

Dynamic Widest Path Selection for Connection Admission Control in Core-Stateless Networks

Avadora Dumitrescu

Tampere University of Technology, P.O. Box 553, FIN-33101, Tampere, Finland

E-mail: avadora.dumitrescu@tut.fi

Abstract

A method for determining the highest available-bandwidth paths in a core-stateless network, with or without restrictions, is proposed. The method keeps track of the possible solution range as it progresses, can operate under number of hops, delay or cost constraints, and induces load balancing and high network resource utilization. The method is combined with a token-based transfer of information and semaphore mechanism to avoid over-allocation and ensure resource allocation coordination along the logical ring of the ingress nodes. The resulted Connection Admission Control procedure alleviates the scalability problem by keeping all the admission control operations in the ingress nodes while preserving the guaranteed service semantics and providing a low-latency, quasi-parallel allocation under light load and a centralized/semaphored allocation under heavy load. The widest path search algorithm is illustrated and compared with other existing algorithms.

1. Introduction

The Quality of Service (QoS) concept has been proposed to capture the performance contract between the service provider and the user applications. The QoS framework brings an increase in computation, storage and communication load (i.e. increased network operation costs) and this should be counterbalanced by the increase in service performances and/or network operation income.

To address the multitude of QoS issues, which can no longer be satisfied effectively by the prevailing best effort paradigm, the Internet Engineering Task Force (IETF) has proposed two different service frameworks, namely Integrated Services (IntServ)[1] and Differentiated Services (DiffServ)[2]. More recently the core-stateless (CS) approach is attempting to provide the guaranteed service semantics while ensuring the scalability [3]. This is achieved by keeping all the per-flow operations in the edge nodes of the domain while the simplified internal core routers' packet admission control and scheduling rely only on their state and on the Dynamic Packet State (DPS) information, which is carried by each packet. An admission control framework is necessary in all the approaches to provide QoS. The QoS Connection

Admission Control (CAC) must tackle at least some of the following tasks (some of the tasks might miss from particular variants of the admission control framework):

- collect and distribute the state information (the information describing the network topology and status);
- find/calculate a feasible/optimal route (path or tree), taking into account restrictions, optimizations required, inaccuracies in state information, etc.;
- ensure that the resources are not over-allocated;
- reserve the resources (explicitly or implicitly).

The main contributions of this paper are in the areas of state information management and distribution, bandwidth allocation coordination and in the procedure for finding the widest paths. These contributions are made within the core-stateless paradigm. Regarding the state information management and distribution, and the coordination of bandwidth allocation, the paper proposes a token-based transfer of information combined with an accumulation procedure, which prevent over-allocation while providing a low latency, quasi-parallel allocation under light load and a higher latency, quasi-semaphorized allocation with the Zero False Negative¹ (ZFN) property under heavy load. Regarding the proposed widest path search procedure, the paper gives some effective criteria for evaluating the distance to the (potential) optimum solution(s) during the search for all-pairs widest-paths. The evaluation criteria, together with a mathematically guaranteed stop criterion enable a more flexible approach in establishing the effectiveness of pursuing the optimum solution, accepting a non-optimum solution or terminating the search before the stop criteria are met.

The remainder of the paper is structured as follows: Section II introduces the general context and related work. Section III describes the mechanism for collecting and distributing the state information. In Section IV, the procedure for finding all-pairs widest paths is presented and some properties of the search process, including an effective stop criterion for the search are mathematically derived. Section V discusses the application of constraints to the widest path search and gives a measure to evaluate the distance to the (potential) optimum solution. An insight of the widest path searching procedure by

¹ The zero false negative (ZFN) property guarantees that a connection request is rejected only when there is no way to satisfy its requirements under the given network context.

examples and discussion, together with comparisons with other procedures is provided in Section VI. Section VII discusses the treatment of the best effort traffic and, in the end, Section VIII draws some conclusions and outlines further work.

2. Problem statement

The problem to be solved in the CAC framework can be defined by a number of variables. An essential variable is the *number and type of metrics* used for defining the QoS and the network operation profit. The metrics most frequently used to define the QoS are: bandwidth, delay, variation of delay experienced by the receiver (jitter), throughput or packet loss rate, reliability or failure rate of the connection, fairness in flow treatment and response time to the service request (latency). The ways in which the QoS metrics combine along the path divide them into additive metrics (e.g. delay, jitter), where the path performance is derived from the sum of the link performances; multiplicative metrics (e.g. reliability, packet loss rate), where the path performance results out of the link performance multiplication; and (concave) min-max metrics (e.g. bandwidth) [4]. The network administration costs and the network operation revenue can be related to metrics like: capacities on links, average number of hops, storage requirement, computational complexity, degree of network utilization, amount of carried traffic, number of flows transferred successfully, etc. The number and type of metrics taken into consideration largely governs the difficulty of finding the solution. It is known that the CAC and routing, performed under multiple (≥ 2) independent additive or multiplicative real-valued constraints or optimizations is a NP-complete problem [4].

From the rather broad range of possible problem parameter combinations [5] we have considered:

- unicast transmission with single path for each flow;
- optimization for the largest available bandwidth, with or without restrictions given by other criteria;
- absolute guarantees;
- global/selected and dynamic state information;
- computation (and communication) distributed among the traffic sources (ingress nodes) of the network domain.

Since we are targeting the path of maximum available bandwidth we do not build the solution around the shortest path. The ubiquitous shortest path routing has the obvious disadvantage that all packets are routed via the same shortest path as long as the routing tables are unchanged. Thus, the links on the shortest path become critical failure points. The shortest path routing distributes the traffic on alternative paths only when congestion has developed on the shortest path. Consequently the traffic load is not evenly distributed in the network but it rather clogs the links one by one in the increasing order of their length or cost. Thus, the network capacity is underutilized

and/or the flows receive a poor QoS as soon as the bandwidth on the shortest path is no longer able to carry effectively all the requested traffic. The preferential distribution of all traffic towards the shortest (or lowest cost) path also brings up the problem that the shortest (or lowest cost) path might get clogged with traffic that does not really need the speedy (or low cost) delivery provided by the shortest (or lowest cost) path. Thus, some traffic that would really require speedy (or low cost) delivery might be denied the service despite the fact that an effective resource allocation could have accommodated it while ensuring the required QoS for all traffic.

