Delay-Margin based Traffic Engineering for MPLS-DiffServ Networks

Mohamed Ashour and Tho Le-Ngoc

Abstract: **This paper presents a delay-margin based traffic engineering (TE) approach to provide end-to-end quality of service (QoS) in multi-protocol label switching (MPLS) networks using differentiated services (DiffServ) at the link level. The TE, including delay, class, and route assignments, is formulated as a nonlinear optimization problem reflecting the inter-class and inter-link dependency introduced by DiffServ and end-to-end QoS requirements. Three algorithms are used to provide a solution to the problem: The first two, centralized offline route configuration and link-class delay assignment, operate in the convex areas of the feasible region to consecutively reduce the objective function using a per-link per-class decomposition of the objective function gradient. The third one is a heuristic that promotes/demotes connections at different links in order to deal with concave areas that may be produced by a trunk route usage of more than one class on a given link. Approximations of the three algorithms suitable for on-line distributed TE operation are also derived. Simulation is used to show that proposed approach can increase the number of users while maintaining end-to-end QoS requirements.**

Index Terms: **Class and route assignments, delay, delay-margin based traffic engineering, differentiated services (DiffServ), multi-protocol label switching (MPLS), quality of service (QoS)-based routing.**

I. INTRODUCTION

In recent years, there has been a steady move towards using internet protocol (IP) routers and switches in building backbone networks. These networks are required to support a wide variety of services with varying quality of service (QoS) requirements. Traditionally a backbone network supported one telecommunication service, which allowed the selection of the backbone technology that can provide the best QoS. However, the use of separate networks decreased resource utility because it prevented the sharing of resources. Grouping all services in one network is also inefficient as it requires operating the network at low load to guarantee the most stringent QoS requirements.

The advance towards IP QoS provisioning used either a flow-based or a class-based approach. The flow-based approach (e.g., integrated services (IntServ)) provides accurate per connection end-to-end QoS but faces a scalability problem. The poor scalability is caused by the need to keep and monitor a per-flow state at each link. Class-based approaches (e.g., differentiated servi-ces (DiffServ) [1]) improve scalability by grouping flows into a limited set of per-hop-behavior (PHB) classes. Providing end-to-end QoS in DiffServ is challenging because DiffServ lacks route definition and resource reservation capabilities. Combining multi-protocol label switching (MPLS) and DiffServ allowed the use of MPLS label switched paths (LSP) to define end-to-end routes in DiffServ. The original MPLS-DiffServ [2] architecture focused on using MPLS LSPs to connect DiffServ networks, which presents a scalability problem close to that of IntServ. In this paper, we aim to maintain scalability by using MPLS to define routes, while using DiffServ to provide per link QoS. This enhances scalability by grouping LSPs at each link into a limited set of DiffServ PHB classes.

End-to-end QoS provision using this MPLS-DiffServ approach faces many challenges. In [3], we presented a delay margin based approach that initially aims to overcome two of these challenges. The first challenge is how to choose routes in the presence of the QoS interdependency between classes. This inter-dependency is caused by work-conserving schedulers. The second is how to assign flows to QoS classes in more efficient manner than the service based QoS mapping associated with DiffServ. The delay margin penalty function introduced in [3] enabled end-to-end flows to choose the route and/or the end-to-end DiffServ class that can best suit their QoS requirements, while providing the least QoS degradation to other network flows. Delay-margin formulation in [3] uses the network topological diversity to increase the number of QoS connections. It allows diverting traffic with low QoS requirements away from parts of the network topology that are needed to support the flows with higher QoS requirements. By doing so the approach allows the network to fully use its topology to accept more users with QoS requirements.

