IFIP Networking 2019 1570512078

A Distributed Algorithm for Throughput Optimal Routing in Overlay Networks

Anurag Rai, Rahul Singh, Eytan Modiano
Laboratory for Information and Decision Systems, MIT, USA
{rai,rsingh12,modiano}@mit.edu

***Abstract*—We address the problem of optimal routing in overlay networks. An overlay network is constructed by adding new overlay nodes on top of a legacy network. The overlay nodes are capable of implementing any dynamic routing policy, however, the legacy underlay has a fixed, single path routing scheme and uses a simple work-conserving forwarding policy. Moreover, the underlay routes are pre-determined and unknown to the overlay network. The overlay network can increase the achievable throughput of the legacy network by using multiple routes, which consist of direct routes and indirect routes through other overlay nodes. We develop an optimal dynamic routing algorithm for such overlay networks called the Optimal Overlay Routing Policy (OORP).**

OORP is derived using the classical dual subgradient descent method, and it can be implemented in a distributed manner. We show that the queue-lengths can be used as a substitute for the dual variables in the algorithm. However, the underlay queue-lengths are unknown to the overlay, so we propose two regression based schemes that learn simplified models of the backlog in the underlay using historical data and use them to estimate the queue-lengths in real time. Simulation results show that near-optimal performance can be achieved without any knowledge of the underlay.

I. Introduction

Throughput optimal routing algorithms[1] have received a significant amount of attention in the literature (e.g. [1], [2], [11], [12]), however, they have had limited success in terms of implementations. One of the main reasons behind the lack of traction is that these policies require additional functionalities that are not supported by the legacy devices. For example, most of these algorithms need the network to be composed of homogeneous nodes that possess the ability to implement a dynamic routing policy. In contrast, many legacy networks use a single path routing scheme with a work-conserving forwarding policy such as FIFO, and hence can support only a fraction of the achievable throughput. Thus, an implementation of a throughput optimal scheme usually requires a complete overhaul of the network. An overlay architecture for a gradual move towards optimal routing was proposed in [6]. This architecture integrates overlay nodes capable of dynamic routing into an underlay network of legacy devices (see Figure 1 for an example). In this paper, we develop a throughput optimal dynamic routing algorithm for such overlay networks.

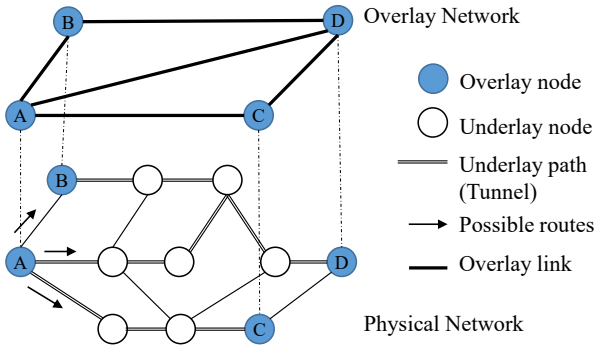


Fig. 1: Overlay network architecture. If the overlay node A has traffic for node D, it can either route it directly using the tunnel from A to D or relay it through other overlay node B or C.

Overlay networks have been used to improve the performance and capabilities of computer networks for a long time. The Internet itself started as a data network built on top of the telephone network. An overlay architecture to improve the robustness of the Internet was proposed in [3], where alternate overlay paths are used to overcome path loss in the underlay network. Placement for the overlay node to improve path diversity was studied in [4]. Architectures for designing overlay networks that improve different quality of service metrics have been proposed in [17]. Currently overlay is being for many applications in the internet such as caching for faster content delivery and improving resilience against DDoS attacks [18].

In [6], the authors study the problem of placing the minimum number of overlay nodes into an existing underlay in order to maximize network throughput. In particular, the authors show that with just a few overlay nodes, maximum network throughput can be achieved. However, [6] also shows that the backpressure routing algorithm of [1], which is known to be optimal in a wide range of scenarios, leads to a loss in throughput when used in an overlay network. Then the authors of [6] proposes a heuristic for optimal routing called the Overlay Backpressure Policy (OBP). An optimal backpressure like routing algorithm for a special case, where the underlay paths do not overlap with each other, was given in [7]. This paper also proposes a threshold based heuristic for general overlay networks. The schemes presented in [6] and [7] are

This work was supported by NSF grant CNS-1524317, and by DARPA I2O and Raytheon BBN Technologies under Contract No. HROO 1 l-l 5-C-0097.

[1]A routing algorithm is throughput optimal if it can stabilize any traffic that can be stabilized by some routing algorithm. In this paper, when we simply say optimality, we will be referring to throughput optimality.

ISBN 978-3-903176-16-4

very similar and were conjectured to be throughput optimal.

In this paper, we provide a counterexample to show that OBP is in fact not throughput optimal and develop a new optimal routing policy. To derive the policy, we notice that the suboptimality of backpressure arises from its failure to accurately account for congestion in the underlay paths. Traditional backpressure doesn't keep track of the packets in the underlay which can lead the overlay nodes to send too many packets into the underlay creating instability of underlay queues. In contrast, our solution uses the underlay queue-length information to keep track of the congestion. This policy implicitly favors underlay paths that are less congested and preserves stability of all the queues.

Our algorithm requires the underlay queue backlog information, but we do not expect such feedback from the underlay nodes. Hence, we propose schemes to infer the sum of the underlay queue-lengths in each path. First, we explain why simple schemes that use the delay experienced by a packet in a path as a measure of backlog in the path can produce bad estimates. This kind of scheme has been used successfully in various versions of TCP such as FAST [20] and Vegas [19], however we will show that this scheme can generate arbitrarily large errors and fail to preserve optimality in OORP. Hence, we propose a new method that trains a linear or a piece-wise linear model of the backlog using historical data, and then use it to predict the backlog information in real time.

This paper is organized as follows. We describe our model in the next section. In section III, we provide a counterexample to the OBP routing policy. In section IV, we develop our optimal routing policy based on the classical dual subgradient descent method which requires the underlay queue-lengths as feedback. In section V, we design different schemes for underlay backlog estimation. Finally, we verify the performance of our algorithm with simulations in Section VI.

