traffic engineering performance between each other, they will change their ETR selections for incoming traffic engineering performance of other edge ASes, but could degrade the delay performance of its own traffic flows. Considering the incentive of cooperative incoming traffic engineering, it is reasonable to assume that each AS has the constraint that it only sacrifices limited delay performance for optimizing the incoming traffic engineering performance of N edge ASes.

Formally, let d_i' and d_i denote the delay of AS_i before and after cooperative incoming traffic engineering respectively,

as shown in Eqs. (10) and (11). $q'_{ij}(s,t)$ and $q_{ij}(s,t)$ are traffic from $RLOC_{is}$ to $RLOC_{jt}$ before and after cooperative incoming traffic engineering, respectively. We formulate the delay constraint that the delay increase after cooperation d_i cannot exceed α_i times of that before cooperation d'_i. $\alpha_i \in [1,+\infty]$ is the *delay increasing factor*, which reflects the delay increase of AS_i due to changing ETR selections for optimizing incoming traffic engineering performance of edge ASes. Considering the fact that $d_{ij}(s,t) = d_{ji}(t,s)$, we combine Eqs. (10)–(12), and have Eq. (13).

$$d'_i = \sum_{j\neq i}\sum_{s=1}^{K_i}\sum_{t=1}^{K_j} q'_{ij}(s,t)d_{ij}(s,t) + q'_{ji}(t,s)d_{ji}(t,s), \forall i. \tag{10}$$

$$d_i = \sum_{j\neq i}\sum_{s=1}^{K_i}\sum_{t=1}^{K_j} q_{ij}(s,t)d_{ij}(s,t) + q_{ji}(t,s)d_{ji}(t,s), \forall i. \tag{11}$$

$$d_i \leqslant \alpha_i d'_i. \tag{12}$$

$$\sum_{j\neq i}\sum_{s=1}^{K_i}\sum_{t=1}^{K_j}(q_{ij}(s,t) + q_{ji}(t,s) - \alpha_i(q'_{ij}(s,t) + q'_{ji}(t,s))d_{ij}(s,t) \leqslant 0. \tag{13}$$

Furthermore, there are another two constraints for the decision variable $q_{ij}(s,t)$. First, the outgoing traffic demand $Q_{ij}(s)$ of each $RLOC$ ($RLOC_{is}$) to AS_j is given and is the sum of traffic from $RLOC_{is}$ to $RLOC_{it}$, $t = 1,2,\ldots,K_j$, as shown in Eq. (3). Second, the traffic from one RLOC of an edge AS to the RLOC of the other edge AS should be positive, as shown in Eq. (14).

$$q_{ij}(s,t) \geqslant 0, \forall i,\ j; \quad s = 1,2,\ldots,K_i; \quad t = 1,2,\ldots,K_j. \tag{14}$$

3.3. Optimization objective

Our motivation is that N interest-independent edge ASes cooperatively optimize incoming traffic engineering performance with each other through ETR selections. However, ETR selections change delay performance of traffic of edge ASes as well. On the one hand, one source edge AS wants to send the traffic to the destination ETR with the minimum delay. On the other hand, one destination edge AS wants other source edge ASes to change their ETR selections to optimize this destination edge AS' incoming traffic engineering performance. If the source edge AS follows the destination edge AS's suggestions of ETR selection, the source edge AS might benefit destination edge AS's incoming traffic engineering performance, but might hurt its own outgoing traffic delay performance. Furthermore, one destination edge AS' incoming traffic engineering performance is affected by many source edge ASes' traffic. Overall, one edge AS's delay performance sacrifice might not necessarily bring benefit to its incoming traffic engineering performance.

With consideration on both utilization and fairness, we choose to use cooperative game theory to solve the cooperative incoming traffic engineering problem. In a cooperative game, two or more players enter the game with their individual utilities and act with each other for a win–win solution. Specifically, the players' individual utilities constitute a social utility in certain form and all the players optimize this social utility. The classic social utility in cooperative game theory is Nash bargaining [22], which is the serial product of all the individual utilities. It can be proved that Nash bargaining can guarantee that all players acquire the maximum and equal individual utilities.

However, the social utility in our scenario cannot be modeled by standard Nash bargaining. Specifically, there are two aspects of performance for each edge AS: traffic engineering performance and delay performance. Each edge AS wishes to optimize its traffic engineering performance which is decided by other edge ASes, and each edge AS can act best for its delay performance but affect traffic engineering performance of other edge ASes. First, these two kinds of performance cannot be perfectly combined by one individual utility. Besides, different edge ASes have different initial incoming traffic engineering performance, which means that the improvement space of incoming traffic engineering performance of edge ASes are different. Also, different edge ASes could sacrifice different delay performance, which means that different edge ASes could contribute different improvements for incoming traffic engineering performance of other edge ASes.

Hence, we propose a heuristic cooperative game social utility inspired by Nash bargaining [22] and its transformation [23]. First, the individual utility of AS_i can be expressed by $(\psi'_i - \psi_i)^{\alpha_i/\sum_{k=1}^{N}\alpha_k}$, where ψ'_i and ψ_i are the network cost of AS_i before and after cooperation, respectively. The normalized delay increase is assigned to the individual welfare by using the exponentiation of the utility gains. The higher α_i, the higher contribution of AS_i to incoming traffic engineering performance for other edge ASes. The social utility is the serial product of individual utility, as shown in Eq. (15). Note that the proposed social utility is a heuristic optimization objective combining both traffic engineering performance and delay performance into together. The social utility guarantees that incoming traffic engineering performance of all the edge ASes can be improved ($\psi'_i - \psi_i > 0$) but does not guarantee absolute fairness like that in Nash bargaining.

$$\begin{aligned}
&\max U = \prod_i (\psi'_i - \psi_i)^{\alpha_i/\sum_{k=1}^{N}\alpha_k};\\
&s.t.\\
&d_i \leqslant \alpha_i d'_i;\\
&Q_{ij}(s) = \sum_{t=1}^{K_j} q_{ij}(s,t), \forall i,\ j; \quad s = 1,2,\ldots,K_i;\\
&q_{ij}(s,t) \geqslant 0, \forall i,\ j; \quad s = 1,2,\ldots,K_i; \quad t = 1,2,\ldots,K_j.
\end{aligned} \tag{15}$$

In Eq. (15), ψ'_i and ψ_i are the function of $q'_{ij}(s,t)$ and $q_{ij}(s,t)$ based on Eqs. (4)–(6) and (8). d'_i and d_i are also the function of $q_{ij}(s,t)$ based on Eqs. (10)–(12). The above problem can be solved by changing $q_{ij}(s,t)$ through ETR selections. Due to monotony of logarithmic function, Eq. (15) can be transformed to Eq. (16) without changing the solution of the original problem.

$$\max lnU = \sum_i \frac{\alpha_i}{\sum_k \alpha_k} \ln(\psi_i' - \psi_i);$$

$$s.t.$$

$$d_i \leqslant \alpha_i d_i'; \tag{16}$$

$$Q_{ij}(s) = \sum_{t=1}^{K_j} q_{ij}(s,t), \forall i,\ j; \quad s = 1,2,\ldots,K_i;$$

$$q_{ij}(s,t) \geqslant 0, \forall i,\ j;\ s = 1,2,\ldots,K_i; \quad t = 1,2,\ldots,K_j.$$

It can be verified that the second derivative of the optimization objective in Eq. (15) is always less than or equal to 0 (proved in Appendix A). So the optimization objective is the convex optimization problem. The constraints in Eq. (15) are all linear, which satisfy the "quality constraints" [24]. For the convex problem with linear constraints, the general method is to obtain the KKT condition. However, if there are N edge ASes and AS_i has K_i RLOCs. There is total $N + \sum_i K_i(N-1)$ (N constraints for $delay_i$ and $\sum_i K_i(N-1)$ constraints for $Q_{ij}(s)$) constraints the problem. The number of constraints increases nonlinearly as the increase of N and K_i, so the computational complexity will grow substantially, the optimal solution cannot be obtained by using optimization software directly within limited time. Therefore, we turn to solve the problem in a distributed manner to reduce the computational complexity, which is specified in the next subsection.

