PLATO: a generic modeling technique for optical packet-switched networks

Shun Yao, Biswanath Mukherjee∗, Sudhir Dixit†

Abstract

This paper presents PLATO, an analytical technique to model the performance in all-optical packet-switched networks with various topologies and node architectures. Using this technique, we propose a two-part algorithm, incorporating mathematical analysis and a numeric method, to model the network’s traffic flow and the local packet-loss probability inside each node. The PLATO technique has demonstrated excellent accuracy with two different representative network topologies and node architectures. It has considerably reduced the time required to evaluate the performance of all-optical packet-switched networks.

1 Introduction

The current ongoing efforts to automate and expedite wavelength and bandwidth provisioning in the optical layer indicate the inevitable trends leading to more intelligent optical networks. With the rapid growth of data traffic, a packet-switched network is a desirable candidate for service providers to meet their versatile traffic demands. Migrating the switching functionality from electronics to optics can resolve the electrical-optical-electrical (OEO) conversion bottleneck in the networks. All-optical packet switching promises to offer a number of layer-two and layer-three functions via packet-level switching in the optical layer. It has a finer bandwidth-allocation granularity, compared with circuit-switched networks, to close the gap between the electrical (IP/MPLS) layer and the optical (WDM) backbone, and provides transparency to data rate and data format [1, 2].

One of the objectives in designing an optical packet-switched network is low packet-loss rate (PLR, sometimes also denoted as probability of packet loss, PPL). Packets may be dropped in contentions, when there are more packets traveling to the output port of a switch than the port can actually accommodate. Various node architectures have been proposed in the past [3, 4, 5, 6, 7], as well as contention-resolution algorithms in time, space, and wavelength domains [8, 9, 10, 11]. Although some analytical studies have been presented for different node architectures, simulation is still the dominant tool for studying the network-wide performance of large, practical networks.

Today’s optical links have a line speed as high as 10 Gbps (OC-192) and many optical packet formats have sizes based on IP packets, which is no more than tens of thousand of bits (with optical burst switching [12] as an exception). Therefore, to simulate the performance of a network, millions to billions of packets need to be generated. The burden of computation becomes much heavier when the network size increases, with multiple wavelengths, and when the PLR is low. In many cases, such large simulations become impractical.

∗S. Yao, and B. Mukherjee are with University of California, Davis, CA 95616. Corresponding author: Shun Yao, Department of Electrical and Computer Engineering, phone: +1-530-752-5129, email: shyao@ece.ucdavis.edu.
†S. Dixit is with Nokia Research Center, 5 Wayside Rd, Burlington, MA 01803.
This paper proposes and develops PLATO (Packet-Loss Analysis with Traffic flOw), a generic analytical modeling technique. It is suitable for a large variety of network topologies and node architectures. This technique combines a numerical method with traffic and probability analysis. The comparison between simulation and analytical results demonstrates that PLATO is both computationally tractable and accurate.

The analysis of optical packet-switching architectures has received some attention in the literature. However, most past work has concentrated on modeling one single switch, or a network with one specific node architecture [13, 14, 10, 15]. A generic approach, independent of topology and node architecture, has been an unchartered problem until now. Our proposed technique is an attempt to contribute to this subject. It is aimed at providing the network designer with an accurate and fast method for evaluating network PLR without running computation-intensive simulations.

After presenting the basic building blocks of the technique (Sections 2 and 3), the paper shows its application to two example networks, with comparison results (Section 4). Section 5 concludes the paper.

2 Basic Algorithms of PLATO

Figure 1 shows the basic approach of PLATO. The model consists of two parts: traffic-flow analysis and local node modeling. In traffic-flow analysis, based on the topology, traffic matrix, and routing algorithm, the traffic intensity and its composition at each input port of every switch is computed. This information is then fed into the PLR model for each node along the path of every source-destination pair, to calculate the corresponding PLR. Because the final model is a multiple-variable system, an iterative approach is used to obtain the actual value of the PLR.