II. Model

We model the network as a graph (N, E) where N is the set of nodes and E is the set of directed links. The links are capacitated and the capacity of a link $(i, j) \in E$ is given by c_{ij}. The nodes can be of two types: underlay or overlay. We represent the set of all underlay nodes by U and the set of all overlay nodes by $\mathcal{O} = N \backslash U$. The network supports a set of commodities, K, where each commodity $k \in K$ is defined by a source-destination pair. The time is slotted and indexed by t. We remove the time index for notational simplicity if removing it doesn't create ambiguity.

A. Overlay

The overlay network consists of the controllable nodes $\mathcal{O}$ which are capable of implementing a dynamic routing algorithm. The links between two overlay nodes can either be a direct edge or a path through the underlay referred to as a *tunnel*. A tunnel l is a sequence of nodes $l_1, l_2, \ldots, l_{|l|}$ where $|l|$ is the length of the tunnel. We represent the set of all the tunnels in the network by L.

Since a tunnel connects two overlay nodes, l_1 and $l_{|l|}$ are overlay nodes, and $l_{|2|}, ..., l_{|l|-1}$ are underlay nodes. When a packet is sent into a tunnel l, node l_1 encapsulates it into a packet destined to node $l_{|l|}$ and forwards it onto the underlay node l_2. The route taken by the tunnel is dictated by the path from l_2 to $l_{|l|}$ which is assigned by the underlay. When the packet reaches $l_{|l|}$, it is decapsulated and enqueued at the node. An example of the different type of links in an overlay network is given in Figure 2. This overlay network consists of one direct link (1,4) and three tunnels (1,3,4), (2,3,4) and (2,3,5).

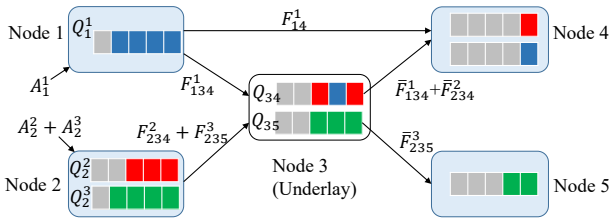


Fig. 2: Example of overlay and underlay queues in the physical network. The variables labelling the edges represent the number of packets transmitted for each commodity on each tunnel. The network consists of three different commodities which are represented by different colors.

Each overlay node i maintains a queue for each commodity k and the backlog is represented by Q_i^k. The number of external commodity k packets that arrive at node i represented by A_i^k. Let F_l^k represents the amount of packets injected into the tunnel l, and $\bar{F}_l^k$ represents the number of packets that exit tunnel l. The quantities are different because a packet sent into the tunnel might not exit the tunnel for several time-steps. Let F_{ij}^k represent the number of commodity k packets that are transmitted on an overlay to overlay link (i, j). Figure 2 illustrates the meaning of these variables on a simple network. The backlog of commodity k packets at overlay node i evolves as follows:

$$Q_i^k(t+1) = \Bigg[Q_i^k(t) - \sum_{j \in \mathcal{O}} F_{ij}^k(t) - \sum_{l \in L: l_1 = i} F_l^k(t) + \sum_{j \in \mathcal{O}} F_{ji}^k(t) + \sum_{l \in L: l_{|l|} = i} \bar{F}_l^k(t) + A_i^k(t) \Bigg]^+$$

Here, $\{l \in L : l_1 = i\}$ are all the tunnels that start at node i, $\{l \in L : l_{|l|} = i\}$ are the tunnels that end at node i, and $[.]^+ = \max(., 0)$. Packets are removed at the destination node, hence the backlog of a commodity at its destination is zero.

We assume that all the traffic arrivals A_i^k are i.i.d. with a mean of λ_i^k. We also assume that the arrival rate vector λ is in the interior of the throughput region of the overlay network Λ [6]. We will be designing a dynamic routing policy that controls F_l^k and F_{ij}^k at each time-step so that both the overlay and the underlay queues stabilize.

B. Underlay

The underlay network consists of the uncontrollable nodes U. These nodes have a static routing policy which assigns a

fixed path between each pair of nodes in the underlay. The paths are assumed to be acyclic and unique, which ensures that all the tunnels are acyclic and that they take a fixed route through the underlay.

An underlay node maintains a queue per outgoing link. The backlog on the queue associated with the link (a, b) is represented by Q_{ab}. The queues have infinite buffer space hence packets are not dropped. When a packet arrives at an underlay node, the node looks up the link assigned to it based on its destination and enqueues it on the corresponding link. Since several tunnels of the overlay network can pass through the same underlay link the underlay queues accumulates packets from several different tunnels and commodities. An example of an underlay queue that is shared by several tunnels is presented in Figure 2. Packets from both the tunnels (1,3,4) and (2,3,4) are queued on the link (3,4).

The underlay employs a work-conserving forwarding scheme that is "universally stable" as defined in [5]. This assumption ensures that if the number of packets injected into the underlay at each time-slot satisfies the capacity constraints of the tunnels, then the underlay queues are deterministically bounded. Specifically, under a universally stable forwarding policy, an underlay queue corresponding to link (a, b) is always deterministically bounded if

$$\sum_{l \in L:(a,b)\in l} \sum_{k} F_l^k(t) < c_{ab} \forall t. \tag{1}$$

Here $\{l \in L : (a, b) \in l\}$ is the set of tunnels that pass through the link (a, b). We refer to such constraints as the *tunnel capacity constraints*. Several work-conserving policies that are universally stable are given in [5].

III. Background

The problem of optimal routing in an overlay network was first studied in [6], where it was shown that backpressure routing, which is known to be throughput optimal in a range of scenarios, is not optimal for overlay networks, and proposed a heuristic called the Overlay Backpressure Policy (OBP). The OBP heuristic was conjectured to be throughput optimal.

For each tunnel l and commodity k OBP keeps track of the *packets in flight* H_l^k, which is the number of packets that have been transmitted into the tunnel by node l_1 but haven't reached node $l_{|l|}$. The weight for each commodity over the tunnel $W_l^k(t)$ is computed as follows

$$W_l^k(t) = Q_{l_1}^k(t) - H_l^k(t) - Q_{l_{|l|}}^k(t).$$

A link (i, j) that connects two overlay nodes can be thought of as a tunnel $l = (i, j)$ with no underlay node, hence the weight is computed as

$$W_l^k(t) = Q_{l_1}^k(t) - Q_{l_{|l|}}^k(t).$$

Then, the commodity with the highest weight sends its packets into the tunnel provided that the weight is positive. A precise description of the OBP is given in Algorithm 1.

