Efficient Two Dimensional-IP Routing: An Incremental Deployment Design

Shu Yang
Tsinghua University

Dan Wang
Hong Kong Polytechnic University

Mingwei Xu
Tsinghua University

Jianping Wu
Tsinghua University

***Abstract*—China Education and Research Network 2 (CERNET2) is deploying Two Dimensional-IP (TwoD-IP) routing. That is, the forwarding decision of each router is not only based on the destination address, but also on the source address. One driving force is that the network needs the ability to divert traffic flows (identified by their source and destination IPs and we call them the VIP flows) to pre-defined paths (we call them the VIP paths). There are also increasing demands on load balancing, security issues, etc. A pure IP based solution is favored over MPLS.**

An important research issue towards TwoD-IP routing is the deployment of the TwoD-IP routing scheme. It is widely known that making changes to the network layer is notoriously difficult. The proposed scheme should have least impact on the current Internet protocols and infrastructure. A node-by-node incremental deployment scheme is highly preferred. Obviously, without full deployment, the resulting paths for traffic diversion may deviate from the required VIP paths. The incremental deployment scheme should minimize such deviation.

In this paper, we formulate the problem as finding a deployment sequence where the VIP traffic flows should follow the VIP paths given 1) the number of nodes to be deployed and 2) the extra burden each router can spare for TwoD-IP routing. We novelly transform our problem to boolean clauses and develop efficient solutions following the MAX-SAT problem.

After deploying part of the routers, network topology and VIP flows may change, which will enlarge the deviation of the resulting paths from the required VIP paths. To reduce the deviation, we devise a new protocol called PaFid, that makes routers adaptively change their forwarding operations according to topology and VIP flow information. We develop a dynamic programming based algorithm for the adaptive forwarding problem.

We evaluate our algorithms using comprehensive simulations with BRITE generated topologies and real world topologies. We conduct a case study on CERNET2 configurations. Compared to an ad-hoc deployment and an arbitrary TwoD-IP forwarding, our algorithms compute a deployment sequence that achieves close to optimal performance after deploying a few nodes. Besides, our adaptive forwarding protocol can greatly improve the performance when network topology and VIP flows change.

I. Introduction

China Education and Research Network 2 (CERNET2), the world's largest IPv6 backbone network (including 59 Giga-PoPs), provides services to end users in Chinese universities across over 22 major cities. CERNET2 is an operational network, yet it also undertakes experimental purposes for new infrastructure and protocols validation.

CERNET2 is currently deploying source IP functionalities for QoS and security reasons. At the edge routers, CERNET2 has deployed SAVI (Source Address Validation Improvement) [1], where the source address of each packet is checked. SAVI guarantees that each packet will hold an authenticated source IP address, and thus enhances the security of the network.

We are now deploying Two Dimensional-IP (TwoD-IP) routing [2]. More specifically, the forwarding decisions of intermediate routers will be based not only on the destination addresses, but also the source addresses. One driving force is policy routing, i.e., the network has the ability to divert traffic flows (identified by their source and destination IPs and we call them the VIP flows) to pre-defined paths (we call them the VIP paths). CERNET2 has two international exchange centers connecting to the Internet, Beijing (CNGI-6IX) and Shanghai (CNGI-SHIX). For example, we find in operation that CNGI-6IX is very congested with an average throughput of 1.18Gbps in February 2011; and CNGI-SHIX is much more spared with a maximal throughput of 8.3Mbps at the same time.

With an overall considerations on security, load balancing, policy routing, we chose to enhance the network with source IP functionalities. We have developed a prototype router (BitEngine 12004) and are designing TwoD-IP routing with support from National Basic Research Program of China (973).

There are many research issues to address for TwoD-IP routing. An important problem is the deployment of the TwoD-IP routing scheme. Unlike SAVI, which was deployed on edge routers, TwoD-IP routing requires upgrade of the CERNET2. It is widely known that making changes to the network layer is notoriously difficult. The proposed scheme should have least impact on the current Internet protocol stack and infrastructure. A node-by-node incremental deployment scheme is highly preferred.

Clearly, if only partial nodes are deployed, a VIP flow may not strictly follow its VIP path. We need to minimize the deviation. In this paper, we define a deviation that is practically meaningful. We formulate a problem where we need to derive a deployment sequence and minimize the deviation given the number of nodes to be deployed. We show that the problem is NP-complete by reducing it to a dense-k subgraph problem. We then novelly transform our problem to boolean clauses and develop algorithms

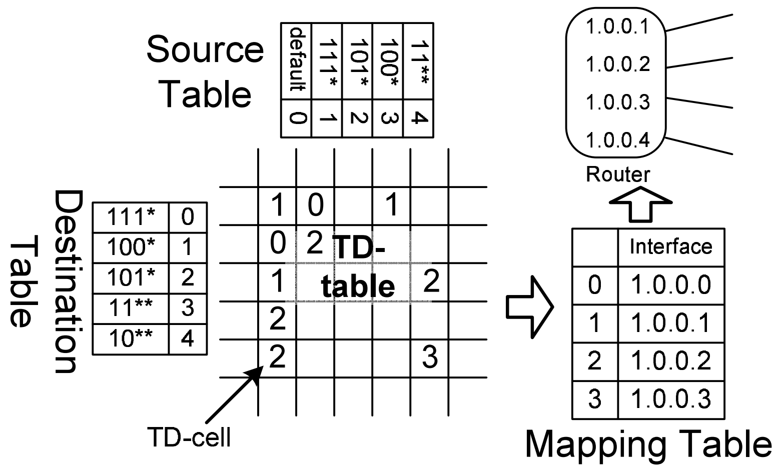


Figure 1: Example of TwoD-IP routing

following the principle of the branch-and-bound algorithm for MAX-SAT. We study several closely related problems for different practical scenarios. We develop efficient algorithms for these problems to compute incremental deployment sequence.

Although we provide guidelines on deployment sequence, the ISPs may do not want to follow the sequence. Even if they follow, network topology and VIP flows may change after deploying. These factors can all enlarge the deviation. To reduce the deviation, we propose to let the deployed routers adaptively change their forwarding operations, according to network topology and VIP information. We develop a protocol call PaFid, which collects the needed information and controls the forwarding operations of routers. We also design a dynamic programming based algorithm, which is optimal for reducing the deviation.

We conduct comprehensive simulations using BRITE generated topologies. We also evaluate our algorithms using the topology of China Education and Research Network (CERNET), a medium scale IPv4 network with 110 routers and 238 links. The results show that our algorithm can achieve close to optimal performance by deploying a few carefully selected nodes. The adaptive forwarding mechanism can greatly improve the performance, when ISPs do not follow the optimal deployment sequence, or network topology and VIP flows change after deploying. Besides, the adaptive forwarding mechanism only stretch a little for the path that VIP traffic flows through.

We carry out a case study with CERNET2 configuration and our primary concerned VIP traffic flows. We suggest the deployment sequence of TwoD-IP routing. By deploying 5 routers, we can successfully divert our concerned VIP traffic flows from the congested CNGI-6IX to CNGI-SHIX.

II. Background and the Optimal Incremental Deployment Problem

A. Background on TwoD-IP Routing

We first give the background of the TwoD-IP routing in our context. In each router, there is a TwoD-IP forwarding table, which is made up of two tables stored in TCAMs and two tables stored in SRAM (see Fig. 1). One table in TCAM stores the destination prefixes (we call it *destination table* thereafter), and the other table in TCAM stores the source prefixes (we call it *source table* thereafter). One table in SRAM is a two dimensional table that stores the indexed next hop of each rule in TwoD-IP (we call it *TD-table* thereafter) and we call each cell in the array *TD-cell* (or in short *cell* if no ambiguity). Another table in SRAM stores the mapping relation of index values and next hops (we call it *mapping-table* thereafter).

When a packet arrives, the router first extracts the source address and destination address. Using the LMF rule, the router finds the matched source and destination prefixes in both source and destination tables that reside in the TCAMs. According to the matched entry, the source table will output a column address and the destination table will output a row address. Combined with the row and column addresses, the router can find a cell in the TD-table, and return an index value. Using the index value, the router looks up the mapping table, and return the next hop that the packet will be forwarded to.

