A New Localized Multi-Constraint QoS Routing Algorithm

Tran Minh Anh and Nguyen Chien Trinh
Posts and Telecommunications Institute of Technology, Hanoi, Vietnam
E-mail: anhtm.dng@vnpt.vn, trinhnc@ptit.edu.vn
Correspondence: Tran Minh Anh
Communication: received 3 June 2016, revised 31 May 2017, accepted 31 June 2017

Abstract: The scheme of Quality of Service (QoS) routing algorithms based on local state information has recently been proposed as an alternative approach to the traditional QoS routing algorithms. By implementing this localized QoS routing algorithm, each source node predetermines and maintains a set of candidate paths for each destination. These sets of paths will help the source node to decide the most appropriate path for a connection request. Hence, it helps to avoid the problems associated with the maintenance of the global network state information. In this paper, we propose a new and effective localized QoS routing algorithm, compare its performance with those of other localized algorithms and a traditional QoS routing algorithm under the same type of network topology, QoS requirements and traffic patterns. The simulations results show that our proposed algorithm can perform better than other routing algorithms.

Keywords: *Localized algorithm, packet delay, packet loss, quality of service (QoS), multi-constraint routing.*

I. Introduction

Nowadays, telecommunication networks develop very strongly and guaranteeing quality for large-scale networks becomes more and more difficult. Therefore, the approach of using local information about congestion probability, network statistics, etc., to build sets of paths for routing information is an effective way of communicating in a network. Because of using local information, this routing technique will decrease the average overall blocking probability of packets transmitted through the network, and make the overall performance of the network better than that by global QoS routing algorithms which update the network state periodically by a link-state algorithm and maintain it up-to-date. The way that global QoS routing algorithms do will lead to a large communication overhead, and inaccurate and out-of-date information of the global state due to large update intervals.

Localized QoS routing helps to avoid these problems by routing based on local information. It means that it performs flow routing using the localized view of the network QoS state. By this approach, each node maintains a predetermined set of candidate paths to each possible destination and routes flows along these paths. The localized routing algorithm will perform differently for each different type of sets of the candidate paths. Therefore, investigating efficient methods to build sets of paths for localized routing algorithms is an important task at present.

In this paper, we first survey some QoS routing approaches and related works. Then we consider the global QoS routing algorithms which need to periodically update the network state, and some other routing algorithms which use local information to route flows. We review the basic problem of QoS routing and then introduce a new routing algorithm which uses some local information such as bandwidth, delay and packet error rate for routing. This algorithm uses an index based on the probability of flow transmit success as a criterion to route as well. We also study the types of sets of paths (and some concerned notions), and use them in our proposed routing algorithm for implementing experiments. We compare the performance of the proposed routing algorithm with some other localized routing algorithms and the traditional global QoS routing algorithm – Widest Shortest Path (WSP) – under the same types of topology, traffic patterns and the same range of traffic loads. Besides, we analyze the impact of types of path sets on the performance of localized routing algorithm through the simulations.

The rest of the paper is organized as follows. Section II introduces some QoS routing approaches and some related works like PSR, BRB, LDCR, etc. Section III describes our novel routing algorithm with its flowchart and pseudocode, and studies the path set types. Section IV presents the simulation results which show that our routing algorithm can perform better than other routing algorithms. Finally, Section V gives conclusion remarks and address some future works.

II. QoS Routing and Related Works

In modern large-scale networks, providing quality guaranteed services is becoming a standard demand for most operators. QoS routing is currently the key concept to achieve such requirements.

In QoS routing algorithms described in [1–12] and references therein, finding a feasible path to satisfy the flow requirements demands some knowledge of the network state; this knowledge should be kept up-to-date and as accurate as possible. It is the job of routing protocols to make this information available for nodes when making routing decisions. A routing strategy consists of two basic tasks. The first task is to collect the link-state information and keep it up-to-date. The second task is to compute the most feasible path for each flow based on the collected state information. The efficiency of any routing strategy depends greatly on the way in which the network state information is collected and maintained. This means that maintaining network state information plays an essential role in any routing process.

In global QoS routing algorithms, nodes need to exchange network state information to get a global view of the network state. Based on the collected information and the current state information, a node can compute a feasible path for the current flow towards the intended destination. Many currently deployed routing algorithms follow this approach, and implement different methods to compute feasible paths. Global routing algorithms usually make a trade-off between the network state update and the accuracy of state information. They also tend to achieve optimal performance by balancing the traffic load across the network and efficiently utilizing available resources. One of the most popular global QoS routing algorithms is the WSP algorithm [2, 5]. WSP was proposed to improve minimum hop (*minhop*) algorithms. It tries to balance the traffic across the network by finding a feasible path with *minhop* count. If more than one path with the same hop count is found, the one with the largest residual bandwidth is selected. WSP is a link-constrained path-optimization routing algorithm as it finds the shortest feasible path. The widest path criterion is used only when there are several paths with the same hop count to choose from. WSP minimizes the usage of network resources by preferring shorter paths over longer ones, which leaves more resources to handle other flows. WSP will be used to compare with our proposed algorithm in Section IV.

Like the above WSP algorithm, the routing algorithm proposed in [9] also uses the Dijkstra algorithm proposed in [13] to choose the path for routing. It builds a new single mixed metric which contains some QoS metrics to evaluate whether a path satisfies for flow or not. It then operates as a modified version of the Dijkstra's shortest path algorithm. It is proved that it will gains a better result against some traditional routing algorithms as showed in [9]. But it must use the global information for computing feasible paths as Dijkstra algorithm requires. Hence, it still causes a large communication overhead as analyzed in Section I.

One more routing algorithm using multi-constraint criteria is QoS routing with Depth-First Search (DFS) [12]. Like the algorithm described in [9], DFS scheme is an source routing algorithm using global network state information. That means that DFS must infer the information from all the network for each demand of QoS attributes (QAs) coming to node. DFS first finds paths that are most likely to be feasible, and then explores these paths. By that way, DFS will help to diminish the Erroneous Decision Rate (EDR) down to a reasonable time limit. However, comparing to localized routing algorithms surveyed below, DFS must collect information from global network state information for each time of path computing like any other global algorithms, and thus it will commit some weak points like [9] as analyzed above, such as large communication overhead and inaccurate and out-of-date information.