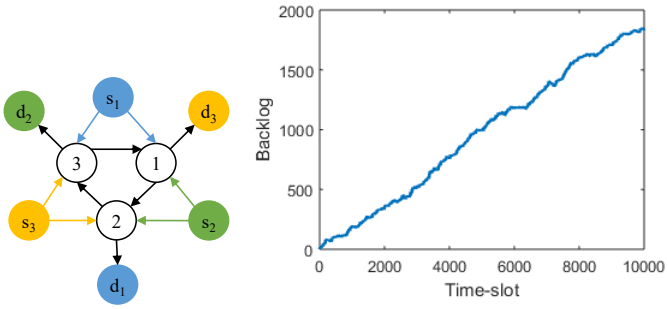
This policy makes sense intuitively because it encourages utilizing the tunnels that have less packets in them. When a tunnel is congested, the number of packets in flight is high, which encourages the overlay nodes to use alternate routes and send packets into the tunnel only when the backlog in the overlay is extremely high. This behavior is common to backpressure-based optimal routing algorithms. Moreover, OBP reduces to backpressure routing when all the nodes are overlay nodes.

Algorithm 1 Overlay Backpressure Policy (OPB):

For each tunnel l at each time-step t:

1) Compute the commodity k^* that maximizes the weight $W_l^k(t)$,
$$k^* \in \arg\max_k W_l^k(t).$$
Ties are broken arbitrarily.
2) Transmit μ packets into the tunnel where
$$\mu = \begin{cases} c_{l_1 l_2} \text{ if } W_l^{k^*}(t) > 0 \\ 0, \text{ otherwise,} \end{cases}$$
where $c_{l_1 l_2}$ is the capacity of the first link of tunnel l.

We present the following counterexample to show that the OBP is not throughput optimal. Consider a network topology given in Figure 3a where all the links are unit capacity. There are three commodities with source s_i and destination d_i, $i = 1, 2, 3$. The source and the destination are overlay nodes, whereas the nodes 1, 2 and 3 (in gray) are underlay nodes. The underlay nodes use the FIFO queuing discipline[2]. Each commodity in this network has two tunnels to the destination, e.g. $(s_1, 1, 2, d_1)$ and $(s_1, 3, 1, 2, d_1)$. Note that the shorter tunnels do not share any links between them, and if the shorter tunnel is chosen by each commodity, this network can support an arrival rate vector of $[1, 1, 1]$.



(a) Topology

(b) Total backlog in the network for arrival rate vector of $[0.8, 0.8, 0.8]$

Fig. 3: Counterexample for throughput optimality of the Overlay Backpressure Policy of [6].

Let us consider Poisson arrivals with the rate of $[0.8, 0.8, 0.8]$, which is clearly supportable in this network. To support this rate OBP has to send most of its traffic through

[2]From [9] we know that FIFO is throughput optimal for a ring which is the underlay topology in this example.

the shorter tunnels. However, as we show below, congestion can lead traffic to use longer tunnels, which leads to instability. A simulation result showing this instability is given in Figure 3b.

This instability is caused by the inability of the algorithm to prioritize the shorter tunnels. The only situation in which the longer tunnel is not used is when there are too many packets in flight inside it. Consider the situation where there number of packets in flight is large for tunnel $(s1, 3, 1, 2, d1)$. This means that the link (3,1) is being used by commodity 1 packets, which creates congestion for commodity 3. Now even higher number of packets in flight is required for tunnel $(s3, 2, 3, 1, d3)$ so that it is not used by commodity 3. This problem continues for the tunnels used by commodity 2, which in turn create congestion for the tunnels of commodity 1. This cyclical nature of increased congestion makes all the commodities unstable.

IV. Optimal Overlay Routing Policy

We begin by modeling the routing problem as a fluid optimization problem. Such models have been successfully used to design and analyze policies for communication networks in papers such as [10], [12]. Our optimization problem will have a zero objective function because we are only concerened with obtaining a routing algorithm and a feasible solution is sufficient for this purpose. We note that the technique from [10], [12] can be used for utility maximization and rate control.

Let f_{ij}^k be the flow assigned to commodity k on the link $(i,j) \in E$, and f_l^k be the flow assigned to commodity k on the tunnel $l \in L$. Let f denote the vector containing all the flow variables. The arrival rate of commodity k at overlay node i is represented by λ_i^k, and we assume that the vector of arrival rates λ is in the interior of the stability region. For simplicity, we will assume λ to be a constant, however, if it is time-variying, we note that the technical results hold as long as the arrival rate is bounded at each time-step and the expected value $E[\lambda(t)]$ exists. The problem of stabilizing the network queues can be formulated as a linear program that finds a feasible flow allocation on all the links and tunnels,

$$\max\ 0 \tag{2}$$

$$s.t. \sum_{l:(i,j)\in l} \sum_k f_l^k \le c_{ij}, \forall (i,j) : i \in U, j \in N \tag{3}$$

$$\sum_{l:(i,j)\in l} \sum_k f_l^k \le c_{ij}, \forall (i,j) : i \in \mathcal{O}, j \in U \tag{4}$$

$$\sum_j f_{ij}^k + \sum_{l:l_1=i} f_l^k - \sum_j f_{ji}^k - \sum_{l:l_{|l|}=i} f_l^k - \lambda_i^k \ge 0, \forall i \in \mathcal{O}, k \tag{5}$$

$$\sum_k f_{ij}^k \le c_{ij}, \forall i,j \in \mathcal{O} \tag{6}$$

$$f_{ij}^k, f_l^k \ge 0, \tag{7}$$

Here, the inequalities (3) are the tunnel capacity constraints which are the fluid version of (1). Each one of these constraints correspond to an uncontrollable link, i.e. a link between two underlay nodes or a link that goes from underlay to an overlay node. Inequalities (4) are the link capacity constraints corresponding to the first link in the tunnel, i.e. the links that go from an overlay node to an underlay node. This link is responsible for controlling the rate received by the underlay links. Inequalities (5) are the flow conservation constraints on the overlay network. Note that the flow conservation constraints are not required for the underlay since each tunnel l is assigned a single route and the flows coming into the underlay are feasible because of (3), i.e. for a tunnel l, when f is a feasible solution, $f_l^k = f_{l_1,l_2}^k = ... = f_{l_{|l|-1},l_{|l|}}^k$. Constraints (6) are the capacity constraints for the overlay links.

