Distributed Out-bound Load Balancing in Inter-AS Routing by Random Matchings

Ravi Musunuri Jorge A. Cobb
Department of Computer Science
The University of Texas at Dallas
Richardson, TX-75083-0688
Email: {musunuri, cobb}@utdallas.edu

Abstract

The Internet is composed of a collection of inter-connected and self-administered Autonomous Systems (ASms). Inter-AS routing is accomplished by having neighboring ASms exchange reachability information via the Border Gateway Protocol. An AS is said to be a transit AS if it allows traffic from other ASms to cross through it. In particular, transit ASms provide transit services for traffic between customer and provider ASms.

In this article, we focus on maximizing the utilization of resources at transit ASms. In particular, inter-AS links have been shown to be a bottleneck. To make better use of inter-AS links, we consider the problem of balancing the load among inter-AS links. We refer to this problem as the Balanced-Flow Assignment ProbleM (B-FAPM). We show that the B-FAPM is NP-hard, and thus, likely intractable. We then present a heuristic protocol, the Balanced-Flow Assignment ProtocoL (B-FAPL), that balances the out-bound traffic loads on inter-AS links. We show via simulation that the B-FAPL effectively balances outgoing traffic over inter-AS links. Our solution is fully distributed and uses random matchings to assign in-bound flows to out-bound inter-AS links.

I. Introduction

The Internet is composed of a collection of inter-connected and self-administered Autonomous Systems (ASms). Routing in the Internet is divided into two categories: intra-AS routing and inter-AS routing. On one hand, intra-AS routing protocols, such as RIP [Malkin, 1998] and OSPF [Moy, 1998], are used to share reachability information between any two routers within the same AS. On the other hand, inter-AS routing protocols are used to advertise reachability information between ASms. The global nature of

inter-AS routing requires that all ASms execute the same inter-AS routing protocol. The protocol chosen for inter-AS routing in the Internet is the Border Gateway Protocol (BGP) [Rekhter and Li, 1995].

Traffic bottlenecks in an AS increase congestion and deteriorate the service provided by the AS to its customers. In particular, inter-AS links commonly cause bottlenecks in transit ASms [Bressoud et al., 2003]. To alleviate this problem, transit traffic through an AS should be balanced among its inter-AS links. In addition, balanced traffic reduces the utilization at each inter-AS link, and thus, each link is able to better absorb temporary increases in traffic.

Future plans for the Internet include the support of real-time applications such as Voice over IP, Internet TV etc. To support these applications, traffic engineering support is required for both intra-AS and inter-AS routing protocols. By providing load balancing across inter-AS links, each inter-AS link has a greater probability of maintaining spare bandwidth to support QoS reservations for real-time applications.

Several methods have been proposed to improve load balancing over intra-AS links [Fortz and Thorup, 2000], [Fortz et al., 2002], [Apostolopoulos et al., 1999], [Guerin et al., 1997]. The load-balancing techniques proposed in [Fortz and Thorup, 2000], [Fortz et al., 2002] change the costs of intra-AS links to direct inter-AS traffic. However, BGP path selection is based on many path attributes. Hence, changing intra-AS costs may not suffice to balance the loads over intra-AS links.

Other solutions attempt to provide QoS in inter-AS routing [Xiao et al., 2002] in a manner similar to QoS extensions proposed for intra-AS routing [Apostolopoulos et al., 1999], [Guerin et al., 1997]. QoS extensions are provided by adding QoS metrics to the original routing messages. However, BGP is a complex protocol, whose path selection is based on many path attributes, and the interaction between these path attributes causes many well-known routing anomalies [Griffin et al., 2002], [Cobb and Musunuri, 2004], [Basu et al., 2002], [Musunuri et al., 2004]. The introduction of additional QoS attributes would increase the complexity of BGP and has the potential of introducing new routing anomalies.

Traffic engineering in BGP [Awduche et al., 2002] may also be implemented by controlling the in-bound and/or the out-bound traffic via service agreements between neighboring ASms. Traffic patterns however may vary over time, in violation of the service agreement.

In this article, we first define the problem of out-bound traffic balancing over inter-AS links. We refer to this problem as the Balanced-Flow Assignment ProbleM (B-FAPM). Next, we show that the B-FAPM is NP-hard. We present a heuristic, the Balanced-Flow Assignment ProtocoL (B-FAPL), to solve this problem. Throughout the article, we focus on the case of transit ASms. However, B-FAPL may be easily extended to the case of stub ASms [Uhlig and Bonaventure, 2004]. B-FAPL uses random matchings [Ghosh and Muthukrishnan, 1996] to assign in-bound flows to out-bound inter-AS links. In addition, B-

FAPL has the desirable properties of being distributed and scalable. Finally, we show via simulation that the B-FAPL effectively balances outgoing traffic over inter-AS links.

II. INTER-AS ROUTING: BGP

In order for each AS to learn a path to all other ASms, neighboring ASms exchange routing information via the Border Gateway Protocol (BGP) [Rekhter and Li, 1995]. A distinguishing feature of BGP is that each router advertises, for each destination prefix, the full path of ASms that are traversed to reach the destination prefix. BGP is thus referred to as a path-vectoring protocol. The motivation for choosing path-vectoring as the basis for BGP, as opposed to more traditional approaches such as link-state or distance vectors, is the avoidance of routing loops and the ability to implement flexible routing policies.

