Fast scheduling for optical flow switching

Zhang, Lei, and Vincent Chan. “Fast Scheduling of Optical Flow Switching.” IEEE, 2010. 1–6. © Copyright 2010 IEEE

http://dx.doi.org/10.1109/GLOCOM.2010.5683194

Institute of Electrical and Electronics Engineers (IEEE)

Final published version

Sun Jan 13 11:50:05 EST 2019

http://hdl.handle.net/1721.1/71913

Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.
Abstract—Optical Flow Switching (OFS) is a promising architecture to provide end users with large transactions with cost-effective direct access to core network bandwidth. For very dynamic sessions that are bursty and only last a short time (∼15S), the network management and control effort can be substantial, even unimplementable, if fast service of the order of one round trip time is needed. In this paper, we propose a fast scheduling algorithm that enables OFS to set up end-to-end connections for users with urgent large transactions with a delay of slightly more than one round-trip time. This fast setup of connections is achieved by probing independent paths between source and destination, with information about network regions periodically updated in the form of entropy. We use a modified Bellman-Ford algorithm to select the route with the least blocking probability. By grouping details of network states into an average entropy, we can greatly reduce the amount of network state information gathered and disseminated, and thus reduce the network management and control burden to a manageable amount; we can also avoid having to make detailed assumptions about the statistical model of the traffic.

I. INTRODUCTION

Optical Flow Switching (OFS) [1] is a key enabler of scalable future optical networks. It is a scheduled, end-to-end transport service, in which all-optical connections are set up prior to transmissions upon end users’ requests to provide them with cost-effective access to the core network bandwidth [2]–[4]. In particular, OFS is advantageous for users with large transactions. In addition to improving the quality of service of its direct users, OFS also lowers access costs for other users by relieving metropolitan and wide area network (MAN/WAN) routers from packet switching large transactions.

To achieve high network utilization in OFS, all flows go through the MAN schedulers to request transmission and are held until the network is ready [2]. These procedures normally account for queuing delays of a few transaction durations at the sender when the network has a high utilization. Some special applications, however, have tight time deadlines and are willing to pay more to gain immediate access. Some examples that may use OFS as a fast transport could be grid computing, cloud computing, and bursty distributed sensor data ingestion. The demand of fast transport of large transactions with low delay calls for a new flow switching algorithm that bypasses scheduling but still obtains a clear connection with high enough probability. To differentiate the new algorithm from the normal scheduling algorithm of OFS which utilizes schedulers on the edges of WAN, we call it “fast scheduling” in this paper.

In [5], the authors have designed and analyzed fast scheduling algorithms for OFS that meet setup times only slightly longer than one round-trip time. The connection is set up by probing independent candidate paths as announced periodically by the scheduler from source to destination and reserving the available paths along the way. To make the analysis of the required number of paths to probe tractable, they assumed statistical models of homogeneous Poisson traffic arrival and exponentially distributed departure processes for all the paths connecting source and destination, which is unrealistic and not robust to model variations. For the heterogeneous traffic case, they would assume the complete statistics of every link are updated periodically in the control plane. This control traffic itself can be large (∼32 Gbps) in a network of the size of the US backbone network and also highly dependent on the statistical models of arrivals and service times.

In this work, we designed a fast scheduling algorithm for OFS which also utilizes the probing approach but does not depend on the assumptions of the statistics of the traffic. The evolution of the network state of each network domain is summarized by one measurable parameter: the average entropy. We chose entropy as the metric because entropy is a good measure of uncertainties. In particular, the higher the entropy, the less certain we are about the network state. In the control plane, the sampled entropy evolution and the set of available paths are broadcast periodically at two different time scales. The entropy evolution is broadcast with a period of its coherence time (that can range from several minutes to several hours, depending on the actual traffic statistics), whereas the set of available paths is broadcast with a period of 0.3 to half of the average transaction time (≥1S). With the updated information of entropy evolution, we can get a close approximation of the average entropy at any time in the next interval between state broadcasts. We have shown that the number of paths we need to probe to satisfy a target blocking probability increases monotonically with the increase of entropy, which makes sense since larger entropy means less certainty on the network state and thus we need to probe more paths. Therefore, the algorithm chooses paths from different network domains by selecting the ones with the least total average entropy. Within one network domain, as the information about its internal states (e.g. the availabilities of each individual links) are aggregated into one single parameter, the entropy, we will lose some of the detailed statistics of the arrival and departure processes, resulting in probing more paths than is necessary if detailed statistics are available. However, this
Algorithm is not dependent on detailed assumptions of the models and thus is much more robust. Moreover, our algorithm reduces control traffic to \(1/M\) at the coarse time scale, where \(M\) is the number of paths or network domains over which we average the evolutions of entropy. By grouping multiple paths or network domains together, we can reduce the control traffic in unit of packets at the coarse time scale into \(1/M\) of that in [5].