A. Dual problem

We now formulate the dual problem so that it can be solved with the subgradient descent method [10], [15]. Let q_{ij} and q_i^k denote the dual variables for the tunnel constraints (3) and the flow conservation constraints (5) respectively, and let q represent the vector containing all the dual variables. We can obtain the Lagrangian function and rearrange the terms to obtain the Lagrangian in the following form:

$$L(f,q) = \sum_l \sum_k f_l^k \left(q_{l_1}^k - \sum_{(i,j)\in l : i \in U} q_{ij} - q_{l_{|l|}}^k \right) + \sum_{(i,j)} \sum_k f_{ij}^k (q_i^k - q_j^k) + \sum_{(i,j):i\in U} q_{ij} c_{ij} - \sum_{i\in\mathcal{O},k} q_i^k \lambda_i^k. \tag{8}$$

Let X be a set such that any $f \in X$ satisfies the constraints (4), (6) and (7). Note that these constraints can be enforced locally by an overlay node using only locally available information. This property will be essential in designing the decentralized algorithm. The dual objective function corresponding to the problem (2) is

$$D(q) = \max_{f\in X} L(f,q).$$

The dual problem is given by,

$$\begin{aligned} \min_q \quad & D(q) \\ \text{s.t.} \quad & q \ge 0. \end{aligned} \tag{9}$$

Since the primal problem (3) is a linear program, the duality gap is zero (Slater's condition [15]). Hence, solution of the dual (9) yields a feasible flow allocation.

B. Distributed solution

The subgradient method works by initializing the dual variables with a value $q(0) \ge 0$, and then iterating on them until it converges to optimal $q^\star$. Each iteration involves computing a subgradient g of D at the current value of the dual variables, then updating the dual variables as follows:

$$q(t+1) = [q(t) - \alpha(t) g(t)]^+. \tag{10}$$

Here $\alpha(t)$ is positive scalar step-size. The dual variables are known to converge to the optimal $q^\star$ if the step-sizes $\alpha(t)$

are chosen appropriately. However, if $\alpha(t) \equiv \alpha$, then the iterates (10) converge to a bounded neighbourhood of $q^\star$ [15].

Let $f_l^{k^*}$ and $f_{ij}^{k^*}$ be the values of flow variables which maximize the Lagrangian $L(f,q)$ over $f \in X$ for a fixed q, i.e. $D(q) = L(f^*, q)$. From [15] we know that a subgradient of $D(q)$ is given by a vector g with entries as,

$$g_{ij} = c_{ij} - \sum_{l:(i,j)\in l} \sum_k f_l^{k^*}, \text{ and} \tag{11}$$

$$g_i^k = \sum_j f_{ij}^{k^*} + \sum_{l:l_1=i} f_l^{k^*} - \sum_j f_{ji}^{k^*} - \sum_{l:l_{|l|}=i} f_l^{k^*} - \lambda_i^k. \tag{12}$$

Now we can use the recursive equation (10) to update the dual variables.

The only necessary step that we haven't covered so far is the computation of $f_l^{k^*}$ and $f_{ij}^{k^*}$. A careful observation of equation (8) and the set X shows that this is a simple optimization problem that can be solved in a decentralized fashion. The objective is a weighted sum of the flow variables, and the constraints that form X are the link capacity constraints. At a high level, for each overlay link, the solution chooses the maximum value of the flow variable that corresponds to the commodity with the highest positive weight. A complete algorithm to compute the optimal flow variables and update the dual variables is given in Algorithm 2.

C. Queue-lengths as dual variables

The subgradient descent algorithm presented in the Algorithm 2 requires the network to explicitly keep track of the dual variables. In order to implement the algorithm in a decentralized fashion, each underlay node i needs to maintain a dual variable q_{ij} for each link (i,j), and each overlay node i needs to maintain a dual variable q_i^k for each commodity k. This is a reasonable assumption for the overlay nodes, but not justified for the uncontrollable underlay. To get around similar problems of not having a dual variable, works such as [13], [12], etc. have proposed approximating them with the corresponding queue lengths. The argument behind this procedure is that the subgradients are proportional to the change in queue-lengths, so that the queue-lengths will move in the same direction as the dual variables. Next, we give an example in which this proportionality does not hold. In spite of this issue, we show that the queue-lengths can provide a good approximation for the dual variables.

We first observe that the dual variable update equations (14) and (15) are the same as the queue update equations when the flows sent into the tunnels f_l^k are feasible for the underlay, i.e. when no queues buildup in the underlay. But when the flows do not satisfy the tunnel capacity constraints, the underlay queues build up, and the flows get reduced from their initial value as they pass through the bottleneck links. This decrease in the flow size is not captured in these dual variable update equations (14), (15). Consider the simple network shown in Figure 4. There is one commodity, $k = 1$, with source node 1 and destination node 4, and a single tunnel $l = (1,2,3,4)$.

Algorithm 2 Optimal Overlay Routing Policy (OORP)

At each time-step t, overlay node i does the following:

Optimal flow variables computation (used to obtain the subgradients):

An overlay to overlay link (i,j) computes the flow variables $f_{ij}^{k^*}$:

- Let $k^{opt} \in \arg\max_k q_i^k - q_j^k$, ties are broken arbitrarily. The weight of commodity k^{opt} in this link is $W_{ij}^{opt} = q_i^{k^{opt}} - q_j^{k^{opt}}$.
- For $k = k^{opt}$,

$$f_{ij}^{k^*} = \begin{cases} c_{ij} \text{ if } W_{ij}^{opt} > 0 \\ 0, \text{ otherwise} \end{cases}$$

- For all $k \neq k^{opt}$, $f_{ij}^{k^*} = 0$.