All the QoS routing algorithms proposed so far require frequent exchange of QoS network state information, which adds considerable overhead and inaccuracy implications to the routing process. As previously discussed, localized QoS routing has recently been proposed as a viable alternative to global QoS routing [14–21]. In localized QoS routing, nodes make routing decisions using only local state information, that is, no global state information needs to be exchanged, thus reducing the overhead computation. It follows a completely different approach to compute feasible paths where each node constructs its own local view about the network and its available resources by monitoring and analyzing local feedback.

The basic concept of localized routing is that a source node infers the network state based on feedback and statistics it has collected locally. Routing decisions are then made with this localized view of the network. Each node is required to maintain a predetermined set of candidate paths to each possible destination. On the arrival of a flow request, the flow is routed along one of these candidate paths with the most feasible path among the path set.

We now survey some localized QoS routing algorithms. The first algorithm discussed here is the Proportional Sticky Routing (PSR) algorithm [16–19]. The main idea behind PSR is that every source node predetermines and maintains a set of candidate paths, P, created by the set of *minhop* paths, $P_{\min}$, and the set of alternative paths, P_{alt}. Then, based on statistics of the number of blocked flows and their flow blocking probability available to it, the source node

distributes proportionally the traffic load to a destination among the candidate paths in this predetermined set.

The PSR algorithm works in observation periods with varied-length cycles. In each period, based on the flow proportion and the flow blocking probability information, the source node selects path for routing flows. The more the number of times a path is selected, the higher its proportion is given, which affects the next selection for flow-in. At the source node, the set of eligible paths, P_{eli}, is set at first based on the set of candidate paths. For each cycle, the flow-in can be routed among paths from P_{eli}. A parameter γ_r, namely flow blocking, is set to determine the number of times of blocking a connection request of a path. If a path has the number of times of blocking in excess of γ_r, that path becomes ineligible. When all the paths P_{eli} become ineligible, the cycle will end and the next cycle begins. PSR selects a path based on its flow proportions as mentioned above. The more the number of times the path is chosen, the higher the chance of selection.

After each observation period, the *minhop* path flow proportions are adjusted to equalize the blocking probability among paths. For the alternative paths with (*minhop* + 1), the minimum blocking probability among the minimum hop paths is also used to control their flow proportion. After that, another period begins. With the proportional distribution like that, the PSR algorithm helps to better balance the network, and more efficiently utilize the network than traditional algorithms. However, PSR delivers the load proportionally to the paths in the determined set. So, it may commit a high congestion when the load is too fragmented at the destination.

Another QoS routing algorithm using local information is Best Reserved Bandwidth (BRB) routing, proposed in [20]. This algorithm also uses the set of candidate paths like PSR, determines and maintains the set of *minhop* and (*minhop*+1) paths from the source to the destination at each node. The main idea behind the BRB algorithm is that it first reserves bandwidth for all the links of a candidate path. Then, it compares among these values of all the paths after the reservation. When a flow is arriving, BRB chooses a path for it, with the maximum value of residual bandwidth in any link in the path. We can see that the BRB algorithm has reserved all the paths for the flow-in. If that value of residual bandwidth satisfies the flow QoS requirements, the path is accepted for flow and the algorithm will reserve for the next flow. If the link does not satisfy the QoS requirements of that flow, bandwidth will be released and returned to the previous links.

To have the path which has the highest minimum residual bandwidth link, the BRB algorithm will continuously compare the value of minimum residual bandwidth in each path among the set of candidate paths. The path that has the largest value will be selected for the next incoming flow. This procedure of BRB helps it to have a good result, but the network can become congested when there are too many paths between the pair of source and destination node, and when there is a large load along these paths.

Using end-to-end delay as a constraint, other localized routing algorithms have recently been proposed in [21], one of which is the Localized Delay-Constrained QoS Routing (LCDR) algorithm. Like other localized routing algorithms, LDCR needs all source nodes to predetermine and maintain the set of candidate paths. Being a source routing algorithm, LDCR requires a source node to select from the candidate path set a path for flows. The path which has the best quality is selected by LDCR to route flow if it satisfies the flow QoS requirements.

To specify the best path, the LDCR algorithm uses the value of average end-to-end delay of that path to decide whether the path is good for routing. It collects all information (blocking probabilities, flow blocking, etc.) at the source node itself. Besides, the value of end-to-end delay of each candidate path is the main metric for comparison. Based on that, the path with the lowest value of average end-to-end delay is selected to route that flow, and that average end-to-end path delay is a constraint for estimating path quality. After a flow is routed, LDCR updates the average delay of that path, and the loop continues with this value of delay. Thanks to this mechanism, LDCR can assure the end-to-end QoS delay for users and optimize the network performance. However, at the same time it causes a higher probability of congestion.

The localized routing algorithms above have some advantages including low time complexity, low flow blocking probability, etc., but they also commit some disadvantages as analyzed above. Next, we will propose a new localized routing algorithm with improved aspects.

III. Proposed QoS Routing Algorithm

1. Methodology

QoS routing uses network state information and resource availability, in addition to the QoS requirements, to meet such demands. QoS routing algorithms that select paths with sufficient residual resources to meet the QoS constraints have played a critical role in meeting the required service level. Furthermore, they should also compromise between resource utilization and network traffic balancing in order to achieve high routing performance.

QoS requirements of a network flow are given as a set of constraints that the routing algorithm should meet to find out a feasible path with path metrics. Metrics

TABLE I
CLASSES OF SERVICES AND QOS STANDARDS

Class Application/Examples	Mean Delay upper bound	Delay variance	Loss Ratio
Class 0 Real-time, jitter-sensitive, high interaction (VoIP, Video Teleconference)	100 ms	< 50 ms	1.00×10^{-3}
Class 1 Real-time, jitter-sensitive, high interaction (VoIP, Video Teleconference)	400 ms	< 50 ms	1.00×10^{-3}
Class 2 Highly interactive, transaction data (e.g signaling)	100 ms	Unspecified	1.00×10^{-3}
Class 3 Interactive transaction data	400 ms	Unspecified	1.00×10^{-3}
Class 4 Low loss only (Short transactions, bulk data, video streaming)	1 s	Unspecified	1.00×10^{-3}
Class 5 Default IP networks applications	Unspecified	Unspecified	Unspecified

define the type of QoS guarantees that the network can support. No requirements can be satisfied if they cannot be mapped onto one metric or a combination of metrics. QoS metrics are generally divided into three types, namely *additive*, *multiplicative* and *concave* (or *convex*), which are defined below.