For example, in Fig. 1, a packet with a destination address of 1001 and source address of 1111 will match 100* in the destination table, and 111* in the source table. The destination table will output 1_{st} row, and the source table will output 1_{st} column. The cell in 1_{st} row and 1_{st} column of TD-table is 2, which corresponds to 1.0.0.2 in the mapping table. Thus the packet will be forwarded to 1.0.0.2.

The updates of TwoD-IP forwarding table can be manually configurable for registered VIP flows. Our current

CERNET2 requirement of diverting a few traffic flows will end up in this way. In the future, when more source IP functionalities are needed and VIP traffic flows are more dynamic, a decentralized (e.g., OSPF-like) or centralized (e.g., OpenFlow-like) mechanism can be developed.

B. The Optimal Incremental Deployment Problem

Let $G = (V, E)$ be a network, where V is the set of nodes, and E is the set of links. In this network we have multiple VIP traffic flows. Let $\mathcal{T}$ denote the set of traffic flows, and $t \in \mathcal{T}$ be a VIP traffic flow. Let the source and destination prefixes of t be $P_s(t), P_d(t)$, t can be represented as $< P_s(t), P_d(t) >$. In this paper, we often omit the destination prefix, i.e., only use the source prefix to identify a flow if there is no ambiguity. For each t, the user expects it to travel on a pre-defined VIP path (which usually is not the shortest path). We use $VIP(t) = \{v_t^0, v_t^1, \dots, v_t^j, \dots\}$ to denote the VIP path for t.

For a destination IP prefix P_d (we omit the traffic t if there is no ambiguity). On a node v, we call it *forwarding* an operation $\mathcal{I}_v^d(P_d)$ to map P_d to a set of next hops $\mathcal{I}_v^d(P_d) = \{a_0, a_1, \dots, a_j, \dots\}$, each of which can lead the packets to the destination (satisfying [3] to be loop-free and failure-tolerant). In conventional routing, the result of a forwarding $\mathcal{I}_v^d(P_d)$ is a single next hop on the shortest path.

For a source IP prefix, we call it *Two Dimensional-IP (TwoD-IP) forwarding* as an operation $\mathcal{I}_v^s(P_s)$ to map P_s to a next hop $a_t \in \{a_0, a_1, \dots, a_j, \dots\}$. *TwoD-IP routing* is that for each packet, the routers perform a forwarding operation and a TwoD-IP forwarding operation to find a next hop on the pre-defined VIP path (note that for packets of non-VIP traffic, a forwarding operation already results in a next hop). Let the path of a packet forwarded by TwoD-IP routing be $\mathcal{L}(P_s) = \{v^0, v^1, \dots, v^j, \dots\}$; here we only use the source prefix P_s to denote a packet for simplicity.

If we *deploy* node v, this node is TwoD-IP forwarding capable, i.e., this node can perform operation $\mathcal{I}_v^s(P_s)$. A *deployment* $\mathcal{G}$ is a set of nodes $V' \subseteq V$ that are deployed to be TwoD-IP forwarding capable. If we do not deploy node v, v will use conventional shortest path routing. In a deployment process, let κ be the number of nodes we want to deploy.

Since a deployment $\mathcal{G}$ may include a subset of nodes, the resulting path for a VIP traffic flow t may deviate from its $VIP(t)$. Note that given $\mathcal{G}$, for a VIP traffic t (identified by its P_s), its path is determined. Let $\mathcal{L}(\mathcal{G}, P_s)$ denote such path. Clearly, we want such deviation to be small. To quantify such deviation, In our formulation, we used hamming distance to represent the deviation between paths. While hamming distance is a well-known indicator of the differences between vectors and was widely used by previous works [4], there exists other meaningful metrics. We will consider other metrics and their comparison in our future work. Note that we do not exclude other meaningful metrics and they should be discussed in our future work. We define $D(\mathcal{L}_i, \mathcal{L}_j) = \max\{|\mathcal{L}_i|, |\mathcal{L}_j|\} - |\mathcal{L}_i \cap \mathcal{L}_j|$ the *distance* between two paths $\mathcal{L}_i$ and $\mathcal{L}_j$. Since shortest path is used if a node is not deployed, we always have $|\mathcal{L}(\mathcal{G}, P_s)| \le |VIP(t)|$. Thus,

Observation 1. $D(VIP(t), \mathcal{L}(\mathcal{G}, P_s)) = |VIP(t)| - |VIP(t) \cap \mathcal{L}(\mathcal{G}, P_s)|$.

Our objective is that given the number of routers that we want to deploy, find a deployment that minimizes the total distance of the paths of the VIP flows and their pre-defined VIP paths.

Problem 1. ***Optimal Deployment:*** *Given κ, find a deployment $\mathcal{G}^o$ where $|\mathcal{G}^o| = \kappa$ so that $\sum_t D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s))$ is minimized.*

In practice, according to different volume of the VIP traffic flows, we may need to assign different weights, i.e., w_t for flow t. In our problem formulation, we can add this weight to the distance and modify $\sum_t D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s))$ to $\sum_t w_t D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s))$ to show different importance of t in the aggregated distance. In this paper, we will focus our study on the unweighted problem. Our analysis and solutions will not change in the weighted version. We will briefly mention the weighted case in our case study.

As said, incremental deployment is highly favored in practice. Let $\mathcal{G}_i = V_i$ and $\mathcal{G}_j = V_j$ be two deployments. We call $\mathcal{G}_j$ *incremental* to $\mathcal{G}_i$ if $V_i \subseteq V_j$. An *incremental deployment* is a series of deployment $\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_j, \dots$, such that $\mathcal{G}_j$ is incremental to $\mathcal{G}_i$ if $i < j$. Thus,

Problem 2. ***Optimal Incremental Deployment:*** *Given $\kappa_0, \kappa_1, \dots, \kappa_j, \dots$, find an incremental deployment, $\mathcal{G}_0^o, \mathcal{G}_1^o, \dots, \mathcal{G}_j^o, \dots$, such that $|\mathcal{G}_i^o| = \kappa_i$, and $\mathcal{G}_i^o$ is an optimal deployment.*

Table I: Notation List

Notation	Definition	Notation	Definition
V	set of nodes	$\mathcal{I}_v^d$	forwarding operation
t	traffic flows	$\mathcal{I}_v^s$	TwoD-IP forwarding
$\mathcal{T}$	set of traffic flows	$\mathcal{G}$	a deployment
$VIP(t)$	VIP path for t	$\mathcal{L}(\mathcal{G}, P_s)$	TwoD-IP path
P_s	source IP prefix	κ	deployed number
P_d	destination IP prefix		

Table II: Forwarding operations for the routers c in Fig. 2

Destination	$\mathcal{I}_c^d(\cdot)$
$P_d =$ (11$*$)	$\{e\}$

(a) Conventional

Destination	$\mathcal{I}_c^d(\cdot)$	Source	$\mathcal{I}_c^s(\cdot)$
$P_d =$ (11$*$)	$\{d, e\}$	$P_s(t_0)$	d
		non-VIP	e

(b) TwoD-IP

We illustrate our definitions with an example topology in Fig. 2. Here we assume a is the source and e is the destination. For a traffic flow the shortest path from a to e

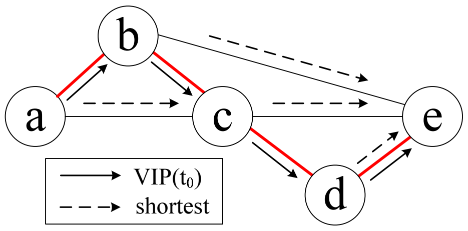


Figure 2: Network topology and VIP paths

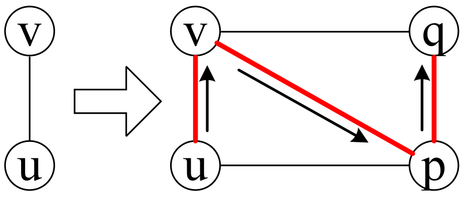


Figure 3: Transformation from a graph

is $\{a, c, e\}$. Assume there is one VIP flow t_0 and one non-VIP flow. Assume the pre-defined VIP path is $VIP(t_0) = \{a, b, c, d, e\}$. As discussed, the VIP flow is identified by its source IP prefix and destination IP prefix. Assume the source and destination prefixes of t_0 are $P_s(t_0) = (00*)$ and $P_d(t_0) = (11*)$; the source and destination prefixes of non-VIP flow is $(10*)$ and $(11*)$. The forwarding operation on node c for conventional routing is shown in Table III(a).