Figure 2 shows a simple network topology. Suppose a packet is going from node 1 to node 4, through node 2 and node 3. This packet could be dropped in any of these nodes, due to contention or lack of sufficient amount of buffering. Here we define $plr_{ij}$ as the end-to-end packet-loss probability of any packet from node $i$ to node $j$, and $p_{ijk}$ as the packet-loss probability at intermediate node $k$ for packets belonging to the $i \rightarrow j$ source-destination pair. $t_{ij}$ is defined as the offered traffic intensity (offered load) between source node $i$ and destination node $j$, and PLR is defined as the network-wide packet-loss probability. The packet-loss probability equals the ratio of the number of lost packets to the total number of offered packets; therefore, “packet-loss rate” and “packet-loss probability” are used interchangeably throughout this paper. The end-to-end packet-loss probability can be obtained by:

$$plr_{ij} = 1 - \prod_{k \in i \rightarrow j} (1 - p_{ijk})$$ (1)

where $k$ denotes the index of nodes along the path from $i$ to $j$ (including $i$ and $j$.)
To compute the network packet-loss probability, all the traffic load from every possible source-destination pair should be taken into consideration. We have

$$PLR = \frac{\sum_{i,j,i \neq j} (t_{ij} \cdot plr_{ij})}{\sum_{i,j,i \neq j} t_{ij}}$$  \hspace{1cm} (2)
To compute $PLR$, we need to determine $plr_{ij}$, and therefore $p_{ijk}$ for any $i \rightarrow j$ pair. This requires a careful inspection of the route that every $t_{ij}$ takes and computation of the packet-loss probability at every intermediate node. Consider $t_{14}$, for example, in Fig. 2. According to the routing information in Table 1, the nodes that a $t_{14}$ packet traverses include nodes 1, 2, 3, and 4. At each node, the packet might be dropped, depending on the contentions it encounters. Figure 3 illustrates the possible contentions at each switch, where the gray arrows represent the contending packet streams. The input/output ports are indexed by the node number they are leading to. In order to compute the packet-loss probability at each switch (in this case, $p_{141}, p_{142}, p_{143}, p_{144}$), we need to find out (1) the contention traffic flows’ intensity and composition, and (2) the PLR model of each switch as a function of input traffic flows. The detailed algorithms are described in the following subsections.

![Figure 3: Possible contentions a $t_{14}$ packet can encounter.](image_url)

### 2.1 Traffic-Flow Analysis

A switch, regardless of its internal architecture, is a system with fixed number of inputs and outputs. The packet loss probability of a packet stream from a given input port is a function of the input traffic flows only. The PLR of a packet can be determined as soon as all the contending traffic flows’ intensity is known, provided there is an accurate model for the specific switch architecture.

Let us take $p_{142}$ for example, as shown in Fig. 4. We follow the $t_{14}$ packets through node 2. The packets are arriving on port 1 and leaving from port 3 at node 2. From the routing table in Table 1, there are three streams contending for output port 3. Each stream is composed of traffic from different source-destination pairs. The top stream consists of packets from $t_{13}$ and $t_{14}$, the middle one from $t_{63}$, and the bottom one from $t_{23}, t_{24},$ and $t_{25}$. Since, before these tributary flows reach node 2, they have already suffered from possible packet loss from the previous nodes along their respective paths, we need to take into account the individual packet-loss rate they have suffered. Therefore, the load of the three streams contending for output port 3 at the input ports of node 2 are:

\[
\begin{align*}
t_{13} \cdot (1 - p_{131}) + t_{14} \cdot (1 - p_{141}) \\
t_{63} \cdot (1 - p_{636}) \\
t_{23} + t_{24} + t_{25}
\end{align*}
\]

According to the switch model specific to the node architecture, with the above values of traffic intensity, we are able to compute the packet-loss probability of $p_{142}$. In this case, $p_{132} = p_{142}$ because $t_{13}$ and $t_{14}$ belong to the same tributary stream contending for output port 3.

It is likely that an optical link has multiple wavelengths, and a node has more than one add/drop port. Nevertheless, the contending traffic flows at a certain node are determined by the given topology, traffic matrix $t_{ij}$, and static routing.
We have demonstrated how to analyze the contending traffic for computing $p_{142}$. Our goal is to compute $\text{plr}_{ij}$, which is given by Eqn. (1). Therefore, we have to perform the same analysis for each $i \rightarrow j$ pair at every switch on its path. This can be performed easily with modern computing facilities. The pseudo-code for the traffic-flow analysis algorithm will follow the completion of local switch models.