Among the best known routing techniques, the Min-Hop Algorithm (MHA) [6] routes an incoming connection along the path with the minimum number of feasible links. The MHA, based on the Dijkstra algorithm, is simple and computationally efficient. However, the choice of the shortest path remains unchanged with link loads, resulting in the use of the same path until saturation. Thus, applying MHA under heavy load results in 'clogged' bottleneck links (on the shortest path), while links of alternate paths are still underutilized.

The best known algorithms for finding the widest paths from a given source node and for incorporating various restrictions in the search procedure are the Widest Shortest Path (WSP) [7] and Shortest Widest Path (SWP) [4] algorithms. Our approach aims at an alternate approach that could also provide the optimum solution and provide features/capabilities un-attainable by WSP or SWP. Such capabilities are: finding the widest paths for all pairs of nodes with a limited increase in computation complexity, enabling the evaluation of the distance to the solution during the search, implicit or easy incorporation of restrictions and more flexible stop criteria.

Besides targeting the maximum available bandwidth, another defining principle of our approach was to tackle the different QoS parameters as much as possible at different levels of the procedural solution. The provision of a high throughput (low packet loss rate) and the fair treatment of flows are achieved by a congestion control mechanism [8]. The delay and the jitter are transformed from restrictions over the whole path into dynamically evaluated local restrictions, and taken care of by the packet scheduling procedure [9]. An essential condition to ensure high throughput and delay guarantees, and for decoupling the delay from the dynamic load is, in our approach, to make sure that the total of bandwidth reservations on each link is below the link capacity.

3. Collecting the state information

The network domain on which the QoS CAC and routing are performed consists of edge and core routers. The edge routers are the ingress and egress points for all the traffic that is transferred through the domain. Consistent with the core-stateless paradigm the core

routers do not maintain per-flow state. Only the ingress nodes of the domain, organized into a logical token-ring, participate to storing, communicating and updating the state information regarding bandwidth reservation. The information regarding the network topology (initialization and changes), including link capacities, is channeled periodically from each core node towards the logical token-ring, where it is circulated among the ingress nodes by a topology information token. The network topology information is thus consistent along the logical token-ring within a round-trip delay of this token.

A. Bandwidth Reservation Tokens (BRTs)

Other tokens, the bandwidth reservation tokens (BRTs), circulating on the same logical ring, carry information regarding the available bandwidths on the links of the domain. The available bandwidth on each link (= the link capacity – the aggregate of the accepted bandwidth reservations for the link) is distributed (during and through the reservation-release process) among different BRTs. Any ingress node can reserve bandwidth on a link by subtracting from the available bandwidth values and releases bandwidth by adding to the available bandwidth values. The overbooking is prevented simply by the fact that the available bandwidth values are kept non-negative. An arbitrary number of BRTs can be used without any danger of overbooking if:

a. the sum of all available bandwidth values carried by the BRTs for any given link ($\sum ab_{ij}^k$, where i, j specify the $i \rightarrow j$ link and k specifies the BRT that carries the respective available bandwidth value) is equal at all times with the effective available bandwidth (or spare capacity) for the link (EFF_SC_{ij} , taken as the difference between the $i \rightarrow j$ link capacity, C_{ij} , and the sum of all the accepted bandwidth reservations on the link):

$$\sum_{k=1}^{No. of BRTs} ab_{ij}^k = EFF_SC_{ij} = C_{ij} - \sum_{l=1}^{nr. of flows} R_{ij}^l \quad (1)$$

where R_{ij}^l is the bandwidth reserved by the l -th flow through the $i \rightarrow j$ link (it is assumed that the CPU bandwidth, i.e. the maximum rate at which the node can pump data into the link, is always greater than the link capacity).

b. upon request, any ingress node can reserve bandwidths on the network links only up to the available bandwidth values specified by the BRT that is currently visiting the node (i.e. any bandwidth reservation on the link $i \rightarrow j$, is restricted to be smaller than ab_{ij}^k , with k the index of the BRT currently in the node).

This mechanism has the advantage that it increases the parallelism of bandwidth allocation and reduces the CAC latency, which is effective mostly in case of plentiful resources, when it would be meaningless to wait for a single semaphore token to complete its round-trip in order to process a request. However, the procedure also has the disadvantage that, by distributing the available

bandwidths of the links among the BRTs and searching for the paths using different ab_{ij}^k sets (with $k=1$ to M , where M is the number of BRTs), it can deny the connections that could have been granted if the search for the available paths was performed on the centralized state information (i.e. based on EFF_SC_{ij}).

B. Accumulation procedure

The erroneous connection denials, which might be due to the distribution of the available bandwidths among BRTs, can be prevented (i.e. Zero False Negative property can be ensured) by an accumulation procedure. For simplicity we present just the simplest accumulation procedure, which is initiated whenever an ingress node is not capable of finding a path for a connection request, based on the information in the current BRT. Once the accumulation procedure is initiated, the node sets up accumulation pots for all the links (ap_{ij}^o , with i, j giving the link for which the accumulation is performed and o giving the index of the ingress node), where it collects/empties the corresponding available bandwidth values (ab_{ij}^k) from all the passing BRTs until the available bandwidths accumulated in the pots enable finding a path. Consequently, the BRTs transiting the accumulation node leave with all ab_{ij}^k values equal to zero. If the accumulation is carried out in just a single ingress node then, within a token-ring round-trip, according to (1), the accumulating node collects in its accumulation pots the exact spare capacities for all links (except for the bandwidths released during the round-trip). If it still cannot find a path for the requested connection, the accumulation node rejects the connection and unloads its accumulation pots into the next BRT, which will now hold the quasi-complete momentary state-information regarding the available bandwidths. A semaphore mechanism is still needed to coordinate the accumulation process so that different ingress nodes do not start and maintain competing accumulation processes. This can be done, in the simplest fashion, by launching a ‘release’ flag along the token-ring, from the node that starts an accumulation process. The ‘release’ flags and accumulation processes could have different priorities and could be associated with the corresponding source and destination. Accordingly, every ingress node that has started an accumulation process could be forced to end it and release the accumulated available bandwidths upon receiving a ‘release’ flag signaling the start of a (competing) accumulation process of higher priority.