To achieve TwoD-IP routing, node c should forward the packets of t_0 to d, and the packets of non-VIP flow to e, that is, $\mathcal{I}_c^s(P_s(t_0)) = d$. We show the forwarding operation and TwoD-IP forwarding operation on node c for TwoD-IP routing in Table III(b).

Assume we deploy one node, i.e., $\kappa = 1$ and we select c, i.e., deployment $\mathcal{G} = \{c\}$. Then $\mathcal{L}(\mathcal{G}, P_s) = \{a, c, d, e\}$, and $D(VIP(t_0), \mathcal{L}(\mathcal{G}, P_s)) = 5 - 4 = 1$. The optimal deployment is $\mathcal{G}^o = \{c\}$.

III. Optimal Incremental Deployment

We will first develop an algorithm for the optimal deployment given the number of nodes to be deployed (Problem 1). We show it can be naturally extended to solve the optimal incremental deployment (Problem 2).

Theorem 1. *Finding the optimal deployment (Problem 1) is NP-Complete.*

Proof: It is easy to see that the decision problem of validating a given optimal deployment is solvable in polynomial time. Therefore, finding the optimal deployment is in NP class. To show this problem is NP-hard, we reduce the dense-k subgraph problem to it; the former is known to be NP-complete [5].

The dense-k subgraph problem is, given a graph (V, E) and a positive integer k, find a subset $\mathcal{G}_k \subseteq V$ where $|\mathcal{G}_k| = k$, so that the number of induced edges is minimized, i.e., $((u, v) \in E) \cap (\mathcal{G}_k \times \mathcal{G}_k)$.

We expand (V, E) to a network (V', E'), where $V' = \bigcup_{(u,v)\in E}\{u, v, p, q\}$, $E' = \bigcup_{(u,v)\in E}\{(u, v), (u, p), (v, p), (v, q), (q, p)\}$. And we add a VIP traffic flow $t = \{u, v, p, q\}$ for each $(u, v) \in E$. Fig. 3 shows the transformation.

We next show that the optimal deployment $\mathcal{G}^o$ where $|\mathcal{G}^o| = \kappa = k$ in (V', E'), is equal to the solution $\mathcal{G}_k$ to dense-k problem. If $u, v \in \mathcal{G}^o$, then $\mathcal{L}(\mathcal{G}^o, P_s) = \{u, v, p, q\}$; if $u \in \mathcal{G}^o, v \notin \mathcal{G}^o$, then $\mathcal{L}(\mathcal{G}^o, P_s) = \{u, v, q\}$; else $\mathcal{L}(\mathcal{G}^o, P_s) = \{u, p, q\}$ (shortest path from u to q). Thus only if $u, v \in \mathcal{G}^o$, $D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s)) = 0$, else $D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s)) = 4 - 3 = 1$. Obviously, $\sum_t D(VIP(t), \mathcal{L}(\mathcal{G}^o, P_s)) = |E| - |((u, v) \in E) \cap (\mathcal{G}^o \times \mathcal{G}^o)|$. So it is equal to maximize $|((u, v) \in E) \cap (\mathcal{G}^o \times \mathcal{G}^o)|$. Obviously, $\mathcal{G}_k = \mathcal{G}^o$. ■

Though the optimal deployment problem is NP-complete, if κ, the number of nodes to be deployed, is a constant, the problem is polynomial-time solvable even we perform exhaustive search. However, if κ is large, a straight-forward exhaustive search is computationally unacceptable. Therefore, when κ is large, we develop a heuristic where we divide κ into small κ' and find the deployment for each individual κ'. This also naturally leads to an algorithm for the optimal incremental deployment problem.

We first define *passing-through property* of each node. Intuitively, v has passing-through property if v is on both the path of the VIP traffic flow and its pre-defined VIP path.

Definition 1. *Node v has passing-through property for traffic t under deployment $\mathcal{G}$, if $v \in VIP(t) \cap \mathcal{L}(\mathcal{G}, P_s(t))$.*

Clearly, the more nodes have passing-through property, the smaller the total distance. Let $\mathcal{C}(v, P_s(t))$ denote the passing-through property of v for traffic t. We develop Algorithm Passing-Through() to evaluate $\mathcal{C}(v, P_s(t))$ of all nodes. Intuitively, evaluation of passing-through properties of all nodes needs to check all possible deployments $\mathcal{G}$, which is exponential. We develop a novel Algorithm Passing-Through() which does not need to perform an exhaust search in the solution space. We will use Algorithm Passing-Through() as a subroutine to solve the optimal deployment problem.

Let θ_v be an indicator variable for node v, if v gets deployed, $\theta_v = 1$, else $\theta_v = 0$. Our idea is that we do not need to make an assignment to θ at the beginning (which will reflect to a specific deployment). We use this abstract θ and transfer the problem to a generalized MAX-SAT problem. Our solution thus does not need to specify the deployment.

We first use an example in Fig. 2 to explain our idea. For example, node c has (or does not have) passing-through property if and only if $\{(\theta_a \wedge \theta_b) \vee \neg\theta_a\}$ is equal to one (or zero). We see that node c has passing-through property in two conditions: 1) a is not deployed or 2) both a and b are deployed. In condition 1), since a is not deployed, the traffic will follow conventional shortest path routing. In condition

2), since a is deployed, a will perform TwoD-IP forwarding and the traffic will flow the VIP path to node b. Since b is also deployed and will perform TwoD-IP forwarding, the traffic will be forwarded to c. Correspondingly, $\{(\theta_a \wedge \theta_b) \vee \neg\theta_a\}$ is equal to 1 if 1) $\theta_a = 0$ or 2) $\theta_a = 1$ and $\theta_b = 1$. We thus provide a mapping between clause satisfaction and our problem. Similarly, we can see that the passing-through property of node a, b, d, e can be transformed to clauses '1', $\{\theta_a\}$, $\{(\theta_a \wedge \theta_b \wedge \theta_c) \vee (\neg\theta_a \wedge \theta_c)\}$, $\{(\theta_a \wedge \theta_b \wedge \neg\theta_c) \vee (\neg\theta_a \wedge \neg\theta_c) \vee (\theta_a \wedge \theta_b \wedge \theta_c) \vee (\neg\theta_a \wedge \theta_c) \vee (\theta_a \wedge \neg\theta_b)\}$ respectively.

We define $s\_child(v,t)$ as the first successor node that is on both $VIP(t)$ and the shortest path. For example, in Fig. 2, $s\_child(a,t_0) = c$, indicating that c is the first successor node on $VIP(t_0) = \{a,b,c,d,e\}$ and also on the shortest path from a to e. Algorithm Passing-Through() computes the passing-through properties of each node as follows.

Algorithm 1: Passing-Through($VIP(t)$)

```
    Output  : $\mathcal{C}(v, P_s(t)), \forall v \in VIP(t)$
1   begin
2       $\mathcal{C}(v_t^0, P_s(t)) \leftarrow$'1'// $v_t^0$ is the source node
3       $s\_child(v_t^0, t) \leftarrow compute\_s\_child(v_t^0, t)$
4       for $i = 1$ to $|VIP(t)| - 1$ do
5           $s\_child(v_t^i, t) \leftarrow compute\_s\_child(v_t^i, t)$
6           if $v_t^i = s\_child(v_t^{i-1}, t)$ then
                $\mathcal{C}(v_t^i, P_s(t)) \leftarrow \mathcal{C}(v_t^{i-1}, P_s(t))$
7           else $\mathcal{C}(v_t^i, P_s(t)) \leftarrow \mathcal{C}(v_t^{i-1}, P_s(t)) \wedge (\theta_{v_t^{i-1}})$
8           for $j = 0$ to $i - 2$ do
9               if $v_t^i = s\_child(v_t^j, t)$ then $\mathcal{C}(v_t^i, P_s(t)) \leftarrow$
                $\mathcal{C}(v_t^i, P_s(t)) \vee (\mathcal{C}(v_t^j, P_s(t)) \wedge (\neg\theta_{v_t^j}))$
```

The input of Algorithm Passing-Through() is a VIP path $VIP(t)$, and the output is the passing-through properties of each node on $VIP(t)$. Basically, Algorithm Passing-Through() follows a dynamic programming structure. We show an example of Algorithm Passing-Through() using Fig. 2 as the input. We show the last round execution of Algorithm Passing-Through() to compute $\mathcal{C}(e, P_s(t_0))$. As shown in Fig. 2, $s\_child(d, t_0) = e$ and d is the predecessor node of e along $VIP(t_0)$. And node b, c satisfy $s\_child(b, t_0) = e$, $s\_child(c, t_0) = e$. So $\mathcal{C}(e, P_s(t_0)) = \mathcal{C}(d, P_s(t_0)) \vee (\mathcal{C}(b, P_s(t_0)) \wedge \neg\theta_b) \vee (\mathcal{C}(c, P_s(t_0)) \wedge \neg\theta_c)$.