Once the traffic analysis is complete, it can be combined with the local model of each switch to render a networked model to compute the global PLR.

### 2.2 Local Switch Modeling

There have been several proposals for optical packet-switch architectures in the literature. Many of them are provided with analytical models. One main characteristic of an optical packet switch is that it does not have optical random-access memory (RAM), unlike the electronic peers which can take advantage of inexpensive RAM for packet buffering. This has resulted in various alternative approaches to resolve contentions, namely buffering (based on fiber delay lines), wavelength conversion, deflection routing, and over-provisioning (offer more capacity to the network).

Since this work does not aim to discuss in depth the modeling of specific switch architectures, but rather a generic approach for network modeling, we will first develop a simple, but representative, example switch model for verification of the PLATO technique. The example switch we adopt is a baseline switch architecture, which is simply an $N \times N$ space switch with no buffers and operates on a single wavelength, as shown in Fig. 5. For the moment, we consider the case when there are three packet streams contending for one output port. Whenever there is a collision, the later-arriving packets will be dropped. For this example, we assume packets are of fixed size and arrive according to Poisson process. Our results will show that the model is also suitable for variable packet sizes with negative exponential
distribution. The traffic load of the streams are $l_1$, $l_2$, and $l_3$, while the packet-loss probabilities are $p_1$, $p_2$, and $p_3$, respectively. Let us first study the packets arriving from stream 1. When a packet arrives, it will not suffer a collision if no other packets are sent within one packet time of its start, as shown in Fig. 6. The colliding packets can only be from stream 2 or stream 3, since stream 1 packets will never contend among themselves. Let $t$ be the time required to send a packet. If stream 2 and/or stream 3 has generated a packet between time $t_0$ and $t_0 + t$ and it has successfully occupied the output port, the end of that packet will collide with the beginning of the shaded one from stream 1. The probability that $k$ packets are generated during one packet time is given by the Poisson distribution:

$$Pr[k] = \frac{G^k e^{-G}}{k!},$$

where $G$ is the average load on the output port, contributed by stream 2 and stream 3:

$$G = l_2(1 - p_2) + l_3(1 - p_3).$$

Therefore, the packet-loss probability for stream 1 is:

$$p_1 = 1 - Pr[0] = 1 - e^{-(l_2(1-p_2)+l_3(1-p_3))} \quad (3)$$

Similarly, we have:

$$p_2 = 1 - e^{-(l_1(1-p_1)+l_3(1-p_3))} \quad (4)$$

$$p_3 = 1 - e^{-(l_1(1-p_1)+l_2(1-p_2))} \quad (5)$$

Equations (3)-(5) form a system of multiple-variable transcendental equations. It is difficult to obtain the analytical solutions for these equations. PLATO adopts a modified iteration algorithm to obtain the numerical solutions. Iteration is useful for solving single-variable nonlinear equations. First, give a set of initial values (for example, zero) to $p_1$, $p_2$, $p_3$. Then plug them into the right-hand side of Eqns. (3)-(5), which will give a new set of values for $p_1$, $p_2$, $p_3$. Repeat the substitution process until the values of $p_1$, $p_2$, $p_3$ converge. (The proof of the convergence of this algorithm is provided in the Appendix.) When this system of equations is applied in the network model, we write a function of $(l_1, l_2, l_3)$, which returns the value of $p_1$. The pseudo-code is as follows:

```plaintext
plr_3stream(l1, l2, l3)
1 p1_new, p2_new, p3_new ← 0
2 p1_old ← 1
3 while Distance(p1_new, p1_old) > e
4   p1_old ← p1_new
5   p2_old ← p2_new
6   p3_old ← p3_new
7   p1_new ← 1-exp(-(l2(1-p2_old)+l3(1-p3_old)))
8   p2_new ← 1-exp(-(l1(1-p1_old)+l3(1-p3_old)))
```

Figure 6: Vulnerable period of the shaded packet.
9 \[ p3_{\text{new}} \leftarrow 1 - \exp(-(l1(1-p1_{\text{old}})+l2(1-p2_{\text{old}}))) \]
10 \text{return } (p1_{\text{new}})

Similar functions can be written (e.g., \( \text{plr}_2\text{stream}() \), \( \text{plr}_4\text{stream}() \) ...) for any number of streams contending for one output port.