II. ENTROPY-ASSISTED PROBING ALGORITHM FOR A SIMPLE NETWORK

There are two states for any path, either occupied or available. Let \(X(t)\) be the probability that the path is blocked (i.e. occupied) at time \(t\). Then we can define the binary entropy of the path at time \(t\) to be:

\[
H_b(t) = -X(t) \log_2(X(t)) - (1-X(t)) \log_2(1-X(t)).
\]  (1)

If the path is available at time zero, then \(H_b(0) = 0\). As time passes, we become less and less certain about the availability of the path, and \(H_b(t)\) increases; until when we totally lose track of the path’s status, \(H_b(t)\) reaches to its maximum and can no longer give us any useful information about the path’s current status except for its long term average load. Take for example, a path with a Poisson traffic arrival process with arrival rate \(\lambda\) and exponential service distribution time with mean \(\mu\). If we know at time \(t\) the path is available, its entropy evolution can be calculated as:

\[
H(t) = -\left(\frac{1}{\rho+1} + \frac{\rho}{\rho+1} e^{-(1+\rho)\mu t}\right) \cdot \log_2\left(\frac{1}{\rho+1} + \frac{\rho}{\rho+1} e^{-(1+\rho)\mu t}\right) - \left(\frac{\rho}{\rho+1} - \frac{\rho}{\rho+1} e^{-(1+\rho)\mu t}\right) \cdot \log_2\left(\frac{\rho}{\rho+1} - \frac{\rho}{\rho+1} e^{-(1+\rho)\mu t}\right),
\]  (2)

where \(\rho = \lambda/\mu\), is the loading factor.

As shown in Fig. 1, if \(\rho\) is smaller than or equal to one, \(H(t)\) keeps increasing until it reaches its maximum of one; if \(\rho\) is greater than one, \(H(t)\) first increases to its maximum of one and then decreases to its steady state value. Since the entropy evolution within each fine interval between broadcasts of the open paths should not exceed the time it reaches its maximum, we limit the fine broadcast interval to be 0.3 to 0.5 of the average transaction time.

In the following analysis, we assume the entropy at any time \(t\) can be approximated from the sample statistics, and study the problem of how to determine the number of paths to probe given the information of entropy. The methods of estimating \(H(t)\) from sample statistics of the online network are proposed in Section IV.

A. Network Model

Figure 2 shows the model of a simple network with one source-destination pair and \(m\) paths between them. The blocking probability of each path is an identically and independently distributed random variable \(X\). Set \(A\) is the set of available paths with size \(N(A)\). Within the \(N(A)\) paths, we randomly pick \(N\) of them such that the total blocking probability of the \(N\) paths is smaller than or equal to some target blocking probability \(P_B\).

Mathematically, to meet \(P_B\), we select \(N\) paths to probe such that:

\[
\prod_{i=1}^{N} X_i \leq P_B < \prod_{i=1}^{N-1} X_i,
\]

that is,

\[
\sum_{i=1}^{N} \log_2 X_i \leq \log_2 P_B < \sum_{i=1}^{N-1} \log_2 X_i.
\]

Take expectation of both sides,

\[
\bar{N} \cdot E[\log_2 X] \leq \log_2 P_B < (\bar{N} - 1) \cdot E[\log_2 X].
\]

Therefore,

\[
\frac{-\log_2 P_B}{E[-\log_2(X)]} + 1 \geq \frac{-\log_2 P_B}{E[-\log_2(X)]}.
\]

The average entropy of the network is

\[
\bar{H} = \frac{H_1 + H_2 + \cdots + H_{N(A)}}{N(A)}.
\]

Take expectation of both sides, we get

\[
E[\bar{H}] = E[H_1 + H_2 + \cdots + H_{N(A)}] = E[H].
\]
B. Problem Formulation

Since \( -\log_2 P_0 \) can be less than \( \bar{N} \) by at most one, in the following study we approximate \( \bar{N} \) by \( -\log_2 P_0 \). As we are interested in determining the average number of paths to probe based on the value of the average entropy, in the following part of this section, we want to find the upper bound of \( \bar{N} \) for a given \( E[\bar{H}] \).