Furthermore, there are some practical issues that should be explained about the optimization objective. First, the parameters ($d_{ij}(s,t)$, $q_{ij}(s,t) \forall i,j$, $s = 1,2,\ldots,K_i$, $j = 1,2,\ldots,K_j$) in the optimization objective Eq. (15) are obtained in a large time scale such as a month or a quarter. In practice, the delay $d_{ij}(s,t)$ (traffic volume $q_{ij}(s,t)$) between RLOCs can be sampled periodically within a period of time such as $d_{ij}(s,t,T_1), d_{ij}(s,t,T_2), \ldots, d_{ij}(s,t,T_M)$ ($q_{ij}(s,t,T_1), q_{ij}(s,t,T_2), \ldots, q_{ij}(s,t,T_M)$), M is the total sample number. $d_{ij}(s,t)$ ($q_{ij}(s,t)$) is time average values based on the samples, as shown in Eqs. (17) and (18). $d_{ij}(s,t)$ ($q_{ij}(s,t)$) reflects the general delay (traffic volume) parameters between RLOCs and is not much affected by the transient and instant delay (traffic volume) change between RLOCs.

$$d_{ij}(s,t) = \frac{1}{M} \sum_{T_m=1}^{M} d_{ij}(s,t,T_m), \forall i,j;\ s = 1,2,\ldots,K_i,\ t = 1,2,\ldots,K_j. \tag{17}$$

$$q_{ij}(s,t) = \frac{1}{M} \sum_{T_m=1}^{M} q_{ij}(s,t,T_m), \forall i,j;\ s = 1,2,\ldots,K_i,\ t = 1,2,\ldots,K_j. \tag{18}$$

Second, this model assumes that the transit network is not heavily loaded, which means that $d_{ij}(s,t)$ is not the function of $q_{ij}(s,t)$. This is because that the transit network bandwidth is typically over-provisioned significantly and ISPs of transit networks can routinely conduct traffic engineering through various methods to avoid potential congestions. Switching the transit path for one pair of edge ASes typically should have little impact on the utilization of the transit links, thus should have little impact on traffic queueing on the transit core networks, but the change to the propagation delay can be non-trivial because of potentially switching to a longer or shorter path (processing delay and transmission delay are relatively trivial compared with propagation delay). Therefore, we consider the propagation delay as the main variable factor of the delay in the transit core network in our model.

Our model can be extended to more complicated models, where parameters (such as delay and traffic) could be dynamic and the optimization objective could be the function of these parameters. These complicated models need the modification of solution methods, which are out of the scope of this paper, and can be proposed and solved in the future.

3.4. Distributed method

In this subsection, we add some artificial variables to decouple the original problem, and propose a distributed method based on Lagrange decomposition.

We divide the variables of Eq. (15) into two categories: incoming traffic variables and outgoing traffic variables. For AS_i and AS_j, $q_{ij}(s,t)$ is the outgoing traffic variables for AS_i and the incoming traffic variables for AS_j. For each edge AS (AS_i), we add a group of artificial incoming traffic variables. $\overline{q}_{ji}(t,s)$ is an artificial incoming traffic variables, which means AS_i's expected traffic from $RLOC_{jt}$ to $RLOC_{is}$. To keep the solution of the original problem the same, we add a constraint for each artificial variables, as shown in Eq. (19).

$$\overline{q}_{ji}(t,s) = q_{ji}(t,s). \tag{19}$$

First, due to the convexity of the optimization objective and the linear constraints, the original problem has strong duality. We can solve it by solving the dual problem [24]. We add the artificial constraint (Eq. (19)) into the optimization objective and formulate the dual problem of Eq. (16), as shown in Eq. (20). Eq. (20) can be proved to be a convex problem [24]. $v_{ji}(t,s)$ is lagrangian multiplier [24]. $\mathbf{v}$ is the set of $v_{ji}(t,s)$.

$$\begin{aligned} L(\mathbf{v}) = {} & \sum_i \frac{\alpha_i}{\sum_k \alpha_k} \ln(\psi_i' - \psi_i) + \sum_i \sum_j \sum_{s=1}^{K_i} \sum_{t=1}^{K_j} v_{ji}(t,s) \\ & \times (q_{ji}(t,s) - \overline{q}_{ji}(t,s)); \\ & s.t. \\ & d_i \leqslant \alpha_i d_i'; \\ & Q_{ij}(s) = \sum_{t=1}^{K_j} q_{ij}(s,t), \forall i,\ j; \quad s = 1,2,\ldots,K_i; \\ & q_{ij}(s,t) \geqslant 0, \forall i,\ j; s = 1,2,\ldots,K_i; \quad t = 1,2,\ldots,K_j. \end{aligned} \tag{20}$$

We replace all the incoming traffic variables with the artificial variables, and keep outgoing traffic variables the same. Then the optimization objective of Eq. (20) is shown in Eqs. (21) and (22).

$$L = \sum_i L_i. \tag{21}$$

$$\begin{aligned} L_i = {} & \frac{\alpha_i}{\sum_k \alpha_k} \ln(\psi_i' - \psi_i) + \sum_{j \neq i} \sum_s \sum_t v_{ij}(s,t) q_{ij}(s,t) \\ & - \sum_{j \neq i} \sum_s \sum_t v_{ji}(t,s) \overline{q}_{ji}(t,s). \end{aligned} \tag{22}$$

Note that Eq. (22) is the function of artificial variables of AS_i and the outgoing variables of AS_i. Hence, it can be independently solved by AS_i with the corresponding constraints, as shown in Eq. (23).

$$
\begin{aligned}
L_i = {} & \frac{\alpha_i}{\sum_k \alpha_k} \ln(\psi_i' - \psi_i) + \sum_{j \neq i} \sum_s \sum_t v_{ij}(s,t) q_{ij}(s,t) \\
& - \sum_{j \neq i} \sum_s \sum_t v_{ji}(t,s) \overline{q}_{ji}(t,s); \\
& s.t. \\
& \sum_{j \neq i} \sum_{t=1}^{K_j} \sum_{s=1}^{K_i} (\overline{q}_{ji}(t,s) + q_{ij}(s,t) \\
& \quad - \alpha_i (q_{ij}'(s,t) + q_{ji}'(t,s)) d_{ij}(s,t) \leqslant 0; \\
& \sum_{t=1}^{K_j} q_{ij}(s,t) = Q_{ij}(s), \forall i, s = 1,2,\ldots,K_i; \\
& \sum_{s=1}^{K_i} \overline{q}_{ji}(t,s) = Q_{ji}(t), \forall j, t = 1,2,\ldots,K_j; \\
& q_{ij}(s,t) \geqslant 0, \forall i,\ j; s = 1,2,\ldots,K_i; \quad t = 1,2,\ldots,K_j. \qquad (23)
\end{aligned}
$$

Hence, the dual problem (Eq. (20)) can be first decomposed. Besides, based on sub-gradient methods [24], we can use Lagrange parameters as interfaces to let the artificial variables and the original incoming traffic variables tend to be the same through iterations. We assume at step k, each edge AS has optimized its own optimization objective (Eq. (23)), and the Lagrange parameters are $v_{ji}(t,s)^k$. If Eq. (24) holds, and ξ is selected satisfying the diminishing step size rules, $v_{ji}(t,s)$ converges in limited steps [25], which means that the artificial variables are the same as the original incoming traffic variables. According to Eq. (20), the derivative of L is as Eq. (25), which can be substituted in Eq. (24). Thus, the specific algorithm can be shown as Algorithm 1.

$$v_{ji}(t,s)^{k+1} = \max\left(0, \xi \frac{\partial L}{\partial v_{ji}(t,s)^k}\right). \qquad (24)$$

$$\frac{\partial L}{\partial v_{ji}(t,s)} = q_{ji}(t,s) - \overline{q}_{ji}(t,s). \qquad (25)$$

Algorithm 1. Distributed method

Algorithm 1 can be expressed as the following: (1) Initialize the Lagrange parameters and the delay increasing factor α for each AS; (2) each AS calculates its individual optimization objective based on Eq. (23); and (3) the Lagrange parameters change according to step 4. Within the limited number of iterations, the Lagrange parameters converge and the algorithm ends.