However, the work presented in [3] had limited traffic engineering capabilities. The limitations emerged from assuming the special case of priority scheduling and the DiffServ end-to-end class assignment in the problem formulation. More importantly the work in [3] did not present the traffic engineering platform that would maximize the use of the delay margin based problem formulations. This traffic engineering is needed to integrate the different components and provide the platform for implementing them. The use of priority queuing limits the network efficiency because it prevents links from adjusting their class delays for the same set of input loads. When priority scheduling is used, a flow with a QoS requirement slightly better than a given link class QoS, has to be assigned to the higher priority class. A flow assigned to a higher priority class can waste capacity by receiving a QoS that is much better than its requirement. Priority queuing has no means of slightly decreasing the QoS of a lower priority class in order to allow it to accommodate a given flow. A general work conserving scheduler, e.g., weighted fair queue (WFQ), has the ability to adjust the delay among its classes. By adjusting the weights, WFQs can provide a wide range of QoS combinations for same set of input traffic. Determining these delay

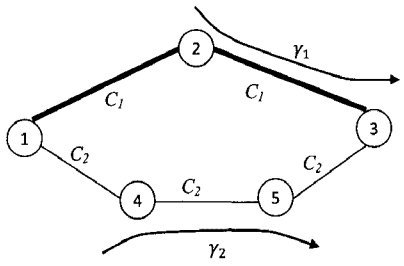
Manuscript received January 20, 2007; approved for publication by Jeonghoon Mo, Division III Editor, March 3, 2007.

M. Ashour is with the Department of Information Technology Engineering, German University in Cairo, Egypt, email: mohamed.ashour@guc.edu.eg.