Theorem 2. *The complexity of Algorithm Passing-Through() is $O(|VIP(t)|^2)$, which is bounded by $O(|V|^2)$.*

Proof: The loop in line 4 has to run for $|VIP(t)| - 1$ times, and the complexity of $compute\_s\_child(v,t)$ is bounded by $O(|VIP(t)|)$. Thus, the theorem gets proved. ■

We now solve the optimal deployment problem. Recall the generalized MAX-SAT problem as: given a set U of variables θ_i, a collection of clauses, where each clause is a disjunction of conjunction of literals (e.g., $(\theta_i \wedge \theta_j) \vee \neg\theta_j$), find a truth assignment such that the number of satisfied clauses is maximized. We develop our Opt-Deploy() following the branch-and-bound algorithm for MAX-SAT [6]. We improve the branch-and-bound algorithm by exploring the search tree in a depth-first order. At each node, the algorithm compares the number of clauses violated (unsatisfied with certainty) by the best assignment (called upper bound ub), with the number of clauses violated by the current assignment plus an underestimation. The underestimation is the number of clauses that become violated if we extend the current partial assignment into a complete assignment. If the current assignment plus the underestimation is greater than the best assignment, the subtree of the node is pruned, else the algorithm searches one level deeper into the tree. If a clause is certain to be satisfied, it will be removed from the union of clauses (Γ). Initially, ub is computed by a local random search procedure call $GSAT()$ [7]. When each step the algorithm searches deeper, a variable in Γ is selected following J-W rule [8], which gives precedence to variable in shorter clauses.

The inputs of Algorithm Opt-Deploy() are ub, Γ and κ. The outputs of the Algorithm Opt-Deploy() are the optimal deployment $\mathcal{G}^o$ and the minimum distance.

Algorithm 2: Opt-Deploy(ub, Γ, κ)

```
    Initialize  : $\Gamma \leftarrow \bigcup_{v \in VIP(t), \theta_v \neq 1} \mathcal{C}(v, P_s(t)), ub \leftarrow$
                  $GSAT(\Gamma)$
    Output      : $< \mathcal{G}^o, ub >$
1   begin
2       if $\Gamma = \emptyset$ or $\Gamma$ only contains violated clauses then
            // return the number of violated clauses
3           return $< \emptyset, violated\_num(\Gamma) >$
4       if $lower\_bound(\Gamma) > ub$ then return $< \emptyset, \infty >$
5       $u \leftarrow select\_variable(\Gamma)$// following J-W rule
6       if $\sum_{v \in V} \theta_v < \kappa$ then // set $\theta_v$ to be 1 in $\Gamma$
7           $< \mathcal{G}, ub_1 > \leftarrow$ Opt-Deploy($ub, \Gamma|\theta_u, \kappa$)
8           if $ub_1 \leq ub$ then $ub \leftarrow ub_1$
9       $< \mathcal{G}', ub_2 > \leftarrow$ Opt-Deploy($ub, \Gamma|\neg\theta_u, \kappa$) // set $\theta_v$ to be 0
10      if $ub_2 \leq ub$ then $ub \leftarrow ub_2$, return $< \mathcal{G}', ub >$
11      else return $< \mathcal{G} \cup \{u\}, ub >$
```

Theorem 3. *The complexity of Algorithm Opt-Deploy() is $O(\binom{|V|}{\kappa} \times |\mathcal{T}| \times \max_t |VIP(t)|^2)$, which is bounded by $O(|V|^{\kappa+2} \times |\mathcal{T}|)$.*

Proof: The algorithm has to check at most $\binom{|V|}{\kappa}$ cases. Each case has at most $|\mathcal{T}| \times \max_t |VIP(t)|$ clauses, each clause has at most $\max_t |VIP(t)|$ variables. ■

This complexity is exponential. However, we can see that if κ is constant, the complexity becomes polynomial and we can perform exhaustive search. But when κ is large, the computing time increases fast with κ. To reduce computing time, we develop a heuristic, which computes Opt-Deploy(ub, Γ, κ) by running Opt-Deploy(ub, Γ, 1) for κ times. The complexity is then reduced to $O(|V|^3 \times |\mathcal{T}| \times \kappa)$.

Note that this heuristic can naturally be used to solve our optimal incremental deployment problem. We call it Inc-

Deploy() for future reference.

Next, we reduce the search space of our problem given a key observation as follows.

Observation 2. $D(VIP(t), \mathcal{L}(\mathcal{G}, P_s)) = 0$ *if and only if for* $0 \leq i \leq |VIP(t) - 2|$, $\theta_{v_t^i} = 1$ *when* $s\_child(v_t^i) \neq v_t^{i+1}$.

Proof: The correctness proof is in Appendix. A. ■

Let $K = \{v_t^i | s\_child(v_t^i) \neq v_t^{i+1}, \forall t, 0 \leq i \leq |VIP(t) - 2|\}$, Observation 2 shows that we only need to deploy nodes in K to guarantee that the paths that VIP flows are identical with pre-defined VIP paths. We call the nodes in K *key* nodes.

Table III: Algorithm Table

Algorithm	Problem Definition	Problem
Opt-Deploy()	Optimal Deployment	Problem 1
Inc-Deploy()	Optimal Incremental Deployment	Problem 2
Adp-Forward()	Adaptive Forwarding	Problem 3

IV. Adaptive Forwarding

The deployment sequence derived from Algorithm Inc-Deploy() is fixed. In practice, the deployment may need to be more flexible. Even the deployment sequence is settled, the network topology or VIP paths may change. As such, we need a more adaptive deployment scheme.

To address this problem, our idea is to make the deployed routers adaptive in the forwarding operation, i.e., $\mathcal{I}_v^s(P_s)$. More specifically, a deployed router may change the TwoD-IP forwarding operation of VIP flows. Let $\mathcal{J}(\mathcal{G}) = \{\mathcal{I}_v^s(P_s(t)) | v \in \mathcal{G}, t \in \mathcal{T}\}$, we call $\mathcal{J}(\mathcal{G})$ an *adaptive forwarding* for deployment $\mathcal{G}$. Our objective is that given $\mathcal{G}$, find an adaptive forwarding that minimizes the total distance between the paths of the VIP traffic flows and their pre-defined VIP paths. Formally,

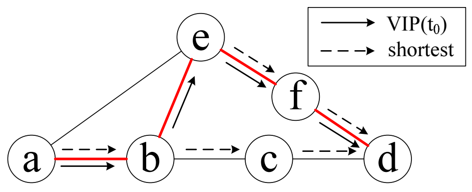


Figure 4: Adaptive forwarding example

Problem 3. ***Adaptive Forwarding:*** *Given a deployment* $\mathcal{G}$*, find an adaptive forwarding* $\mathcal{J}(\mathcal{G})$ *that minimizes the total distance* $\sum_t D(VIP(t), \mathcal{L}(\mathcal{G}, P_s))$.

We use an example in Fig. 4 to illustrate the definition. Here we assume a is the source and d is the destination. The shortest path from a to d is $\{a, b, c, d\}$. Assume there is one VIP flow t_0 whose VIP path is $\{a, b, e, f, d\}$. After deploying node a, i.e., $\mathcal{G} = \{a\}$, the distance between the path of t_0 and the pre-defined VIP path is $D(VIP(t_0), \mathcal{L}(\mathcal{G}, P_s(t_0))) = 5 - 3 = 2$. Within adaptive forwarding, node a can divert the traffic of t_0 to node e, i.e., $\mathcal{I}_a^s(P_s(t_0)) = e$, then the traffic of t_0 flows along $\{a, e, f, d\}$, and the distance decreases to be 1.