There are some practical issues about the above algorithm that deserve to be explained. First, as presented previously in this section, cooperative incoming traffic engineering has a large time scale, such as a month or a quarter. For each incoming traffic engineering cooperation, edge ASes joining the cooperative game provides its AS information (delay $d_{ij}(s,t)$ and traffic $q_{ij}(s,t)$ between its own RLOCs and RLOCs of other edge ASes), its current network cost (ψ_i'), and its delay performance sacrifice tolerance (α_i). After the algorithm runs, each AS updates its newly RLOC choices of its outgoing traffic to other edge ASes and also obtain its network cost change from ψ_i' to ψ_i.

Second, the algorithm can be run in central servers or be performed in a distributed manner by edge ASes. When the algorithm is run in central servers (note that when there are many edge ASes joining the cooperative game, more than one central server are needed to run the distributed method because the calculation complexity could be higher than one server can support), edge ASes only submit its information but do not participate in running the algorithm. In theory, there are possibilities that some selfish edge ASes provide fake information to change the social utility, so that it can acquire more traffic engineering performance improvement with less delay performance sacrifice. In practice, however, the social utility is usually private and cannot be acquired by each edge AS. Hence, it is difficult for a specific edge AS to know how to change its own parameters to best benefit its own performance because the information submission interval is quite large. Besides, considering the fact that the algorithm runs in a large time scale, it is also impractical for a specific edge AS to try submitting different information to best benefit its own performance. Furthermore, some parameters can be actually submitted by two edge ASes, which can be regarded as a verification. For example, traffic volume $q_{ij}(s,t)$ can be submitted by both AS_i and AS_j. Hence, we believe that there is little

Iutput:
The delay increasing factor: $\alpha_i, i = 1,2,\ldots N$.
The traffic of each RLOC from its belonging AS to another AS: $Q_{ji}(t), \forall i,j; s = 1,2,\ldots,K_i, t = 1,2,\ldots K_j$.
Output:
The traffic between RLOCs of ASes: $q_{ji}(t,s)$.
1: Initial the step-size ξ.
2: Initial Lagrange parameters $v_{ji}(t,s)$.
3: Each edge AS independently optimizes its optimization objective $L_i, i = 1,2,\ldots,N$ based on Eq. (23).
4: Update $v_{ji}(t,s) = \max(0, v_{ji}(t,s) - \xi(q_{ji}(t,s) - \overline{q}_{ji}(t,s)), \forall i,j; s = 1,2,\ldots,K_i, t = 1,2,\ldots,K_j$.
5: **if** All the $v_{ji}(t,s)$ no long change **then**
6: End algorithm.
7: **else**
8: Go to step 3.
9: **end if**

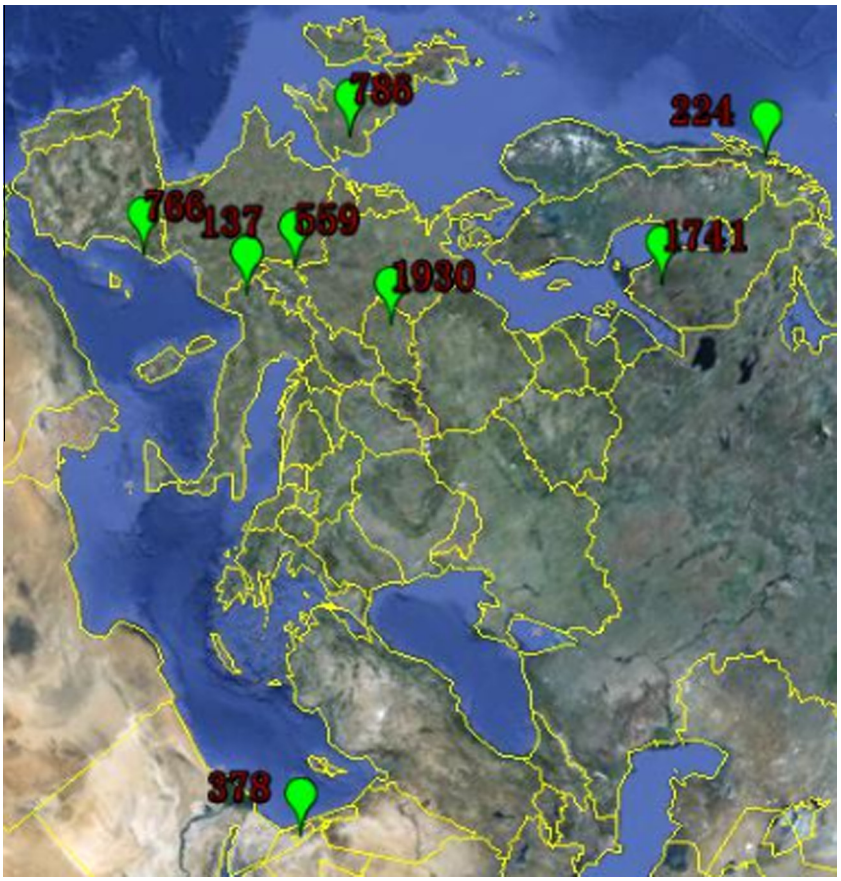

Fig. 2. RTD measurement experiment based on PlanetLab.

motivations for an edge AS to provide false information for its own interests.

When the algorithm is distributedly performed by edge ASes, there are a set of common $v_{ji}(t,s)$ for each two RLOCs ($RLOC_{is}$ and $RLOC_{jt}$) of two specific edge ASes (AS_i and AS_j), and a common ξ for all the edge ASes. Then AS_i optimizes its own optimization objective L_i in Eq. (23), and $v_{ji}(t,s)$ will change as the iterations. Until $v_{ji}(t,s) = 0, \forall i,\ j,s = 1,2,\ldots,K_i,\ t = 1,2,\ldots,K_j$, the algorithm ends, which means that $q_{ji}(t,s) = \overline{q}_{ji}(t,s), \forall i,\ j,s = 1,2,\ldots,K_i,\ t = 1,2,\ldots,K_j$ in Eq. (25). Note that the change of $v_{ji}(t,s)$ is decided by both AS_i and AS_j, and $v_{ji}(t,s) = 0$ represents the success negotiation of AS_i and AS_j for the traffic $q_{ji}(t,s)$. In this distributed method, each edge AS does not have to submit its private AS information to all the other ASes or central servers. Edge ASes perform the algorithm by the interface parameters v and ξ.

Each edge AS just optimizes its artificial optimization objective L_i and decides whether its current incoming traffic ($q_{ji}(t,s)$) is the same as its expected incoming traffic ($\overline{q}_{ji}(t,s)$), as shown in Eq. (25). Then AS_i uses the difference between $\overline{q}_{ji}(t,s)$ and $q_{ji}(t,s)$ to perform its next individual optimization no matter whether AS_i wants to modify its own optimization objective. At this time, one edge AS could want to cheat in the cooperation, which could result in frequent negotiation between other edge ASes, but other edge ASes also could refuse to cooperate with this edge AS. Hence, we believe that the distributed method also provides incentives for edge ASes to honestly cooperate with other edge ASes.