Let \( f_X(x) \) be the density function of \( X \). Since we only pick paths out of the available set \( A \), and \( H(t) \) increases within one broadcast interval, the blocking probabilities of each path in \( A \) is smaller than 0.5. Therefore, for a sampled entropy of \( h_0 \), we have the following conditions for \( f_X(x) \):

\[
\begin{align*}
\int_0^{0.5} f_X(x)dx & = 1, \\
E[H_b(X)] & = h_0.
\end{align*}
\]

Let \( C \) be the set of all density functions \( f_X(x) \) that satisfy the above conditions. The upper bound of \( \bar{N} (\bar{N}_{\text{max}}) \) can be obtained by solving the following optimization problem:

\[
\begin{align*}
\text{(P1)}: \bar{N}_{\text{max}} = \max_{f_X(x) \in C} -\log_2 P_2 & \quad \text{subject to} \quad f_X(x) \geq 0, \quad \text{for } x \in [0, 0.5], \\
& \int_0^{0.5} f_X(x)dx = 1, \\
& E[H_b(X)] = h_0.
\end{align*}
\]

which can be solved by first solving:

\[
\begin{align*}
\text{(P2)}: \min_{f_X(x) \in C} E[-\log_2(X)] & \quad \text{subject to} \quad f_X(x) \geq 0, \quad \text{for } x \in [0, 0.5].
\end{align*}
\]

In fact, we only need to consider the discrete random variable solution of \( \text{P2} \) (See Appendix), and then \( \text{P2} \) can be reformulated to a Linear Programming (LP) problem. To see this, let \( x_1, x_2, ..., x_n \) be any \( n \) different possible values for \( X \) in \([0, 0.5]\), and \( y_1, y_2, ..., y_n \) be the probability weights for \( x_1, x_2, ..., x_n \) (i.e. \( P_r\{X = x_i\} = y_i \)). Then (4) can be rewritten as:

\[
\begin{align*}
\sum_{i=1}^{n} y_i = 1, \\
\sum_{i=1}^{n} y_i H_b(x_i) = h_0, \\
y_i \geq 0, \quad \text{for } i \in \{1, ..., n\}.
\end{align*}
\]

Subjecting to the above conditions, we need to minimize \( \sum_{i=1}^{n} y_i [\log_2(x_i)] \). To further transform the conditions and the problem, we define the following vectors:

\[
\begin{align*}
1 & = \begin{bmatrix} 1 & 1 & 1 & ... & 1 \end{bmatrix}, \\
y & = \begin{bmatrix} y_1 & y_2 & ... & y_n \end{bmatrix}, \\
h & = \begin{bmatrix} H_b(x_1) & H_b(x_2) & ... & H_b(x_n) \end{bmatrix}, \\
g & = \begin{bmatrix} -\log_2(x_1) & -\log_2(x_2) & ... & -\log_2(x_n) \end{bmatrix}, \\
H & = \begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix}.
\end{align*}
\]

Then, the conditions in (6) are equivalent to:

\[
\begin{align*}
1^T y & = 1, \\
h^T y & = h_0, \\
y_i \geq 0, \quad \text{for } i \in \{1, ..., n\}.
\end{align*}
\]

Let \( \mathcal{Y} \) be the polyhedron defined by \( y \) subjecting to conditions in (7). With the new representations, \( \text{P2} \) can be transformed into:

\[
\text{(P3)}: \min_{y \in \mathcal{Y}} g^T y,
\]

which is an LP problem. For the LP problem of minimizing \( g^T y \) over \( y \in \mathcal{Y} \), if there exists an optimal solution, there exists a basic feasible optimal solution, denoted by \( y^* \). For a basic feasible solution, there are \( n \) linearly independent active constraints on \( y^* \). In conditions (7), we already have two such constraints, \( 1^T y = 1 \) and \( h^T y = h_0 \). Therefore, we need \((n-2)\) other constraints such that \( y_i = 0 \). Intuitively, since \( g \geq 0 \) and \( y \geq 0 \), the minimum of \( g^T y \) is achieved by letting the \((n-2)\) \( y_i \)’s for the largest \((n-2)\) \( g_i \)’s equal to zero.