To find the adaptive forwarding, we devise a centralized protocol, that can change the forwarding operations of each router according to the topology and VIP flow information. We also develop a dynamic programming based algorithm, that computes the adaptive forwarding operations.

A. PaFid Overview

Unlike in the optimal deployment scheme, where we can simply deploy, here we need to deploy a new protocol to facilitate the computing and changing of the forwarding operations. In this section, we devise PaFid (**P**rotocol for **A**daptive **F**orwarding during **I**ncremental **D**eployment), which is a centralized protocol that can collect topology and VIP flow information, and return the computed results to each router. PaFid is based on centralized control, because the controller can easily obtain the global topology and VIP flow information, and flexibility change the forwarding operations on each router.

In Fig. 5, we show the information flow of PaFid. Within PaFid, deployed routers sends the topology information to the controller, users send VIP flow information to the controller, and the controller computes the adaptive forwarding, then returns the forwarding operations to each router. We list the three main communication procedures of PaFid as following.

- **Topology Information:** The deployed routers should notify the controller of the routers being deployed. The notification should be sent once a router is deployed, and sent periodically to maintain a soft state on the controller. And one of the deployed router should send the global network topology information to the controller (we assume that link state routing protocol (e.g., OSPF) is used).
- **VIP Flow Information:** User can subscribe VIP flows to the controller. The VIP flow information includes the routers sequence that the flow passes by. The controller is responsible for VIP flows validation, including user identity validation and path validation, i.e., satisfying [3] to guarantee loop-free. Note that "users" generally refers to the entities that set up the VIP flows, routers themselves can also be "users".
- **Forwarding Operation information:** After computation, the controller sends back the forwarding operations to each router. And routers install the forwarding operations into their forwarding tables.

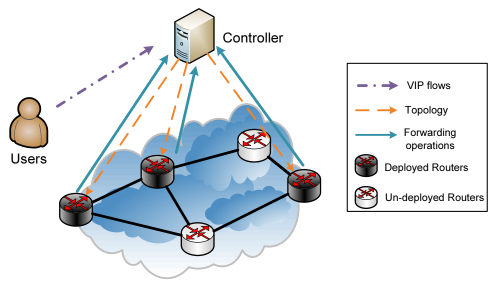


Figure 5: Information flow of PaFid

B. Adaptive Forwarding Algorithm

After collecting the topology and VIP flow information, the controller of PaFid should compute the adaptive forwarding. In this section, we develop a dynamic programming based algorithm *Adp-Forward()* which is optimal.

To be loop-free and failure-tolerant, we adopt the deflection rules in [3]. In this paper, due to page limit, we use the simplest "one hop down" rule, i.e., a router can divert vip traffic to any neighbor given that the neighbor is closer to the destination. Algorithms using other rules can be found in [9]. To avoid high stretch of VIP paths, we define ϵ as the maximum stretch ratio, such that $\frac{|\mathcal{L}(\mathcal{G},P_s)|}{|VIP(t)|} \leq \epsilon$.

Algorithm 3: Adp-Forward($\mathcal{G}, \epsilon$)

```
   Output : $J$
1  begin
2      for $t \in \mathcal{T}$ do
3          $\lambda = \epsilon \times |VIP(t)|, d = v_t^{|VIP(t)|-1}$
4          $\mathcal{R} = V - \{d\}, f(d,i) = 1, 0 \leq i \leq \lambda - 1$
5          while $\mathcal{R} \neq \emptyset$ do
6              Select $u \in \mathcal{R}$ that is closest to $d$
7              if $u \in \mathcal{G}$ then
8                  Find $\mathcal{N} \subset V$, $v \in \mathcal{N}$ if $(u,v) \in E$, and $v$ is
                   closer to $d$
9                  $f(u,i) = \min_{v\in\mathcal{N}} f(v,i-1) + I_{u\in VIP(t)}$,
                   $1 \leq i \leq \lambda - 1$
10                 $f(u,0) = 0$
11                 $\mathcal{I}_v^s(P_s(t)) = w$, where
                   $w \in \mathcal{N}, f(w,i-1) = \min_{v\in\mathcal{N}} f(v,i-1)$
12             else
13                 Find $w$, the successor of $u$ along shortest path
14                 $f(u,i) = f(w,i-1) + I_{u\in VIP(t)}, 1 \leq i \leq \lambda - 1$,
                   $f(u,0) = 0$
15             $\mathcal{R} = \mathcal{R} - u$
16     return $\mathcal{J}(\mathcal{G}) = \{\mathcal{I}_v^s(P_s(t))\}, t \in \mathcal{T}, v \in \mathcal{G}$
```

The input of Adp-Forward() is a deployment $\mathcal{G}$, i.e., the subset of routers that have sent notifications to the controller, and the maximum stretch ratio ϵ. The output of Adp-Forward() is the adaptive forwarding, i.e., the forwarding operations for all VIP flows on each deployed router. In Algorithm Adp-Forward, $I_{u\in VIP(t)}$ is an indicator function, such that $I_{u\in VIP(t)} = 1$ if $u \in VIP(t)$, else $I_{u\in VIP(t)} = 0$.

Theorem 4. *The complexity of Adp-Forward() is* $O(|V| \times log(|V|) \times |\mathcal{T}|)$.

Proof: The loop in line 2 runs for $|\mathcal{T}|$ times, the loop in line 5 runs for $|V|$ times. The complexity of the action (line 8) to find the lowest cost neighbor to reach the destination is $O(|V|)$. However, for each VIP path, we can perform a ranking algorithm on all node of V based on their costs to reach the destination. The lowest complexity of ranking algorithms will be $0(|V| \times log(|V|))$. ■

V. Implementation

We realize the algorithm Inc-Deploy() as an offline software that runs on PC hosts. We also implement the PaFid on routers and a PC host that acts as the centralized controller. On a commercial router (BitEngine 12000) that has been equipped with TwoD-IP, we realize the interfaces through which we can change the forwarding operations of the router. We implement PaFid, such that VIP flows information is maintained in the controller, and the deployed routers will send both topology and VIP flow information to the controller. We realize the controller as a module in OpenFlow [10], which can provide centralized control on forwarding operations of routers over the network. After collecting the needed information, the controller will send back the forwarding operations to routers through the predefined interfaces.

VI. Performance Evaluation

A. Simulation Setup

We evaluate the performance of our TwoD-IP routing scheme using both BRITE [11] generated topologies and CERNET. A case study on CERNET2 is in the next section.

1) Topology: We generate topologies with nodes from 50 to 400. To set up a VIP flow, we first randomly select a pair of source-destination nodes, and then randomly select its VIP path which 1) is not the shortest path and 2) satisfies the rules in [3] to prevent routing loops. The number of VIP flows is between 10 and 100. The maximum stretch ratio of adaptive forwarding is set to be 1.5. The default values and other parameters of our evaluation are in Table IV.

Table IV: Parameter Table

VIP flows	**No. Nodes**	links/new node	Mode
30	150	3	Router Only
Model	Placement	α / β	ϵ
Waxman	Random	0.15/0.2	1.5

We also use the topology of CERNET, which is a medium-scale IPv4 network with 110 routers and 238 links.

We compare our algorithm with random deployment (RD). We admit that random deployment is artificial, and service providers may attempt other schemes. We thus compare with human-like deployment (HD), that only selects key nodes to deploy (see Observation 2). Note however, HD is based on an observation within the contribution of our study. Besides, we also set full deployment (FD) as a benchmark for comparison. Our evaluation metric is the averaged normalized distance (or in short *a-distance* thereafter) between the paths computed by our algorithms and the pre-defined VIP paths. Formally, a-distance equals to $\frac{\sum_t D(\mathcal{L}(\mathcal{G},P_s),VIP(t))}{\sum_t |VIP(t)|}$. The smaller the a-distance, the better. If a-distance equals to zero, the performance is identical to the FD, i.e., the computed paths are identical to the pre-defined VIP paths. The results shown in this section are averaged by ten random and independent experiments.

