A New Path Computation Algorithm and its Implementation in NS2

Davide Adami
CNIT Research Unit - Dept. of Information Engineering
University of Pisa
Via Caruso, 16 PISA – ITALY
e-mail: davide.adami@cnit.it

Christian Callegari, Stefano Giordano, Michele Pagano
Dept. of Information Engineering
University of Pisa
Via Caruso, 16 PISA – ITALY
e-mail: {c.callegari, s.giordano, m.pagano}@iet.unipi.it

Abstract— Originally conceived as a fast forwarding technique, MPLS provides support for Traffic Engineering and network survivability. Constrained-based path computation is a key building block for Traffic Engineering in MPLS networks, since it allows to select a path that satisfies assigned QoS requirements. In this paper, we introduce a novel path computation procedure which aims at improving the performance of the well-known Wang-Crowcroft algorithm by means of some heuristics. Moreover, the two algorithms have been developed in NS2 as an extension of OSPF-TE\ns and integrated with RSVP-TE\ns. Finally, the paper shows how the developed software module can be used to satisfy a set of LSP allocation requests with multiple QoS constraints.

Keywords— MPLS, NS2, Path computation, Wang-Crowcroft

I. INTRODUCTION

In recent years, new advanced network architectures have been proposed to satisfy the Quality of Service (QoS) requirements of heterogeneous applications running on the Internet [1]. Meanwhile, Multiprotocol Label Switching (MPLS) [2], which was originally conceived as a fast forwarding technique, has been deployed by a great number of Internet Service Providers (ISP) mainly due to its Traffic Engineering (TE) and Virtual Private Networking capabilities [3][4]. Constrained-based path computation is a fundamental building block for TE in MPLS and GMPLS networks, since it allows to select or determine a path which satisfies a single or multiple QoS requirements. In particular, path computation may be performed either at the time of, or ahead of, service provisioning. The former is called on-line path computation, the latter off-line path computation. The hybrid case is also possible when path computation is performed both off-line (long-term allocations) and on-line (for new incoming requests). Path computation may be performed according to a centralized or distributed architecture [5]. In the first case, all path computations, for a given domain, are performed by a single, centralized PCE (Path Computation Element). This may be a dedicated server (e.g. an external PCE node), or a designated router (e.g. a composite PCE node). On the other hand, a “distributed path computation model” refers to a domain or network that may include multiple PCEs and where the computation of paths is shared among them. A given path may in turn be computed by a single or multiple PCEs. Due to the complexity and great number of available options, powerful and flexible simulation tools are necessary to help network designers and administrators in analyzing the efficiency of different path computation schemes for MPLS networks. Unfortunately, up to now a full open-source simulation tool has not been realized, so, in the last two years, our efforts have been focused on the implementation of new software modules for the Network Simulator (NS2) [6].

In previous works, the authors of this paper developed RSVP-TE\ns [7][8] and OSPF-TE\ns [9] simulators. Within OSPF-TE\ns, CSPF (Constrained Shortest Path First), an algorithm that allows to compute the best path by using cost and bandwidth as QoS metrics, was implemented.

This work is mainly focused on the extensions added to the OSPF-TE\ns simulator to support two path computation algorithms with bandwidth and delay constraints: Wang-Crowcroft [10] and Wang-Crowcroft with Sorting, a new algorithm which aims at improving Wang-Crowcroft performances by means of some heuristics.

The paper is organized as follows. Section II, after a short overview on QoS metrics, discusses the two path computation algorithms implemented in this work. Section III describes the main functionalities added to OSPF-TE\ns. Section IV shows how the new path computation modules can be used to study the allocation of Label Switched Paths (LSPs) with multiple QoS constraints in a complex MPLS network scenario. Finally, section V concludes the paper with some final remarks.

II. PATH COMPUTATION ALGORITHM

The basic function of a PCE is to find a network path that satisfies multiple constraints. Given the QoS metrics, the network status and the QoS requirements of a path computation request, the problem is to determine a path that maximizes the chances of the request to be successful with the minimum negative impact on the network ability to accept future requests.

A. QoS metrics

Routing metrics are links features used to represent a network. Given the metric \( d(i,j) \), associated to the link \((i,j)\), and the path \( p=(i,j,k, \ldots, l,m) \), the metric \( d \) is:
additive if \( d(p)=d(i,j)+d(j,k)+...+d(l,m) \)
multiplicative if \( d(p)=d(i,j)d(j,k)...d(l,m) \)
concave if \( d(p)=\min[d(i,j),d(j,k);...;d(l,m)] \)

Routing protocols usually characterize a network through a single metric (the cost of the link). In the case of QoS routing protocols and constrained-based path computation algorithms, instead, the network is described by means of multiple metrics. The most common ones are the following:

- **cost**: it is an additive metric, because the cost of a path is the sum of the costs of the links.
- **Bandwidth (or residual bandwidth)**: it is a concave metric. Indeed, we define the bandwidth of a path as the minimum of the residual bandwidth of all links on the path (bottleneck bandwidth).
- **delay**: it is an additive metric, since the delay of a path is the sum of the delays of its links. It consists of three components: propagation delay, transmission delay and queuing delay.