As a consequence, for any chosen set of discrete values of \( X \), the optimization problem \( \text{P3} \) can always reduces to a problem where only two of the \( y_i \)’s are greater than or equal to zero. In other words, there are only two possible values for \( X \). Therefore, one optimum \( f_X(x) \) that minimizes \( \text{P2} \) can be written as:

\[
\begin{align*}
f_X(x) & = \alpha \delta(x-x_1) + (1-\alpha) \delta(x-x_2), \\
\text{where } \alpha & \in (0, 1), \quad x_1 \in (0, 0.5), \quad \text{and } x_2 \in (0, 0.5). \quad \text{Hence, } \text{P2} \text{ can be reduced to:}
\end{align*}
\]

\[
\begin{align*}
\min_{\alpha \in [0,1], x_1, x_2 \in [0,0.5]} & \quad \alpha H_b(x_1) + (1-\alpha) H_b(x_2) + \alpha \log_2(x_1) - (1-\alpha) \log_2(x_2) = 1 \\
\text{subject to: } & \quad \alpha H_b(x_1) + (1-\alpha) H_b(x_2) = h_0.
\end{align*}
\]

With this transformation, the optimal solution to \( \text{P2} \) can be readily solved as:

\[
\begin{align*}
E[-\log_2(X)] & = \left\{ \begin{array}{ll}
-\log_2[H_b^{-1}(h_0)] & \frac{1}{1-h_0} \log_2(h_0) \quad \text{if } h_0 \leq h_A,
\frac{1}{1-h_0} \log_2(h_0) + h_0 - h_A & \text{if } h_0 > h_A \\
\end{array} \right.
\end{align*}
\]

where \( H_b^{-1}(h_0) \) is the inverse function of \( H_b(x) = h_0 \) for \( x \in (0,0.5) \). \( h_A \) is the solution to

\[
H_b^{-1}(h) \log_2 \frac{1}{1-H_b^{-1}(h)} = \frac{h - 1}{\log_2 H_b^{-1}(h) + 1},
\]

and, numerically, \( h_A \approx 0.4967 \).

Substituting (9) into (5), we obtain \( \bar{N}_{\text{max}} \) in \( \text{P1} \) as:

\[
\bar{N}_{\text{max}} = \left\{ \begin{array}{ll}
-\log_2(h_0) & \frac{1}{1-h_0} \log_2(h_0) \quad \text{if } h_0 \leq h_A,
\frac{1}{1-h_0} \log_2(h_0) + h_0 - h_A & \text{if } h_0 > h_A \\
\end{array} \right.
\]

Figure 3 plots \( \bar{N}_{\text{max}} \) and \( \bar{N}_{\text{app}} \) which is defined as:

\[
\bar{N}_{\text{app}} = -\log_2\left(\frac{P_0}{-\log_2[H_b^{-1}(h_0)]}\right),
\]

for \( P_0 = 10^{-4} \). \( \bar{N}_{\text{app}} \) is the same as \( \bar{N}_{\text{max}} \) for \( h_0 \leq h_A \) and is smaller than \( \bar{N}_{\text{max}} \) for \( h_0 > h_A \). Both \( \bar{N}_{\text{app}} \) and \( \bar{N}_{\text{max}} \) increase as \( h_0 \) increases. For \( h_0 \) smaller than 0.1, we know the paths in \( \mathcal{A} \) have low blocking probabilities. Therefore the average number of paths we need to probe is only one or two, that is, \( \bar{N}_{\text{max}} < 2 \). For \( h_0 \) close to 1, we are less certain about the availability of the paths in \( \mathcal{A} \). Thus, we end up with...
probing more of them. The largest difference between \( \bar{N}_{\text{max}} \) and \( \bar{N}_{\text{app}} \) happens at point B in Fig. 3, where \( \bar{N}_{\text{app}} \) is smaller than \( \bar{N}_{\text{max}} \) by:

\[
\frac{\bar{N}_{\text{max}} - \bar{N}_{\text{app}}}{\bar{N}_{\text{max}}} \bigg|_{h=1.45} = 0.145.
\]

This leads to difference of only one or two paths between them for \( P_B = 10^{-4} \), for which case \( \bar{N}_{\text{app}} \) can be taken as a good approximation of \( \bar{N}_{\text{max}} \). In fact, for the entropy technique to be useful, most of the time the network will be operating with entropy less than \( h_A \), where the two expressions are equal.