T. Le-Ngoc is with McGill University, Montreal, Canada, email: tho.le-ngoc @mcgill.ca.

1229-2370/08/$10.00

operating points added two dimensions of coupling to the problem formulation in [3] a link coupling and a route coupling. On the link level, increasing the QoS of a class decreases the QoS received by the other link classes. On the route level, decreasing the QoS of given class on a given link may require increasing the QoS at other links across the route in-order to insure that the route end-to-end QoS is not violated. The route coupling also provides traffic engineering tools by allowing the network to increase the QoS at unloaded links hence allowing loaded links to support less stringent QoS requirement. The splitting also allows each class to manage its own delay limits. These added traffic engineering tools come with challenge of how to determine the values of these QoS operating points for each queue. Several approaches dealt with adjusting the WFQ inter-class delay. These approaches were mainly concerned with adjusting the weights given a local delay requirement for each class. Techniques in [4] and [5] adapt WFQ weights to changes in input traffic patterns. The weight adaptation in WFQ can be based on the queue length [6], the moving average input rates [7], and the delays of served packets [8]. The delay assignment in this paper is more concerned with partitioning the end-to-end delay of flows to local per link delays is a way that will maximize the network wide delay margin function. It has to consider the interdependency between *classes* (due to WFQ) because it affects the partitioning of the end-to-end delays. For generality, this paper assumes the presence of a proper WFQ adjustment algorithm that is capable of achieving the delay operation points, but is not related to any particular method to achieve this delay. The idea to split end-to-end QoS classes into a set of local QoS classes, and to let each of these classes locally monitor and guarantee its local QoS bound was introduced in [9]. Several techniques dealt with the partitioning of different QoS aspects. A partitioning of the loss rate guarantees while optimizing bottleneck utility using a heuristic was proposed in [9]. Similar to [9], the work in [10] splits the guarantees equally on all links. The QoS partitioning problem is solved in [11] using greedy add and greedy move algorithms that are based on proofs in [12] and [13]. Both algorithms in [11] iteratively move a small amount of resources from the highest-cost gradient link to the lowest one. The distributed version of the delay assignment algorithm in this paper uses a similar concept of incremental adjustment, but it operates in a DiffServ environment. The work in [9], [10], and [14] assumes an IntServ flow-based environment where each flow is allowed to have any QoS partition at any link by reserving the appropriate amount of resources. An algorithm for network-wide QoS partitioning of flows sharing the same class/link was introduced in [15]. The algorithm focuses on maximizing the delay partition at each link without considering that a better configuration can be obtained by allowing some links to decrease their delay to provide slack for other links. These approaches are more concerned with partitioning the QoS requirement in a way that would minimize the resource usage and do not fit a class-based QoS provisioning where there is no reservation and flows assigned to a given class receive the same QoS. This paper takes a different approach of trying to partition the end-to-end delay seen by a route instead of partitioning the route end-to-end delay requirement. The partitioned end-to-end delay is used to define the delay operation point of each class. Due to the lack of reservation, a delay partition on one class has to consider the coupling between classes introduced by the DiffServ service-provision model. The aim is to push end-to-end delays of all routes as far away from their requirement as possible, i.e., to maximize their end-to-end *delay margin*. The combination of MPLS and DiffServ also provides another important traffic engineering capability. Because the DiffServ class is inferred from the MPLS header, LSPs can switch their DiffServ classes with the switching of MPLS labels. In [3] flows are assigned the same DiffServ class on all links. The *fixed* end-to-end DiffServ class assignment limits the number of end-to-end QoS choices to the number of supported DiffServ classes. The limited end-to-end QoS choices waste capacity, because flows choose end-to-end QoS classes that are much better than their requirements. It also prevents saving link capacity elsewhere in the network by allowing flows that receive better QoS at a given link to be assigned to a lower QoS classes at other links. The closest approach to the one in this paper was presented in [16]. The approach in [16] maintains the end-to-end QoS by allowing the promotion and demotion of DiffServ drop precedence based on the cost seen by a given flow. The approach in this paper has the advantage of considering a network-wide cost and provides more flexibility by allowing the switching of DiffServ classes. This paper focuses on providing a platform that combines all the traffic-engineering capabilities provided by the delay margin penalty function. The proposed approach aims to exploit the inter-queue and inter-link QoS dependencies in multi-class networks to increase the amount of end-to-end network traffic. The information on QoS interdependency is used to find a network configuration that provides a flow with the largest possible QoS margin (for its requirement), while minimizing the decrease in the QoS margins of the others. Combining the choice of route, per-link DiffServ class assignment, and values of class delay operating points aims to overcome the resource inefficiency of fixed end-to-end DiffServ class mapping and make use of the work conserving scheduler ability to provide a wide range of QoS combinations. The approach increases the end-to-end QoS choices, hence increasing the number of network users. To maintain scalability, the approach uses end-to-end flow state and local QoS calculation at each link, and does not requires resource reservation and does not maintain per flow state information. To the best of our knowledge, no existing solution exploits the queue dependencies to find the class delay operating points, variable class assignment, and the route configuration in a multi-class DiffServ networks. The problem of combining routing and QoS partitioning was considered in [17] but it used an IntServ model and the assumption of unicast links. A combined approach based on inaccurate traffic information was introduced in [18]. Although the solutions in [17] and [18] take a similar approach to this paper by combining routing and QoS partitioning, they are flow-by-flow approaches that operate in an IntServ environment. The paper starts in the next section by an example that explains the value of changing the delay operating points. The section then presents the nonlinear optimization problem formulation that combines defining the delay, class and route assignments. It is shown in the Appendix that areas of the objective are concave. A solution is proposed based on three algorithms. The first two algorithms, *centralized offline route configuration* and *link-class delay assignment*, operate in the *convex* areas of the feasible region to iteratively reduce the objective function using a gradient-



$c_1 = 12$ BW Units

$c_2 = 6$ BW Units

Trunk	Source	Destination	γ	D
1	2	3	1	150
2	4	5	2	200
2	1	3	4	300

Fig. 1. Example network.

based approach. Both solutions are based on a decomposition of the function gradient into per-link, per-class components. Each component captures the effect of all flows using this class and the interacting flows. The third algorithm is a heuristic used at the LSP level to promote/demote LSP at different links in order to move across the *concave* areas. The algorithms are suitable for the centralized component of traffic engineering (TE). Based on the nonlinear optimization problem, we present approximations for *distributed on-line* operation and propose the TE approach that integrates the algorithms functionality and provides the platform for their operation. The TE approaches assume the use of currently proposed IP QoS routing extensions [19], and the extended functionality of MPLS-DiffServ introduced in [20]. The performance of the proposed approach is investigated by simulation in Section V.