With several coordinated accumulation processes in progress, the left-side equality of (1) becomes:

$$\sum_{k=1}^{No. of BRTs} ab_{ij}^k + \sum_{\substack{o=ingress\ node \\ with\ acc.\ proc.}} ap_{ij}^o = EFF_SC_{ij} \quad (2)$$

derived from the distribution of the effective spare capacity of the link (EFF_SC_{ij}) between the available

bandwidths carried by BRTs (ab_{ij}^k) and the accumulation pot(s) of the node(s) having an accumulation process in progress (ap_{ij}^o).

C. Regulating the BRTs contents and number

A simple mechanism for the automatic regulation of the number of BRTs can be implemented based on two requirements. Firstly, the number of BRTs should be large enough to minimize the time spent by an ingress node waiting for a BRT upon receiving a request (*wait_BRT*). Secondly, the next BRT should not arrive during the processing of a request, whose minimum duration can be evaluated (*min_proc*). The regulation of the number and spacing of the BRTs can be performed by taking care that, whenever the incoming BRTs are spaced with more than *wait_BRT*, new BRTs are launched and that, whenever the incoming BRTs are spaced with less than *min_proc*, they are retained in the node. Furthermore, whenever more than one BRT are in the same ingress node at the same time they are merged. An advantage in using the *wait_BRT* and *min_proc* variables for regulating the number and spacing of the BRTs is that the process can be dynamically adjusted according to the changing requests and processing requirements.

A procedure for (evenly) distributing the available bandwidths between the BRTs as well as a procedure to ensure the maintenance of consistent information and to provide resilience in case of BRT loss should also be deployed. A straightforward way for minimizing the probability of token loss, detecting the token loss and recovering the lost information is to operate the ingress-node token-ring as a resilient overlay network [10] [11] [12]. Also the ingress nodes could keep track of the reservations and releases they have made and an accumulation procedure (triggered by the detection of token loss or performed periodically) along the token ring could check, for every link, the equality:

$$\sum_{k=1}^{No. of BRTs} ab_{ij}^k + \sum_{\substack{o=ingress\ node \\ with\ acc.\ proc.}} ap_{ij}^o = C_{ij} - \sum_{l=1}^{nr. of\ flows} R_{ij}^l \quad (3)$$

where the aggregate available bandwidth in circulation along the token-ring, the first term on the left hand side, is collected from BRTs; the accumulation along the token ring, second term on the left hand side, is collected from the ingress nodes; the link capacity (C_{ij}) is collected from topology information tokens; and the bandwidth reservations made by all ingress nodes (R_{ij}^l) is collected from the ingress nodes. The advantage of such a re-initialization procedure is that it discards/releases automatically any reservations made by an ingress node that has failed.

The whole procedure of managing the distribution of available bandwidth information among the BRTs and among the ingress nodes attempting to realize a connection ensures that distributed, quasi-parallel

bandwidth allocation takes place under light load conditions, when the available bandwidths are quasi-evenly distributed among the BRTs, while under heavy load the available bandwidths start to concentrate in fewer BRTs or ingress nodes, towards the limit situation when, under heavy congestion, the whole spare capacity for all links could be concentrated in a single BRT/ingress node. It should be pointed out that it is also possible to have simultaneously a low latency, quasi-parallel allocation for some un-congested links (whose available bandwidths are distributed among BRTs) and a higher latency, quasi-semaphorized allocation for the links under heavy load (whose available bandwidths are concentrated in one single BRT). It also should be mentioned that using several BRTs does not improve the worst case latency (as compared with the case of a single semaphore token) but it significantly decreases the average latency, mostly under light and distributed load.

4. Finding the Widest Path

The available bandwidth information, provided by the previously described state information management procedure, is structured in the spare-capacity matrix of the first order $\|SC\|^1$, with the elements given by $sc_{ij}^1 = ab_{ij}^k$ (with i, j corresponding to the $i \rightarrow j$ link and k corresponding to the current BRT). At the limit of the accumulation process, when all the available bandwidths are accumulated in the node performing the path finding:

$$sc_{ij}^1 = C_{ij} - \sum_{l=1}^{nr. of\ flows} R_{ij}^l \quad (4)$$

For all the pairs of nodes without a direct link, including the diagonal elements of the matrix, sc_{ij}^1 is 0. Since the reservation rule is that the spare-capacities are maintained in the non-negative domain of values, i.e.

$$\sum_{l=1}^{nr. of\ flows} R_{ij}^l < C_{ij} \quad (5)$$

it results that, in general, $\|SC\|^1$ is a sparse, non-symmetric matrix (the capacities and reservations are not equal in both directions of a link) with non-negative elements.

Starting from $\|SC\|^1$, which gives the spare link-capacities through a single link, the largest spare link-capacities through paths consisting of more than one link can be derived step-by-step by the following matrix computation:

$$sc_{ij}^k = \max_{m=1 \dots N} \left\{ \min \left\{ sc_{im}^{k-1}, sc_{mj}^1 \right\} \right\} \quad (6)$$

where N is the number of nodes in the domain of search. The computation is similar with the matrix multiplication having the element multiplication replaced by the min operator and the summing replaced by the max operator. The min operator is applied to the pair of spare capacity

values for a (k-1)-link path (having sc_{im}^{k-1} spare capacity) and for the respective link (with sc_{mj}^1 spare capacity) to be concatenated in the k-link path. The max operator is applied to all the resulted alternative k-link paths (which can be up to N).

In order to avoid including looped paths into the calculation, the elements of higher-order spare-capacity matrices (sc_{ij}^k , for $k>1$) are associated with the list of nodes on the corresponding path. The spare-capacities corresponding to the situation where the node to be added on the path is already in the list, $j \in \{i \rightarrow \dots \rightarrow m\}$, are not taken into consideration by the max operator in (6), thus eliminating the loops.

The k-th order spare-capacity matrix $\|SC\|^k$ gives the largest spare-capacities on paths of k links between any two nodes (because the partial paths that are eliminated by the max operator are only those which have wider alternatives). Consequently, the problem of finding the maximum spare-capacity path $n \rightarrow p$ can be solved by looking up the sc_{np}^k values for various k. The computational complexity of the problem is given by the number of nodes in the domain (spare-capacity matrix dimension = $N*N$), by the number of non-zero matrix elements and by the matrix order up to which the search is performed. The computational complexity of the problem can be reduced if there is a constraint on the solution. For example if one has to find the maximum spare-capacity which is above a certain constraint value, the first-order spare-capacity matrix can be purged of all spare capacity values below the constraint value, thus becoming sparser.