Each overlay to underlay link (i,j) computes the flow variable f_l^k for all $l : (l_1, l_2) = (i,j)$:

- Let

$$(l^{opt}, k^{opt}) \in \underset{l:(l_1,l_2)=(i,j),k}{\arg\max} q_{l_1}^k - \sum_{(a,b)\in l: a\in U} q_{ab} - q_{l_{|l|}}^k. \tag{13}$$

  Ties are broken arbitrarily. Let the weight of commodity k^{opt} in the tunnel be

$$W_l^{opt} = q_{l^{opt}_1}^{k^{opt}} - \sum_{(a,b)\in l^*: a\in U} q_{ab} - q_{l^{opt}_{|l^{opt}|}}^{k^{opt}}.$$

- For $(l,k) = (l^{opt}, k^{opt})$,

$$f_l^{k^*} = c_{ij} \text{ if } W_l^{opt} > 0, \text{ and 0 otherwise}$$

- For all $(l,k) : l \neq l^{opt}$ or $k \neq k^{opt}$, $f_l^{k^*} = 0$.

Data transmission:

Transmit $f_{ij}^{k^*}$ amount of commodity k traffic into each overlay to overlay link (i,j) and $f_l^{k^*}$ amount of commodity k traffic into each tunnel l.

Dual variables update:

Performed by an overlay node i:

$$q_i^k(t+1) = \left[q(t) - \alpha(t) \left(\sum_j f_{ij}^{k^*} + \sum_{l:l_1=i} f_l^{k^*} - \sum_{j:(j,i)\in E} f_{ji}^{k^*} - \sum_{l:l_{|l|}=i} f_l^{k^*} - \lambda_i^k \right) \right]^+ \tag{14}$$

Performed by an underlay node i:

$$q_{ij}(t+1) = \left[q(t) - \alpha(t) \left(c_{ij} - \sum_{l:(i,j)\in l} \sum_k f_l^{k^*} \right) \right]^+ \tag{15}$$

Suppose that at a certain iteration, $q_1^1 > q_4^1$, hence $f_l^{k^*} = 3$. This flow into the tunnel gets bottlenecked at link $(2,3)$ so node 3 only receives a flow of 1. In this situation, equation (15) predicts that the queue-length for q_{34} would increase because

a flow of size 3 was sent into the tunnel and the capacity of the link is 2, however this queue can only decrease or stay unchanged at 0.

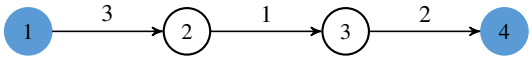


Fig. 4: Link $(3,4)$ never builds a queue as the flow gets bottlenecked by (2,3).

To capture this reduction of the flow sizes in the tunnel, we model the queuing in the network as follows:

$$\hat{q}_i^k(t+1) = \left[\hat{q}(t) - \alpha(t)\left(\sum_j {f_{ij}^k}^* + \sum_{l:l_1=i} {f_l^k}^* - \sum_{j:(j,i)\in E} {f_{ji}^k}^* - \sum_{l:l_{|l|}=i} \epsilon_l^k(i){f_l^k}^* - \lambda_i^k\right)\right]^+ \quad (16)$$

$$\hat{q}_{ij}(t+1) = \left[\hat{q}(t) - \alpha(t)\left(c_{ij} - \sum_{l:(i,j)\in l}\sum_k \epsilon_l^k(i,j){f_l^k}^*\right)\right]^+ \quad (17)$$

where $\epsilon_l^k(i), \epsilon_l^k(i,j) \in [0,1]$ represent the reduction suffered by the corresponding flows before arriving at node i. These quantities are implicitly determined by the network at each time-step depending on the scheduling policy in the underlay. In the example presented above, for any work conserving scheme, $\epsilon_l^k(3,4) = 1/3$. We will show that for any value of ϵ in the set $[0,1]$ the queue-lengths will converge to the optimal dual variables. Let g be the true subgradient of D at q, and $\hat{g}$ be the approximate subgradient after the reduction, then we can represent the queuing equation as

$$\hat{q}(t+1) = \left[\hat{q}(t) - \alpha(t)\hat{g}(t)\right]^+,$$

and $\hat{g} \geq g$.

Before we prove the convergence, we state the following preliminary lemma.

Lemma 1. *The vector $q^* = 0$ is an optimal solution to the dual problem (9).*

Proof. Since the objective of the primal problem is 0, a feasible solution to the primal is given by any feasible flow allocation f_{ij}^k. Since $q = 0$ is a feasible dual solution, and any feasible f_{ij}^k together with $q = 0$ satisfy the complementary slackness condition (Theorem 4.5 in [14]), the proof follows. □

This shows that the optimal solution corresponds to queue lengths equal to zero, which makes sense intuitively because any feasible flow allocation in the fluid domain doesn't require queuing.

Let G be a constant such that it bounds the Euclidean norm of the subgradients of the dual function $D(q)$ for all possible values of q, i.e. $G > \|g\|$. From equations (11)-(12), we can see that the subgradients are bounded because the flow variables are bounded by link capacities and arrival rates are bounded by assumption. So G is finite. For simplicity we fix $\alpha(t) = 1$ and present the following convergence result.

Theorem 1. *Let us approximate the dual variables q with the queue-lengths $\hat{q}$ that evolve according to equations (16)-(17). Using the dual subgradient descent algorithm with $\alpha(t) = 1$, the queue lengths converge to a bounded set.*

Proof. We will show that $||\hat{q}(t+1) - q^*||^2 < ||\hat{q}(t) - q^*||^2$ when $q(t)$ is outside the set S. Because $q^* = 0$ from Lemma 1, it suffices to show that $||\hat{q}(t+1)||^2 < ||\hat{q}(t)||^2$.