B. Simulation Results

1) Optimal Incremental Deployment: Fig. 6(a) shows a typical incremental deployment process and compares different deployment algorithms. There are 150 nodes and 30 VIP flows. We deploy three nodes in every incremental step. In Fig. 6(a), we see that even we do not deploy any router, the a-distance is still around 50%. This is because the shortest path can travel through a few VIP nodes already. When we deploy more nodes, the a-distance decreases. However, Algorithm Inc-Deploy() performs better than RD and HD. For example, after we deploy 33 nodes, the nodes chosen by Algorithm Inc-Deploy() match the nodes on the VIP paths very well, and the a-distance is only 2.02%. On the contrary, if we randomly choose 36 nodes to deploy, the a-distance is 39.08%, if randomly choose 36 key nodes to deploy, the a-distance is 17.82%. When we look into the details of the simulation trace, we see that there are only 4 nodes that do not match the VIP nodes using Algorithm Inc-Deploy(), while there are 68 VIP nodes not covered using RD, 31 VIP nodes not covered by HD. We emphasize again that the essence of incremental deployment is to demonstrate the benefits of TwoD-IP routing when deploying as few nodes as possible. Clearly, our algorithm achieves this.

In Fig. 6(a), we see that for the paths computed by RD, the a-distance to the VIP paths can even increase after more nodes are deployed. For example, when 36 nodes are deployed, the a-distance is 36.78%. After we deploy another 3 nodes, the a-distance increases to 37.36%. On the contrary, we see in Fig. 6(a), this never happens to Inc-Deploy(). We also evaluate adaptive forwarding Adp-Forward() in Fig. 6(a). We see that its impact is quite small. Both HD and Inc-Deploy() will achieve optimal performance after deploying all key nodes, however, the a-distance of Inc-Deploy() decreases much faster than HD.

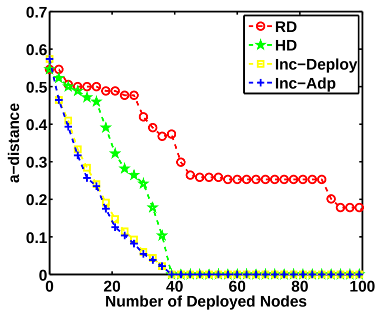
Fig. 6(b) shows the impact of the incremental step size. We compare the performance for four step sizes, 1, 2, 5, 10 (i.e., 1, 2, 5, 10 nodes are deployed in each incremental step). We see that there is not much difference between different step sizes. Therefore, our heuristic algorithm, using small step size to reduce the computational complexity, is satisfactory.

In Fig. 7, we study the impact of the network size. We see that for Inc-Deploy() and HD, the a-distance decreases when the network size increases. For RD, the a-distance increases when the network size increases. This is because when the network is larger, the number of nodes that belong to VIP paths increases slower than the total number of nodes. Since Inc-Deploy(), and HD always choose from this set of nodes, the performance improves. On the contrary, RD selects randomly from all nodes, making its performance decrease. In Fig. 7, we also compare different deploy algorithms within two different deployment ratios, 10%, 20% (i.e., 10%, and 20% of all nodes are deployed). Clearly, the more nodes deployed, the shorter the a-distance. And by deploying 10% more nodes within RD, the improvement (i.e., the gap between 10% and 20% deployment ratio) is smaller than Inc-Deploy() and HD. Although the a-distance both Inc-Deploy() and HD decrease with network size, the a-distance of Inc-Deploy() drops more quickly to be zero, especially when deployment ratio is low. The result indicates that better performance can be achieved by deployed carefully selected nodes based on our algorithm, especially at the initial stage of incremental deployment.

In Fig. 8, we study the impact of the number of VIP traffic flows. In practice, when the number of VIP traffic flows increases, we will also deploy more nodes to be TwoD-IP capable. Thus, we set an increasing ratio x between the number of VIP flows and the nodes deployed, i.e., every x additional nodes will be deployed when there is one additional VIP traffic flow. We compare different deploy algorithms within two different ratios 1 and 0.2. We see that the a-distance decreases when the number of VIP flows increases for all algorithms. This shows that deploying more nodes has higher positive impact. RD performs the worst, even the increasing ratio of RD is 1, its performance is still worse than Inc-Deploy() with increasing ratio of 0.2. When the ratio is high, the a-distance of HD is quite small. However, when the ratio is low, the a-distance of HD is almost as worse as RD.

2) Adaptive Forwarding: We use Inc-Adp(), RD-Adp and HD-Adp to denote adaptive forwarding running on nodes selected by Inc-Deploy(), RD and HD.

Fig. 9 shows the impacts of adaptive forwarding on a random deployment process (RD or HD) that does not follow the optimal incremental deployment. We deploy three nodes in every incremental step. We see that adaptive forwarding improves both RD and HD. For example, if we randomly choose 90 nodes to deploy, the a-distance is 34.71%, however, the a-distance decreases to be 27.65% if we use adaptive forwarding between these 90 nodes; if we randomly choose 12 key nodes to deploy, the a-distance is 28.49%, and the a-distance decreases to be 22.93% if adaptive forwarding is used. In the extreme case, adaptive forwarding improves RD by over 7% maximally, while improves HD by 5.5% maximally. This is because RD will choose non-key nodes, which can also act as relay nodes to further decrease the a-distance.


(a) Comparison of different deployment algorithms

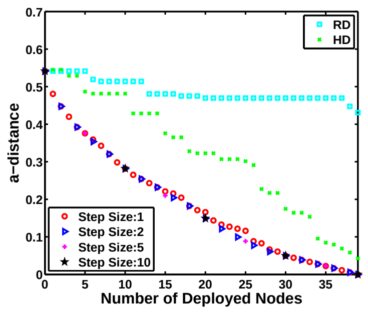

(b) Impact of step size

Figure 6: a-distance as a function of number of deployed nodes

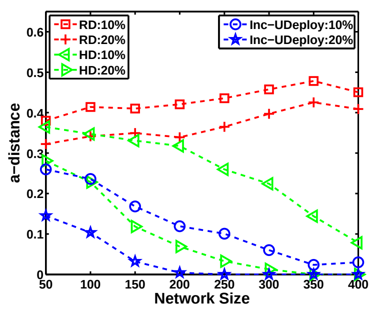


Figure 7: a-distance as a function of network size

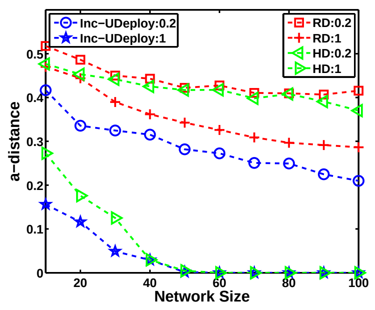


Figure 8: a-distance as a function of number of VIP flows

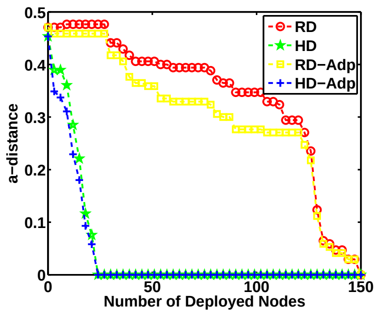


Figure 9: a-distance as a function of number of deployed nodes

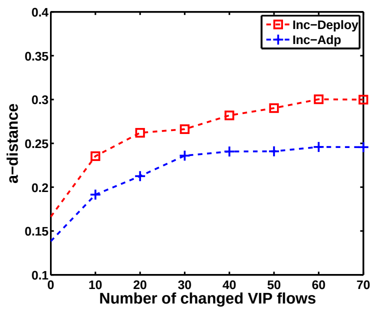


Figure 10: a-distance as a function number of vip flows

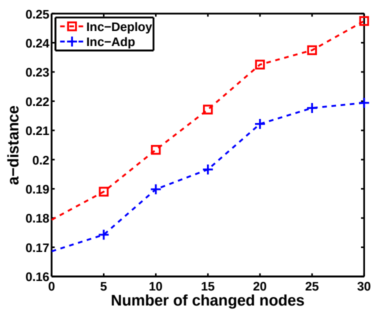


Figure 11: a-distance as a function of number of changed nodes

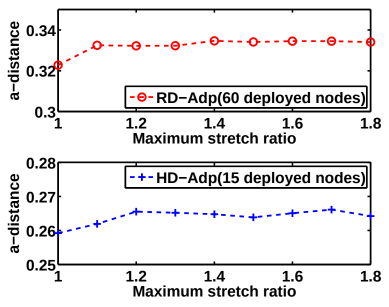


Figure 12: a-distance as a function of the maximum stretch ratio

In Fig. 10, we study the benefits of adaptive forwarding when VIP flows change. After setting up 30 VIP flows and choosing 30 nodes to deploy according to Inc-Deploy(), we generate 10-70 new VIP flows. We see that when the number of VIP flow increases, the a-distance will increase. The more the new VIP flows join in, the larger the a-distance. However, with adaptive forwarding, the a-distance increases slower, i.e., the gap between Inc-Deploy() and Inc-Adp() becomes larger when more VIP flows generated. This is because adaptive forwarding can regulate the VIP flows that are away from the right VIP paths.