While the computation of the first two metrics is quite straightforward, it is not easy to properly evaluate the end-to-end delay, because one of its components, the queuing delay, is not a deterministic quantity. Hence, by using the Network Calculus theory [11], we computed an upper bound of the delay. In particular, since the queuing delay is strongly dependent on the queuing discipline adopted in the network nodes, we supposed that all nodes handle their output queues according to a WFQ (Weighted Fair Queuing) scheduling discipline. The upper bound of the delay \( D_i \), experienced by a packet, which belongs to the generic flow \( I \) and goes through \( K \) nodes, may be computed as:

\[
D_i = \frac{M}{r_i} + \sum_{j=1}^{K} S_j^i 
\]

where

- \( M \) is the maximum burst size
- \( r_i \) is the guaranteed rate for flow \( i \)
- \( S_j^i \) is the delay due to node \( j \), which may be calculated as:

\[
S_j^i = \frac{L_{MAX}}{R_j} + \frac{L_i}{r_i} 
\]

where:

- \( L_{MAX} \) is the maximum packet size
- \( L_i \) is the maximum size of a packet belonging to flow \( i \)
- \( R_j \) is the output link bandwidth.

B. **Wang- Crowcroft Algorithm (WC)**

The Wang-Crowcroft path computation algorithm (WC) aims at finding a path which satisfies multiple QoS constraints, given in terms of bandwidth (\( B_{MIN} \)) and delay (\( D_{MAX} \)).

Every link \((i,j)\) of the network is characterized by two parameters: \( b_{ij} \) (residual bandwidth) and \( d_{ij} \) (propagation delay). Unlike the original version of the algorithm, which takes into account only the propagation delay, our implementation will consider all the components of the delay, as described in section 2.A.

The algorithm consists of the following steps:

1. set \( d_{ij} = \infty \) if \( b_{ij} < B_{MIN} \)
2. compute the path \( P \) with the minimum delay (applying the Dijkstra algorithm)
3. calculate the delay \( D* \) of \( P \)
4. compare the delay with \( D_{MAX} \): If \( D* < D_{MAX} \) select the path, otherwise the request cannot be satisfied.

C. **Wang- Crowcroft with Sorting (WCS)**

When a centralized PCE has to setup several LSPs with bandwidth and delay constraints, the number of requests that can be allocated may depend on the order in which they are processed. To better explain this concept, let us consider the topology shown in figure 1, where (bandwidth \( b_{ij} \) (Mbps) - delay \( d_{ij} \) (ms)) are the metrics associated to each link and the MTU is equal to 1500 bytes.

\[\text{Figure 1. Network Topology}\]

Suppose that two LSPs, with (\( B=2\text{Mbps}, D=100\text{ms})\), (\( B=2\text{Mbps}, D=80\text{ms})\)) constraints should be allocated. Given the network topology and the links metrics in figure 1, it is apparent that after the pruning phase, which excludes the links with a residual bandwidth less than 2 Mbps, two paths may be selected \( P_1 = A\_B\_C\_F \) and \( P_2 = A\_B\_D\_E\_F \). By means of the formula (1), we can easily obtain, for the delays, \( D_1 = 74.53 \) ms and \( D_2 = 94.65 \) ms. Hence, according to the WC algorithm, the first request will be satisfied by allocating an LSP along the path \( P_1 \) (the “best one”), whereas the second one will be rejected. Clearly, by analyzing the requests in reverse order, both of them would have been accepted. Therefore, we introduced a new algorithm, based on a heuristic approach (see figure 2 for details), which tries to improve the performances of the WC algorithm, when all the path computation requests can not be satisfied. The basic idea behind the WCS algorithm is to analyze the path computation requests after re-ordering them based on their bandwidth and delay requirements.
Figure 2. WCS Algorithm

In particular, given a random sequence of \( N \) path computation requests, our algorithm checks whether they can be satisfied as a whole by applying the WC algorithm. If some of them is rejected \((Z < N)\), we compute the number of requests that can be accepted when:

1) they are sorted top-down according to the bandwidth \( B_{MIN} \) requirements;
2) they are sorted bottom-up according to the delay \( D_{MAX} \) requirements.

Finally, the best solution according to a pre-defined cost function (e.g. the maximum number of reservation requests that can be accepted) is chosen. Therefore, the WCS algorithm always performs not worse than the original WC algorithm.