We have,

$$||\hat{q}(t+1)||^2 = \|\hat{q}(t) - \hat{g}\|^2$$

Since $\hat{g} \geq g$,

$$\begin{aligned}||\hat{q}(t+1)||^2 &\leq \|\hat{q}(t) - g\|^2 \\ &= \|\hat{q}(t)\|^2 - 2\hat{q}(t)^T g + \|g\|^2\end{aligned}$$

Our algorithm chooses g to be a subgradient of $D(.)$ at $\hat{q}(t)$. So,

$$D(x) \geq D(\hat{q}(t)) + (x - \hat{q}(t))^T g, \forall x \in \mathbb{R}^m,$$

wehre m is the dimension of $\hat{q}$. Taking $x = 0$,

$$D(\hat{q}(t)) \leq \hat{q}(t)^T g(f(t)^*)$$

So, $||\hat{q}(t+1)||^2 \leq ||\hat{q}(t)||^2 - 2D(\hat{q}(t)) + G^2$. Hence when, the $\hat{q}$ is far away from the optimal, specifically when $D(\hat{q}(t)) > \frac{1}{2}G^2$, it moves towards the optimum in the next time-step. □

Hence, we will use queue-lengths instead of the dual variables in the implementation of OORP. This will allow us to use the policy presented in Algorithm 2 without having to perform the dual variables update.

V. Estimation of Underlay Queues-lengths

In the previous section we showed that the dual subgradient descent algorithm can be used to compute a feasible rate for each commodity on each link. We also showed that the queue lengths can be used to approximate the subgradient. However, typically legacy devices are not be able to send queue-lengths to the sources, hence we explore methods to estimate them from only the data that is available at the overlay. To make the problem simpler, we consider networks where the control packets have a high priority and consume negligible capacity, hence, the feedback between the overlay nodes is immediate.

From equation (13) we can see that in order to compute the subgradients we only need the total backlog in the tunnel, i.e. we don't need the length of individual queues. Hence, the goal of the estimation approaches is to approximate the tunnel backlog

$$b_l(t) = \sum_{(a,b)\in l: a\in U} q_{ab}(t).$$

A. Delay based estimation

A natural approach to estimate the total backlog in a tunnel is by using the time it takes for a packet to traverse it. Similar approach has been used by many versions of TCP, such as Vegas [19] and FAST [20], to estimate the congestion along a path. Although this approach is simple and does not require cooperation from the underlay, the queue-length estimates obtained by this method can be arbitrarily bad.

Consider a FIFO queue that is empty at time zero. As shown in Figure 5(a), it has an incoming rate of 2 and outgoing capacity of 1. We want to estimate the queue-length at time t by using packet delays. To see the problem with this approach, let us consider a situation when 2 packets arrive at the queue at every time-slot for the first τ time-slots, and no arrivals happen after that. In this situation, the actual queue length grows at the rate of 1 for the first τ time-slots, and then it decreases at the rate of 1 packet per time-slot until the queue is empty. On the other hand, the delay increases at the rate of $\frac{1}{2}$, and the last packet (that arrives at the τth time-slot) sees a delay of 100 because there are 99 packets in the queue at that time. So at time 2τ when the queue is emptied, the packet received will have suffered a delay of τ time-slots giving a queue-length estimate of τ, whereas the actual queue-length at that time is zero. Furthermore, the estimate stays bad until another arrival happens. This problem is illustrated in Figure 5(b). These arbitrarily bad estimates lead to sub-optimality of OORP which we will observe in the simulations.

A simple way to improve the estimate is to send empty probe packets when real packets are not available for some time period $\mathcal{T}$. A similar approach has been shown to achieve throughput optimality in some special network settings in [21]. This approach quickly identifies when a queue becomes empty in the absence of new data packets, and the control algorithm can react accordingly. Although this approach corrects the estimate within $\mathcal{P}$ time-slots, it can still suffers from the arbitrarily bad estimation errors. As shown in Figure 5(c), at time 2τ the estimate is τ whereas the actual queue-length is zero.

B. Learning based estimation

In order to motivate a simple model for tunnel backlog, we first observe that the packets in flight, i.e. the number of packets that has entered tunnel l but hasn't exited it, $h_l(t)$, has information about the backlog in the underlay. For example, when tunnel l does not intersect with other tunnels $h_l(t) = b_l(t)$. However, when two or more tunnels intersect, each tunnel can have a fraction of packets from another tunnel along with its own packets, so $b_l(t)$ can be larger than $h_l(t)$. Moreover, when $h_l(t)$ is large, a large number of packets are accumulated at a few bottleneck links. So we model tunnel backlog as a function of the packet in flight:

$$b_l(t) = f(h(t)),$$

where $h(t)$ is a vector of the packet in flight in all the tunnels in the network. Note that $h_l(t)$ is readily available to the overlay network at each time-slot.

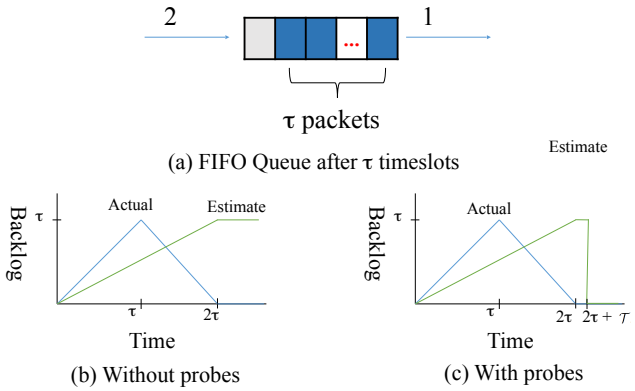


Fig. 5: The actual queue-length of a single FIFO queue and its estimate calculated using delay. The arrivals happen at the rate of 2 packets per time-slot for the first τ time-slots, and there are no arrivals after that. Service rate is fixed at 1 packet per time-slot.

However, in order to train this model, we also need the data on the dependent variable $b_l(t)$ which might not be available to the overlay. For such a situation, we make a second observation: the delay experienced by a packet that exits a tunnel at time t gives a good estimate of the backlog in the tunnel at the time when the packet entered the tunnel, i.e. at time $t-x$ where x is the delay experienced by the packet in the tunnel. So, $b_l(t-x) \approx cx$ where c is the bottleneck capacity of the tunnel. For example, in Figure 5 we can see that the delay experienced by the packet that entered at time τ (which is the packet exiting at time 2τ) gives a good estimate of the backlog in the tunnel at time τ. Hence, we can use the delay of a packet entering at time $t-x$ to approximate the tunnel backlog at that time. Note that we cannot use this quantity in real time because the delay experienced by the packet entering a tunnel at time $t-\tau$ is not available until the packet exits the tunnel at time t. Also note that, we have already shown in the previous subsection that using the (delayed) delay estimate can lead to instability.