II. MULTI-CLASS DELAY-MARGIN PENALTY MINIMIZATION (MC-DMPM)

Consider the example network with the associated parameters shown in Fig. 1, and assume that trunk traffic is the Poisson distributed with an average of γ, and that delay is calculated using an M/M/1 queuing model. Using a similar example, we showed in [3] that delay margin maximization allows users to choose the appropriate end-to-end route and QoS class. When the delay requirements are set to those shown in Fig. 1, the priority scheduler used in [3] fails to find a routing configuration that can accommodate all three trunks. The best configuration provided by the priority scheduling in [3] will achieve a delay of 325 ms by placing trunk 3 on the lower-priority class on route 1–2–3. In this case, trunk 1 will have an end-to-end delay of 100 ms, which is well below its requirement. When a queuing scheme such as WFQ is allowed to change the *delay operating points* of the two classes on link 2–3, it can increase the delay operating point of the lower-delay class from 100 ms to 150 ms, and consequently decrease that of the higher-delay class from 200 ms to 175 ms. In this example, the ability to adjust the delay operating points provided more flexibility and allowed both trunks 1 and 3 to satisfy their end-to-end delay requirements. This provides a trade-off in finding the appropriate class assignment and delay operation point for each class.

As shown in the example in Fig. 1, enhancing the network operation requires using a network TE [21] approach that uses global network state knowledge, non-shortest-paths, and multi-path routes to find an optimal network configuration. The aim of this TE is not limited to searching for a network route configuration but extends [3] to add the determination of the class delay operating points and the per-link class assignment. The approach should provide LSPs with the flexibility of choosing different DiffServ classes at different links in order to guarantee their QoS at the minimum degradation to the QoS of other flows. To formulate the problem, consider a network supporting a set I of QoS classes and described by the directed graph $G = (V, E)$ with a set V of nodes connected to each other using a set E of directional links; each link $e \in E$ is represented by a source-destination pair (s_e, t_e), where $s_e \in V$ and $t_e \in V$ are the source and destination nodes, respectively. Each link has a total capacity C_e. Each link QoS class $i \in I$ carries an amount of traffic of $\lambda_{e,i}$. Traffic using class i on link e experiences an average delay $d_{e,i}$. A multi-class scheduling approach such as WFQ is used to serve traffic heading to link e. The average delay feasible region for a given link can be represented as a linear combination of the link delays weighted by the class loads, i.e.,

$$\lambda_e d_e = \sum_i \lambda_{e,i} d_{e,i} \tag{1}$$

where λ_e is the total link traffic and d_e is the link average delay if this total traffic was served by a *single* queue. The network supports a set M of end-to-end *trunks*; each trunk $m \in M$, denoted by the source-destination pair $(\hat{s}_m, \hat{t}_m)$, wishes to transfer a total amount of traffic γ_m from the source node[1] $\hat{s}_m$ to the destination node $\hat{t}_m$ using the set R_m of all possible *routes* within a required trunk delay limit of $\hat{D}_m$. The trunk m can select a subset of routes, $P_m \subset R_m$ where each route $p \in P_m$ is used to transport a partial amount of traffic, $\gamma_{m,p} > 0$ and is defined by a set $E_{m,p}$. Each element in $E_{m,p}$ represents a link-class pair $\{e, i\}$ that represents a QoS class i in link e. The formulation in this way is more general than traditional DiffServ QoS architecture used in [3] since it enables a flow to attain different QoS class assignments on different links along the route. The end-to-end delay $\hat{d}_{m,p}$ of each route p is $\hat{d}_{m,p} = \sum_{\{e,i\} \in E_{m,p}} d_{e,i}$ and must be kept below a given delay requirement $\hat{D}_m$. It is desired to keep $\hat{d}_{m,p}$ as smaller than $\hat{D}_m$ as possible. As an indicator for each route p of trunk (or user) m, a *delay-margin penalty* function $U_{m,p}$ is defined as