Each BGP router establishes a peering session with other BGP routers. A peering session is said to be internal if both peers are contained in the same AS. A peering session is said to be external if the peers are located in different ASms, and furthermore, they are joined directly by an inter-AS link. BGP routers with external peering sessions are said to be border routers, because they lie at the "border" of the AS.

Assume a router R is located in AS v, and it receives an advertised path P from a peer, where path P leads to destination prefix d. Then, the advertised path contains the following attributes.

- $local\_pref$: A preference value indicating the ranking of P in the local routing policy of AS v. A larger preference value indicates a greater preference for the path. This attribute is exchanged only if two peers belong to the same AS.
- AS_path: Sequence of ASms along the path to reach destination prefix d from the current AS v.
- MED: For a pair of ASms connected by more than one link, the Multi-Exit Discriminator (MED) value indicates the preference of one link over another. A smaller MED value indicates a greater link preference.
- $next\_hop$: The IP address of the next-hop border router. If the router R is an interior router, then $next\_hop$ is the IP address of the border router that is the exit point from AS v. If the router R is a border router, then $next\_hop$ is the IP address of the border router that is the entry point into the neighboring AS.

From each peer, a router receives a path (potentially empty) to reach each destination prefix. From this set of paths, the router must choose the "best" path and adopt it as its own path. The best path to reach some destination d is chosen according to the algorithm given in Fig. 1 [Basu et al., 2002]. If a router adopts a new path, i.e., if its best path is not its previously chosen path, then the router informs each of its peers about the newly chosen path.

best(input A: set of paths advertised by peers to reach d)

{

1) A is reduced to only those paths with largest $local\_pref$ value.
2) If $|A| > 1$, then reduce A to those paths with least AS_path sequence length.
3) If $|A| > 1$, then separate A into disjoint subsets, where all paths in a subset exit via the same neighboring AS. Reduce each subset to those paths with smallest MED value. Set A to the union of the reduced subsets.
4) If $|A| > 1$, then:
   a) If A has at least one path whose $next\_hop$ is an external peer, then the router reduces A to those paths whose $next\_hop$ is an external peer.
   b) If A has no paths whose $next\_hop$ is an external peer, then the router reduces A to those paths whose intra-AS cost from itself to the path's border router is the least.
5) Finally, if $|A| > 1$, then use some deterministic tie breaker to reduce A to a single element.
6) The best path is the single element in A.

}

Fig. 1. Best Path Selection Algorithm

III. Problem Definition

Consider an AS v that provides transit service for traffic destined to l prefixes. We denote these prefixes as $p_1, p_2, \ldots, p_{l-1}, p_l$. We assume AS v contains m border routers, which are denoted by $b_1, b_2, \ldots, b_{m-1}, b_m$.

We assume each inter-AS link is assigned an agent in charge of balancing the traffic load. Throughout the article, we use the terms agent and inter-AS link interchangeably. Let AS v contain n agents, which are represented as $a_1, a_2, \ldots, a_{n-1}, a_n$. The outgoing capacity of an agent a_i is denoted by $c(a_i)$. Furthermore, the set of destination prefixes that are reachable from agent a_i (through its external peer) is denoted by $pf(a_i)$.

Each agent a_i maintains two matrices, *t-in*$_i$ and *t-out*$_i$, as shown in the Fig. 2. Matrix *t-in*$_i$ stores the in-bound traffic information of agent a_i, while matrix *t-out*$_i$ stores the out-bound traffic information of agent a_i.

Matrix *t-in*$_i$ is indexed by destination prefix, and it returns the in-bound traffic volume of agent a_i destined to this prefix, as shown in Fig. 2. Before explaining matrix *t-out*, we define the term $flow$.

Definition 1: A flow is a tuple $\langle a_j, p_x \rangle$ representing the traffic entering via agent a_j and destined to

t-in$_i$	
prefix	traffic
...	...
p_x	*t-in*$_i[p_x]$
...	...

t-out$_i$	
$\langle agent, prefix \rangle$	traffic
$\langle \dots \rangle$	...
$\langle a_j, p_x \rangle$	*t-out*$_i[a_j, p_x]$
$\langle \dots \rangle$	...

Fig. 2. Traffic Matrices at Agent i

prefix p_x.

Each row in *t-out*$_i$ stores a flow and the corresponding amount of traffic of the flow that exits via agent a_i. The total out-bound traffic at an agent a_i is denoted by $t(a_i)$. That is,

$$t(a_i) = \sum_{j,x} \textit{t-out}_i[a_j, p_x]. \tag{1}$$

The load at agent a_i is calculated as follows.

$$load(a_i) = \frac{t(a_i)}{c(a_i)} \tag{2}$$

Note that the selection of a best path, according to Fig. 1, is influenced by intra-AS link costs, as follows. From step 4(a) in Fig. 1, border routers prefer a path advertised by an external peer, provided the paths advertised by internal peers are equally preferable until step three of the algorithm. Those routers not choosing a path via an external peer, from step 4(b), choose the path advertised by the nearest border router according to intra-AS cost values.

In general, the intra-AS cost value [Cisco Systems, 1997] assigned to each link is inversely proportional to the capacity of the link, and does not consider traffic demands. Since loads on inter-AS links depend on the choice of intra-AS cost values, BGP may not provide balanced loads on out-bound inter-AS links.