Consider a network (N, L), where N is the set of nodes and L is the set of links in the network. Denote by L_{ij} the link from node i to node j. A path that consists of s links from node a to node b is symbolized as $P_{ab} = (a, p_1, p_2, \ldots, p_{s-1}, b)$, with $s > 1$. When node a connects directly to node b, $s = 1$ and $P_{ab} = (a, b)$. Denote by $w(i, j)$ the weight of link L_{ij} corresponding to the metric of QoS that can be used as a constraint in routing algorithm.

Then, the weight of the whole path, denoted by $w(P_{ab})$, is defined, respectively for metric of additive, multiplicative, concave or convex type, as:

$$w(P_{ab}) = w(a, p_1) + w(p_1, p_2) + \cdots + w(p_{s-1}, b), \quad (1)$$

$$w(P_{ab}) = w(a, p_1) \times w(p_1, p_2) \times \cdots \times w(p_{s-1}, b), \quad (2)$$

$$w(P_{ab}) = \min\left[w(a, p_1), w(p_1, p_2), \ldots, w(p_{s-1}, b)\right], \quad (3)$$

$$w(P_{ab}) = \max\left[w(a, p_1), w(p_1, p_2), \ldots, w(p_{s-1}, b)\right]. \quad (4)$$

In reality, bandwidth is a metric of concave type. It means that the bandwidth of P_{ab} (used in comparison in the next section) is the minimum bandwidth of all links which take part in that path. Delay, cost and hop count are metrics of additive type, and reliability is a metric of multiplicative type.

A QoS routing algorithm usually uses one or more of the above metrics as constraints for the routing problem. If it uses only one metric, it will be a mono-constraint routing algorithm, which compares the value of only one QoS metric of flow-in to the value of that metric from the network. For example, a bandwidth-constrained routing finds a path whose bottleneck bandwidth can satisfy the required bandwidth value. If it uses more than one metric, it will use the results from comparisons of all those metrics between demands from service and support from the network. For example, a bandwidth-delay constrained problem finds the path that meets the required bandwidth and has the least delay which is better than the required value of flow-in delay.

2. Requirements for QoS Routing

Nowadays, new telecom services need high level of QoS parameters such as interactive services, multimedia, Video on Demand (VoD), etc. Therefore, network providers must assure the quality of the network for customers who use these high quality services, support the increasing demands by applications and provide different classes of services to meet diverse QoS requirements. These service requirements impose strict constraints on transporting networks to ensure end-to-end performance guarantee.

The QoS requirements are in a set of constraints for which a routing algorithm must satisfy to find out feasible paths for flows. A QoS parameter specifies types of network quality (i.e., bandwidth, delay, etc). If there is no QoS requirement, there is no corresponding constraint for routing.

To analyze the characteristics of services and their QoS requirements, the ITU-T, in its Recommendation Y.154x [22], has proposed the requirements for QoS standards as given in Table I.

We can see that the telecom networks are now carried out with a large mix of traffic and various QoS requirements. Therefore, QoS routing has to select paths which satisfy QoS requirements. For a very large and complex network, path selection is much complicated and challenging, because multiple QoS constraints must be met while ongoing

dynamic changes in the network resources and topology must be kept track. It is the responsibility of routing algorithms to tackle these problems while achieving high routing performance. Among various QoS performance metrics, bandwidth, delay and packet loss are currently the most significant ones [22]. Although using many metrics will make computation become a burden, it will help make routing more flexible and effective. Hence, in this paper, we use bandwidth, delay and packet-error-rate metrics from local information in our algorithm. Accordingly, we call the proposed QoS routing algorithm, to be described next, Bandwidth-Delay-PacketErrorRate (BDP).

3. Proposed BDP Algorithm

The proposed routing algorithm– BDP– also requires every nodes to maintain a predetermined set of candidate paths to each possible destination, and keeps track of only one set of candidate paths, R, (the result of finding shortest paths). Previous localized QoS routing algorithms use different information for routing. In particular, BRB reserves the whole path for the current flow and the next flow deals with the residual bandwidth after the reservation in any link in the path. PSR uses the number of blocked flows as the only available information at the source node. LDCR relies on the average end-to-end path delay in order to make routing decisions, update the average delay of the selected path, which directly reflects the actual quality of the path. Unlike the above algorithms, our BDP algorithm uses the values of bandwidth, delay and packet-error-rate from local node information to be criteria for choosing path. Hence, BDP becomes more flexible than BRB, LDCR or PSR, which will be used for later performance comparison.

In BDP, we propose to associate every path $P \in R$ with a variable $P.Criteria[1, 2, 3] = P.[Bandwidth, Delay, PER]$, and a path index, Pt_{index}. The way to build Pt_{index} will be discussed later. The set R will be sorted for the index Pt_{index}, denoted by $R(Pt_{\text{index}})$.

Upon flow arrival, BDP performs routing by taking the following steps:

1) Select path P with $\max(Pt_{\text{index}})$ from the set R (first place in the set R, after sorting).
2) Check the demand of the flow from Service Level Agreement, SLA, and use it for choosing path. Denote $SLA.[Bandwidth, Delay, PER]$, or SLA.
3) Compare $P.Criteria$ with SLA.
   - If $P.Criteria \geq SLA$, then choose this P. The way to compare will be discussed in Section III.5.
   - If $P.Criteria < SLA$: select the next P from the set R (the $\max(Pt_{\text{index}})$ of the rest of R). The loop will be exited when finding out the path which has $P.Criteria \geq SLA$. If no path has quality satisfying SLA, the arriving flow is cancelled, obviously.
   - Increase Pt_{index} if route P is accepted, or decrease Pt_{index} otherwise.