1) Linear model using least squares regression: We develop a simple linear model of the form

$$b_l(t) = \alpha_l^T h(t), \tag{18}$$

were h is the vector of the packets in flight in all the tunnels, α_l is a vector of model parameters, and $(.)^T$ represents the transpose of the vector.

Let H be the matrix of historical packets in flights and B_l be the vector of historical backlogs in tunnel l (either actual or estimated through delay). Row i of H is the packets in flights in all the tunnels at a certain time-slot t, and the ith entry in B_l is the backlog in tunnel l at the same time-slot. It is well known that the vector of parameters α_l that fits the linear model and minimizes the Euclidian norm of the error for the samples is given by [22],

$$\alpha_l = (H^T H)^{-1} H^T B_l.$$

Note that this model is not a special case of the piecewise linear model that is generated by the MARS algorithm described below because MARS is a greedy algorithm and it is not guaranteed to minimize the error.

2) Piecewise linear model using the MARS algorithm: Multiple Adaptive Regression Splines (MARS) described in [23] is a popular algorithm to fit a piecewise linear function to data. MARS builds the piecewise linear model by taking a linear combination of "hinge" functions of the form $\max(h_i - x_i, 0)$ or $\max(x_i - h_i, 0)$. Here h_i is an independent variable such as the packets in flight, and x_i is a constant that represents the location where the two lines (pieces) connect, known as the "knot". The set of possible basis function B comprises of all the hinge functions with a knot at each value of each input variable. With a single variable h_1 and p samples (of h_1) given by $x_1, ..., x_p$, the set of basis functions is given by $B = \{\max(h_1 - x_i, 0), \max(x_i - h_1, 0)\}_{i \in \{1,2,...,p\}}$.

The model produced by MARS has the form

$$b_l = f(h) = \sum_{i=1}^{K} (\alpha_l)_i B_i,$$

where $B_i \in B$ is the ith basis function, α_l is the vector of weights which determines the slope of the lines and $(\alpha_l)_i$ is its ith element, and K is the number of basis functions in the model.

The MARS algorithm is a greedy iterative algorithm that operates in two phases. In the first phase, the algorithm adds one basis function per iteration until the maximum number of basis functions allowed, a parameter to the algorithm, is reached. At each iteration, all the basis functions are tested one at a time by choosing the slope using the least squares regression method. Then the basis function that gives the largest decrease in error is added to the model. This usually causes the model to over-fit. Thus, in the second phase MARS removes one basis function at a time. Again, the removal is done greedily such that the basis function that adds the least error is removed in each iteration. The removal of the basis function continues until a generalized cross validation condition is satisfied.

VI. Simulation Results

We consider the network given in Figure 6 to study the effect of estimating the backlog in the tunnels. In this network, all the links are bidirectional, composed of two unidirectional links. The links between an overlay and an underlay node have capacity 2 in each direction. All other links have unit capacity in both directions. We will simulate the network with two commodities. The first commodity is defined by the source-destination pair (1,3) and the second is defined by (2,4). For these commodities the network supports a max-flow vector of $\lambda_{\max} = [2, 2]$. The simulation is performed at various load levels and the arrivals are Poisson distributed. The underlay uses shortest path routing.

The source nodes 1 and 2 are connected to two underlay nodes, hence, they have two tunnels to every other overlay nodes. Notice that in order to achieve a throughput close to $\lambda_{\max}$ node 1 must send its traffic through node 2 and have it forward it to node 3, and similarly node 2 must send some of its traffic through node 1. Also, we can see that sending packets in the wrong tunnel can cause the network to become congested and lose throughput. This make this network particularly challenging to stabilize under high load.

For the purpose of training our learning models, we obtained the delay and packets in flight data by using the delay based method for 10^5 time-slots at 80%. To generate the piecewise linear model, we used an open-source Matlab implementation of the MARS algorithm called ARESLab from [24] with default parameters.

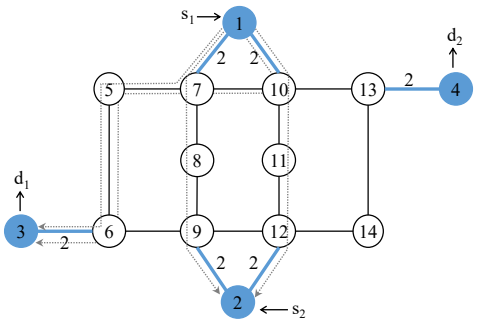


Fig. 6: Physical network topology with overlay (blue) and underlay (white) nodes. The underlay network uses shortest path routing creating a total of eighteen tunnels between the overlay nodes. The dotted lines show the tunnels from node 1 to nodes 2 and 3.

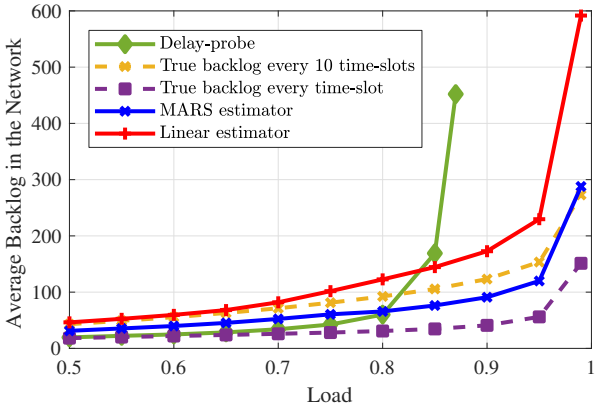


Fig. 7: Performance comparison of OORP under various tunnel-backlog estimation schemes. The estimation methods (solid lines) perform very well without any knowledge of the underlay network.

The results of the simulation are given in Figure 7. We plot the average queue-lengths of all the queues in the network (both underlay and overlay) at various load levels. Average backlog is a good measure of performance as it can show instability in the network as well as the average delay ex-

perienced by packets since average delay is proportional to average backlog.