### 2.3 Constructing the Network Model

Now we have the complete information on traffic intensity and its composition at every input port of every switch, as well as the model to compute packet-loss probability at any intermediate node of any source-destination pair. We are ready to compute \( \text{plr}_{ij} \) and \( \text{PLR} \).

Let us look back at the example of \( \text{plr}_{14} \). From Eqn. (1):

\[
\text{plr}_{14} = 1 - (1 - p_{141})(1 - p_{142})(1 - p_{143})(1 - p_{144})
\]

where

\[
\begin{align*}
p_{141} &= 0 \\
p_{142} &= \text{plr}_3\text{stream} \left( t_{13}(1 - p_{131}) + t_{14}(1 - p_{141}), \\
&\quad t_{63}(1 - p_{636}), \; t_{23} + t_{24} + t_{25} \right) \\
p_{143} &= \text{plr}_2\text{stream} \left( t_{14}(1 - p_{141})(1 - p_{142}) + \\
&\quad t_{24}(1 - p_{242}), \; t_{34} \right) \\
p_{144} &= \text{plr}_2\text{stream} \left( t_{14}(1 - p_{141})(1 - p_{142}) \cdot \\
&\quad (1 - p_{143}) + t_{24}(1 - p_{242})(1 - p_{243}) \\
&\quad + t_{34}(1 - p_{343}), \; t_{54}(1 - p_{545}) + t_{64}(1 - p_{645}) \right)
\end{align*}
\]

With flow analysis as described earlier, it is possible to construct similar equations for every \( p_{ijk} \), \( \forall i,j,k; \; i \neq j; \; k \in i \rightarrow j \). This leads to a system of nonlinear equations with all \( p_{ijk} \) as the variables. To solve them, we apply the numerical iteration method again, as previously described in the pseudo-code. This algorithm will not only give the network PLR, but also give the packet-loss rate at every switch of any source-destination pair, \( \text{plr}_{ijk} \). Thus, it enables us to investigate the “hot spots” in the network.

Here is the pseudo-code of the algorithm:

\[
\text{Main}(T, e) \]
\[
1 \; \text{Init}(q) \\
2 \]
\[
3 \; \text{PLR}_\text{old} \leftarrow 0 \\
4 \; \text{PLR}_\text{new} \leftarrow 1 \\
5 \\
6 \; \text{while Distance}(\text{PLR}_\text{new}, \text{PLR}_\text{old})>e \\
7 \; \quad \text{do PLR}_\text{old} \leftarrow \text{PLR}_\text{new} \\
8 \; \quad \text{PLR}_\text{new} \leftarrow \text{Step}(T, q) \\
9 \\
10 \; \text{return } \text{PLR}_\text{new}
\]

where \( T \) is the network topology and \( e \) is the numerical iteration error limit. \( \text{Init}(q) \) initializes every
The element of array $q$ as 0. Distance($a$, $b$) returns $|a - b|/a$ and Step($T$, $q$) returns the network PLR.

Step($T$, $q$)
1 for $i ← 1$ to Node_Number($T$)
2 do for $j ← 1$ to Node_Number($T$)
3 do $tmp_p ← 1$
4 for each vertex $k$ in Route[$i$][$j$]
5 do Compute($p[i][j][k]$, $q$)
6 $tmp_p ← tmp_p * (1 - p[i][j][k])$
7 $plr[i][j] ← 1 - tmp_p$
8 $\sum_{i,j,i\neq j} p_{r_{ij}} \cdot t_{ij}$
9 $PLR_{new} ← \frac{\sum_{i,j,i\neq j} p_{r_{ij}} \cdot t_{ij}}{\sum_{i,j,i\neq j} t_{ij}}$
10 $\sum_{i,j,i\neq j} t_{ij}$
11 for $i ← 1$ to Node_Number($T$)
12 do for $j ← 1$ to Node_Number($T$)
13 do for $k ← 1$ to Node_Number($T$)
14 do $q[i][j][k] ← p[i][j][k]$
15
16 return $PLR_{new}$