C. Simulation Results and Theoretical Bounds

Simulation results are presented to evaluate the performance of determining the number of probing paths based on average entropy value. The simulation is based on the model in Fig. 2. The basic idea is to simulate a simple network of one source-destination pair with \( m \) paths in between. A randomly drawn blocking probability with uniform distribution in \([0, 0.5]\) is assigned to each path. As shown in Fig. 4, two forms of \( N \) as functions of average entropy value \( h \) are plotted, \( N_r \) and \( N_o \). To get \( N_r \), a set of paths are randomly selected from the pool of \( m \) available paths until the total blocking probability of the selected paths is smaller than the target blocking probability \( P_B \). \( N_r \) is taken as the average of the numbers of paths of such repeated processes. On the other hand, to get \( N_o \), paths are picked in ascending order of their blocking probabilities, that is, the path with lowest blocking probability is picked first, followed by the one with the second lowest blocking probability, etc., until their total blocking probability is smaller than \( P_B \). Then, in the same manner as for \( N_r \), \( N_o \) is taken as the average number of selected paths over many runs. In particular, \( N_o \) can be considered as the analogue of the case from [5] for heterogeneous traffic arrival and departure processes.

In Fig. 4, \( N_r \) is bounded by \( \bar{N}_{\text{max}} \) for \( h > 0.82 \), and is slightly bigger than \( \bar{N}_{\text{max}} \) for \( h < 0.82 \). However, the latter case can be justified by the approximation we applied in the problem formulation in Section II-B, where \( \bar{N}_{\text{max}} \) is actually confined by (3). Indeed, observing Fig. 4, even when \( N_r \) is larger than \( \bar{N}_{\text{max}} \), \( N_r \) is always smaller than \( \bar{N}_{\text{max}} + 1 \). In addition, \( \bar{N}_{\text{app}} \) is no smaller than \( N_r \) by one for all \( h \) values, which suggests it is a good approximation to \( N_r \) as well. On the other hand, \( N_o \) is smaller than \( N_r \) for all \( h \in (0, 1) \): rounding up to integer values, \( N_o \) is smaller than \( N_r \) by one for \( h \in (0, 0.3) \), and is only half of \( N_r \) for \( h \in (0.3, 0.77) \). Nevertheless, this is justifiable as we have to sacrifice some performance in order to avoid detailed assumptions of network statistics and to reduce the amount of network control and management messaging.

III. ENTROPY-ASSISTED FAST SCHEDULING ALGORITHM FOR A GENERAL NETWORK

Section II studied the entropy-assisted probing algorithm for a simple network. In this section, we first extend the probing algorithm to a network with two-link paths and finally to a general mesh network.

A. Information Theoretical Analysis

Consider a path with two links \( L_1 \) and \( L_2 \) as shown in Fig. 5. Their blocking probabilities themselves can be considered as random variables \( X_1 \) and \( X_2 \). For \( L_1 \), it has two states, either 0 or 1, where 0 means that the link is available and 1 means the link is occupied. As we know that \( P(r_l = 1) = X_1 \), the entropy of \( L_1 \) can be easily calculated as \( H(L_1) = H_b(X_1) \). Similarly, entropy of \( L_2 \) is \( H(L_2) = H_b(X_2) \).