A property of the spare-capacity matrix that helps in evaluating the distance to the (potential) optimum solution is that both the minimum and the maximum spare-capacity values in higher-order matrices are bounded by the minimum and maximum spare-capacity values in lower-order matrices.

Pinch theorem: Given a higher-order spare-capacity matrix $\|SC\|^k$ ($\forall k>1$), constructed by (6), then, for the non-zero elements of the matrix:

$$\min_{i,j} \{sc_{ij}^k\} \geq \min_{i,j} \{sc_{ij}^{k-1}\} \quad (7)$$

and

$$\max_{i,j} \{sc_{ij}^k\} \leq \max_{i,j} \{sc_{ij}^{k-1}\} \quad (8)$$

Proof of (7): $\min_{i,j} \{sc_{ij}^2\} \geq \min_{i,j} \{sc_{ij}^1\}$ because when constructing $\|SC\|^2$, the exterior max operator in (6) is bound to generate values of at least $\min_{i,j} \{sc_{ij}^1\}$. By

induction (7) is true. In other words, the minimum spare-capacity path (having more than one link) contains the minimum spare-capacity link and, as long as there are alternative paths to this minimum spare-capacity path then (7) holds, with the equality being true when there are

no other/alternative paths to the path containing the minimum spare-capacity link or when all alternative paths have the same minimum spare-capacity.

Proof of (8): $\max_{i,j} \{sc_{ij}^2\} \leq \max_{i,j} \{sc_{ij}^1\}$ because when constructing $\|SC\|^2$, the interior min operator in (6) generates a value lower than $\max_{i,j} \{sc_{ij}^1\}$ unless the link

concatenated with the maximum spare-capacity link also has the maximum spare-capacity, in which case

$\max_{i,j} \{sc_{ij}^2\} = \max_{i,j} \{sc_{ij}^1\}$. By induction (8) is true. In

other words, as long as the link added to the k-1 order maximum spare-capacity path does not have the maximum spare-capacity, then

$\max_{i,j} \{sc_{ij}^k\} < \max_{i,j} \{sc_{ij}^{k-1}\}$. The equality holds in (8)

only when the links added to the maximum spare-capacity path also have the maximum spare-capacity.

The search for the maximum $n \rightarrow p$ spare-capacity can be stopped/concluded according to the following theorem.

Ceiling theorem: The maximum spare-capacity between two nodes n and p is the maximum value from the spare capacities obtained by (6) for $n \rightarrow p$ paths having 1 to s links, $\max_{k=1..s} \{sc_{np}^k\}$, when:

$$\max_{k=1..s} \{sc_{np}^k\} \geq \max_{i,j} \{sc_{ij}^s\} \quad (9)$$

Proof: $\max_{i,j} \{sc_{ij}^s\} \geq \max_{i,j} \{sc_{ij}^{s+n}\}$ for any $n \geq 1$

according to the pinch theorem. If (9) is true, then:

$$\max_{k=1..s} \{sc_{np}^k\} \geq \max_{i,j} \{sc_{ij}^s\} \geq \max_{i,j} \{sc_{ij}^{s+n}\} \quad (10)$$

i.e. the maximum $n \rightarrow p$ spare-capacity value found in spare-capacity matrices up to s-order (which can be and frequently is different from the maximum $n \rightarrow p$ spare capacity found in the s-order matrix), cannot be lower than any $n \rightarrow p$ spare-capacity in matrices of higher order than s (as long as it is higher or equal to the maximum value in the s-order spare capacity matrix). Therefore, the maximum spare capacity already found cannot be increased further.

If the stop criterion given by the ceiling theorem is applied for several source-destination pairs (i.e. the search is stopped only when the condition given by the ceiling theorem is satisfied for all the source-destination pairs) then the widest path is found for all the considered source-destination pairs. Since the algorithm generates simultaneously the widest paths for all the possible source \rightarrow destination pairs, the computational effort for finding several widest paths is the same as the highest computational effort for finding a single widest path in the set. However, the parallelism in calculating several widest paths in the same run can be applied directly to bandwidth reservation only if the different paths found do

not share links, otherwise the algorithm should be applied sequentially. Since, at the limit, the algorithm can find all-pairs widest paths for a given spare-capacity matrix we call it the All-Pairs-Widest-Path-Search (APWPS).

5. Widest Path search with constraints

Besides the bandwidth constraints (when searching for a path with a minimum-accepted bandwidth), which can be applied by purging the first-order spare-capacity matrix of all values less than the minimum-accepted bandwidth, one or more additive and/or multiplicative constraints – simple or combined into cost functions – can also be applied in the upper-matrix computation (6). For example if the path is delay constrained (i.e. when only the paths with delays below a maximum accepted delay are useful), then an estimated or measured (upper) delay value is associated with every link. When a new link is concatenated to an existing path, the extended path is accepted only if the delay over it (i.e. the delay over the (k-1)-link path + the delay of the added link) is less than the maximum accepted delay. Otherwise the extended path is not taken into account. A similar selection procedure can be applied to a number of different additive and/or multiplicative constraint criteria or to a (cost function) combination of constraint criteria. Other min-max selection criteria, different from the bandwidth, could also be added into the path selection process. However, it should be mentioned that, while applying the un-constrained procedure ensures finding the widest path (ZFN property), including constraints into the procedure might prevent it from finding the (optimum) path, depending on the number, nature and way of applying the constraints. It should also be mentioned that finding the path that best satisfies the constraint might not be always the most effective approach. For example, when delay constraints are used, targeting the shortest delay path might clog this path with traffic that does not really require such a speedy delivery and prevent satisfying later requests for low delay paths. It is clearly better to provide paths that satisfy the delay requirement (with some margin) without being the shortest delay paths. Thus, the requests for short delay transfer are less affected by previous reservations for traffic with less stringent delay requirements.