4. Delay evaluation of ETR selections

From previous sections, edge ASes sacrifice their delay performance for incoming traffic engineering performance of other edge ASes. The delay performance loss is caused by ETR selections of source edge ASes. When source ASes select different RLOCs of the destination ASes, they route their corresponding traffic flows through different paths with different delay. Hence, it is critical to measure delay variations of different paths between edge ASes.[3]

In this section, we measure delay between RLOCs of different ASes based on PlanetLab. PlanetLab is a global research network supporting the development of new network services, which consists of 1173 nodes at 561 sites worldwide. We can manipulate these nodes through a struct called "slice" [26]. We deployed this experiment within Europe, which has the highest density of nodes in PlanetLab.

The purpose of this experiment is to record delay between edge ASes. However, the nodes in PlanetLab are

[3] Obviously, if there is no delay differences when edge ASes change ETRs, edge ASes are willing to optimize incoming traffic engineering of other ASes, because it will not bring any loss for themselves.

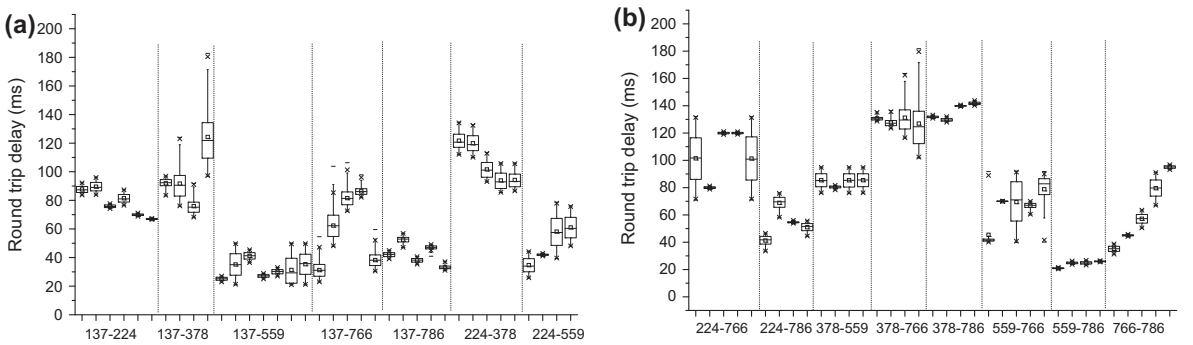

Fig. 3. Round trip delay between two ASes.

widely dispersed, and generally there is only one node in one edge AS. It is difficult to let one node send traffic through different RLOCs due to the fixed intra-domain routing protocols. Considering the fact that delay of traffic flows between two edge ASes is mainly caused by its transmission through the transit ASes, so delay between transit ASes also partially reflects delay between the edge ASes within the transit ASes. Hence, we measure the delay between transit ASes.

4.1. Experiment settings

The experiment scenario is depicted in Fig. 2. The red numbers represent the AS numbers in experiment while the green marks represent geographic positions of these ASes. We choose the group of ASes with the AS number {137 224 378 559 766 786} because there are more nodes available to login. We can login some nodes in one AS, and send packets to nodes of other ASes. Command "traceroute" is used to find out the packet path from one AS to another AS. For example, we login one node in AS 224 and send a packet from this node to another node in AS 137. This packet goes through a series of routers belonging to AS 224, AS 1741, AS 1930, AS 559 and AS 137. The router information can be recorded. Based on the router information, we can roughly identify the edge routers of different ASes as follows. For example, the "trace-route" packet goes through router 1 (AS 224), router 2 (AS 224), router 3 (AS 1741), router 4 (AS 1741), router 5 (AS 1741), router 6 (AS 1930), router 7 (AS 559) and router 8 (AS 137) (Router names have been omitted). Then we take router 2 as the edge router resided in AS 224 which connects to AS 1741, and router 3 as the edge router resided in AS 1741 which connects to AS 224.

Then, we login different nodes in the source AS respectively and send "trace-route" to different nodes in the destination AS. After identifying the edge routers between two ASes, we use "ping" to periodically record round trip delay (RTD) between different RLOCs of different ASes, and accumulate RTDs once an hour within one months.

4.2. Experiment results

We show the path RTD statistics between 6 ASes (AS 137, AS 224, AS 378, AS 556, AS 766, and AS 786), respectively in Fig. 3. We use boxplots to display RTD statistical properties (Each box, between the min. and the max., displays the first quartile, the median, and third quartile). Each box represents the statistic RTD of one path between two specific ASes. The median RTD can be regarded as the delay of this path in a long time scale and corresponds to the time scale of traffic engineering.

From Fig. 3, we find that the median RTDs of different paths of two specific ASes are different in most cases. This difference could be relatively large such as between AS 766 and AS 786 and could be quite small such as between AS 559 and AS 786. The root cause of the above observation is that changing paths brings different routing distances and routing hops, which lead to different propagation delay and switching delay.

The purpose of this experiment is to show the impact of ETR selections of edge ASes on delay performance. The delay of different paths between two ASes can be roughly regarded as the delay when two edge ASes change their ETR selections, as specified at the beginning of Section 4. If two ASes initially choose paths best for their delay performance, their ETR selections form the routing path with minimum RTD (For example in Fig. 3, if AS 766 exchanges traffic with AS 786 and they both minimize delay, they will choose the first path to exchange traffic and the RTD is less than 40 ms). When they decide to cooperatively optimize incoming traffic engineering performance between each other, they change the ETR selections and could exchange traffic in other paths, which results in delay increase. This delay increase could be relatively large and affect the bandwidth between two ASes, which can be regarded as the delay performance loss. Hence, it is reasonable to consider the delay performance variations when we optimize incoming traffic engineering through ETR selections.

5. Evaluation of cooperative incoming traffic engineering

In this section, we evaluate the performance of cooperative incoming traffic engineering driven by delay data from PlanetLab. We first provide simulation settings, followed by statistic performance evaluation and impact of parameters on performance. The simulation parameters

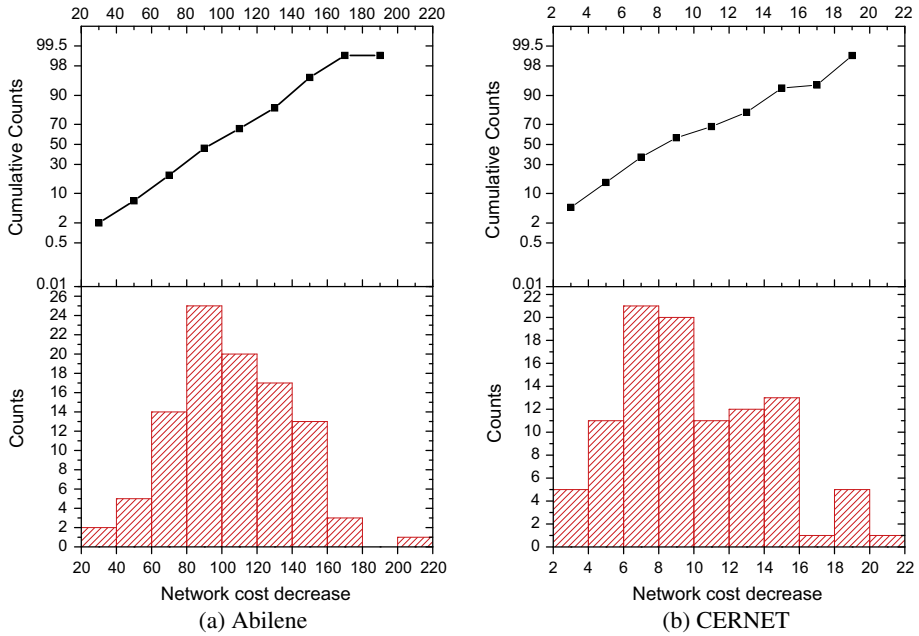

Fig. 4. The total network cost decrease of six edge ASes. *X*-axis is the total network cost decrease of the six edge ASes. *Y*-axis is the counts (cumulative counts). $\alpha_i = 1.2, i = 1, 2, \ldots, 6$.

are collected from different realistic scenarios as possible as we can, but still could deviate from the reality.