The above observation leads us to define the Balanced-Flow Assignment ProbleM (B-FAPM) as follows. Given the *t-in* matrix associated with each agent, the *t-out* matrix at each agent must be found such that following conditions hold.

1) For all i and x, *t-in*$_i[p_x] > 0$ implies

$$\textit{t-in}_i[p_x] = \sum_{j, j \neq i} \textit{t-out}_j[a_i, p_x]$$

2) For all i,j, and x, *t-out*$_i[a_j, p_x] > 0$ implies both of the following.

- Prefix p_x is reachable through an external peer at the agent a_i, i.e. $p_x \in pf(a_i)$.
- *t-in*$_j[p_x] > 0$.

3) The standard deviation (σ_L) of the loads at the agents should be minimized, where

$$\sigma_L = \sqrt{\frac{\sum_{i=1}^{n} (\overline{load} - load(a_i))^2}{n}} \tag{3}$$

Where $\overline{load}$ denotes the average load at all the agents.

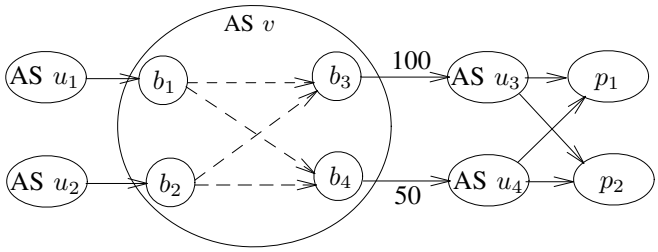

Fig. 3. Example Autonomous System v

A. Example

To explain the problem more clearly, let us consider the example shown in Fig. 3. AS v consists of four border routers and four inter-AS links. Each inter-AS link between $(AS\ u_i, b_i)$ is associated with an agent a_i. Agents a_1 and a_2 have the in-bound traffic to the reach destination prefixes p_1 and p_2, while agents a_3 and a_4 have the external paths to reach destination prefixes p_1 and p_2.

As shown in Fig. 3, the out-bound capacities of agents a_3 and a_4 are 100 and 50 units, respectively. The *t-in* matrices of agents a_1 and a_2 are shown in Fig. 4. Agent a_1 receives 30 units of traffic destined to prefix p_1 and 15 units of traffic destined to prefix p_2. Agent a_2 receives 10 units of traffic destined to prefix p_1 and 20 units of traffic destined to prefix p_2. Matrix *t-in* is empty (i.e., all elements are zero) for both a_3 and a_4.

AS v should route its in-bound traffic such that the out-bound traffic load on the inter-AS links is as balanced as possible. One such solution, shown in the Fig. 5, is as follows. Agent a_1 routes its in-bound traffic destined to p_1 through the agent a_3 and the in-bound traffic destined to p_2 through the agent a_4. Agent a_2 routes its in-bound traffic destined to p_1 through a_4 and the in-bound traffic destined to p_2

t-in$_1$	
prefix	traffic
p_1	30
p_2	15

t-in$_2$	
prefix	traffic
p_1	10
p_2	20

Fig. 4. Traffic-in Matrices

t-out$_3$	
$\langle agent, prefix \rangle$	traffic
$\langle a_1, p_1 \rangle$	30
$\langle a_2, p_2 \rangle$	20

t-out$_4$	
$\langle agent, prefix \rangle$	traffic
$\langle a_1, p_2 \rangle$	15
$\langle a_2, p_1 \rangle$	10

Fig. 5. Traffic-out Matrices

through a_3. Total out-bound traffic at a_3 is equal to 50 units. Hence, the load at the agent a_3 is equal 50%. Similarly, the total out-bound traffic at a_4 is equal to 25 units. Hence, the load at the agent a_4 is also equal to 50%.

B. Assumptions

We use the following assumptions in our B-FAPM.

- Internal BGP (IBGP) uses the full-mesh peering scheme [Musunuri et al., 2004], i.e., every border router advertises its chosen best path to every other router inside its AS.
- Agents are time synchronized. In particular, different phases in our protocol are time synchronized.
- To support traffic engineering, each AS can create Multi Protocol Label Switching (MPLS) tunnels between any entry border router and any exiting border router.
- In-bound traffic, i.e., *t-in* matrices at the agents, is known a-priori and is static. Every AS maintains an estimate of the in-bound traffic information. However, traffic estimates might be different during normal and peak times of the day. These differences can be addressed by solving the problem for each of these times using the in-bound traffic information collected during each of these.

IV. Complexity of B-FAPM

Before presenting our heuristic, we show that B-FAPM is NP-hard by reducing an instance of the Generalized Assignment Problem (GAP) [Shmoys and Tardos, 1993] to an instance of B-FAPM. An instance of the GAP is defined as below:

Given the following:

- J : jobs.
- M : parallel machines.
- $t_{j,m}$: processing time of job j on machine m.
- $c_{j,m}$: cost of processing job j on machine m.
- T_m : total available processing time on machine m.

GAP, if possible, assigns each job to a machine, such that the processing time at a machine m does not exceed T_m and the total processing cost is minimized. GAP remains NP-hard if the processing costs are ignored [Lenstra et al., 1990]. Furthermore, the problem remains NP-hard even with the additional simplifying assumption that processing time is constant on all machines, i.e., under the assumption that $t_{j,m}$ is independent of m [Chekuri and Khanna, 2000]. We thus ignore processing costs, and assume a constant processing time t_j for each job j.