Compute($p[i][j][k]$, $q$)
1 $b ←$ the input port index when packet flow from $i$ to $j$ goes through node $k$
2 find all the other packet flows and their loads, which go through node $k$ using output port $b$
3 merge the flows from the same input port to output port $b$ in node $k$
4 compute $p[i][j][k]$ according to the number of conflicting flows by fitting them into suitable local switch models

where Node_Number($T$) returns the total number of nodes in network $T$, and Route[$i$][$j$] contains the route from node $i$ to node $j$.

3 Delay-Line Model

To verify the accuracy of the PLATO technique with a different node architecture, this study also presents a sample switch model that implements fiber delay lines as buffers for the packets. Figure 7 shows the switch architecture. There is one fiber delay line dedicated to each output port, including the local drop port. Although this is not a very economical way of utilizing buffers, it simplifies the modeling process and can still capture sufficiently well the effect of buffering. To further reduce the model complexity, we impose that each packet is allowed to enter the delay line only once during contention. Packets could be dropped due to a busy delay line or a busy output port when it emerges from the delay line. Each delay line is 5-km long in the numerical example here to ensure that (1) it can accommodate a full packet and (2) when the packet emerges from the end of the delay line, the previous packet occupying the preferred output port has finished transmission. It is necessary to develop new
functions as local switch models, namely \texttt{bufplr\_nstream}(l_1, l_2, \ldots, l_n), where \( n \) is the number of contending streams, to take into account the effect of buffering.

Figure 8 shows an example of 3 packet streams contending for one output port. Let us follow the packets arriving from stream 1 and investigate the packet-loss probability they suffer. Packets from stream 1 are contending with packets from stream 2, stream 3, and the delay line. We denote \( l_k \) as the load on stream \( k \), \( b_k \) as the load on the delay line, \( b_{k}\) as the probability that a packet from stream \( k \) needs to be buffered due to contention on the output port, and \( bb_{k} \) as the probability that a packet from stream \( k \) will find the delay line occupied. We will also use the functions previously written for the baseline switch, \texttt{plr\_nstream}(l_1, l_2, \ldots, l_n), to compute \( b_1 \). There are two scenarios where the packet from stream 1 is dropped: (1) it finds the output port and the delay line busy at the same time, and (2) it is successfully buffered but finds the output port busy after emerging from the delay line. Therefore, the packet-loss probability for stream 1 packets, \( p_1 \), can be written as:

\[
p_1 = b_1 \cdot (bb_1 + (1 - bb_1) \cdot \text{plr\_4stream}(l_b, l_1, l_2, l_3))
\]  

(6)

where

\[
bb_1 = \text{plr\_3stream}(l_1b_1, l_2b_2, l_3b_3)
\]  

(7)

\[
bb_2 = \text{plr\_3stream}(l_2b_2, l_1b_1, l_3b_3)
\]  

(8)

\[
bb_3 = \text{plr\_3stream}(l_3b_3, l_1b_1, l_2b_2)
\]  

(9)

\[
b_1 = \text{plr\_4stream}(l_1, l_2, l_3, l_b)
\]  

(10)

\[
b_2 = \text{plr\_4stream}(l_2, l_1, l_3, l_b)
\]  

(11)

\[
b_3 = \text{plr\_4stream}(l_3, l_1, l_2, l_b)
\]  

(12)

\[
l_b = l_1b_1(1 - bb_1) + l_2b_2(1 - bb_2) + l_3b_3(1 - bb_3)
\]  

(13)
From Eqns. (6)-(13), together with the functions written for the baseline switch architecture and the iteration method, we can construct function `bufplr_3stream(l_1, l_2, l_3)` to compute the packet-loss probability $p_1$ with one dedicated delay line. Similarly, `bufplr_nstream(l_1, l_2, ..., l_n)` can be written and plugged into the traffic-flow model, and the network PLR model can be constructed.