In Fig. 11, we study the benefits of adaptive forwarding when topology changes. After deploying 30 nodes, we randomly revoke 5-10 existed or add 5-30 new nodes[1]. The a-distance will increase when more nodes are added. This is because when new nodes are added, the forwarding action (e.g., the next hop on the shortest path) of each node may change. Thus the a-distance will increases because the deployed nodes are computed based on the original topology. With adaptive forwarding, the a-distance increases slower. The result is similar with Fig. 10, indicating that through adaptive forwarding, we can achieve substantial benefits when VIP flows or network topology change.

In Fig. 12, we study the impact of the maximum stretch ratio on adaptive forwarding. We set maximum stretch ratio to be 1.0-1.8 when randomly deploying 60 nodes or 15 key

[1]the influenced VIP flows will be regenerated

nodes. Fig. 12 shows that the a-distance increases when the maximum stretch ratio increases from 1.0 to 1.2. However, the a-distance remain almost the same when the maximum stretch ratio is larger than 1.2. This is because adaptive forwarding only slightly stretch the path, by at most 20%.

3) Simulation Results on CERNET: We further validate our results using CERNET. Due to page limit, we only evaluate optimal incremental deployment algorithms. Fig. 13(a) shows an optimal incremental deployment process by deploying two nodes at each step. We see that the a-distance is 26.7% even we do not deploy any router, this is because of the low connectivity of CERNET topology. When we deploy more nodes, the a-distance decreases and Inc-Deploy() performs much better than RD and HD. After deploying 8 nodes, the a-distance of Inc-Deploy() is 1.78%, the a-distance of RD is still 22.49%, and the a-distance of HD is 5.92%. Fig. 13(b) shows the impact of the number of VIP flows. Here, we set the increasing ratio to be 0.1 and 0.5, i.e., deploying 0.1 and 0.5 nodes after adding a VIP flow. In Fig. 13(b), a-distance decreases with the number of VIP flows. Inc-Deploy() decreases faster and performs the best. HD achieves nearly optimal result when the ratio is high, but much worse when the ratio is low.

VII. TwoD-IP Routing for CERNET2: A Case Study

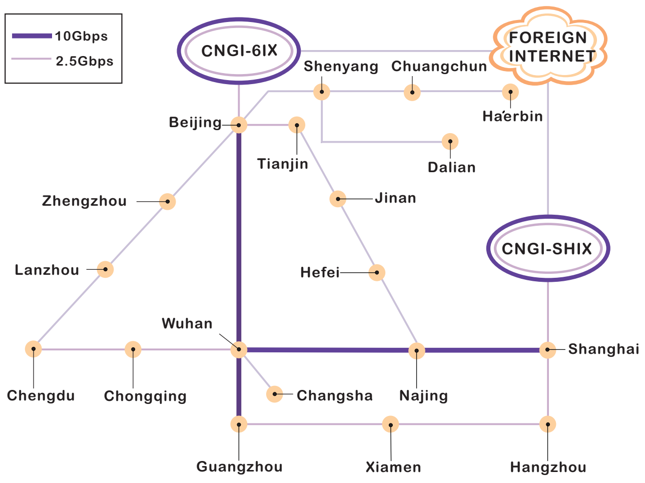


Figure 15: CERNET2 topology

We conduct a case study with the real topology and prefix information of CERNET2. Our work also serves as a reference for the future deployment of TwoD-IP routing on CERNET2.

We want to move the out-going International traffic of three universities, i.e., THU (in Beijing, with 38 prefixes), HUST (in Wuhan, with 18 prefixes) and SCUT (in Guangzhou, with 28 prefixes) to CNGI-SHIX (Shanghai portal). There will be three VIP paths, $VIP(t_0)$ ={Beijing, Tianjin, Jinan, Hefei, Nanjing, Shanghai}, $VIP(t_1)$ ={Wuhan, Nanjing, Shanghai} and $VIP(t_2)$ ={Guangzhou, Xiamen, Hangzhou, Shanghai}.

In the CERNET2 scenario, we apply the weighted version of our problems and each VIP traffic flow is assigned a weight that is proportional to the traffic volume. In our case, we set the weights to $w_{t_0} = 19.6, w_{t_1} = 5.1, w_{t_2} = 11.0$ (for details of the weight assignment, please refer to [9]).

Fig. 14(a) shows the optimal incremental deployment process, where we deploy nodes one by one. We compare Inc-Deploy() and HD algorithm (here nodes are randomly selected from the VIP paths, the candidate node set includes Beijing, Tianjin, Jinan, Hefei, Nanjing, Shanghai, Wuhan, Guangzhou, Xiamen, Hangzhou). For Inc-Deploy(), the deployment sequence we suggest is Guangzhou, Wuhan, Jinan, Tianjin and Beijing. We see that after deploying the router of Guangzhou (occupies 30.8% of the total traffic), the a-distance falls from 100% to 69.2%. For HD to achieve similar performance, we need to deploy routers at five cities.

VIII. Related Work

There is little work on two dimensional routing since IP routing won over circuit based routing such as PNNI [12]. Because of the important semantic in source address, recent years see more research on giving sources control.

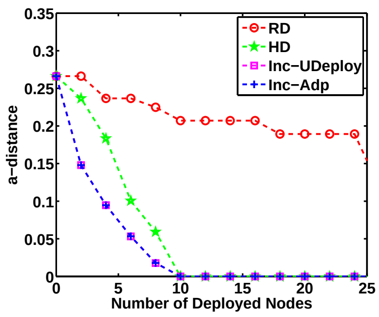
IP (loose/strict) source routing [13] allows the sender to take full control of the routing path. However, due to security reason [14], source routing is disabled in most networks. In addition, source routing is inefficient because of the additional IP option header, and it hands most control to the end users, which is unfavorable for ISP operators.

MPLS [15] is often used to manage traffic per-flow. However, due to the control and management overheads, MPLS raises concern for scaling when the number of label switching paths (LSPs) increases [16]. The more the LSPs, the heavier the system burden [17]. MPLS only supports limited number of LSPs (one Cisco moderate router supports 600 LSPs, according to data in 2005 [18]).

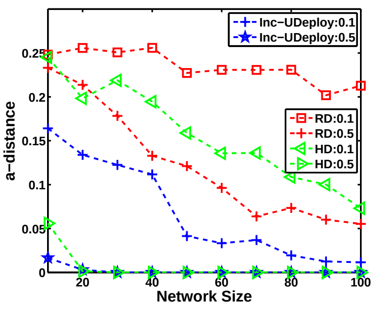
There are many other routing schemes that have been integrated with source address lookup. Such as Policy based routing (PBR) [19] , Customer-specific routing [20], Mutli-topology routing [21], and even overlay routing [22], which is beyond the network layer. However, for an ISP, a light weight, pure IP-based, more network controllable solution is favored.

Due to security and accounting problems, CERNET2 has deployed SAVI to validate the source address of each packet at the edge points of the network. Currently, SAVI has been installed by more than 100 university campus network. Confirmed SAVI users are more than 900,000 [23]. CERNET2 then decides to make full use of source address for better reliability, security and traffic distribution by integrating the source address lookup into IP routing.