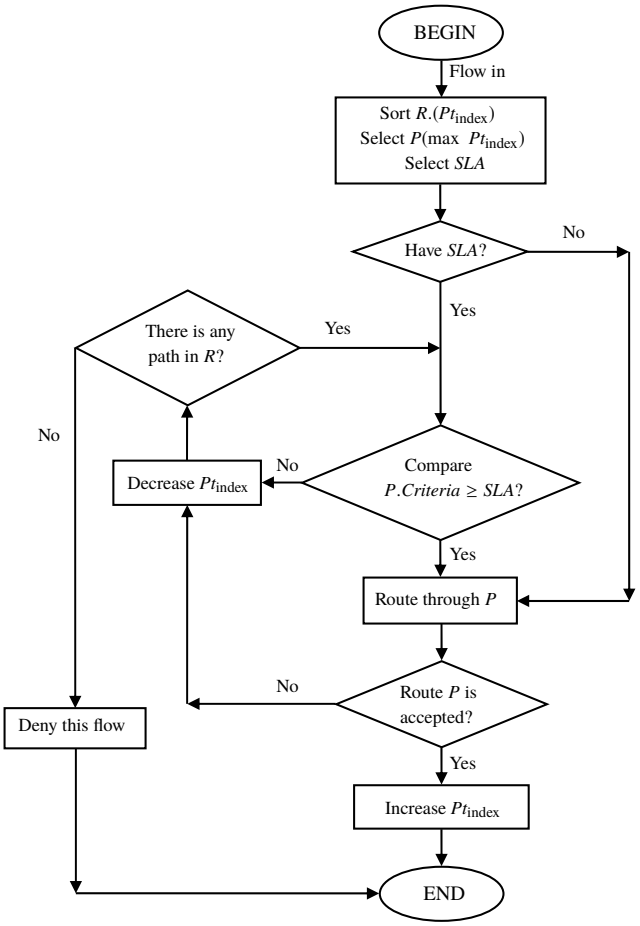

Figure 1. Flowchart of BDP.

The flowchart of the BDP algorithm is illustrated in Figure 1 and its pseudo-code is given in Algorithm 1.

Unlike other localized routing algorithms, BDP only changes the index Pt_{index} of this path. When the flow is accepted or unaccepted along the selected path, Pt_{index} is updated increment or decrement accordingly.

The path index Pt_{index} is initially set to 1 at first. When the path is selected and $P.Criteria > SLA.[Bandwidth, Delay, PER]$, indicating that the candidate path has good (or bad) quality, we increase (or decrease) Pt_{index}. After that, the flow that follows will use this value of Pt_{index} as criterion to choose path, and the next loop re-begins. Mathematically, if the flow is transmitted well through a path, then

$$Pt_{\text{index}} = \text{Last } Pt_{\text{index}} + \frac{1}{\sum \text{times be selected}}. \quad (5)$$

Algorithm 1: Pseudo-code of BDP algorithm

```
Initialize
Building
Set Pt_index = 1, ∀ P ∈ R; Set T = 100
procedure BDP
    Range R{Pt_index} for flow-in
    Set P.success = false
    Collect SLA for flow-in
    P = first {P.Pt_index : P ∈ R}
    while !(P.success or R(end)) do
        if (P.Criteria ≥ SLA) then
            Route flow along path P
            if P is accepted then
                Compute P.Pt_index (increase Pt_index(≤ 1))
                P.success = true
            else if P = next{P.Pt_index : P ∈ R} then
                Compute P.Pt_index (decrease Pt_index(≥ 0))
            end if
        else
            P = next {P.Pt_index : P ∈ R}
            Compute P.Pt_index (decrease Pt_index(≥ 0))
        end if
    end while
end procedure
```

Otherwise, if it is discarded, then

$$Pt_{\text{index}} = \text{Last } Pt_{\text{index}} - \frac{1}{\sum \text{times be selected}}, \quad (6)$$

where $0 \le Pt_{\text{index}} \le 1$. After the above calculation, if Pt_{index} becomes greater than 1 then set it to 1, or if smaller than 0 then set to 0. Once the path has been selected more than T times (T is pre-determined), the path increases $1/T$ for each time being chosen, and decreases $1/T$ for each time being discarded.

Moreover, the value T will be set first in accordance with the ability of the network. A higher T gives more accurate Pt_{index}. Typically, we set $T = 100$ and hence the interval becomes 0.01. It means that when the path P is selected more than 100 times, each time of being selected after that, Pt_{index} increases 0.01 (but not exceeding 1), or decreases 0.01 (not below 0) vice versa. Therefore, after one flow has been processed, Pt_{index} changes accordingly to the success/fail of that path, and the "value" of that path changed correspondingly. It affects the probability of being chosen for the next as well.

4. Static and Dynamic Sets of Paths

This section will present some notions which concern path sets mentioned above. In localized routing algorithms, path sets play a big role to route flow along.

1) Depth of connection:

Consider a network $G(N, L)$, where N is the number of nodes and L is the number of links. Consider two nodes i and j in $G(N, L)$ and assume that there are many connectable paths between them. Using the Dijsktra algorithm, we find out easily the shortest path between i and j. Denote by $minhop_{ij}$ the minimum of hops of that shortest path, and by d the depth of connection which decides the length of that path.

Let r be the number of paths between i and j that have the number of hops not greater than $(minhop_{ij} + d)$. The set of these r paths is denoted by $R_{ij}^d = R_{ij}^d(1, \ldots, r)$, which is just the predetermined set of candidate paths used in localized routing. We note the following:

- The connection (or path) does not contain any repeated node;
- $minhop_{ij} \ge 1$;
- For $0 \le d \le (N-2)$: When $d = 0$, we have the set R_{ij}^0 as the set of shortest paths between i and j. When d increases, the set R_{ij}^0 enlarges the number of paths as well.

2) Using d to build set of paths:

A node will use network information to build all sets of paths between all pairs of two nodes with the fixed value of the depth of connection d. In the BDP algorithm, we set $d = 0$ at the beginning, because we do not have any information about the load in the network. Accordingly, this value of d is used to first build the sets of paths. Then, under some concrete conditions (as mentioned next), the node will collect new information from the global network to rebuild the set of candidate paths between that pair with a new depth of connection: $d_{\text{new}} = d_{\text{old}} + \alpha$, with α is an integer. This update will enlarge the set, and thus help achieve better routing.

3) Static path set:

If a localized routing algorithm, e.g. BDP, uses one fixed value of d to build the set of paths, then the path set is said to be static. In that case, the algorithm works without changing d, that is $\alpha = 0$ and hence $d_{\text{new}} = d_{\text{old}}$.

4) Dynamic path set:

If d is not set unchanged, then the path set is said to be dynamic, and its size changes depending on real condition of the network. The algorithm will rebuild the path set with changing the depth of connection d. In this case, $d_{\text{new}} > d_{\text{old}}$, because $\alpha \ne 0$. With $\alpha = 1$, typically, we have $d_{\text{new}} = d_{\text{old}} + 1$. This will assure better satisfaction for demands of users.