The pseudo-code is as follows:

```plaintext
def bufplr_3stream(l_1, l_2, l_3):
    lb = lb_3stream(l_1, l_2, l_3)
    b1 = plr_4stream(l_1, l_2, l_3, lb)
    b2 = plr_4stream(l_2, l_1, l_3, lb)
    b3 = plr_4stream(l_3, l_1, l_2, lb)
    bb1 = plr_3stream(l_1*b1, l_2*b2, l_3*b3)
    p1 = b1*(bb1+(1-bb1)*plr_4stream(lb, l_1, l_2, l_3))
    return p1
```

where `lb_3stream(l_1, l_2, l_3)` computes $l_b$:

```plaintext
def lb_3stream(l_1, l_2, l_3):
    lb_new = 0.5
    lb_old = 0
    while Distance(lb_new, lb_old) > e:
        lb_old = lb_new
        b1 = plr_4stream(l_1, l_2, l_3, lb_old)
        b2 = plr_4stream(l_2, l_1, l_3, lb_old)
        b3 = plr_4stream(l_3, l_1, l_2, lb_old)
        bb1 = plr_3stream(l_1*b1, l_2*b2, l_3*b3)
        bb2 = plr_3stream(l_2*b2, l_1*b1, l_3*b3)
        bb3 = plr_3stream(l_3*b3, l_1*b1, l_2*b2)
        lb_new = l_1*b1*(1-bb1)+l_2*b2*(1-bb2)+l_3*b3*(1-bb3)
    return lb_new
```

4 Numerical Results and Discussion

Using the above algorithms, we are now able to model the network's performance (packet-loss rate) with a given topology and a static routing algorithm. We apply the algorithms to two topologies: the six-node topology in Fig. 2 and the NSF network topology in Fig. 9. Shortest-path routing is used in both topologies. Our first example switch architecture is the baseline switch with no contention resolutions, as shown in Fig. 5. Each switch has one local add port and one local drop port. Offered traffic is injected to the network from the transmitter at the local add port. Each transmitter is operating at OC-48 line speed. Packets are of size 12000 bits.

Figure 10 shows a comparison between simulation and analytical results. PLR is plotted against average offered transmitter load, which is the average fraction of busy time over total simulated time for the transmitter at the local add port of every switch. The traffic injected from a transmitter is uniformly distributed among all the rest of the nodes as destinations. The model provides excellent matching with the simulation results. Figure 11 gives a zoom-in view of the light-load region, since the
Figure 9: The NSF network topology with link cost.

Figure 10: PLR with baseline switch architecture for the 6-node topology and the NSF topology.
Figure 11: A zoom-in view of the light-load region of Fig. 10.

Figure 12: PLR with buffering switch architecture for the NSF topology.
low PLR region is more important. Both figures show that the PLATO technique offers accurate results of PLR with both light and medium load, with different topologies.

Figure 12 shows the PLR results with buffering for the NSF topology. The analytical model demonstrates a very close match with the simulation results, indicating that the PLATO technique has great accuracy with different switching architectures.

We also carried out simulations with variable packet size. The packet-size distribution is negative exponential, with mean of 12000 bits. The results show no noticeable difference from the fixed-size case.

5 Conclusion

This paper presented PLATO, an analytical modeling technique for evaluating packet-loss probability for optical packet-switched networks. This technique can accurately model different topologies and node architectures.

The traffic-analysis part provides an algorithm to decompose all the traffic flows and to construct a system of equations that represent the contentions at every output port of every node. The switch-modeling part models each switch in such a way that the packet-loss probability of any input packet stream can be computed as a function of the intensities of all contending streams. These functions representing local packet-loss probability are called by the system of equations from the traffic-flow analysis. Through numerical iterations, the values of the packet-loss rate of all the flows finally converge and give the network-wide packet-loss rate.

Comparison between analytical and simulation results shows excellent accuracy of the PLATO technique. It is also demonstrated that PLATO performs well with different topologies and node architectures; therefore, it is a good candidate as a generic modeling technique for optical packet-switched networks, and it should be customized for each specific application.