From information theory [6], the joint entropy of \( L_1 \) and \( L_2 \) is

\[
H(L_1, L_2) = H(L_1) + H(L_2) - I(L_1; L_2). \quad (12)
\]

\( I(L_1; L_2) \) is the mutual information between \( L_1 \) and \( L_2 \). Given the joint probability mass function (PMF) of \( L_1 \) and \( L_2 \) \( (P_{L_1, L_2}) \) and their marginal PMFs \( (P_{L_1}, P_{L_2}) \), \( I(L_1; L_2) \) can be written as:

\[
I(L_1; L_2) = \sum_{L_1, L_2} P_{L_1, L_2}(l_1, l_2) \log_2 \left( \frac{P_{L_1, L_2}(l_1, l_2)}{P_{L_1}(l_1)P_{L_2}(l_2)} \right) \quad (13)
\]
I(L_1; L_2) can be interpreted as the correlation between L_1 and L_2. It equals to zero if L_1 and L_2 are independent, and increases with the increase of the correlation between L_1 and L_2. When the state of L_2 can be fully determined by that of L_1, or vice versa, I(L_1; L_2) is at its maximum, and I(L_1; L_2) = H(L_1) = H(L_2). For a path in a general network, if L_1 is serving one transaction, there is a high chance that L_2 is also taken by the same transaction. Therefore, I(L_1; L_2) > 0 for most cases. Indeed, in the long haul network if there are no merging and departing traffic in the mid-span, we can assume I(L_1; L_2) = H(L_1) = H(L_2).

Define a new random variable M_2 to represent the state of the whole path. M_2 is 0 if the whole path is available, and 1 if occupied. Then we have the following theorem:

Theorem 1: H(M_2) ≤ H(L_1) + H(L_2) − I(L_1; L_2)

This theorem can be easily extended to a path with three or more links. Similarly, we define a random variable M_n to represent the state of the whole path of n links. Then we have:

Theorem 2: H(M_n) ≤ ∑_{i=1}^{n} H(L_i) − ∑_{i=1}^{n-1} I(L_i; L_{i+1})

B. Extension to a network with two-link paths

Consider the network in Fig. 5 where L_1 and L_2 each represents a group of links, if we know E[H(L_1)] = h_1, E[H(L_2)] = h_2, and E[I(L_1; L_2)] = i, we have E[H(M_2)] ≤ E[H(L_1) + H(L_2) − I(L_1; L_2)]. Substituting h_0 = h_1 + h_2 − i into (10), we can find \( \bar{N}_{max} \) for this network. In particular, a larger mutual information corresponds to a smaller upper bound of E[H(M_2)], thus a tighter \( \bar{N}_{max} \).

Simulations were carried out to test the probing method for paths with two hops. Correlation between L_1 and L_2 was introduced by defining the following conditional probability:

\[
P_{X_2|X_1}(x_2|x_1) = \begin{cases} 
\beta & \text{if } x_2 = x_1 \\
1 - \beta & \text{if } x_2 \neq x_1
\end{cases}
\]

As shown in Fig. 6, three cases with different \( \beta \) values were tested to achieve the same total blocking probability \( P_B = 10^{-5} \). \( \bar{N}_{max} \), defined in (10), is the upper bound of the expected number of paths to probe using \( h_1 + h_2 - i \) as the upper bound of the total entropy. \( N_1 \) is the simulated average number of paths to probe for \( \beta = 0.9 \), \( N_2 \) is for \( \beta = 0.99 \), and \( N_3 \) is for \( \beta = 0.999 \).

As expected, \( \bar{N}_{max} \) is a tight bound of \( N_1 \) and \( N_2 \) for which \( \beta \) equals to 0.999 and 0.99, respectively. For the cases of \( \beta = 0.9 \), as we lose track of the state of the network when \( H(M_2) \) exceeds one, we should operate at the region where \( H(M_2) \) is small. For example, even if \( H(L_1) = 0 \) for \( \beta = 0.9 \), the bound of \( H(M_2) \) is \( H(L_2|L_1) = H_0(0.9) = 0.469 \). Therefore, the line of \( N_1 \) starts from \( H(M_2) = 0.469 \).

C. Entropy-Assisted Probing in a General Network

The above described algorithm of determining the number of paths to probe can be easily extended to paths with three or more links. However, in a mesh network, even with the average entropy of each link and the mutual information between adjacent links, we still need to first figure out which route is most likely to be available. This can be done through a modified Bellman-Ford algorithm.

Take the network shown in Fig. 7 for example, consider node 1 to be the destination. The length of each arc \( d_{ij} \) is taken as the expected average entropy of the corresponding link \( L_{ij} \), that is, \( d_{ij} = E[H(L_{ij})] \). \( d_{ij} = \infty \) if there is no arc between node i and node j. \( D_h^i \) is the length of the shortest walk from node i to node 1 within h steps. The mutual information between link \( L_{ij} \) and link \( L_{jk} \) is represented as \( I_{ijk} = I(L_{ij}; L_{jk}) \). In Algorithm 1, the Bellman-Ford algorithm is slightly modified to incorporate the mutual information into the total length of a route.