Next, we reduce an instance of GAP into an instance of B-FAPM. Let the B-FAPM have $J+M$ agents and one destination prefix. Let there be J in-bound flows, one per each of J agents. Also, let there be an additional M agents, each of which can reach the destination prefix. We map each of job in GAP to a distinct in-bound flow in B-FAPM, and each machine in GAP to a distinct agent that can reach the destination prefix.

We next address processing times. If a job j is mapped to a flow f, then the constant processing time, t_j, of job j on any machine corresponds to the in-bound traffic volume (bit rate) of flow f in B-FAPM. Lastly, the total available processing time on machine m, T_m, corresponds to the capacity (bit rate) of its corresponding agent. B-FAPM assigns flows to agents such that the capacity of each agent is not exceeded, i.e., so that the available processing time on each machine is not exhausted.

V. Balanced-Flow Assignment Protocol

We next present a distributed heuristic solution to assign the in-bound flows to the out-bound inter-AS links such that the load on the inter-AS links is as balanced as possible. We refer to the heuristic as the Balanced-Flow Assignment ProtocoL (B-FAPL). In-bound flows are given in the form of the *t-in* matrices. B-FAPL finds the out-bound traffic assignment in the form of *t-out* matrices.

Agents in the B-FAPL participate in three phases: the initialization phase, the random matching phase and the flow transfer phase. All these phases are time synchronized at all agents. In the initialization phase, every agent assigns each of its in-bound flows from the *t-in* matrix to an agent at the nearest border router. The nearest border router can be found by using the algorithm in Fig. 1. This is same behavior as in the original BGP protocol.

Next, B-FAPL, at each agent, iteratively calls the next two phases to balance the out-going loads. The random matching and flow transfer phases, shown in Figs. 6 and 7, were motivated by the load balancing algorithms in [Ghosh and Muthukrishnan, 1996], [Ghosh and Muthukrishnan, 1994]. B-FAPL takes two input parameters: P_m and $iter$. The probability of an agent choosing another particular agent in the random matching phase is denoted as P_m. Parameter $iter$ denotes the number of iterations that

each agent should call the random matching and the flow transfer phases. The value of $iter$ depends on the the network topology and the in-bound traffic. The value of P_m is assigned between 0.5 and 0.7.

Random Matching Phase at a_i:

choose a random number p between 0 to 1
if $p \leq P_m$
    randomly choose another agent a_j
    $M_i := \{(a_i, a_j)\}$
inform a_j that a_i has chosen to match with it;
wait for all other agents to choose their match;
if any other a_k has chosen a_i to match with;
    $M_i := M_i \cup \{(a_i, a_k)\}$
for every j and k,
    if $(((a_i, a_j) \wedge (a_i, a_k)) \in M_i) \wedge (a_k < a_j)$
        $M_i := M_i - \{(a_i, a_j)\}$

Fig. 6. Random Matching Phase

In the random matching phase, every agent participates in choosing another agent with whom to match. In the flow transfer phase, matching agents transfer flows between each other. Next, we explain each of these phases in detail.

A. Random Matching Phase

The pseudo-code for the random matching phase at an agent a_i is as shown in Fig. 6. Matching edges are selected in two steps. In step one, each agent generates a uniform, real random variable between 0 and 1. If the generated random variable is less than or equal to P_m, then the agent randomly chooses an agent and creates a tentative matching between itself and the chosen agent. In step two, if an agent is involved in more than one tentative matching, then each agent removes all its tentative matchings except the matching with the smallest id agent.

Random matching is simple, efficient, and does not require any centralized entity coordination. Time complexity of the random matching phase is constant.

Flow Transfer Phase at a_i:

01. if $((a_i, a_j) \in M) \wedge (load(a_i) > load(a_j))$
02. $traf_t := (t(a_i) \cdot c(a_j) - t(a_j) \cdot c(a_i))/(c(a_i) + c(a_j))$
03. $cpf := pf(a_i) \cap pf(a_j)$
04. $xfer := \emptyset$
05. for each agent agt and prefix pfx
06. if $(pfx \in cpf) \wedge$ (*t-out*$_i[agt, pfx] = traf_t$)
07. $xfer := xfer \cup \langle agt, pfx,$ *t-out*$_i[agt, pfx]\rangle$
08. *t-out*$_i[agt, pfx]$:= 0
09. $traf_t$:= 0
10. let *s-out*$_i$ contain the flows of *t-out*$_i$ sorted in non-decreasing order of traffic.
11. k := 1
12. while(($traf_t > 0$) $\wedge$ ($k <$ rows(*s-out*$_i$))
13. let $\langle agt, pfx\rangle :=$ *s-out*$_i[k]$
14. if $(pfx \in cpf) \wedge$ (*t-out*$_i[agt, pfx] \leq traf_t$)
15. $xfer := xfer \cup \langle agt, pfx,$ *t-out*$_i[agt, pfx]\rangle$
16. $traf_t$:= $traf_t$ − *t-out*$_i[agt, pfx]$
17. *t-out*$_i[agt, pfx]$:= 0
18. if $(pfx \in cpf) \wedge$ (*t-out*$_i[agt, pfx] > traf_t$)
19. if $(split = 1 \vee (split = 2 \wedge$ *t-out*$_i[agt, pfx] \geq Thr))$
20. $xfer := xfer \cup \langle agt, pfx, traf_t\rangle$
21. *t-out*$_i[agt, pfx]$:= *t-out*$_i[agt, pfx] - traf_t$
22. $traf_t$:= 0
23. else if $(split = 3)$
24. $traf_t$:= 0
25. $k := k + 1$
26. transfer the flows in $xfer$ to a_j