5) Impact of path set type on the performance:

Clearly, we can confirm that a dynamic set of paths will make the algorithm more flexible than a static one (in case

of no new node or no shutdown node). In spite of using the same condition of rebuilding, using a dynamic set allows the algorithm to enlarge the path set, i.e. to use more paths (although they are "longer", they are "possibly better") and this will give better performance.

An important thing to be discussed next is the condition of rebuilding. The condition must be designed in compliance with each algorithm. For BDP, we propose the following: if half of path set has $Pt_{\text{index}} = 0$ (i.e., half number of paths has relatively "weak" quality), then the set of paths should be rebuilt. We will see the effect of rebuilding in the next section, where at one node i there are largely different values of d, corresponding to each destination in the network. The comparison of static and dynamic path sets will be shown by simulation later, and we will see their impact on the operation of the localized routing algorithm.

5. Metric Selection and Comparison

As previously mentioned, we choose bandwidth, delay, and bit(packet)-error-rate as the three main metrics for comparison to choose paths. There are some ways to compare, but the relevant and popular way will be discussed below.

We make the comparison of three metrics independently. First, we compare the bandwidth of the path with the demanded bandwidth of the flow-in. If it is true, we then compare the delay of that path with the demanded delay for that flow. Finally, we compare the value of bit(packet)-error-rate of the path with the value of the flow request. If we have three values being true, the path is proved to have "enough" quality for flow, and thus will be chosen. It is noted that $P.Bandwidth \geq F.Bandwidth$, $P.Delay \leq F.Delay$ and $P.PER \leq F.PER$, where *Bandwidth* is determined as the minimum residual bandwidth of any link on path, *Delay* is the sum of all delay of propagation from all links on this path, and *PER* is the sum of all PER on all links of path, as illustrated in Algorithm 2.

Algorithm 2: Comparison with Three Constraints

```
procedure COMPARE3(p, f)
    if p.Bandwidth ≥ f.Bandwidth then
        if p.Delay ≤ f.Delay then
            if p.PER ≤ f.PER then
                p is chosen
            else
                p is discarded
            end if
        else
            p is discarded
        end if
    else
        p is discarded
    end if
end procedure
```

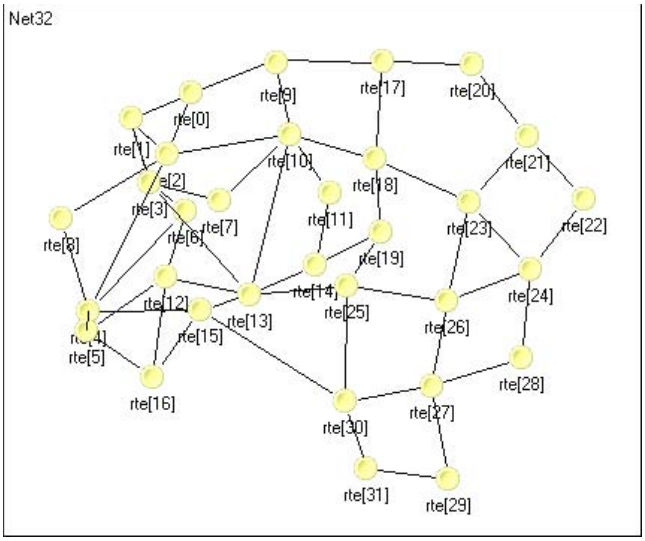

Figure 2. Network topology of 32 nodes.

IV. Performance Evaluation

In this section, we study the performance of the BDP algorithm (with dynamic and static path sets) and compare it with the performance of the BRB, LDCR and PSR algorithms. Besides, we also compare it with the popular global QoS routing algorithm, WSP, which searches for a feasible path with shortest path. All the experiments will be set under the same condition.

1. Simulation Model

We use `OMNeT++` [23] which is a commonly used simulator. To evaluate the performance, we collect all of simulation parameters as vectors, scalars and histograms to compare. The setup of simulation experiments is the same as in [16–21]. In particular, the network is built with 32 nodes. Network links are all bidirectional with the same capacity $C = 150$ Mbps in each direction, the same value of delay $D = 20$ ms, and packet error rate $PER = 200 \times 10^{-6}$. Flows arrive to each source node according to a Poisson process with rate λ. Destination nodes are selected randomly (each node can be source or destination). The flow duration is exponentially distributed with mean $1/\mu$, flow bandwidths are uniformly distributed within [0.5–4] MB. The required value set for the delay of each packet is randomly distributed between 20 ms and 250 ms, and the *PER* of each packet as 1.000×10^{-3}. The network topology is shown in Figure 2.

As analyzed in [24, 25], the offered network load is

$$\rho = \frac{\lambda N b h}{\mu L C},$$

where N is the number of nodes, b is the average bandwidth required by a flow, h is the average path length (in number of hops) and L is the number of links in the network. In the experiments, we set $N = 32$, $L = 108$, $h = 3.137$, $1/\mu = 60$ s. Since the performance of routing algorithms may vary across different load conditions, our simulation experiments consider several load conditions by varying the value of λ to create experiments with load from low to high.

To compare the performance, we choose the flow blocking probability, P_{block}, as the performance criterion, similar to [16–19]. The flow blocking probability is calculated based on the most recent 100.000 flows. The time of simulation is set about of 20 minutes, which corresponds to more than 2.5 million of flows emitted. The flow block probability is calculated as

$$P_{\text{block}} = \frac{|B|}{|T|}, \tag{7}$$

where $|T|$ is the total number of flows and $|B|$ is total number of blocked flows.

We also calculate the overall end-to-end delay of the network when using the BDP algorithm with both types of path sets and compare with WSP under low and high load scenarios.

2. Simulation Results

Figure 3 shows the performance of BDP (with dynamic path set) against BRB, LDCR, PSR and WSP in terms of flow blocking probability under load ρ, ranging from 0.2 to 0.5. We can see that, under a low load ($\rho \leq 0.25$), the difference in the performance among the routing algorithms is rather small, because finding available path with sufficient bandwidth is easy and flows are almost accepted.