5.1. Simulation settings

- *Network.* We assume 6 edge ASes exchange traffic between each other and cooperatively optimize incoming traffic engineering between each other. The network is shown in Fig. 1, where $N = 6$. Each edge AS has two RLOCs which is the typical scenario in reality [1].[4]
- *Topology of edge ASes.* Topology structures and positions of RLOCs (edge routers) affect the performance of cooperative incoming traffic engineering. The ideally realistic topology parameters should include the topology of edge ASes, the link weights inside the edge AS, traffic demands between edge ASes and traffic load on links in an edge AS. However, there are almost no public topologies of edge ASes including its link weights (edge ASes mainly belong to enterprises and the topologies are generally not released). Hence, to obtain more realistic and accurate topology parameters, we choose two public topologies Abilene [27] and CERNET [28] as the topologies for the edge ASes in the simulation instead, which can provide a more detailed information about topology and OSPF weights. We assume that DNVY and KSCY are regarded as two edge routers in Abilene while Qingdao and Shanghai are regarded as two edge routers in CERNET. The intra-domain routing policy is OSPF with suggested links weights from [27,28]. The link capacity inside each edge AS is 10 Gbps.
- *Inter-domain traffic demand between ASes.* We assume that the total cooperative incoming traffic for each AS is 10 Gbps, which is the incoming traffic demand of each edge AS. Each RLOC has the traffic demand 1 Gbps for a destination AS, for example $Q_{ij}(1) = Q_{ij}(2) = 1$ Gbps. The traffic demand is uniformly destined to the routers inside the edge AS. We omit the intra-domain traffic because intra-domain traffic is trivial compared to the inter-domain traffic for edge ASes.
- *Delay between RLOCs of edge ASes.* The six edge ASes needs C_6^2 inter-domain delay data (each two edge ASes need delay data), which corresponds to our experiments in PlanetLab. Hence, we use the delay in Fig. 3 as the inter-domain delay between the six edge ASes. In our simulation, there are four paths between each two edge ASes (because there are two RLOCs of each edge ASes), but the delay data in our experiment could have more than 4 paths (For example, there are 6 paths of RTDs between AS 137 and AS 224 in Fig. 3). In this condition, we choose the first four groups of delay data as the inter-AS delay between RLOCs of two ASes. We randomly choose 100 groups of delay data to drive our simulations.[5]

[4] The experiment in [37] demonstrates that more xTRs will benefit more incoming traffic engineering performance, but the improvement from one xTR to two xTRs is much larger than the improvement from two xTRs to more than two xTRs. In reality, the most typical case is that one edge AS has two RLOCs, only 17% of edge ASes in the Internet have more than two RLOCs [1].

[5] Here we use RTDs data in the experiment as the one-way delay in the simulation because it is hard to directly obtain the one-way delay in our experiment. We only use the simulation to show the impact of delay variations on incoming traffic engineering performance, so RTD can also reflect the one-way delay to some extent.

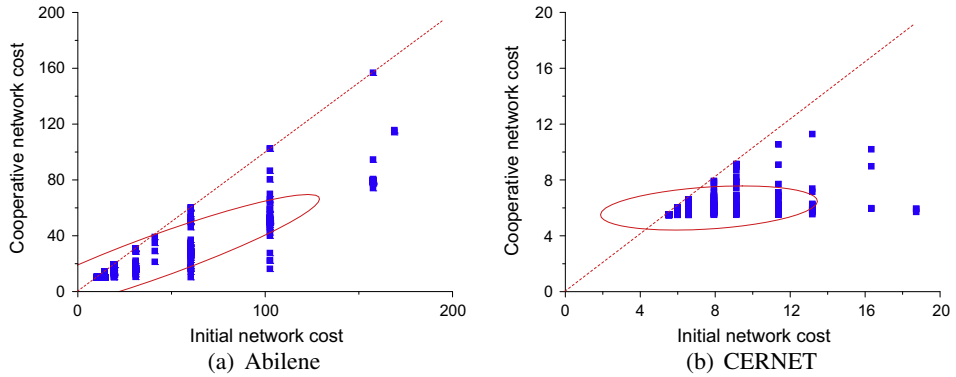

Fig. 5. The individual network cost before and after cooperation. *X*-axis is the initial network cost before cooperation of one edge AS. *Y*-axis is the network cost after cooperation of one edge AS. $\alpha_i = 1.2,\ i = 1, 2, \ldots, 6$. The line in each subfigure is $y = x$. The point below the line represents that the network cost decreases after cooperation compared with that in the initial state before cooperation. The point above the line represents that the network cost increases after cooperation compared with that in the initial state before cooperation. The range of ellipse includes 90% of points in the simulation.

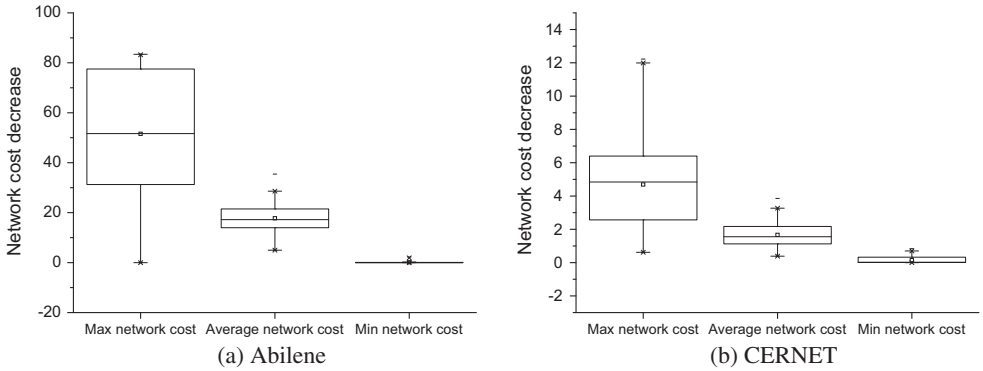

Fig. 6. The network cost decrease statistics of the six edge ASes. Maximum network cost represents the largest network cost of the six edge ASes before cooperation. Average network cost represents the average network cost of the six edge ASes before cooperation. Minimum network cost represents the minimum network cost of the six edge ASes before cooperation. *Y*-axis is the network cost decrease after cooperation. $\alpha_i = 1.2, i = 1, 2, \ldots, 6$.

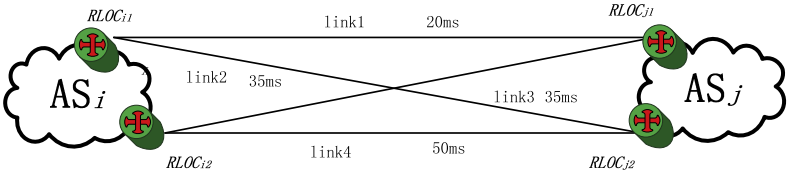

Fig. 7. Delay in artificial simulation setting.

5.2. Performance evaluation

5.2.1. Total incoming traffic engineering performance

We first evaluate cooperative incoming traffic engineering performance. Fig. 4(a) and (b) shows the total CDF of network cost decrease of the six edge ASes. First, the topology of edge ASes has a great impact on the network cost decrease, because intra-domain routing protocols deicide how incoming traffic is distributed over intra-domain links of edge ASes. Abilene and CERNET are different topologies, which means that the same incoming traffic distribution over RLOCs leads to different load distributions of links and different network cost.