Incremental design is advocated [24] for network layer proposals. Many protocols and new algorithms can be in-

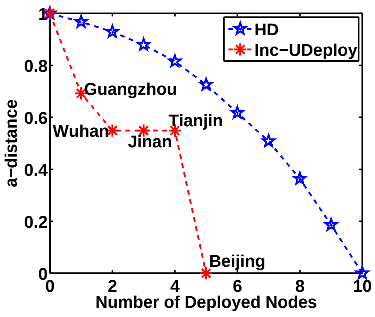


(a) a-distance as a function of number of deployed nodes on CERNET



(b) a-distance as a function of number of VIP flows on CERNET

Figure 13: Simulation on CERNET



(a) a-distance as a function of number of deployed nodes on CERNET2

Figure 14: Simulation on CERNET2

crementally deployed [3][25][26]. Our problem is specific and unique to TwoD-IP routing. Thus a design from scratch is needed.

IX. Conclusion and Future Work

In this paper, we presented a study of TwoD-IP routing where the forwarding decisions is not only based on the destination IP addresses but also on the source IP addresses. Our focus is on incremental deployment requirement, a practical concern of CERNET2. We formulated our problem such that we need to find a deployment sequence given the number of deployed nodes. We proved the problem to be NP-complete. We then novelly transformed our problem to boolean clauses and developed an efficient algorithm following the principles of branch-and-bound algorithm for MAX-SAT.

We evaluated our algorithms comprehensively using CERNET and other topologies. We showed that by deploying a few nodes suggested by our deployment sequence can successfully manage the traffic flows, and adaptive forwarding mechanism can flexibility manage the traffic flows when topology or VIP flows change. We then presented a case study on CERNET2 and provided a fine deployment sequence.

Based on our specific consideration, we use hamming metric to represent the deviation between paths. While there are other meaningful metric in graph theory, our next work includes trying different metrics for other considerations. We assertively use summation of individual path distance to express the total deviation, validation

References

[1] J. Wu, J. Bi, M. Bagnulo, F. Baker, and C. Vogt, “Source address validation improvement framework,” Internet Draft, Mar 2011, draft-ietf-savi-framework-04.txt.

[2] M. Xu, J. Wu, S. Yang, and D. Wang, “Two dimensional ip routing architecture,” Internet Draft, Mar 2012, draft-xu-rtgwg-twod-ip-routing-00.txt.

[3] X. Yang and D. Wetherall, “Source selectable path diversity via routing deflections,” in *Proc. ACM SIGCOMM’06*, New York, NY, Sept 2006.

[4] L. Trevisan, “When hamming meets euclid: the approximability of geometric tsp and mst,” in *Proc. ACM STOC’97*, El Paso, TE, May 1997.

[5] U. Feige, G. Kortsarz, and D. Peleg, “The dense k-subgraph problem,” *Algorithmica*, vol. 29, pp. 410–421, 2001.

[6] T. Alsinet, F. Manya, and J. Planes, “Improved branch and bound algorithms for max-sat,” in *Proc. Theory and Applications of Satisfiability Testing (SAT’03)*, Portofino, Italy, May 2003.

[7] B. Selman, H. Levesque, and D. Mitchell, “A new method for solving hard satisfiability problems,” in *Proc. of the tenth national conference on Artificial intelligence (AAAI’92)*, San Jose, CA, July 1992.

[8] J. Hooker and V. Vinay, “Branching rules for satisfiability,” *Journal of Automated Reasoning*, vol. 15, pp. 359–383, 1995.

[9] S. Yang, D. Wang, M. Xu, and J. Wu, “Efficient two dimensional-ip routing: An incremental deployment design,” Tsinghua University, Tech. Rep., July 2011. [Online]. Available: http://www.wdklife.com/tech.pdf

[10] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar 2008.

[11] “Brite: Boston university representative internet topology generator,” http://www.cs.bu.edu/brite.

[12] *Private Network-Network Interface Specification v1.0(PNNI)*, ATM Forum Technical Committee, Mar. 1996.

[13] D. Estrin, T. Li, Y. Rekhter, K. Varadhan, and D. Zappala, “Source Demand Routing: Packet Format and Forwarding Specification (Version 1),” RFC 1940 (Informational), Internet Engineering Task Force, May 1996.

[14] C. Perkins, “IP Encapsulation within IP,” RFC 2003 (Standards Track), Internet Engineering Task Force, Oct. 1996.

[15] E. Rosen, A. Viswanathan, and R. Callon, “Multiprotocol Label Switching Architecture,” RFC 3031 (Standards Track), Internet Engineering Task Force, Jan. 2001.

[16] S. Yasukawa, A. Farrel, and O. Komolafe, “An Analysis of Scaling Issues in MPLS-TE Core Networks,” RFC 5439 (Informational), Internet Engineering Task Force, Feb. 2009.

[17] C. Metz, C. Barth, and C. Filsfils, “Beyond mpls ... less is more,” *Internet Computing, IEEE*, vol. 11, no. 5, pp. 72 –76, Sep 2007.

[18] S. Kandula, D. Katabi, B. Davie, and A. Charny, “Walking the tightrope: responsive yet stable traffic engineering,” in *Proc. ACM SIGCOMM’05*, Philadelphia, Pennsylvania, USA, Aug 2005.

[19] *Policy-Based Routing (white paper)*, Cisco, 1996.

[20] J. Fu and J. Rexford, “Efficient ip-address lookup with a shared forwarding table for multiple virtual routers,” in *Proc. ACM CoNEXT’08*, Madrid, Spain, Dec 2008.

[21] N. Wang, K.-H. Ho, and G. Pavlou, “Adaptive multi-topology igp based traffic engineering with near-optimal network performance,” in *Proc. IFIP-TC6 NETWORKING’08*, Singapore, May 2008.

[22] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient Overlay Networks,” in *Proc. ACM SOSP’01*, Banff, Canada, October 2001.

[23] “Cngi-cernet2 savi deployment update,” http://www.ietf.org/proceedings /80/slides/savi-2.pdf, CERNET, March 2011.

[24] N. M. K. Chowdhury and R. Boutaba, “A survey of network virtualization,” *Computer Networks*, vol. 54, pp. 862–876, April 2010.

[25] Q.Li, D.Wang, M.Xu, and J.Yang, “On the scalability of router forwarding tables: Nexthop selectable fib aggregation,” in *Proc. IEEE INFOCOM’11, mini-conference*, Shanghai, China, Apr 2011.

[26] K. Ramakrishnan, S. Floyd, and D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP,” RFC 3168 (Standards Track), Internet Engineering Task Force, Sep. 2001.

APPENDIX

A. Proof for Observation 2

Proof: fff It’s easy to prove that it is a sufficient condition. With precondition that $\mathcal{C}(v_t^0, P_s(t))$ is 1, suppose that for all $j < k$, $\mathcal{C}(v_t^j, P_s(t))$ are 1. Then if $s\_child(v_t^{k-1}, P_s(t)) = v_t^k$, $\mathcal{C}(v_t^k, P_s(t))$ must be 1, else according to Observation 2 we have $\theta_{v_t^{k-1}} = 1$, thus $\mathcal{C}(v_t^k, P_s(t))$ is 1. So we can conclude that all clauses $\mathcal{C}(v, P_s(t))$ are 1.

Then we prove that the condition is necessary. According to Algorithm 1, if $s\_child(v_t^j, P_s(t)) \neq v_t^{j+1}$ and $\theta_{v_t^j} \neq 1$, there must exist $k_0 < j$ such that $s\_child(v_t^{k_0}, P_s(t)) = v_t^{j+1}(s\_child(v_t^{k_0}, P_s(t)) \neq v_t^{k_0+1})$ and $\theta_{v_t^{k_0}} \neq 1$. Indicating there must be $k_1 < k_0$ such that $s\_child(v_t^{k_1}, P_s(t)) \neq v_t^{k_1+1}$ and $\theta_{v_t^{k_1}} \neq 1$. Thus there is a integer sequence $k_0 > k_1 > k_2...$ until $k_l = 0(l > 0)$, obviously, if $s\_child(v_t^0, P_s(t)) \neq v_t^1$, $\theta_{v_t^0}$ must equal to 1 (else $\mathcal{C}(v_t^1, P_s(t))$ will be 0). Thus the assumption is false. ■