When the load is high ($\rho > 0.3$), some differences reveal. Many flows drop or/and fail to reach the destination node, then the flow blocking probability grows rapidly. The reason is in the case of the BDP algorithm, paths are selected based on the probability of predetermined paths, then the index of that path increases, assuring that this path is good and may support relatively the next flow. It means that it has all-ready available quality in the links on the path selected. Unlike BRB, BDP uses delay and PER as criteria to compare, so all chosen flows almost satisfy the demand of delay and PER at destination. It helps diminish the flow blocking probability of that path and the value of end-to-end delay of those flows.

Moreover, the index value which is set for choosing path helps to avoid the congestion of flows that come at nodes

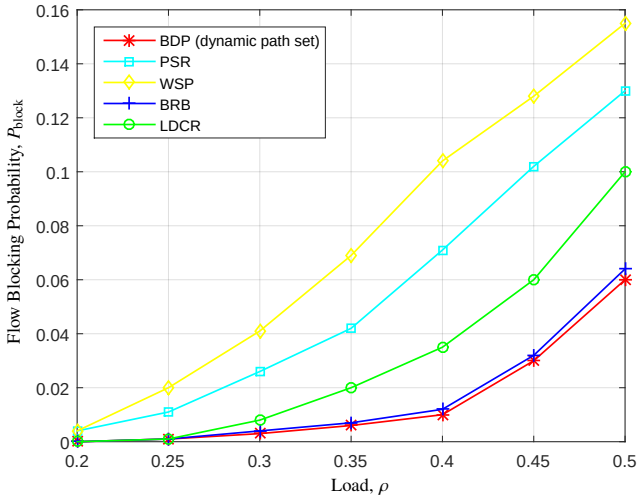

Figure 3. Performance in terms flow blocking probability.

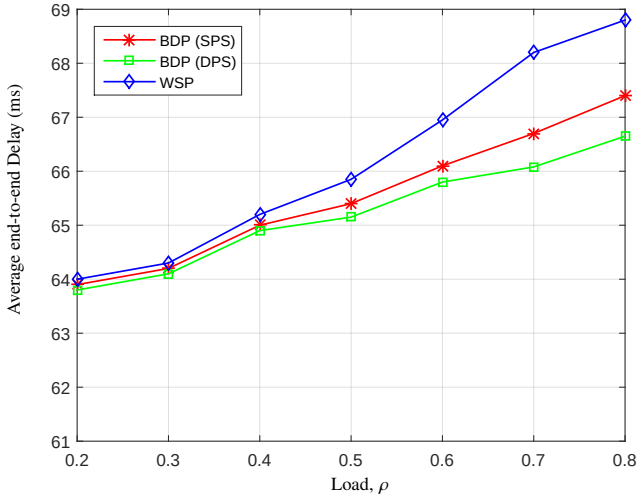

Figure 4. Performance in terms of average end-to-end delay of flows.

simultaneously, particularly when the load increases and the links begin to become congested. If congestion happens, flows will be redirected to other paths and the index decreases at once. Then, the source node might reduce the use of these paths which have low index values. Therefore, the flow blocking probability of BDP is considerably lower than that of BRB, LDCR, PSR and WSP.

In the next experiments, we compare the performance of BDP, with dynamic path set (DPS) and static path set (SPS), with that of WSP (using Dijkstra algorithm with weighted links to route). We evaluate the performance in terms of the average end-to-end delay under different load ($\rho = 0.2$ to 0.8), as showed in Figure 4. It can be seen that as the load augments, the value of end-to-end delay increases as well. When $\rho < 0.4$, the three algorithms perform similarly, with an end-to-end delay of about 64 ms and 65 ms. When

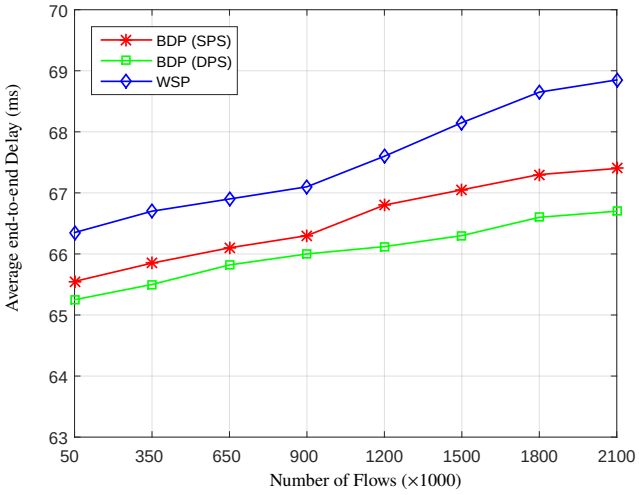

Figure 5. Performance in terms of average end-to-end delay of flows at fixed load $\rho = 0.8$.

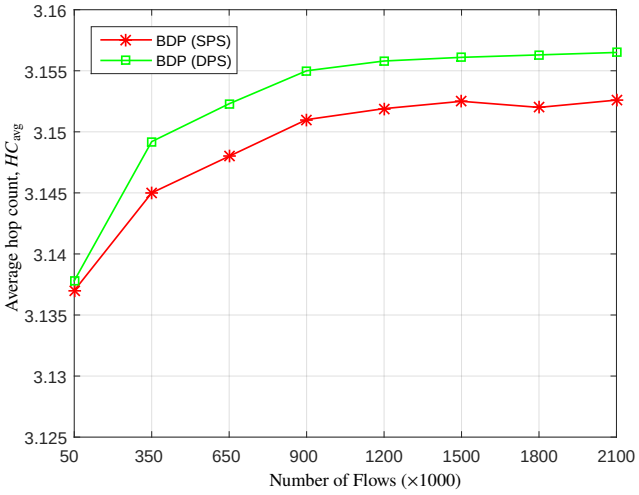

Figure 6. Performance in terms of average hop count at $\rho = 0.4$.

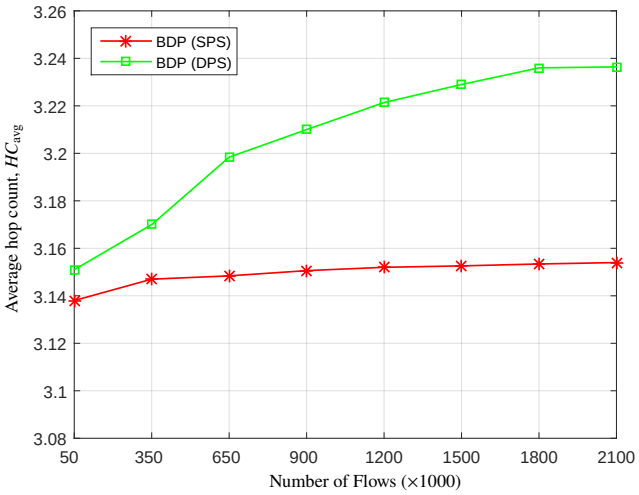

Figure 7. Performance in terms of average hop count at $\rho = 0.8$.