Fig. 7. Flow Transfer Phase

B. Flow Transfer Phase

Fig. 7 shows the pseudo-code of the flow transfer phase at an agent a_i. Let us assume that agents a_i and a_j have a matching between them, and $load(a_i)$ is greater than $load(a_j)$. In the flow transfer phase, matching agents share their out-going traffic volume with each other. The agent with the higher load calculates the transferable amount of traffic (denoted by $traf_t$). If agent a_i's load is greater, then it should transfer traffic to a_j such that loads at a_i and a_j become equal after the transfer. The transferable amount of traffic from a_i to a_j is calculated by equating the loads at a_i and a_j after the transfer, as follows.

$$\frac{t(a_i) - traf_t}{c(a_i)} = \frac{t(a_j) + traf_t}{c(a_j)}$$

$$t(a_i) \cdot c(a_j) - traf_t \cdot c(a_j) = t(a_j) \cdot c(a_i) + traf_t \cdot c(a_i)$$

$$traf_t = \frac{t(a_i) \cdot c(a_j) - t(a_j) \cdot c(a_i)}{c(a_i) + c(a_j)}$$

Next, agent a_i calculates the common set of prefixes (cpf) that are reachable from both a_i and a_j. This information is available locally at agent a_i, because, we assumed every border router advertises its best path to every other router inside its AS. From steps 4 to 25, agent a_i marks the flows that are transferable. In the end, agent a_i transfers to agent a_j all the flows that are marked.

The actual traffic transferred may be less than $traf_t$ due to following. First, both a_i and a_j should have a non-empty cpf, i.e., the set of prefixes reachable by both agents. If cpf is empty, then a_i may not be able to transfer any traffic to a_j. Second, the actual traffic transferred also depends on the flow-splitting policy of the ISP. Some ISPs support splitting of all the flows [Fortz and Thorup, 2000], i.e., part of the incoming traffic of a flow may exit via some agent, while the remaining part may exit via a different agent. Some ISPs support constrained splitting, in which, a flow is allowed to be split only if the traffic of that flow exceeds some threshold, Thr, while other ISPs do not allow any flow to be split [Ben-Ameur and Gourdin, 2003].

From step 5 to 9, agent a_i searches the *t-out*$_i$ matrix to find a flow whose traffic volume is exactly equal to $traf_t$. If the agent is successful in finding such flow, then it marks that flow as transferable (i.e., adds the flow to set $xfer$) and assigns the required traffic volume, $traf_t$, to zero. If a_i is unsuccessful in finding such a flow, then the marking process continues from step 10. These steps are necessary to avoid unnecessary flow splits.

At step 10, the flows are sorted in order of non-decreasing traffic. The remaining steps iterate over these flows from the lowest traffic flow to the highest traffic flow. The iterations continue until there are no more flows, or until a_i finds enough flows to transfer $traf_t$ units of traffic.

For some flow $\langle agt, pfx \rangle$, if pfx is in set cpf and its traffic *t-out*$_i[agt, pfx]$ is smaller than the remaining $traf_t$ (or equal to $traf_t$), agent a_i adds the flow, $\langle agt, pfx \rangle$, and its traffic volume, *t-out*$_i[agt, pfx]$, to the set of flows to transfer. Also, agent a_i reduces $traf_t$ by the amount of traffic transferred, i.e., $traf_t -$ *t-out*$_i[agt, pfx]$.

For some flow $\langle agt, pfx \rangle$, if pfx is in set cpf and its traffic *t-out*$_i[agt, pfx]$ is greater than the remaining $traf_t$, then there are three cases to consider. These cases depend on the splitting policies of the ISPs. In Fig. 7, variable $split$ stores the splitting policy of the ISP, where 1 = splitting allowed, 2 = threshold splitting, and 3 = no splitting.

Splitting occurs under two conditions: either splitting is allowed ($split = 1$) or there is constrained splitting and the flow has enough traffic to be split ($split = 2 \wedge$ *t-out*$_i[agt, pfx] \geq Thr$). If either of these holds, the flow is split. Thus, the flow, $\langle agt, pfg \rangle$, and the remaining traffic to be transferred, $traf_t$, are added to the set of flows to be transferred. The output traffic of this flow is reduced by the amount that will be transferred (*t-out*$_i[agt, pfx] :=$ *t-out*$_i[agt, pfx] - traf_t$, and $traf_t$ is set to zero.

On the other hand, if the ISP does not allow splitting ($split = 3$), then $traf_t$ is set to zero. This is because all other flows in the iteration will have non-decreasing traffic, and therefore are to be large to be transferred without splitting.

In the flow transfer phase shown in Fig. 7, from line one to four, it takes only constant time. From line five to twenty five, each agent scans each row in the *t-out* matrix twice and sorts *t-out* matrix once. The number of rows in the *t-out* matrix of an agent is at most equal to the total number of prefixes reachable via that agent. Hence, the worst case time complexity of the flow transfer phase iteration is equal to $O(max(\forall i, |pf(a_i)log(pf(a_i))|))$, where $|pf(a_i)log(pf(a_i))|$ is equal to number of prefixes reachable via some agent a_i.