**Algorithm 1 Modified Bellman-Ford Algorithm**

1. **Initialize**
   - \( D_h^i = 0 \) for all \( h \)
   - \( D_1^i = d_{ii} \) for all \( i \neq 1 \)

2. **repeat**
   - \( D_{h+1}^i = \min_j[d_{ij} - E[I_{ijk}] + D_h^j] \) for all \( i \neq 1 \) and \( h > 0 \), where \( k \) is the node to which node \( j \) is linked to in \( D_h^j \).

3. **until** \( D_h^i = D_h^{i-1} \) for all \( i \).

After running the algorithm to find the shortest route from source to destination, we can take the “length” (as defined in the modified Bellman-Ford algorithm) of the shortest route between them as the approximation of the upper bound of the average entropy. Then we can determine how many paths we need to probe along the shortest route. This method can be generalized for networks with multiple domains each with different traffic statistics and thus entropies.
IV. Estimation of Network Information

In the previous sections we have discussed the fast scheduling algorithm for OFS with the assistance of the entropies of network domains. Here we discuss the methods to collect the required information $E[H(t)]$ and $E[I(t)]$ for a network in operation.

As both $H(t)$ and $I(t)$ start from zero at $t = 0$, we work on a set of paths that are all available at $t = 0$. We continue working on the same set to measure how the number of blocked paths increases with time, and thus we can find $H(t)$ and $I(t)$.

We assume there are many links between two neighboring nodes. The blocking probability for each link is an identically and independently distributed random variable $X$, as shown in Fig. 2. To gather the necessary statistics for $H(t)$, we sample the network periodically at time interval $\tau$. Suppose at time $t = 0$ we sample each path of the network, and from the sample results we can divide those paths into two sets, $A$ and $B$. $A$ is the set of available paths. $B$ is the set of blocked paths. At the $i$th sampling epoch from $t = 0$, the number of occupied paths are noted as $N_i(i\tau)$. Then the blocking probability $X(i\tau)$ can be estimated as $\hat{X}(i\tau) = N_i(i\tau)/N(i\tau)$. Hence, entropy at that epoch can be obtained as $H(i\tau) = H_0(\hat{X}(i\tau))$. The $H(t)$ obtained in this way is noisy as we only sample paths from one set $A$. The fluctuations can be averaged out by taking a running time average of $\hat{H}(t)$ over a period less than the coherence time of the traffic statistics. Basically, every $\delta$ time, we start with a new set $A$ with available paths, and keep sampling it to get $H(t)$. Using $H_j(t)$ for the $j$th $H(t)$ obtained starting from $t = j\delta$, $H(t)$ can be obtained by averaging the past $k$ $H(t)$’s:

$$\bar{H}(t) = \frac{\sum_{j=0}^{k-1} H_j(t - j\delta)}{k}.$$  

We assume the number of sampled $H(t)$ we take average over, $k$, is large enough so that we can approximate $E[H(t)]$ by $\bar{H}(t)$. We also assume the length of the averaging period $k\delta$ is much smaller (e.g. less than one tenth) than the coherence time of $H(t)$ so that $\bar{H}(t)$ is a good predication for the $H(t)$ in the next broadcast interval.

$\bar{I}(t)$ can be obtained in a similar way. We first get $\bar{I}(t)$ through sampling two neighboring links at their sharing node, and estimating $P_{X_1X_2}$ by its empirical distribution. Then we can get $\bar{I}(t)$ by taking the average of $k$ consecutive $I(t)$’s, that is,

$$\bar{I}(t) = \frac{\sum_{j=0}^{k-1} I_j(t - j\delta)}{k}.$$  

V. CONCLUSIONS

In this paper, we designed a new entropy-assisted fast scheduling algorithm for OFS which is not dependent on any assumptions of the network statistical models, and greatly reduces network control and management messaging. We studied how to determine the average number of paths to probe based on the updates of information about network state for different network domains and regions in the form of entropy. We also showed that this can be easily extended to a general network by introducing another measure, the mutual information, to describe the correlation of adjacent links. Finally, we discussed how the required information can be gathered from actual networks.