$\rho > 0.4$, the BDP algorithm, in both cases of DPS and SPS, works more efficiently.

In Figure 5, we evaluate the end-to-end delay performance at a fixed load, $\rho = 0.8$. It can be seen that when the number of flows increases in the load ρ of 0.8, the average end-to-end delay increases gradually until a stable value. In our two cases of using BDP, with both dynamic and static path sets, the average end-to-end delay is high but kept in a margin of 66.3–67.3 ms, as compared to the margin of 68–68.8 ms in the case of WSP. It means that with high load, the congestion happens more frequently, and hence this value increase highly, especially in the case of WSP. In the case of BDP, the flows must change path frequently based on the index Pt_{index}. When congestion happens, the index of the regular path diminishes, then, the path is changed. Therefore, the average end-to-end delay is better.

Next, we will observe the change of the average hop count, HC_{avg}, when using the BDP algorithm, with both static and dynamic path sets, under low load ($\rho = 0.4$) and high load ($\rho = 0.8$). Under a low load at $\rho = 0.4$, both cases show only a small change of HC_{avg}, as shown in Figure 6. They both starts around 3.137 (which is the average path length h) and the static path set achieves its maximum of around 3.156. The change is mainly from "short" paths (with 1 or 2 hops) or from path set which has one or two paths between a pair of source-destination nodes. On the other hand, the use of "long" paths (with more than 4 hops) which happen frequently has made HC_{avg} longer than the average path length of the network.

At a high load of $\rho = 0.8$, the difference between the two cases is significant, as shown in Figure 7. The average hop count when the dynamic path set is used is high because it must rebuild the path set many times when it commits congestion. When the load is high, the congestion happens more frequently and it makes the routing algorithm rebuild more times. That leads to the change of the path length, as well as the average hop count. When the static path set is used, HC_{avg} is almost kept around 3.15, slightly greater than the average path length ($h = 3.137$) and achieves its maximum of around 3.238.

We now observe the change of the depth of connection, d, after the experiments (which use BDP with the dynamic path set) have run for 20 minutes. Generally, we observe the value d of 31 pairs from source node 0 (viewed in Figure 2) to the other 31 destination nodes. First, we take the value of *minhop* between each pair (it means at $d = 0$). After 20 minutes of operation, we pause all the processes and record the values of all d of all pairs from source node

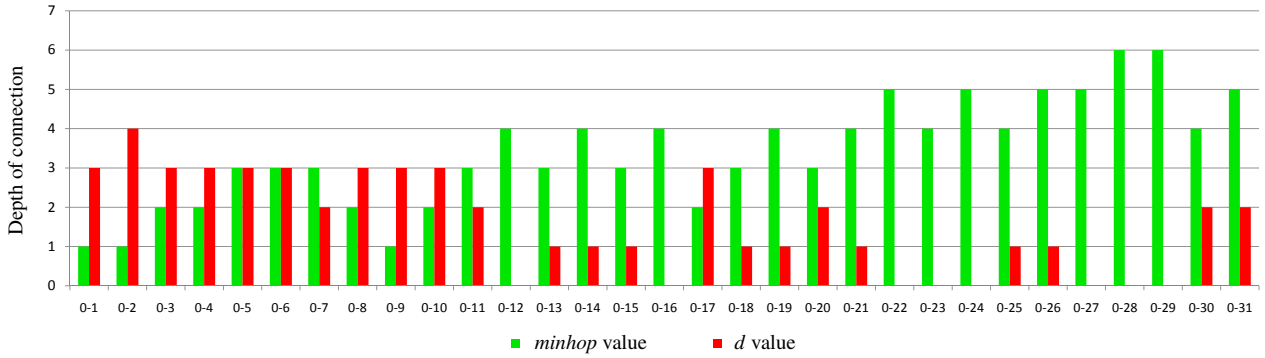

Figure 8. Depth of connection from path set of node 0.

0. The simulation result is shown in Figure 8.

It can be seen that the pairs, which have *minhop* equal to 1 or 2, tend to augment the depth of connection d. Conversely, the long paths (with high *minhop*) or pairs which have lots of paths connected among them do not often change d. In other words, they do not often commit congestion. Clearly, with the mechanism of changing the depth of connection like dynamic path set, we can see that the bandwidth blocking probability goes down. That involves the better quality of the network. Therefore, we can have better performance than the case when we use the static path set or other case such as WSP in some experiments which have been done.

Overall, BDP (with both types of path sets) has better performance than other state-of-the-art algorithms (BRB, LDCR, PSR or WSP) in the simulation experiments.

3. Complexity and Overhead

The overhead of global QoS routing can be attributed to two factors: (i) the computation complexity required to find feasible paths, and (ii) the excessive signaling overhead resulting from the exchange of network state information. And as analyzed in [1–3], the global QoS routing algorithm like WSP, which uses Dijkstra algorithm, takes at least $O(N \log N + E)$ time, where N is the number of nodes, and E is the number of links (edges) from that network.

Localized routing algorithms select path from the predetermined set of candidate paths R whose size is $|R|$. In the PSR algorithm [16–19], the path selection is an invocation of a Weighted-Round-Robin like Path Selector (WRRPS), whose worst-case time complexity is $O(|R|)$. The BRB and BDP algorithms require order of better than that, because they use only the shortest paths as explained above. In addition these localized routing algorithms require updating information, which takes a constant time $O(1)$.

Hence, regarding the communication overhead, BDP and other localized routing algorithms require very little over and above computing the blocking probability based on acceptance or rejection of a path, while at the same time, global algorithms require a huge amount of overhead to keep the link state information updated. In conclusion, the computation of our case at source node is much smaller than the traditional WSP.