Second, the cooperative performance (which is defined as the network cost decrease after cooperation) is sensitive to link delay. We take the Abilene topology as the example. When edge ASes have the topology of Abilene, there are more than 90% percent of simulation runs where the sum of network cost decrease of six edge ASes is between 60 and 180. There are 7 simulation runs, where the sum of network cost decrease of six edge ASes is less than 60. There is one simulation run, where the

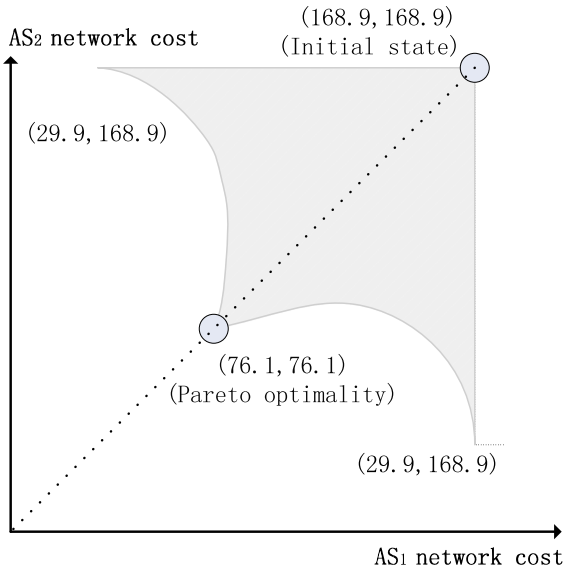

Fig. 8. Pareto optimality in 2 ASes scenario in Abilene. The topology of the edge AS is Abilene. $\alpha_1 = \alpha_2 = 1.1$. Both ASes receives 10 Gbps traffic from the other AS. The path delay is shown in Fig. 7.

sum of network cost decrease of six edge ASes is more than 200.

Actually, the link delay decides ETR selections before cooperation of edge ASes. Sometimes, edge ASes route traffic best for their individual delay performance, which also results in the balanced incoming traffic of the edge AS. In this condition, edge ASes have balanced incoming traffic even without cooperation, and there is no much improvement of cooperative incoming traffic engineering. Similarly, the link delay also could lead to very unbalanced incoming traffic of a specific AS. In this condition, cooperative incoming traffic engineering can greatly decrease the network cost with a higher probability.

5.2.2. Individual incoming traffic engineering performance

Fig. 5 shows the network cost change of each edge AS. First, network cost of all the edge ASes will not increase after cooperation. Furthermore, higher network cost in the initial state before cooperative incoming traffic engineering leads to higher network cost decrease after cooperation in statistic. The higher initial network cost (before cooperation) means that some links inside the edge AS are in heavier load, which also represents the more unbalanced incoming traffic of the edge AS. Considering the fact that network cost is a nonlinear function of the link load, to balance the incoming traffic of edge ASes which have a more unbalanced initial incoming traffic will decrease the heavier load of the links inside the edge AS and will reduce more network cost. This conclusion also can be drawn from Fig. 6.

5.3. Pareto optimality

We further consider the pareto optimality in a simple scenario that two edge ASes cooperate to optimize the other's network cost performance. The one way delay is shown in Fig. 7 (The longest RTD is more than 3 times over the shortest one in Fig. 3, so we choose the longest delay is 2.5 times over the shortest one) and the parameters of inter-domain traffic and α are shown in Fig. 8. When two edge ASes do not cooperate, their initial individual network cost is 168.9. When they cooperate to optimize the other AS's network cost, the shadow area shows the cooperative feasible region. their individual network cost decreases to 76.1, which is minimum under the condition that they have equal network cost decrease. Hence, we verify that the cooperative solution is Pareto optimality.

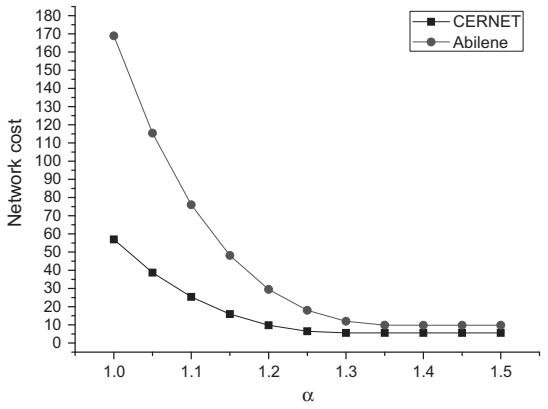

Fig. 9. Impact of α on network cost when α of two ASes are the same. Two ASes have the same topology. The delay between two ASes is shown in Fig. 7. Each AS receives 10 Gbps from the other AS, and the sending traffic of each RLOC of each AS is 5 Gbps. The received traffic of each RLOC is uniformly distributed over the inside routers of the AS. *X*-axis is α. *Y*-axis is the network cost of two ASes.

5.4. Impact of α on network cost

Fig. 9 shows the impact of α on network cost when α of two ASes are the same. First, the network costs of the two ASes are the same because they have the same parameters and have the same status in the optimization objective (Eq. (15)). As α gradually increases, each AS's network cost goes down when $\alpha \leqslant 1.3$ in Abilene (1.25 in CERNET). This is because each edge AS can sacrifice more delay performance of some flows for improving incoming traffic engineering performance. When $\alpha > 1.3$ and increases, each AS's network cost becomes optimal, the further increase of α can no longer decrease its network cost.

Fig. 10 shows the mpact of α on network cost when α of ASes are different. In Fig. 10(a) where two ASes exchange traffic between each other, when $\alpha_1 = \alpha_2 = 1.1$, the network cost of AS_1 and AS_2 are the same (76.1). As α_1 increases and α_2 keeps the same which means that AS_1 is willing to sacrifice more delay performance for cooperative incoming traffic engineering. The network cost of AS_1 decreases and the network cost of AS_2 increases. The increase of α_1 changes the cooperative status between AS_1 and AS_2. AS_1 becomes more and more cooperative and should decrease more network cost from cooperation, which is contributed by AS_2. But AS_2 can less decrease its network cost. Thus, the network cost of AS_2 increases compared to the scenario when $\alpha_1 = \alpha_2$. When $\alpha_2 = 1.2$ and $\alpha_2 = 1.3$, we have similar conclusions.