The dotted lines show the performance of the OORP algorithm when the exact queue-lengths are known to the overlay. This simulates the situation when the underlay is cooperative and sends the backlog information to the sources at certain time intervals. As expected the best performance is obtained when the true backlog is known at every time-slot. We can also see that the performance of OORP degrades gradually as the true backlog is obtained less frequently, i.e. every 10 time-slots. This shows that if the underlay nodes are designed to send the backlog information to the overlay, it can make the optimal implementation of OORP very simple.

We can see that the delay-probe method of estimating the tunnel backlog results in the sub-optimality of OORP. This method uses delay experienced by a packet exiting a tunnel at time t as an estimate of the backlog in the tunnel at time t. When there is no new packets to send for T (10 for this experiment) time-slots, this scheme sends empty probe packets in order to probe the delay. In the simulation, these probe packets did not consume any capacity. Intuitively, this scheme gives a delayed estimate of the congestion in the tunnels, where the delay in the estimate is proportional to the congestion. Thus, when the network is heavily loaded the estimates are highly inaccurate leading the network to become unstable under OORP.

We also can see that the regression based estimation methods achieve stability and good performance. The linear model obtains very good performance for loads as high as 95%. The piecewise linear model generated using the MARS algorithm performs even better. At most load levels, this method achieves a better performance than even the scheme that allows the underlay to send the true backlog every 10 time-slots. This shows that we can use OORP with this estimation scheme to achieve good performance in an overlay network setting without the cooperation from the underlay network. We refer the readers to [8] for more simulation results.

VII. Conclusion

We showed that the existing algorithms for routing traffic in an overlay network are suboptimal, and developed a throughput optimal policy called the Optimal Overlay Routing Policy (OORP). This policy is distributed and works without the knowledge of the underlay topology. Our algorithm requires the total backlog in each underlay tunnel as feedback, which might not be available to the overlay nodes. Hence we proposed different approaches to estimating underlay congestion. Simulations results show that estimating congestion using probing mechanism is effective but suboptimal, and the regression based approaches maintain throughput optimal of OORP and perform very close to the case with the true values of the tunnel backlog are known.

References

[1] L. Tassiulas and A. Ephremides. "Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks." IEEE Transactions on Automatic Control, Dec. 1992.

[2] B. Awerbuch and T. Leighton. "A Simple Local-Control Approximation Algorithm for Multicommodity Flow." Proceedings 34th IEEE Conference on Foundations of Computer Science, Oct. 1993.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. "Resilient overlay networks." In Proceedings of ACM SOSP, October 2001.

[4] J. Han, D. Watson, and F. Jahanian. "Topology aware overlay networks." In Proceedings IEEE INFOCOM, March 2005.

[5] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu, and J. Kleinberg. "Universal-stability results and performance bounds for greedy contention-resolution protocols." Journal of the ACM 48, 1 (January 2001), 39-69.

[6] N. M. Jones, G. S. Paschos, B. Shrader and E. Modiano, "An Overlay Architecture for Throughput Optimal Multipath Routing," in IEEE/ACM Transactions on Networking, vol. 25, no. 5, pp. 2615-2628, Oct. 2017.

[7] G. S. Paschos and E. Modiano. "Throughput optimal routing in overlay networks." In Proceedings of the Allerton Conference, 2014.

[8] A. Rai. "Towards practical policies for network control." Doctoral dissertation, http://hdl.handle.net/1721.1/120417, 2018.

[9] L. Tassiulas and L. Georgiadis. "Any work-conserving policy stabilizes the ring with spatial re-use." Networking, IEEE/ACM Transactions on 4.2 (1996): 205-208.

[10] S. H. Low, and D. E. Lapsley. "Optimization flow control—I: basic algorithm and convergence." IEEE/ACM Transactions on Networking, 1999.

[11] M. J. Neely, E. Modiano, and C. Li. "Fairness and Optimal Stochastic Control for Heterogeneous Networks." In Proceedings of IEEE INFOCOM, March 2005.

[12] X. Lin, N. B. Shroff and R. Srikant. "A tutorial on cross-layer optimization in wireless networks." IEEE Journal on Selected Areas in Communications, vol. 24, no. 8, pp. 1452-1463, Aug. 2006.

[13] S. H. Low, L. L. Peterson, and L. Wang. 2002. "Understanding TCP Vegas: a duality model." Journal of the ACM 49, 2 (March 2002).

[14] D. Bertsimas and J. Tsitsiklis. "Introduction to Linear Optimization." Athena Scientific, 1997.

[15] D. Bertsekas, A. Nedic, and A. E. Ozdaglar. "Convex Analysis and Optimization." Athena Scientific, 2003.

[16] L. Georgiadis, M. J. Neely, and L. Tassiulas. "Resource Allocation and Cross-Layer Control in Wireless Networks." Foundations and Trends in Networking, 2006.

[17] Z. Li, and P. Mohapatra. "QRON: QoS-aware routing in overlay networks." IEEE Journal on Selected Areas in Communications 22.1 (2004): 29-40.

[18] R. K. Sitaraman, M. Kasbekar, W. Lichtenstein, and M. Jain. "Overlay Networks: An Akamai Perspective." John Wiley & Sons, 2014.

[19] L.S. Brakmo and L.L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," IEEE Journal on Selected Areas in Communications, Oct 1995.

[20] Wei, D. X., Jin, C., Low, S. H., Hegde, S. "FAST TCP: motivation, architecture, algorithms, performance. IEEE/ACM transactions on Networking," 14(6), 1246-1259, 2006.

[21] G. Paschos, M. Leconte, and A. Destounis. "Routing with Blinkers: Online Throughput Maximization without Queue Length Information." In Proceedings of the International Symposium on Information Theory, 2016.

[22] Bertsekas, Dimitri P., and John N. Tsitsiklis. Introduction to probability. Vol. 1. Belmont, MA: Athena Scientific, 2002.

[23] Friedman, Jerome H. "Multivariate adaptive regression splines." The annals of statistics (1991): 1-67.

[24] Gints Jekabsons, "ARESLab: Adaptive Regression Splines toolbox," http://www.cs.rtu.lv/jekabsons/regression.html, ver. 1.13.0, retrieved July 2, 2017.