V. Conclusion and Future Works

In this paper, we proposed a new localized QoS routing algorithm, BDP, to choose paths using only flow information collected locally at the source node. We have done simulation experiments to compare the performance of BDP (with dynamic path set and with static path set), with the BRB, LDCR, PSR and WSP algorithms. These experiments have already shown a better performance of BDP which uses three parameters of QoS to be criteria with better time complexity and very low communication overhead. We have also proposed a flow chart and a pseudo code of the BDP algorithm with bandwidth, delay and packet-error-rate metrics to route flows through the network. We collect the local information of blocking probability to update the path index. This index directly decides the routing, hence makes better quality of routing, and hence better working of the network.

It is of interest to investigate the effect of using more QoS parameters at nodes, especially when the telecommunication services require very high quality. From that, we will adjust the algorithm to make routing more flexible. Another possible development of this algorithm is to investigate network balancing in building routing policies. The algorithm should take care of it, and the balancing factor should be considered as a new parameter in routing. Finally, it is of interest in the way of building sets of candidate paths, based on knowledge of network as well as reduction of computing at nodes. With more sophisticated selection of set of paths, the algorithm will operate more effectively and more reliably.

REFERENCES

[1] C. Pornavalai, G. Chakraborty, and N. Shiratori, "QoS based routing algorithm in integrated services packet networks," in *Proceedings of International Conference on Network Protocols*. IEEE, 1997, pp. 167–174.

[2] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: problems and solutions," *IEEE Network*, vol. 12, pp. 64–79, 1998.

[3] ——, "Distributed quality-of-service routing in high-speed networks based on selective probing," in *Proceedings of 23rd Annual Conference on Local Computer Networks (LCN'98)*. IEEE, 1998, pp. 80–89.

[4] F. A. Kuipers and P. F. Van Mieghem, "Conditions that impact the complexity of QoS routing," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 717–730, 2005.

[5] R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *Global Telecommunications Conference*. IEEE, 1997, pp. 1903–1908.

[6] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[7] R. A. Guerin and A. Orda, "QoS routing in networks with inaccurate information: theory and algorithms," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 350–364, 1999.

[8] Q. Ma and P. Steenkiste, "Quality-of-Service Routing for Traffic with Performance Guarantees," in *IFIP International Workshop on Quality of Service*, 1997, pp. 115–126.

[9] P. Khadivi, S. Samavi, T. Todd, and H. Saidi, "Multi-constraint QoS routing using a new single mixed metric," in *IEEE International Conference on Communications*, 2004, pp. 2042–2046.

[10] A. Kulhari, C. Gandhi, and A. Jaiswal, "Multi-constraint Multipath QoS Routing Using Swarm Intelligence," in *International Conference on Computational Intelligence and Communication Networks*. IEEE, 2014, pp. 468–471.

[11] X. Wang, S.-H. Yu, J. Dai, and T. Luo, "A Multiple Constraint Quality of Service Routing Algorithm Base on Dominating Tree," in *International Conference on Computational Intelligence and Software Engineering (CiSE 2009)*. IEEE, 2009, pp. 1–4.

[12] D.-W. Shin, E. K. Chong, and H. J. Siegel, "A multi-constraint QoS routing scheme using the depth-first search method with limited crankbacks," in *IEEE Workshop on High Performance Switching and Routing*. IEEE, 2001, pp. 385–389.

[13] "Dijkstra Algorithm." [Online]. Available: https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html

[14] N. C. Trinh and T. M. Anh, "About a new localized multi-criterion routing algorithm–a solution to assure quality of network," in *Proceedings of the International Conference on Communications, Management and Telecommunications (ComManTel)*. IEEE, 2015, pp. 223–228.

[15] ——, "Dynamic set of paths for localized QoS routing with bandwidth-constraint," in *Proceedings of the International Conference on Electronics, Information, and Communications (ICEIC)*. IEEE, 2016, pp. 1–6.

[16] S. Nelakuditi, Z.-L. Zhang, and D. H. Du, "On selection of candidate paths for proportional routing," *Computer networks*, vol. 44, no. 1, pp. 79–102, 2004.

[17] S. Nelakuditi, Z.-L. Zhang, R. P. Tsang, and D. H.-C. Du, "Adaptive proportional routing: a localized QoS routing approach," *IEEE/ACM Transactions on networking*, vol. 10, no. 6, pp. 790–804, 2002.

[18] S. Nelakuditi and Z. L. Zhang, *Localized Quality of Service Routing for the Internet*. Kluwer Academic, June 2003.

[19] S. Nelakuditi and Z.-L. Zhang, "On selection of paths for multipath routing," in *International Workshop on Quality of Service*. Springer, 2001, pp. 170–184.

[20] T. A. Al Ghamdi and M. E. Woodward, "Novel localized QoS routing algorithms," in *IEEE 9th Malaysia International Conference on Communications*, 2009, pp. 199–204.

[21] F. M. Aldosari and F. Alradady, "Localized QoS routing with end-to-end delay guarantees," in *Tenth International Conference on Information Technology: New Generations (ITNG)*. IEEE, 2013, pp. 464–472.

[22] ITU, *Internet protocol aspects – Quality of service and network performance*, ITU-T Std. Y.1500–Y.1599, 1999. [Online]. Available: http://www.itu.int/ITU-T/recommendations/index.aspx?ser=Y

[23] A. Varga, "The OMNeT++ Discrete Event Simulation System, the European Simulation Multiconference," Prague, Czech Republic, 2001.

[24] A. Shaikh, J. Rexford, and K. G. Shin, "Load-sensitive Routing of Long-lived IP Flows," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '99. ACM, 1999, pp. 215–226.

[25] J. L. Rexford and A. Shaikh, "Efficient precomputation of quality-of-service routes," Oct. 14 2003, US Patent 6,633,544.

Tran Minh Anh received his B.Eng. degree in Electronics and Telecommunication Engineering from Danang University of Technology in 1995. In 2001, he received his M.Sc. degree in Telecommunication Engineering from Hanoi University of Science and Technology. His research interests include New Generation Networks (NwGN) Trends, telecom engineering, network load balancing, network routing techniques.

Nguyen Chien Trinh is a Professor of Electro-Communication Engineering at the Posts and Telecoms Institute of Technology (PTIT), Hanoi, Vietnam. He received his B.E. degree from the University of Electro-Communications, Odessa, Ukraine in 1989, M.Eng. and Ph.D. degrees in University of Electro-Communications, Tokyo, Japan, in 1999 and 2005, respectively. His research interests include new generation networks (NwGN) technologies, network traffic analysis and modeling, network resource allocation, network QoS, and network security.