The constraint criteria (or a cost function) can also be used to limit the depth of search (in the number of links on the path) by using them to stop upper-order matrix computations before the ceiling theorem is satisfied. The most straightforward restriction that can be applied is in terms of maximum acceptable number of hops, which can be enforced by stopping the computation, if the widest path was not already found, at the spare-capacity matrix order corresponding to the maximum acceptable number of hops. In case of using more general cost functions, the depth of search can also be decided by evaluating the

potential benefits of continuing the search. According to the pinch theorem, the maximum potential improvement to the maximum spare-capacity found for $n \rightarrow p$ through paths of up to s links is $\max_{i,j} \{sc_{ij}^s\} - \max_{k=1..s} \{sc_{np}^k\}$,

because the maximum spare-capacity in higher-than- s -order matrices is bound by $\max_{i,j} \{sc_{ij}^s\}$. Then, if the

minimum supplementary cost of adding more links to the path (which can be obtained from the cost matrix) counterbalances the potential improvement in the spare capacity on the path, the search can be stopped even before finding the maximum spare-capacity path (before the ceiling theorem is satisfied).

6. Examples and discussions

The network domain used to illustrate and discuss the properties of the widest path search algorithm is depicted in Fig. 1.

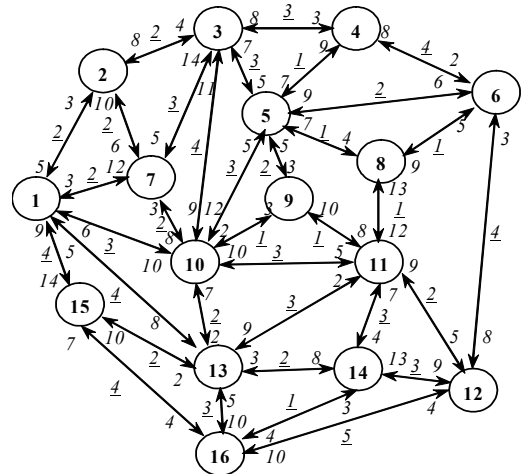


Fig. 1 Topology of the network domain where the widest path algorithm was applied.

The domain has 16 nodes and the edge of the domain, on which the token-ring is organized, consists of the ingress/egress nodes 1, 2, 3, 4, 6, 12, 15, 16. The links are bidirectional, with different available bandwidths for the two transmission directions, rendering the first-order spare-capacity matrix – given in Table 1 – asymmetric. In Fig. 1 the cost values in are underlined and the available bandwidth values are placed next to the arrow pointing in the corresponding transfer direction. The link costs, taken with the same value for both directions of transfer, render a symmetric first-order cost matrix, given in Table 2, but asymmetric cost matrices can equally be used. Normalized integer values have been considered to ease the presentation and discussion of the solutions. Although the APWPS operates similarly with real values, it is computationally efficient to map the real values to integer ones.

AB	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	3	0	0	0	0	12	0	0	10	0	0	8	0	14	0
2	5	0	4	0	0	0	6	0	0	0	0	0	0	0	0	0
3	0	8	0	3	5	0	10	0	0	9	0	0	0	0	0	0
4	0	0	8	0	7	2	0	0	0	0	0	0	0	0	0	0
5	0	0	7	9	0	6	0	4	3	12	0	0	0	0	0	0
6	0	0	0	8	9	0	0	9	0	0	0	8	0	0	0	0
7	3	10	14	0	0	0	0	0	0	8	0	0	0	0	0	0
8	0	0	0	0	7	5	0	0	0	12	0	0	0	0	0	0
9	0	0	0	0	7	0	0	0	0	2	8	0	0	0	0	0
10	6	0	11	0	5	0	3	0	3	0	5	0	2	0	0	0
11	0	0	0	0	0	0	0	13	10	8	0	5	9	4	0	0
12	0	0	0	0	0	3	0	0	0	0	9	0	0	13	0	10
13	5	0	0	0	0	0	0	0	0	7	2	0	0	8	10	10
14	0	0	0	0	0	0	0	0	0	0	7	9	3	0	0	4
15	9	0	0	0	0	0	0	0	0	0	0	0	2	0	0	4
16	0	0	0	0	0	0	0	0	0	0	0	4	5	3	7	0

Table 1. First-order spare-capacity matrix for Fig. 1, with available-bandwidth values given for the transfer direction <row index>→<column index>

c	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1		2					2			3			4		4	
2	2		2				2									
3		2		3	3		3			4						
4			3		1	4										
5			3	1		2		1	2	3						
6				4	2			1				4				
7	2	2	3							2						
8				1	1						1					
9					2					1	1					
10	3		4	3		2		1		3		2				
11							1	1	3		2	3	3			
12					4						2		3		5	
13	4								2	3			2	2	3	
14										3	3	2				1
15	4											2				4
16												5	3	1	4	

Table 2. First-order cost matrix for the network domain depicted in Fig. 1.

The progress towards the solution when searching for the path 1→6 is outlined in Table 3. The columns 2 and 3 give the maximum and minimum available bandwidth values in the spare-capacity matrix and the columns 4 and 5 give the maximum and minimum costs in the associated cost matrix. The effect of the pinch theorem can be observed in the maximum and minimum available bandwidth columns (2) and (3). Columns 6, 7 and 8 track the evolution of the ‘temporary’ solution series sc_{16}^k (with $k=1$ to 15). The stop criterion given by the ceiling theorem was not employed. Instead, the spare capacity matrix and cost matrix computations were carried until the 15th order (i.e. until all possible paths were explored) to illustrate the APWPS operation. Had the stop criterion (given by the ceiling theorem) been applied, the search would have been stopped at the 13th order. Also the solution was searched under the restriction that the cost of

the path should be below 200 (i.e. practically no cost restriction was applied). The found solution (1→13→14→11→8→5→6, with a bandwidth of 6 and a cost of 13) is different from the shortest path solution obtained by the Dijkstra algorithm.

Order	max. band.	min. band.	max. cost	min. cost	1 to 6 band.	1 to 6 cost	1 to 6 path (cost < 200)
1	14	2	5	1	-	-	-
2	12	2	9	2	-	-	-
3	10	2	13	3	5	8	1-10-5-6
4	10	2	17	4	5	8	1-10-11-8-6
5	9	2	21	6	5	10	1-10-11-8-5-6
6	9	2	23	9	6	13	1-13-14-11-8-5-6
7	9	2	26	12	6	15	1-13-14-12-11-8-5-6
8	9	2	28	15	5	23	1-13-14-12-11-10-3-5-6
9	8	2	30	17	4	23	1-13-14-12-11-10-3-5-8-6
10	7	2	32	19	-	-	-
11	7	2	34	20	-	-	-
12	7	2	38	24	-	-	-
13	6	2	39	28	-	-	-
14	6	2	39	32	-	-	-
15	-	-	-	-	-	-	-