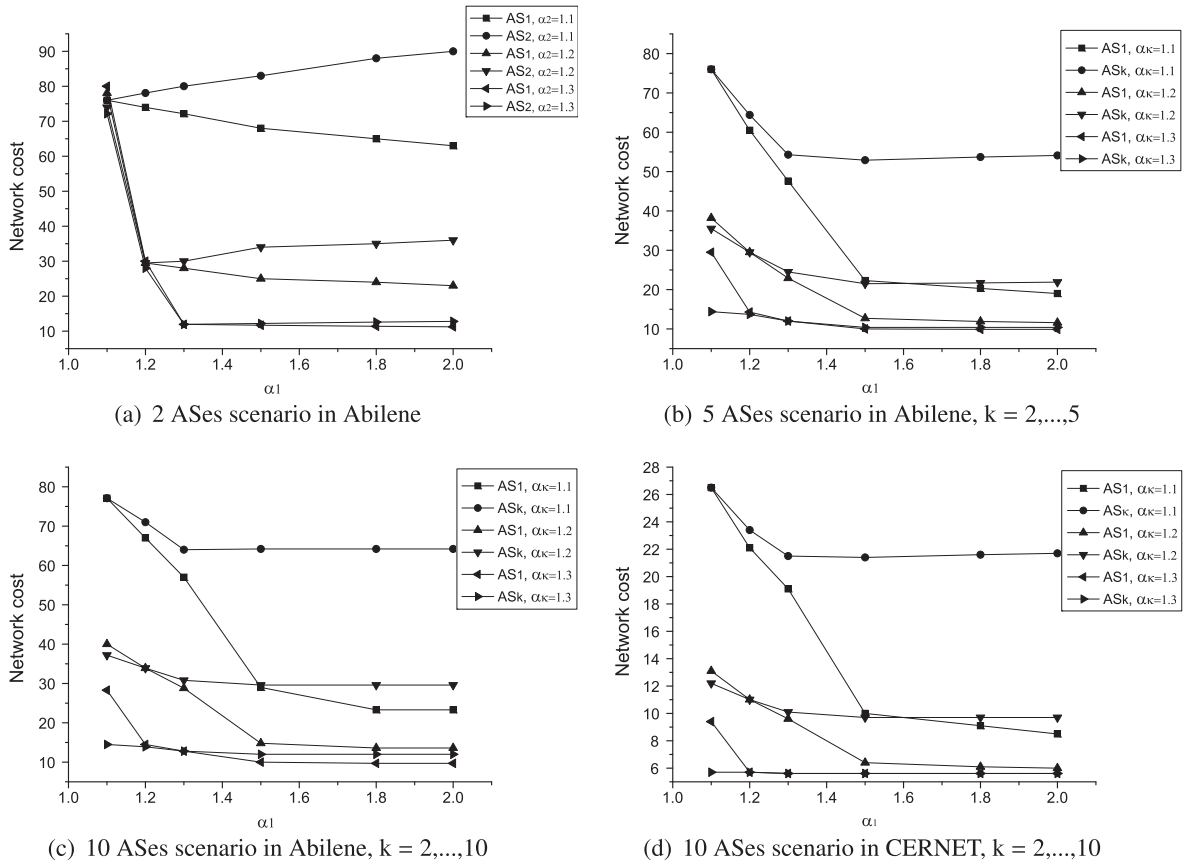

Fig. 10. Impact of α on network cost when α of ASes are different. The ASes have the same topology. The delay between each two ASes is shown in Fig. 7. Each AS receives 10 Gbps which is uniformly from the other ASes, and is uniformly distributed over the inside routers of the AS. *X*-axis is α. *Y*-axis is the network cost of the ASes.

When the number of ASes increases, changing the cooperative status of one AS in the cooperative group becomes more difficult. In the scenario of two ASes, the increase of α of one AS will be conducive to lowering its network cost while the network cost of the other AS will remarkably increases (although this increase will not exceed that in the initial state).

In the scenario of more ASes, the increase of α will be still conducive to lowering its network cost, but the network costs of the other ASes will slightly increase. This is because the increase of α of one AS will not have so much contributions to all the ASes on average. When there are ten ASes cooperating with each other, the network cost of other ASes even ceases to rise, as shown in Fig. 10(b)–(d).

6. Related work

There are some related work about cooperative traffic engineering in BGP and LISP. [6,8,9,29–34] studied the cooperation between ISPs in BGP. [30–32] focused on cooperative traffic engineering between ASes belonging to one ISP. [6,8] modeled the cooperative traffic engineering in BGP protocol. [29,33] considered the traffic engineering between different ASes with the same optimization objectives. [9] analyzed how neighboring ISPs can cooperate with each other for inter-domain traffic engineering. [35] provided a cooperative mechanism for traffic engineering between ASes. [34] proposed a cooperative method based on a series of learning process.

Some works have proposed methods for incoming traffic engineering under LISP. [36] presented an open and flexible solution that allows an ISP using identifier/locator separation to engineer its inter-domain traffic. [37] focused on incoming traffic engineering based on mapping assignment under LISP. These works are applicable in a cooperative Internet environment. In a simple two-AS scenario, two edge ASes can directly negotiate with each other for joint incoming traffic load balance optimization, which makes sense when traffic between two edge ASes are balanced (e.g., similar bit rates). This scenario has been analyzed in previous works [1,38]. Specifically, [1] considered the selfishness of ASes and modeled traffic engineering under LISP as a noncooperative game theory in a two-AS scenario and proposed an enforced load balance method for incoming traffic engineering. In [38], the author extends the cooperation from two edge ASes to multi-ASes. Specifically, a multi-AS cooperation is done by each two edge ASes which have significant mutual traffic between each other. Both [1,38] use abstract utilities for the optimization objective. Different from these works, our model is formulated by specific network parameters instead of abstract utility. Besides, the concept

of multi-AS cooperation in [38] is to decompose the multi-AS cooperation into a serial of binary-AS cooperation process. Our work builds an optimization objective, for which all edge ASes join the cooperation and optimize the same optimization objective simultaneously.

7. Conclusion

In this paper, we first analyze the potential conflict of ETR selections of edge ASes under LISP when they pursue both delay performance and incoming traffic engineering performance. Then, we propose the cooperative incoming traffic engineering framework for edge ASes in the transit edge separation network, where edge ASes sacrifice limited delay performance for optimizing incoming traffic engineering performance between each other. We formulate the cooperative incoming traffic engineering with a heuristic cooperative game optimization objective, where delay performance loss is considered as the contribution of edge ASes. Furthermore, we evaluate delay variations between different RLOCs with experiments on PlanetLab, and verifies the rationality of considering delay performance when optimizing incoming traffic engineering performance. Finally, we evaluate the performance of cooperative incoming traffic engineering by simulations driven by real Internet delay data. Simulations show that our algorithm fairly alleviates the unbalance of incoming traffic in a selfish multi-AS scenario.

Future work lies in the following two directions. First, it is necessary to systematically analyze the gaming phenomenon in the Internet, especially when different aspects of performance of ASes are considered. Secondly, because the number of ASes is neither very small (2 or 3) nor very large (infinite), it is interesting to explore the cooperative mechanism design among the finite number of ASes.

Acknowledgements

We thank our editor Doctor Chadi Barakat and the anonymous reviewers for their detailed and insightful comments that greatly improved the paper. We thank Heli Wu and Zheng Yang for PlanetLab Europe setup and slice allocations. This work is supported by the National Basic Research Program of China (973 Program) under grants 2013CB329105.

Appendix A

We prove Eq. (15) is a convex function. The first derivative of Eq. (15) is shown in Eq. (26). The second derivative of Eq. (15) is shown in Eq. (27).

$$\frac{\partial z}{\partial q_{ji}(t,s)} = \prod_{j\neq i}(\psi_i' - \psi_i)^{\sum_k^{\alpha_j}\alpha k} \frac{\alpha_i}{\sum_k \alpha_k} \times (\psi_i' - \psi_i)^{(\sum_k^{\alpha_i}\alpha k - 1)} (-1)\frac{\partial \psi_i}{\partial q_{ji}(t,s)}. \tag{26}$$

$$\frac{\partial^2 \psi_i}{\partial q_{ji}(t,s)^2} = \prod_{j\neq i}(\psi_i' - \psi_i)^{\sum_k^{\alpha j}\alpha k}(-1)\frac{\alpha i}{\sum_k \alpha_k} \times \left(\frac{\alpha i}{\sum_k \alpha k} - 1\right)(-1)(\psi_i' - \psi_i)^{(\sum_k^{\alpha i}\alpha k - 2)}\left(\frac{\partial \psi_i}{\partial q_{ji}(t,s)}\right)^2 + (\psi_i' - \psi_i)^{(\sum_k^{\alpha i}\alpha_k - 1)}\frac{\partial^2 \psi_i}{\partial q_{ji}(t,s)^2}. \tag{27}$$

$$\frac{\partial \psi_i}{\partial q_{ji}(t,s)} = \frac{\partial \psi_i}{\partial r_s}\frac{\partial r_s}{\partial q_{ji}(t,s)}. \tag{28}$$

$r_i(s)$ is the total incoming traffic volume into sth RLOC of AS_i. So $r_i(s)$ is the linear combination of $q_{ji}(t,s)$,as shown in Eq. (29). As a result, Eq. (30) holds.

$$r_i(s) = \sum_j \sum_{t\in N_j} q_{ji}(t,s). \tag{29}$$

$$\frac{\partial r_i(s)}{\partial q_{ji}(t,s)} = 1. \tag{30}$$

Because ψ_i is the network cost the AS_i, which is nonlinearly increase as the load increase. That is, as $r_i(s)$ increases, the network cost increases nonlinearly, which is shown in Eqs. (31) and (32).