VI. Simulation Study

In this section, we will the study performance of our B-FAPL on the synthetic ISP networks. We will use two example ISPs to compare the performance. In the ISP-1 example, we assume that the AS v has 50 border routers, 25 neighboring ASms, and 300 destination prefixes. In the ISP-2 example, we assume that the AS v has 70 border routers, 35 neighboring ASms, and 1000 destination prefixes. In both the examples, we also assume the following.

- The intra-AS cost values between the pair of border routers is randomly distributed between 10 and 30 units.
- Each neighboring AS will have a path to a randomly chosen set of 5% to 10% of the total destination prefixes.

- Each border router randomly creates an inter-AS link with 10% to 20% of the total neighboring ASms.
- The out-bound capacity of the inter-AS links is randomly distributed between 20 and 60 units in the increments of 10 units.
- Values of input parameters p_m, $iter$ are 0.7 and 100 respectively.

Before presenting the simulation results, lets consider another *coordinated approach* to create the matchings. In the coordinated matching, a centralized entity helps in creating the matchings instead of every agent distributively choosing the matchings. We will use the coordinated matching with full splitting of flows for comparison in our simulation study. In each iteration, central entity divides the agents into two sets A_1 and A_2, where the set A_1 consists of top 50% of the agents with higher loads and the set A_2 consists of bottom 50% of the agents with lower loads. Central entity creates the matchings such that no two agents from the same set A_i are matched. Intuitively, coordinated matching with full splitting should perform better than our B-FAPL, which uses the randomized matchings. But the simulation results show that the performance gain is very small. Next, we will present the simulation results on the ISP-1 example.

We created 300 in-bound random flows from the neighboring ASms with traffic volume ranging from 5 to 20 units. Graph, shown in the Fig. 8, presents variation in σ_L value as the number of iterations ($iter$) increased to 100.

In the graph 8, we compared the σ_L values of three flavors of our B-FAPL, the coordinated matching with full splitting (CM-FS) and the original BGP. Three flavors of B-FAPL include the random matching with full splitting (RM-FS), the random matching with constrained splitting (RM-CS), and the random matching with no splitting (RM-NS). In the RM-CS, threshold value, Thr, is equal to 12.5 units, i.e., flow is allowed to split if it belongs to top 50% of the flows with the higher traffic volume.

Original BGP protocol greedily assigns the in-bound flows to the agents without balancing the loads at the agents. Value σ_L obtained from the original BGP is shown as the straight line. After 100 iterations, the RM-NS, even with no flow splitting policy, decreases the σ_L value up to 52% as compared to the original BGP. If we allow splitting of all the flows, the RM-FS decreases the σ_L value up to 65%. But, If we allow constrained splitting, which allows splitting of only 50% of the flows with higher traffic volume, RM-CS decreases the σ_L value up to 59%. This is important because [Feamster et al., 2003], "in the Internet, traffic destined for the top 10% of prefixes accounts for 70% of the out-bound traffic ". Hence, we can get the balanced loads on the inter-AS links by splitting only a few number of flows. As expected, the RM-FS performs better than the RM-CS and the RM-CS performs better than the RM-NS.

The CM-FS performs slightly better than RM-FS protocol during the first 50 iterations. Reason for this performance gain is as follows. In the CM-FS, there is a better chance of two agents with high load difference being matched. Hence, there will be a higher reduction in the σ_L value. After 50 iterations RM-FS performs slightly better than the CM-FS. Performance of B-FAPL is comparable to CM-FS, which requires centralized entity coordination.

In all three flavors of the B-FAPL, the σ_L value is decreased significantly during the first 30 iterations. Hence, the number of iterations required is relatively linear to the number of border routers.

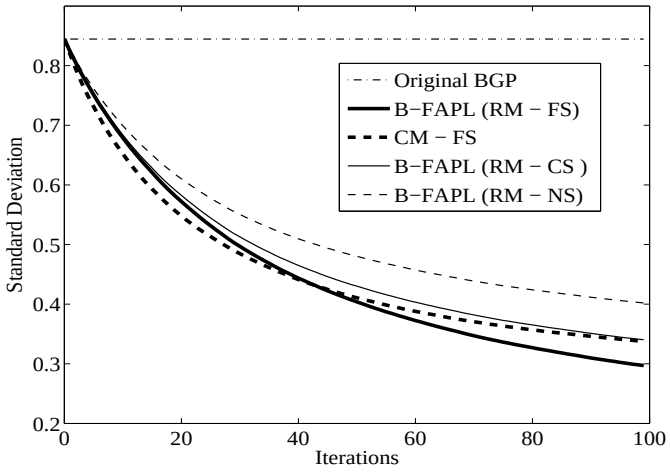

Fig. 8. σ_L Vs $iter$ on ISP-1

In the current Internet, routing table of the BGP aware router contains around 90,000 prefixes [Bressoud et al., 2003], [Feamster et al., 2003]. But Feamster et al. [Feamster et al., 2003] suggested ways to group the prefixes to reduce the scale of the problem. As mentioned before, very few popular prefixes account for major portion of the out-going traffic volume. Hence, we can further reduce the size of the traffic assignment problem by considering only popular prefixes.