The work presented here can be pursued in several directions. On the local-switch modeling side, the study assumes Poisson arrival. In practice, packet arrival is most likely self-similar or long-range dependent. Packet sizes are closely related with IP traffic, in which the packet-size distribution has a hard-to-capture bipolar nature. Moreover, we only considered two simple node architectures, while there exist a large number of complex architectures incorporating wavelength conversion and sophisticated buffering schemes. The simple architectures and contention resolutions are the main contributors to the high packet-loss rate. These simplified assumptions were adopted to facilitate our analytical modeling. We have presented a more extensive simulation-based study in [16], in which both self-similar packet arrival and IP-like packet size distribution were used. It has been shown that, with more sophisticated contention resolutions, a much lower packet-loss rate can be achieved. On the network-traffic-analysis side, the PLATO technique is based on static routing. Dynamic routing, such as deflection routing, has been proposed as one of the important contention-resolution techniques. The traffic-flow-analysis algorithm could be extended to include dynamic routing. We believe that our present work can serve as an excellent foundation for further exciting research along the lines outlined in the previous paragraphs.

Appendix

Iteration is a common numerical method to solve certain nonlinear equations [17]. Let us begin with the single-variable case. Suppose the equation to solve is \( f(x) = 0 \). It can be re-written as \( x = \varphi(x) \). We start with an estimate of the root \( x_0 \), and plug it into the equation to obtain \( x_1 = \varphi(x_0) \). Generally, \( x_1 \neq x_0 \). Then, plug \( x_1 \) back into the equation and obtain \( x_2 = \varphi(x_1) \). By repeating this substitution, we have a series of approximate solutions: \( x_0, x_1, x_2, \ldots, x_n, \ldots \) If we have a pre-defined tolerable error limit \( \varepsilon \), then for certain functions \( \varphi(x) \), this process is limited. The condition to end the process is
\[ |x_{n+1} - x_n| < \varepsilon \]. \(x_{n+1}\) is the approximation of the root with error less than \(\varepsilon\). The sufficient condition for the iteration process to converge is \(|\varphi'(\xi)| < 1, \forall \xi\).

For a system of nonlinear equations of \(k\) variables
\[
\begin{align*}
x_1 &= \varphi_1(x_1, x_2, \ldots, x_n) \\
x_2 &= \varphi_2(x_1, x_2, \ldots, x_n) \\
& \vdots \\
x_k &= \varphi_k(x_1, x_2, \ldots, x_k)
\end{align*}
\]
the initial estimates of the solutions are:
\[ x_{1,0}, x_{2,0}, \ldots, x_{k,0} \]
and the condition of convergence is:
\[
\left| \frac{\partial \varphi_i(x_1, x_2, \ldots, x_k)}{\partial x_j} \right| < 1 \quad i, j = 1, 2, \ldots, k \tag{14}
\]

In the system of equations developed from both traffic-flow analysis and local switch models, the variables are packet-loss probabilities, which are between 0 and 1. All the equations can satisfy Eqn. (14). Take \(\text{plr}_3\text{stream}(l_1, l_2, l_3)\) in Fig. 5, for example. This function performs iteration to solve:
\[
\begin{align*}
p_1 &= 1 - e^{-[l_2(1-p_2)+l_3(1-p_3)]} \\
p_2 &= 1 - e^{-[l_1(1-p_1)+l_3(1-p_3)]} \\
p_3 &= 1 - e^{-[l_1(1-p_1)+l_2(1-p_2)]}
\end{align*}
\]

The partial differentials of \(p_1\) are:
\[
\begin{align*}
\frac{\partial p_1}{\partial p_2} &= -l_2 \cdot e^{-[l_2(1-p_2)+l_3(1-p_3)]} \\
\frac{\partial p_1}{\partial p_3} &= -l_3 \cdot e^{-[l_2(1-p_2)+l_3(1-p_3)]}
\end{align*}
\tag{15}
\]

Since \(l_i, p_i < 1\), we have:
\[
\begin{align*}
\left| \frac{\partial p_1}{\partial p_2} \right| &< 1 \\
\left| \frac{\partial p_1}{\partial p_3} \right| &< 1
\end{align*}
\tag{16}
\]

Therefore, they are all converging.

References