Table 3. Evolution of the minimum and maximum values in the spare-capacity matrix and cost matrix, together with the evolution of the 1→6 solution found by APWPS with increasing order

The Dijkstra algorithm, operating on the same cost matrix (given in Table 2) generates the solution 1→10→9→11→8→6, with a bandwidth of 3 and a cost of 7. When searching for the minimum hop path (by employing the Dijkstra algorithm and using unitary costs on all the links) the solution found was 1→10→5→6, with a bandwidth of 5 and a cost of 8. It can be seen that the minimum hop solution was also the first element in the ‘temporary’ solution series, sc_{np}^k , obtained by APWPS. This illustrates that the APWPS algorithm inherently incorporates the number of hop restriction in the spare-capacity matrix order, thus being always able to find the widest path for the minimum number of hops. Obviously, APWPS is able to generate the widest path for any given number of hops when incorporating the number of hop restriction in the criterion for ending the search. However, when other restrictions are incorporated into APWPS the optimum solution might not be guaranteed depending on the type, number and way of applying the constraint. For example, when searching for the widest 1→6 path in the specified domain, under the restriction that the cost should be smaller or equal to 7, APWPS finds no solution. This happens because the APWPS eliminates from search the alternate partial paths with smaller available bandwidths. In this particular case at order 3 the segment 1→10→9→11 is eliminated by the wider alternate 1→7→10→11, which still obeys the cost restriction. However, for the general case, when applying restrictions within the cost range associated with the sc_{16}^k series, the solution is selected accordingly from the series of ‘temporary’ solutions.

Table 4 gives the solutions found by the APWPS with the maximum available-bandwidth as target and with maximum available-bandwidth together with minimum number of hops as targets (in this case the search is stopped at first solution found, i.e. the widest path for the minimum number of hops). The rows of the table illustrate the evolution of the solution found when unitary bandwidth reservations are made on the links of the paths found. First row gives the solution obtained using the first-order spare-capacity matrix given in Table 1. The following rows give the solution found after reserving a bandwidth of 1 on the solution path of the preceding row (i.e. solutions generated using a first-order spare-capacity matrix where a bandwidth of 1 was subtracted from the available bandwidths of the links belonging to the precedent solution).

step	APWP(available bandwidth)					APWP(available bandwidth + nr.hops)				
	path found	band	hops	cost	dead link	path found	band	hops	cost	dead link
1	1-13-14-11-8-5-6	6	6	13		1-10-5-6	5	3	8	
2	1-10-5-6	5	3	8		1-10-5-6	4	3	8	
3	1-10-11-8-6	5	4	8		1-10-5-6	3	3	8	
4	1-10-5-6	4	3	8		1-10-5-6	2	3	8	
5	1-10-11-8-6	4	4	8		1-10-5-6	1	3	8	10-5
6	1-10-5-6	3	3	8		1-10-11-8-6	5	4	8	
7	1-10-11-8-6	3	4	8		1-10-11-8-6	4	4	8	
8	1-13-14-12-6	3	4	13		1-10-11-8-6	3	4	8	
9	1-10-5-6	2	3	8		1-13-14-12-6	3	4	8	
10	1-10-11-8-6	2	4	8		1-10-11-8-6	2	4	8	
11	1-7-3-4-6	2	4	12		1-7-3-4-6	2	4	12	
12	1-13-14-12-6	2	4	13		1-13-14-12-6	2	4	13	
13	1-10-5-6	1	3	8	10-5 & 5-6	1-13-11-8-6	1	4	9	8-6
14	1-13-11-8-6	1	4	9	8-6	1-7-3-5-6	1	4	8	5-6
15	1-7-3-4-6	1	4	12	4-6	1-7-3-4-6	1	4	12	4-6
16	1-13-14-12-6	1	4	13	12-6	1-13-14-12-6	1	4	13	12-6

Table 4. Evolution of the widest 1→6 path found by APWPS variants when step-by-step unitary bandwidth allocation was performed on the selected widest paths.

For comparison we performed the same calculation using the Dijkstra algorithm with the cost minimization as target and with the number of hops minimization as target. The results are shown in Table 5.

step	DIJKSTRA(cost)					DIJKSTRA(nr.hops)				
	path found	band	hops	cost	dead link	path found	band	hops	cost	dead link
1	1-10-9-11-8-6	3	5	7		1-10-5-6	5	3	8	
2	1-10-9-11-8-6	2	5	7		1-10-5-6	4	3	8	
3	1-10-9-11-8-6	1	5	7	10-9	1-10-5-6	3	3	8	
4	1-10-5-6	5	3	8		1-10-5-6	2	3	8	
5	1-10-5-6	4	3	8		1-10-5-6	1	3	8	10-5
6	1-10-5-6	3	3	8		1-2-3-4-6	2	4	11	
7	1-10-5-6	2	3	8		1-2-3-4-6	1	4	11	4-6
8	1-10-5-6	1	3	8	10-5	1-2-3-5-6	1	4	9	1-2 & 5-6
9	1-10-11-8-6	2	4	8		1-10-11-8-6	5	4	8	
10	1-10-11-8-6	1	4	8	1-10 & 8-6	1-10-11-8-6	4	4	8	
11	1-2-3-5-6	1	4	9	5-6	1-10-11-8-6	3	4	8	
12	1-2-3-4-6	2	4	11		1-10-11-8-6	2	4	8	
13	1-2-3-4-6	1	4	10	1-2 & 4-6	1-10-11-8-6	1	4	8	1-10 & 10-11 & 8-6
14	1-13-14-12-6	3	4	13		1-13-11-12-6	2	4	13	
15	1-13-14-12-6	2	4	13		1-13-11-12-6	1	4	13	13-11
16	1-13-14-12-6	1	4	13	12-6	1-13-14-12-6	1	4	13	12-6

Table 5. Evolution of the widest 1→6 path found by Dijkstra variants when step-by-step unitary bandwidth allocation was performed on the selected widest paths.

It can be seen that APWPS targeting the maximum available bandwidth distributes the reservations among the widest paths, thus postponing link blocking as long as possible. Moreover, due to the distribution of the reservations among the widest paths, the number of links

blocked until the 1→6 transmission capability is exhausted is limited to the minimum. In comparison, the Dijkstra algorithm blocks more links for the same traffic transferred between 1 and 6. Even when the APWPS incorporates the minimization of number of hops (by stopping at the first path found with increasing the number of hops), it still postpones blocking the links until there is no other way to maintain the minimum number of hops but to block a link. While APWPS targeting the maximum available bandwidth provides better load balancing and reduced link blocking, the APWPS that is also targeting the minimum number of hops provides a smaller jitter when the bandwidth reservation range does not induce a change in number of hops.