III. IMPLEMENTATIONS IN THE NETWORK SIMULATOR

In this section, we describe the most important functionalities added to the OSPF-TE/ns simulator. The main enhancements concern the following components:

- Path Computation algorithms: in addition to CSPF, two path computation algorithms (WC, WCS) have been developed;
- Extensions for establishing LSPs with QoS constraints;
- MPLS-based recovery of LSPs with QoS constraints.

A. LSP Establishment

The following command allows to set-up a LSP by using the WC path computation algorithm:

\[
<\text{Ingress-LER}> \text{create-crlsp-WC} <\text{Source}> <\text{Egress-LER}> <\text{SessionID}> <\text{FlowID}> <\text{TunnelID}> <\text{Bandwidth}> <\text{MaxDelay}> <\text{Buffer}> <\text{TTL}>
\]

When this command is inserted in the simulation script, the OSPF-TE agent of the ingress node performs the WC algorithm to compute a path which satisfies the constraints \( B_{MIN} = \langle \text{Bandwidth} \rangle \) and \( D_{MAX} = \langle \text{MaxDelay} \rangle \). If a path is found, the hop list is passed to the RSVP-TE agent, which inserts it in the Explicit Route Object (ERO) of the Path message. Then, an LSP, with reserved bandwidth \( \langle \text{Bandwidth} \rangle \) and identified by the tunnel ID \( \langle \text{TunnelID} \rangle \), is created between the \( \langle \text{Ingress-LER} \rangle \) and the \( \langle \text{Egress-LER} \rangle \). After the LSP has been established, the OSPF-TE agent is invoked again and Opaque LSAs (Link State Advertisements) are sent to all the MPLS domain nodes to announce that the OSPF-TE databases have to be updated with the new values of the available link bandwidth. The WCS algorithm, instead, requires that one command for each LSP setup request is inserted in the NS2 script. The syntax of the command is the following:

\[
<\text{Central-LSR}> \text{create-crlsp-WCS} <\text{Source}> <\text{Ingress-LER}> <\text{Egress-LER}> <\text{SessionID}> <\text{FlowID}> <\text{TunnelID}> <\text{Bandwidth}> <\text{MaxDelay}> <\text{Buffer}> <\text{TTL}>
\]

As a consequence, the \( \langle \text{Central-LSR} \rangle \) will execute the WCS algorithm. It is relevant to highlight that when a centralized approach is adopted, the path computation algorithm returns the LSP set-up order. Later, the commands to set-up the LSPs must be scheduled according to such order.

B. LSP Recovery

The following command enables fast-rerouting with the WC algorithm:

\[
<\text{Ingress-LER}> \text{reroute-WC} <\text{Source}> <\text{Egress-LSR}> <\text{Dest}> <\text{OldSID}> <\text{SessionID}> <\text{FlowID}> <\text{TunnelID}> <\text{Rate}> <\text{Bucket}> <\text{TTL}>
\]

If this command is inserted in the simulation script, the \( \langle \text{Ingress-LER} \rangle \) RSVP-TE agent invokes the OSPF-TE agent as soon as a PathError is processed due to a link failure. Thus, the OSPF-TE agent performs the WC algorithm to compute an alternative route for the failed LSP. Once the new route has been calculated, the LSP is established.

IV. SIMULATIONS

This section shows the results of some tests performed to validate the new modules added to the simulator.

Figure 3 shows the network topology chosen for the tests. All the nodes, except for the traffic sources \((0, 1, 2, 3)\) and
receivers (17, 18, 19, 20), are considered as part of an MPLS domain.

Moreover, to validate the dynamic behaviour of the simulator, a failure on the link 12_13 is forced at \( t = 16s \). As a consequence, the corresponding LSP is released and flow 4 packets are forwarded along the shortest path computed by OSPF. Meanwhile, the ingress node (LSR7) computes a path for the backup LSP by executing the WC algorithm. The final situation, after the establishment of the recovery LSP along the path 7_6_5_8_10_16, is shown in figure 5.

### Table I LSP allocation requests

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Traffic Flow</th>
<th>Source Node</th>
<th>Dest. Node</th>
<th>Bandwidth (Kbps)</th>
<th>Delay (ms)</th>
<th>Ingress Node</th>
<th>Egress Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.0</td>
<td>1</td>
<td>0</td>
<td>17</td>
<td>400</td>
<td>330</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>8.0</td>
<td>2</td>
<td>1</td>
<td>18</td>
<td>550</td>
<td>370</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>9.0</td>
<td>3</td>
<td>19</td>
<td>2</td>
<td>500</td>
<td>300</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>12.0</td>
<td>4</td>
<td>3</td>
<td>19</td>
<td>300</td>
<td>120</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>13.0</td>
<td>5</td>
<td>2</td>
<td>20</td>
<td>350</td>
<td>360</td>
<td>6</td>
<td>15</td>
</tr>
</tbody>
</table>

### Table II WC algorithm output

<table>
<thead>
<tr>
<th>Traffic Flow/ LSP Number</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4 5 8 10 16</td>
</tr>
<tr>
<td>2</td>
<td>4 6 11 14 15 16</td>
</tr>
<tr>
<td>3</td>
<td>16 15 14 11 6</td>
</tr>
<tr>
<td>4</td>
<td>7 12 13 15</td>
</tr>
<tr>
<td>5</td>
<td>6 11 14 13 15</td>
</tr>
</tbody>
</table>

Figure 4 shows a snapshot of the traffic flows in the network at \( t=15s \), when all the LSPs have been established.