Next, we will consider a more realistic ISP-2 example with 1000 prefixes. In the ISP-2, we created 500 in-bound flows randomly from the neighboring ASms with the traffic volume ranging from 5 to 20 units. Graph, shown in the Fig. 9, presents the simulation results on the ISP-2 example. Results obtained for ISP-2 example are very similar to the results in the previous example. Hence, B-FAPL performs well even when the scale of the problem increased.

VII. Load Balancing in Multi-homed Stub ISPs

We can divide ISPs in the current Internet into two different categories. This division is based on whether the ISP transits traffic from neighboring ISPs or the ISP uses services from neighboring ISPs

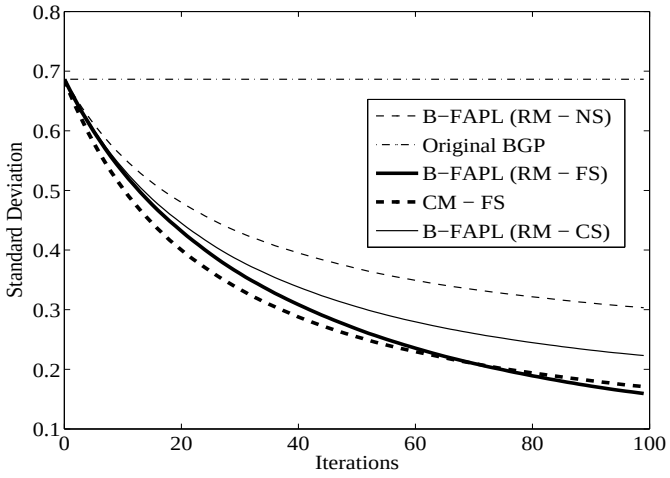

Fig. 9. σ_L Vs $iter$ on ISP-2

t-in$_i$ at source i	
prefix	traffic
...	...
p_x	*t-in*$_i[p_x]$
...	...

t-out$_k$ at agent k	
$\langle source, prefix \rangle$	traffic
$\langle \ldots \rangle$	...
$\langle s_j, p_x \rangle$	*t-out*$_i[s_j, p_x]$
$\langle \ldots \rangle$	...

Fig. 10. Traffic Matrices

to send and receive its traffic. Former category is known as the *transit* ISP and later category is known as the *stub ISP*. Stub ISPs could be multi-homed and single-homed. A multi-homed ISP is connected to more than one upstream provider ISP, where as single-homed ISP is connected to only one upstream provider ISP.

Until now, our article considers load balancing only in transit ISPs. We can extend our work to provide inter-AS outgoing load balancing in multi-homed stub ISPs. For multi-homed stub ISPs, we need to make two important changes to B-FAPM defined in section III. First, we need to change the definition of a flow as following.

Definition 2: A flow is a tuple $\langle s_j, p_x \rangle$ representing the traffic from a source a_j and destined to prefix p_x.

Second, each border router should be associated with an agent instead of inter-AS links. Updated traffic matrices are shown in Fig. 10. Matrix *t-in*$_i$ stores the in-bound traffic information of source s_i, while matrix *t-out*$_k$ stores the out-bound traffic information of agent a_k.

Now, agents iteratively improve the *t-out* matrices by using B-FAPL defined in section V.

VIII. Related Work

Many works address the problem of traffic engineering in intra-AS routing. Fortz et al. [Fortz and Thorup, 2000] studied the problem of assigning intra-AS costs in order to balanced the load on all links. Their solution is based on the local search heuristic. Other solutions [Apostolopoulos et al., 1999], [Guerin et al., 1997] provide QoS by adding QoS metrics to the original routing messages. These solutions do not consider the inter-AS traffic and they don't balance the loads on inter-AS links.

In [Xiao et al., 2002], the authors propose a QoS extension to BGP. In their solution, each BGP update message carries an Available Bandwidth Index (ABI) metric. Their technique is scalable and efficient. However, BGP is already a complex protocol and plagued with many forms of routing anomalies (see [Griffin et al., 2002], [Cobb and Musunuri, 2004], [Basu et al., 2002], [Musunuri et al., 2004]) due to the interaction between path attributes. The introduction of additional QoS attributes would increase the complexity of BGP and has the potential of introducing new routing anomalies. Awduche et al [Awduche et al., 2002] suggested that inter-AS traffic engineering is possible by controlling in-bound and out-bound traffic. But they did not provide any solution to control the traffic.

Bressoud and Rastogi [Bressoud et al., 2003] solved an optimization problem, in which, for each incoming flow, an AS selects an outgoing inter-AS link such that capacity constraint of the inter-AS link is obeyed and intra-AS routing link cost of all incoming flows is minimized. This work considers inter-AS traffic. However, they don't balance the outgoing loads on the inter-AS links and their solution is centralized as opposed to our distributed solution.

Authors in [Uhlig and Bonaventure, 2004] designed an out-bound traffic engineering technique for stub ASms. Their solution is based on an evolutionary algorithm, which solves a multi-objective optimization problem. Their solution deals only with multi-homed stub ASms, as opposed to our solution, which can be used in both stub and transit ASms. Also, their solution requires a centralized coordination entity. B-FAPL is a distributed protocol, and it does not require any centralized coordination. However, B-FAPL does not deal with multi-objective optimizations.