We have also performed the same calculations that generated the Tables 4 and 5 using this time an implementation of WSP [8]. The results are given in Table 6.

step	WSP(available bandwidth)					WSP(available bandwidth + nr.hops)				
	path found	band	hops	cost	dead link	path found	band	hops	cost	dead link
1	1-13-14-11-8-5-6	6	6	13		1-10-5-6	5	3	8	
2	1-10-5-6	5	3	8		1-10-5-6	4	3	8	
3	1-10-11-8-6	5	4	8		1-10-5-6	3	3	8	
4	1-10-5-6	4	3	8		1-10-5-6	2	3	8	
5	1-10-11-8-6	4	4	8		1-10-5-6	1	3	8	10-5
6	1-10-5-6	3	3	8		1-10-11-8-6	5	4	8	
7	1-10-11-8-6	3	4	8		1-10-11-8-6	4	4	8	
8	1-13-14-12-6	3	4	13		1-10-11-8-6	3	4	8	
9	1-10-5-6	2	3	8		1-13-14-12-6	3	4	13	
10	1-7-3-4-6	2	4	12		1-7-3-4-6	2	4	12	
11	1-10-11-8-6	2	4	8		1-10-11-8-6	2	4	8	
12	1-13-14-12-6	2	4	13		1-13-14-12-6	2	4	13	
13	1-10-5-6	1	3	8	10-5 & 5-6	1-7-3-4-6	1	4	12	4-6
14	1-7-3-4-6	1	4	12	4-6	1-7-3-5-6	1	4	8	5-6
15	1-13-11-8-6	1	4	9	8-6	1-13-11-8-6	1	4	9	8-6
16	1-13-14-12-6	1	4	13	12-6	1-13-14-12-6	1	4	13	12-6

Table 6. Evolution of the widest 1→6 path found by WSP variants when step-by-step unitary bandwidth allocation was performed on the selected widest paths.

It can be seen that in both variants (when targeting just the maximum available bandwidth and when targeting both the maximum available bandwidth and the minimum number of hops) APWPS gives similar results with WSP (the slight differences coming from the fact that in our WSP implementation we did not order equally wide paths in increasing order of cost). What are then the advantages of using APWPS? Since the ratio between the APWPS and the WSP computational complexities when searching for paths emerging from a single source node is approximately $(N*N)/E$ (with N the number of nodes in the domain and E the number of links, taken in both directions if the available bandwidths are different depending on the transfer direction, i.e. in the worst case $E=N*(N-1)$, at the limit of a completely connected network with asymmetric bi-directional links; e.g. in the case of our example the ratio is 256/70), the APWPS is always more complex when performing a single source search. However, APWPS is able to generate the solution for all pairs of nodes with basically the same complexity as for the worst case of single source search. This feature is of interest for the routing problem only when the links

of the different paths generated at once by APWPS are not overlapping. Nevertheless, APWPS has the advantage that it can evaluate both the maximum (potential) improvement of the solution and the minimum cost to be paid for it, while proceeding towards the solution. This enables a more flexible approach in establishing the effectiveness of pursuing the optimum solution, accepting a non-optimum solution or terminating the search before the stop criteria are met. For example, the decision to stop the search can be taken if it is known that the temporary solution (with, lets say an available bandwidth of 5) can only marginally be improved (e.g. up to 6) at a high cost (min cost is high), but the search might be continued if it is known that a high available bandwidth improvement (e.g. from 5 to 25) could be obtained at an acceptable cost (min cost is low).

The use of APWPS is effective when its parallelism in finding the solutions to all the node pairs can be exploited. Such situation occurs when re-allocation and re-routing are performed. Re-allocation and re-routing of relatively long-lasting flows might offer better QoS because the paths were chosen in the context of the allocation moment. For a significantly changed network load distribution these previously allocated paths might no longer be the best choices. In case of a re-allocation for a multitude of long-lasting flows, the APWPS parallelism can be exploited to generate at once the solution for all the flows running through the domain (incorporating provisions of multi-allocation on the same link).

Other cases when APWPS might be efficient are the cases of multicast or generalized group transmission. APWPS parallelism and its flexibility in incorporating constraints can be used to search more effectively for a solution when multiple flows with heterogeneous characteristics and requirements must be established at once. Furthermore, when using a hierarchical decomposition of the network, the efficiency of APWPS increases as the domains become smaller (in the number of nodes) and with higher connectivity.

The computational complexity of the APWPS might be reduced (at the expense of redundant memory use) by distributing the state information among (and within) the ingress nodes. Each ingress node (I) could store a different spare-capacity matrix for each possible egress node (E): $\|SC_{IE}\|^1$. These spare-capacity matrices, different, at the limit, for each ingress-egress node pair, would contain only the information on the links that could be part of a path between the respective ingress and egress nodes. Upon receiving a request for an A→B path, the path search could be operated using the spare-capacities in the $\|SC_{AB}\|^1$ matrix, which is much smaller than the matrix for the whole network domain.

While the Zero False Positive (ZFP) property² is provided by obeying the rule of non-negative spare-capacity values (see (4) and (5)), it should also be taken into consideration that the traffic to be transferred has, most of the times, highly variable parameters. Taking the traffic parameters into account in the CAC process could either reduce the network utilization (by providing generous reservation margins) or could jeopardize the capability to ensure the QoS. Our approach was to eliminate all issues related to traffic parameter variations, flow policing, throughput control, ensuring fairness, delay and jitter control, etc., from the CAC process and let those issues be handled at other procedural levels (see packet handling procedures described in [8] and [9]). Consequently, the CAC procedure has only to take care that the selected path has the required characteristics (bandwidth + imposed restrictions) and it does not care if the flows are in- or out-of-profile. However, in order to ensure an effective operation, the total reservation limit over any link should keep a ‘reservation margin’ below the link capacity.