Figures 6 and 7 respectively show the throughput and the delay experienced by flow 4 packets during the whole simulation.
In both graphs we can identify five phases:

1) From t = 0s to t = 8s: all the flows are forwarded according to traditional IP routing (path 7, 6, 11, 14, 16). Since the path followed by flow 4 is not congested, there are no losses (the throughput at the receiver is 300Kbps), while the average delay is about 165ms.

2) From t = 8s to t = 12s: at t = 8s an LSP (550Kbps) bound to the flow 2 is created on the path followed by flow 4. As a consequence, since the available bandwidth (150Kbps on the link 11_14) is not enough, flow 4 experiences some losses and the throughput reduces to about 150Kbps, while the average delay increases up to 340ms.

3) From t = 12s to t = 16s: flow 4 is bound to an LSP, the throughput corresponds to the reserved bandwidth (300Kbps) and the average delay (60ms) satisfies the requirement of 120ms.

4) t = 16s: link 12_13 fails. As a result, the flow is forwarded, once again, according to layer 3 routing. Hence, we have a peak in the average delay and the throughput suddenly decreases.

5) From t = 16s to t = 20s: the flow is bound to the backup LSP. Therefore, the throughput is equal to the reserved bandwidth and the average delay is about 90ms.

B. Second Simulation Scenario

A further set of simulations has been performed to validate the WCS path computation algorithm. The links metrics and the LSP set-up requests are reported in Table III and IV, respectively, while the output of the WC algorithm is shown in Table V. Note that the fourth request is rejected due to a bandwidth error.

<table>
<thead>
<tr>
<th>Link</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upstream Node</td>
<td>Downstream Node</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Other</td>
<td>Other</td>
</tr>
</tbody>
</table>

Table III Link metrics

<table>
<thead>
<tr>
<th>Flow Number</th>
<th>Source Node</th>
<th>Dest. Node</th>
<th>Bandwidth (Kbps)</th>
<th>Delay (ms)</th>
<th>Ingress Node</th>
<th>Egress Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>17</td>
<td>400</td>
<td>400</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>18</td>
<td>600</td>
<td>320</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>19</td>
<td>700</td>
<td>400</td>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>20</td>
<td>600</td>
<td>250</td>
<td>7</td>
<td>15</td>
</tr>
</tbody>
</table>

Table IV LSP allocation requests

By using the WCS algorithm, the best solution is obtained when the requests are sorted according to their bandwidth requirements. As clearly shown in table VI, the processing order 3, 4, 2, 1 permits to satisfy all the requests.

<table>
<thead>
<tr>
<th>Traffic Flow/ LSP Number</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4 5 8 10 16</td>
</tr>
<tr>
<td>2</td>
<td>4 6 11 14 15 16</td>
</tr>
<tr>
<td>3</td>
<td>6 7 12 13 14 16</td>
</tr>
<tr>
<td>4</td>
<td>No Path – Bandwidth Error</td>
</tr>
</tbody>
</table>

Table V WCS algorithm output

A second simulation has been performed by using the links metrics and the LSP allocation requests shown in Table VII and VIII, respectively.
Constraint-based path computation is a key function in MPLS and GMPLS networks. Several algorithms have been proposed in literature to satisfy the QoS requirements of LSPs allocation requests and to adopt traffic engineering strategies. The paper describes the design and development of a new NS2 software module for the simulation of path computation algorithms in MPLS networks. Different types of algorithms have been implemented and integrated in the simulator. In particular, a new algorithm (WCS) has been proposed to improve the performance of WC when a centralized approach is used. Finally, the paper reports the results of some simulations, performed in a test network scenario, to show how the developed software module can be used to satisfy a set of LSP allocation requests with multiple QoS constraints.

### V. CONCLUSION

The output of the WC algorithm is shown in Table IX. If the requests are analyzed in the order 1,2,3,4 the second and the fourth are rejected because their delay constraints cannot be satisfied.

The authors would like to thank Marco Meini for his work in support of the development activities related to the simulator. This work was partially supported by Euro-NGI Network of Excellence funded by the European Commission.

### ACKNOWLEDGMENTS

### REFERENCES