$$\frac{\partial \psi_i}{\partial r_i(s)} > 0. \tag{31}$$

$$\frac{\partial^2 \psi_i}{\partial r_i(s)^2} > 0. \tag{32}$$

Eq. (33) holds because of the increasing delay factor α is normalized. Eq. (34) holds because the network cost before cooperation is not less than that after cooperative incoming traffic engineering between edge ASes. We substitute Eqs. (32)–(34) into Eq. (27) and prove that $\frac{\partial^2 z}{\partial q_{ji}(t,s)^2} < 0$.

$$\frac{\alpha_i}{\sum_k \alpha_k} < 1, \quad \forall i. \tag{33}$$

$$\psi_i' - \psi_i > 0, \quad \forall i. \tag{34}$$

References

[1] S. Secci, K. Liu, G.K. Rao, B. Jabbari, Resilient traffic engineering in a transit-edge separated internet routing, in: ICC, 2011, pp. 1–6.
[2] B. Quoitin, C. Pelsser, O. Bonaventure, S. Uhlig, A performance evaluation of BGP-based traffic engineering, Int. J. Network Manage. 15 (3) (2005) 177–191.
[3] N. Spring, R. Mahajan, T. Anderson, The causes of path inflation, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, August 2003, pp. 113–124.
[4] J.W. Stewart, BGP4: Inter-Domain Routing in the Internet, AddisonWesley, 1998.
[5] B. Quoitin, C. Pelsser, O. Bonaventure, S. Uhlig, A performance evaluation of BGP-based traffic engineering, Int. J. Network Manage. 15 (3) (2005) 177–191.
[6] G. Shrimali, A. Akella, A. Mutapcic, Cooperative interdomain traffic engineering using nash bargaining and decomposition, IEEE/ACM Trans. Networking 18 (2) (2010) 341–352.
[7] J. Winick, S. Jamin, J. Rexford, Traffic Engineering between Neighboring Domains, 2002.

[8] R. Mahajan, D. Wetherall, T. Anderson, Mutually Controlled Routing with Independent ISPs, NSDI, April 2007.
[9] R. Mahajan, D. Wetherall, T. Anderson, Negotiation-based routing between neighboring ISPs, in: Proc. 2nd Conference on Symposium on Networked Systems Design Implementation, 2005, pp. 29–42.
[10] R.T. Ma, D.M. Chiu, J. Lui, V. Misra, D. Rubenstein, Internet economics: the use of Shapley value for ISP settlement, IEEE/ACM Trans. Networking 18 (3) (2010) 775–787.
[11] D. Farinacci, V. Fuller, D. Lewis, D. Oran, Locator/id separation protocol (LISP), Internet Draft, January 2013. <http://tools.ietf.org/html/rfc6830>.
[12] N. McKeown, Software-defined networking, in: INFOCOM, 2009.
[13] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J.D. Thornton, D.K. Smetters, E. Yeh, Named data networking (ndn) project, Xerox Palo Alto Research Center-PARC, 2010.
[14] O. Bonaventure, Reconsidering the Internet Routing Architecture, Internet Draft, March 2007. <http://www.ietf.org/internet-drafts/draft-bonaventureirtf-rrg-rira-00.txt>.
[15] R. Moskowitz, P. Nikander, Host Identity Protocol (HIP) Architecture, RFC 4423, May 2006.
[16] E. Nordmark, M. Bagnulo, Level 3 Multihoming Shim Protocol, Internet Draft, May 2006. <www.ietf.org/internet-drafts/draft-ietf-shim6-proto-07.txt>.
[17] http://tools.ietf.org/html/rfc6830.
[18] Y. Wang, J. Bi, J. Wu, Empirical analysis of core-edge separation by decomposing Internet topology graph, in: GLOBECOM, 2010, pp. 1–5.
[19] Facebook LISP Deployment. <http://www.nanog.org/meetings/abstract?id=1633>.
[20] B. Fortz, M. Thorup, Internet traffic engineering by optimizing OSPF weights, in: INFOCOM (Tel-Aviv, Israel), March 2000.
[21] B. Fortz, J. Rexford, M. Thorup, Traffic engineering with traditional IP routing protocols, Commun. Mag. 40 (10) (2002) 118–124.
[22] R.B. Myerson, Game Theory: Analysis of Conflict, Harvard Univ. Press, Cambridge, MA, 1991.
[23] H. Boche, M. Schubert, N. Vucic, S. Naik, Non-symmetric nash bargaining solution for resource allocation in wireless networks and connection to interference calculus, in: Proc. 15th European Signal Processing Conference, 2007.
[24] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge Univ. Press, Cambridge, 2004.
[25] S. Boyd, L. Xiao, A. Mutapcic, Subgradient Methods, Lecture Notes of EE392o, Stanford University, Autumn Quarter, vol. 2004, 2003.
[26] PlanetLab Website. <http://www.planet-lab.org/>.
[27] Abilene Website. <http://totem.run.montefiore.ulg.ac.be/datatools.html>.
[28] CERNET Website. <http://www.edu.cn/bw_6470/20130228/t20130228_909080_1.shtml>.
[29] X. Xiao, L.M. Ni, Internet QoS: a big picture, IEEE Network 13 (2) (1999) 8–18.
[30] M. Afergan, J. Wroclawski, On the Benefits and Feasibility of Incentive Based Routing Infrastructure in ACM SIGCOMM PINS, September 2004, pp. 197–204.
[31] R. Mortier, I. Pratt, Incentive based inter-domain routing, in: Proceedings of the Internet Charging and QoS Technology Workshop, 2003, pp 308–317.
[32] J. Feigenbaum, C. Papadimitriou, R. Sami, S. Shenker, A BGP-based mechanism for lowest-cost routing, Distrib. Comput. 18 (1) (2002) 61–72.
[33] S. Machiraju, R.H. Katz, Verifying Global Invariants in Multi-Provider Distributed Systems, in: Proc. SIGCOMM Workshop on HotNets, November 2004, pp 149–154.
[34] K. Suksomboon, P. Pongpaibool, Y. Ji, C. Aswakul, PC-Nash: QoS provisioning framework with path-classification scheme under Nash equilibrium, Comput. J. 54 (6) (2011) 931–943.
[35] P. Jacob, B. Davie, Technical challenges in the delivery of interprovider QoS, IEEE Commun. Mag. 43 (6) (2005) 112–118.
[36] D. Saucez, B. Donnet, L. Iannone, O. Bonaventure, Interdomain traffic engineering in a locator/identifier separation context, in: INM 2008, 2008.
[37] Li Ke, S. Wang, X. Wang, Edge router selection and traffic engineering in LISP-capable networks, Commun. Networks 13 (6) (2011) 612–620.
[38] S. Secci, K. Liu, B. Jabbari, Efficient inter-domain traffic engineering with transit-edge hierarchical routing, Comput. Networks (2012).

Yaodong Zhang received his B.S. degree from Beijing University of Posts and Telecommunications in 2010. He is currently a Ph.D. student in the Department of Electronic Engineering at Tsinghua University, Beijing, China. His current research interest is in network traffic engineering.

Yue Wang received his B.S. and Ph.D. degrees from the Electronic Engineering Department of Tsinghua University in 1999 and 2005, respectively. He is now an associate professor at Tsinghua University. His research interests include computer networks, data fusion, and complex networks.

Dan Pei received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1997 and 2000, respectively, and the Ph.D. degree from the University of California, Los Angeles (UCLA), in 2005. He is now an associate professor at Tsinghua University. His current research interests are network measurement and security.

Jian Yuan received his Ph.D. degree in electrical engineering from the University of Electronic Science and Technology of China, in 1998. He is currently a professor in the Department of Electronic Engineering at Tsinghua University, Beijing, China. His main research interest is in complex dynamics of networked systems.