7. Treatment of Best-Effort traffic

The treatment of best effort traffic could be provided under the same framework simply by replacing the sum of all reservations over the each link with the effective total traffic through each link (Eff_traf_{ij}). The Eff_traf_{ij} values could be periodically reported to the ingress-node tokenring by the topology information tokens. By performing the path search and bandwidth allocation using Eff_traf_{ij} , the acceptance of best effort traffic is adapting to the network load context. Higher network utilization is

provided when $Eff_traf_{ij} < \sum_{l=1}^{nr. \text{ of flows}} R_{ij}^l$, while the best

effort traffic is not un-necessarily clogging the network in the opposite case. Furthermore, the best effort traffic does not affect the QoS for the reserved traffic if a proper packet scheduling and forwarding procedure ensures the isolation [8]. On the other hand, the non-blocking property for the best effort traffic can be assured by providing a reservation margin, M, to the link capacity:

$$\sum_{l=1}^{nr. \text{ of flows}} R_{ij}^l < C_{ij} - M \quad (11)$$

Congestion management and fairness in flow treatment can also be provided by the packet forwarding procedure [8].

² The Zero False Positive (ZFP) property ensures that, when a connection request is accepted, the required bandwidth is guaranteed for the solution path.

8. Conclusions

A connection admission control (CAC) framework, consistent with the core-stateless paradigm, was presented. The core-stateless semantics is upheld by keeping all the operations, decisions and signaling exclusively in the edge nodes of the domain. The core nodes not only do not perform any signaling or CAC operations, but they also do not keep any record of the reservations and any routing tables. The routing path information [13], together with all the other parameters needed for packet handling in the core routers [8] [9] are carried by the dynamic packet state (DPS).

The management of the state information and the coordination of bandwidth allocation required in the CAC framework are achieved along the (logical) token-ring of ingress nodes. Several bandwidth reservation tokens (BRTs) together with an accumulation procedure are used to provide both a low-latency, quasi-parallel allocation under light load conditions and a centralized/semaphorized allocation under heavy load conditions. The main targeted resource and consequently the dominant metric followed in path selection is the available bandwidth and the path selection procedure ensures that the path with the maximum available bandwidth is found. The presented all-pairs-widest-path search (APWPS) procedure incorporates intrinsically the number of hops constraint and can easily include other constraints (including delay and/or cost). The APWPS ensures global max-min fairness and is capable of keeping track of the distance to the (potential) optimum solution (and of the incremental cost) along the path searching procedure. APWPS avoids over-allocation (by keeping the spare capacity values in the non-negative domain), provides maximum resource utilization (when the applied constraints do not affect the selection of widest partial paths) and minimizes/postpones link blocking (as long as possible under the given constraints). Load balancing is provided by the inherent switch from the widest path once the reserved bandwidth changes the widest path status (i.e. other path might become the widest once bandwidth reservation is performed on the current widest path).

The APWPS performs at least as well as the WSP algorithm under the same constraints, with the advantage of keeping track of the distance to the (potential) optimum solution along the path search process and the disadvantage of an increased computational complexity. While various methods can be employed in reducing the computational complexity, the tracking of the distance to the (potential) optimum solution enables a more flexible approach in establishing the effectiveness of pursuing the optimum solution, accepting a non-optimum solution or terminating the search before the stop criteria are met. Furthermore, the APWPS is capable of finding the widest paths simultaneously for all the node pairs of the domain, while maintaining the same computational complexity as

for determining the hardest-to-find node-to-node widest path. Although this parallelism cannot be exploited effectively in the general case, there are situations (like massive re-allocation or group to group multicast) where APWPS's parallelism can be exploited in the search towards a globally optimum solution.

Further work is needed to provide a detailed analysis of the multiple-token state information management in a realistically complex network context. Also the properties and the operation of APWPS should be analyzed in real, complex network contexts (large number of nodes and links, arbitrarily distributed request bursts, etc.). Procedures to employ effectively the APWPS in massive re-allocation and group-to-group multicast problems should also be developed.

9. References

- [1] R. Braden, D. Clark, S. Shenker., "Integrated Services in Internet Architecture, an Overview," RFC1633, October, 1997.
- [2] S. Blake, D. Black., M. Carlson, E. Davies., Z. Wang, W. Weiss, "An architecture for Differentiated Services," RFC2475, Dec.1998.
- [3] I. Stoica, H. Zhang, "Providing Guaranteed Services without Per-flow Management", Proc.ACM SIGCOMM,1999, pp.81-94.
- [4] Z. Wang, J. Crowcroft, "Quality-of-service Routing for Supporting Multimedia Applications", IEEE J. on Selected Areas in Communications, vol. 14, no 7, Sept 1996, pp.1228-1234
- [5] S. Chen, K. Nahrstedt, "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions", IEEE Network, vol. 12, issue 6, Nov./Dec. 1998, pp. 64 – 79.
- [6] D.O.Awduche, L.Berger, D.Gain, T.Li, G.Swallow, and V.Srinivasan., "Extensions to RSVP for LSP Tunnels." draft-ietf-mppls-rsvp-lsp-tunnel-04.txt, September 1999
- [7] R.Guerin, D.Williams, and A.Orda., "QoS Routing Mechanisms and OSPF Extensions", Proc. of 2nd Global Internet Miniconference, (joint with GLOBECOM'97), November 1997
- [8] A. Dumitrescu and J. Harju, "Assuring fair allocation of excess bandwidth in reservation based core-stateless networks", Proc. of 28th Annual IEEE International Conference on Local Computer Networks LCN, Oct. 2003, pp. 64 – 70
- [9] A. Dumitrescu and J. Harju, "Delay management in core-stateless networks", Proc. of NEW2AN'2004, Feb. 2004, pp. 156-160.
- [10] D. Anderson, H.Balakrishnan, M.F. Kaashoek and R. Morris, "Resilient Overlay Networks", in Proc. of ACM SOSP, October 2001
- [11] S. Bhatnagar and B. Nath, "Distribution Admission Control to Support Guaranteed Services in Core-Stateless Networks", in *Proceedings* of INFOCOM 2003, vol.3, March-April 2003, pp. 1659-1669.
- [12] S. Bhatnagar, B. Nath, A. Acharya, "Distributed Admission Control for Heterogenous Multicast with Bandwidth Guarantees", 11th Intl Workshop on Quality of Service (IWQoS 2003), June 2003, pp. 115-136.
- [13] I. Stoica and H. Zhang, "LIRA: A model for service differentiation in the Internet", Proc. of NOSSDAV'98, July 98.