IX. Summary and Concluding Remarks

BGP is the standard inter-AS routing protocol in the Internet. To improve the utilization of resources at transit ASms, we defined B-FAPM and proved that B-FAPM is NP-hard. We proposed a heuristic B-FAPL, which assigns the in-bound flows to the inter-AS links such that out-bound load on the inter-AS links is as balanced as possible. B-FAPL is efficient and distributed. We also extended our work to provide inter-AS load balancing for the case of multi-homed stub ASes.

Future directions for extending our work are as follows. In B-FAPL, each agent creates the matchings without knowledge about the loads at other agents. We would like to investigate matching techniques in which every agent will have partial knowledge about the loads at some random set of other agents. This type of ivestigation is useful, If ASes are using route-reflection clustering to mitigate scalability problems in distributing external BGP paths inside the AS.

In addition, we have assumed that the in-bound flows are static. We would like to investigate the removal of this restriction from B-FAPL to provide online traffic engineering.

REFERENCES

[Apostolopoulos et al., 1999] Apostolopoulos, G., Guerin, R., Kamat, S., Orda, A., Przygienda, T., and Williams, D. (1999). QoS routing mechanisms and OSPF extensions. *IETF RFC-2676*.

[Awduche et al., 2002] Awduche, D. O., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X. (2002). Overview and principles of internet traffic engineering. *IETF RFC-3272*.

[Basu et al., 2002] Basu, A., Ong, C.-H. L., Rasala, A., Shepherd, F. B., and Wilfong, G. (2002). Route oscillations in IBGP with route reflection. In *Proc. of ACM SIGCOMM conference*, pages 235–247.

[Ben-Ameur and Gourdin, 2003] Ben-Ameur, W. and Gourdin, E. (2003). Internet routing and related topology issues. *SIAM Journal on Discrete Mathematics*, 17(1):18–49.

[Bressoud et al., 2003] Bressoud, T. C., Rastogi, R., and Smith, M. A. (2003). Optimal configuration for BGP route selection. In *Proc. of IEEE INFOCOM conference*, volume 2, pages 916–926, San Francisco.

[Chekuri and Khanna, 2000] Chekuri, C. and Khanna, S. (2000). A PTAS for the multiple knapsack problem. In *Proceedings of the ACM-SIAM symposium on Discrete algorithms*, pages 213–222.

[Cisco Systems, 1997] Cisco Systems (1997). Configuring OSPF. *Documentation*.

[Cobb and Musunuri, 2004] Cobb, J. A. and Musunuri, R. (2004). Covergence of inter-domain routing. In *Proc. of IEEE Globecom Conference*, pages 1353 – 1358.

[Feamster et al., 2003] Feamster, N., Borkenhagen, J., and Rexford, J. (2003). Guidelines for interdomain traffic engineering. *SIGCOMM Comput. Commun. Rev.*, 33(5):19–30.

[Fortz et al., 2002] Fortz, B., Rexford, J., and Thorup, M. (2002). Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40(10):118–124.

[Fortz and Thorup, 2000] Fortz, B. and Thorup, M. (2000). Internet traffic engineering by optimizing OSPF weights. In *Proc. of IEEE INFOCOM conference*, volume 2, pages 519–528, Tel Aviv, Israel.

[Ghosh and Muthukrishnan, 1994] Ghosh, B. and Muthukrishnan, S. (1994). Dynamic load balancing in parallel and distributed networks by random matchings (extended abstract). In *Proc. of ACM symposium on Parallel algorithms and architectures*, pages 226–235, New Jersey.

[Ghosh and Muthukrishnan, 1996] Ghosh, B. and Muthukrishnan, S. (1996). Dynamic load balancing by random matchings. *J. Comput. Syst. Sci.*, 53(3):357–370.

[Griffin et al., 2002] Griffin, T. G., Shepherd, F. B., and Wilfong, G. (2002). The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2):232–243.

[Guerin et al., 1997] Guerin, R., Orda, A., and Williams, D. (1997). QoS routing mechanisms and OSPF extensions. In *Proc. of IEEE Globecom conference*, pages 1903–1908, Phoenix.

[Lenstra et al., 1990] Lenstra, J. K., Shmoys, D. B., and Tardos, E. (1990). Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46(3):259–271.

[Malkin, 1998] Malkin, G. S. (1998). RIP version 2. *IETF RFC-2453*.

[Moy, 1998] Moy, J. (1998). OSPF version 2. *IETF RFC-2328*.

[Musunuri et al., 2004] Musunuri, R., , and Cobb, J. A. (2004). Covergence of IBGP. In *Proc. of IEEE ICON Conference*.

[Rekhter and Li, 1995] Rekhter, Y. and Li, T. (1995). A border gateway protocol. *IETF RFC-1771*.

[Shmoys and Tardos, 1993] Shmoys, D. B. and Tardos, E. (1993). An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62(3):461–474.

[Uhlig and Bonaventure, 2004] Uhlig, S. and Bonaventure, O. (2004). Designing bgp-based outbound traffic engineering techniques for stub ases. *ACM SIGCOMM Computer Communications Review*, 34(5):89–106.

[Xiao et al., 2002] Xiao, L., Lui, K.-S., Wang, J., and Nahrstedt, K. (2002). QoS extension to BGP. In *Proc. of IEEE ICNP conference*, pages 100–109, Paris, France.