APPENDIX

Theorem 3: Let $X_n$ be a continuous random variable with density function $f_{X_n}(x) \in C$. Let $X^*_n$ be the discrete random variable that is the best discrete solution in $P_3$. Then, $E[-\log_2(X_n)] \geq E[-\log_2(X^*_n)]$.

Proof: Assume $E[-\log_2(X_n)] < E[-\log_2(X^*_n)]$ and $\epsilon = E[-\log_2(X^*_n)] - E[-\log_2(X_n)]$. Let $Y_n$ be a sequence of discrete random variables defined by

$$Y_n = \left\lfloor \frac{nX_n}{n} \right\rfloor,$$

for $n = \{1, 2, ..., \}$. Then, we have $Y_n \to X_n$ almost surely, and $Y_n \leq Y_{n+1}$ for any $n \geq 1$. Since $\log_2(x)$ and $H_0(x)$ are both monotonic functions of $x$ for $x \in (0, 1/2)$, by monotone convergence theorem, we have $E[-\log_2(Y_n)] \to E[-\log_2(X_n)]$ and $E[H_0(Y_n)] \to E[H_0(X_n)] = h_0$.

Now, define another sequence of discrete random variables $\hat{Y}_n^\alpha = Y_n + \left(\frac{1}{n} - Y_n\right)\alpha$. Let $g_n(\alpha) = E[H_0(\hat{Y}_n^\alpha)]$. Then $g_n(\alpha)$ is a continuous and monotonically increasing function of $\alpha$, with $g_n(0) = E[H_0(Y_n)]$ and $g_n(1) = E[H_0(1/n)] = 1$. Because $Y_n \leq X_n$, $g_n(0) = E[H_0(Y_n)] \leq h_0$. Therefore, there exists $\alpha_n^* \in [0, 1]$, such that $g_n(\alpha_n^*) = h_0$. However, as $g_n(0) = E[H_0(Y_n)] \to E[H_0(X_n)] = h_0$, we have $\alpha_n^* \to 0$.

Therefore, we have $E[-\log_2(\hat{Y}_n^{\alpha_n^*})] = E[-\log_2(Y_n)]$$,$ which implies $E[-\log_2(\hat{Y}_n^{\alpha_n^*})] \to E[-\log_2(X_n)]$$.$

Now we have constructed a sequence of discrete random variables $\hat{Y}_n^{\alpha_n^*}$ that converges to $X_n$. Therefore, we can find a $N$ such that $E[-\log_2(\hat{Y}_N^{\alpha_N^*})] < E[-\log_2(X_n)] + \epsilon = E[-\log_2(X^*_n)]$. But $Y_N^{\alpha_N^*}$ is a discrete random variable, so we must have $E[-\log_2(Y_N^{\alpha_N^*})] \geq E[-\log_2(X^*_n)]$, leading to contradiction with the assumption. Q.E.D.

REFERENCES

[1] V. Chan, G. Weichenberg, and M. Medard, “Optical flow switching,” in Broadband Communications, Networks and Systems, 2006. BROAD- NETS 2006. 3rd International Conference on, pp. 1–8, Oct. 2006.

[2] G. Weichenberg, V. Chan, and M. Medard, “Design and analysis of optical flow-switched networks,” in Optical Communications and Networking, IEEE/OSA Journal of, vol. 1, no. 3, pp. B81–B97, Aug 2009.

[3] G. Weichenberg, V. Chan, E. Swanson, and M. Medard, “Throughput-cost analysis of optical flow switching,” in Optical Fiber Communication, 2009. OFC 2009. Conference on, Mar 2009, pp. 1–3.

[4] G. Weichenberg, V. Chan, and M. Medard, “Performance analysis of optical flow switching,” in Communications, 2009. ICC 2009. IEEE International Conference on, Jun 2009, pp. 1–6.

[5] V. Chan, A. Ganguly, and G. Weichenberg, “Optical flow switching with time deadlines for high-performance applications,” in Globecom 2009, Dec 2009.

[6] T. M. Cover and J. A. Thomas. Elements of Information Theory. Wiley-Interscience, Hoboken, N.J., 2006.