$$U_{m,p} = \left(1 - \hat{d}_{m,p}/\hat{D}_m\right)^{-1}. \tag{2}$$

This objective function providing an indicator for the overall network is defined as the weighted sum of the delay-margin penalty functions of all trunks,

$$U = \sum_{m \in M} \sum_{p \in P_m} \frac{\gamma_{m,p}}{\gamma_m} U_{m,p} \tag{3}$$

where the weight denotes the ratio of the partial amount of trunk m traffic, $\gamma_{m,p} > 0$ carried over the route p to the total amount of traffic γ_m of trunk m. Our objective is to find the optimum operating delay points for each link-QoS class,

[1] The sign ^ is used to indicate an end-to-end variable.

the sets of routes, $E_{m,r}$, and the corresponding allocated traffic amounts $\gamma_{m,r}$ for all trunks in order to achieve the lowest possible weighted sum of the delay-margin penalty functions $U_{m,p}$. In other words, the *multi-class delay-margin penalty minimization* (MC-DMPM) problem is formulated as follows. Find $E_{m,p}$, $\gamma_{m,p}$, and $d_{e,i}$ for $\forall e \in E$, $\forall m \in M$, $\forall i \in I$, and $\forall p \in P_m$ to

$$\min \left(\sum_{m \in M} \sum_{p \in P_m} \frac{\gamma_{m,p}}{\gamma_m} U_{m,p} \right) \tag{4}$$

while satisfying the following constraints:

$$\sum_{i \in I} \lambda_{e,i} < C_e, \forall e \in E \text{ and } \forall i \in I, \tag{5}$$

$$\gamma_m = \sum_{p \in P_m} \gamma_{m,p}, \ \forall m \in M, \tag{6}$$

$$\sum_{t_e = t} \lambda_{e,m,i} - \sum_{s_e = t} \lambda_{e,m,i} = \begin{cases} -\gamma_m, & \text{if } t = \hat{s}_m, \\ \gamma_m, & \text{if } t = \hat{t}_m, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

$$\hat{d}_{m,p} \leq \hat{D}_{m,p}, \tag{8}$$

$$\lambda_e d_e = \sum_i \lambda_{e,i} d_{e,i}, \tag{9}$$

$$d_{e,i}(w = 1) < d_{e,i} \tag{10}$$

where $\lambda_{e,m,i} = \sum_{(p \mid \{e,i\} \in E_{m,p})} \gamma_{m,p}$, $\hat{d}_{m,p} = \sum_{\{i,e\} \in E_{m,p}} d_{e,i}$, and w is the weight in the weighted fair queue. The above formulation has the form of a multi-commodity flow optimization problem and its delay-margin objective function in (3) is similar to that previously presented in [3]. This penalty function goes to infinity as the path delay approaches the delay requirement under the end-to-end delay constraint (8). Using admission at network edges to prevent accepting calls as the penalty approaches ∞ ensures that the function is only valid in the range form $\hat{d}_{m,p} = 0$ to $\hat{d}_{m,p} = \hat{D}_m$ and hence will always remain in the delay feasible region. Another advantage is that, for any end-to-end delay requirement $\hat{D}_m < \infty$, the capacity constraint (5) is also implied in the penalty function formulation, since any link delay $d_{e,i}$ would go to infinity as the λ_e approaches C_e and hence, will cause $\hat{d}_{m,p}$ to approach $\hat{D}_m$. Constraints (6) and (7) are required in any flow optimization problem as they guarantee that all the traffic sent by a given trunk is received at the trunk destination and that all traffic input to a node is equal to the traffic output except at the source and destination. The problem formulation extends [3] to add the finding of the set of *delay operating point* $d_{e,i}$ to the route and class selection. Adding the optimization of $d_{e,i}$ adds another dimension to the problem. Because flows are allowed to switch their classes with the switching of MPLS labels, the constraint is interpreted differently in this paper. In all the values of $d_{e,i}$ in the sum $\sum_{\{e,i\} \in E_{m,p}} d_{e,i}$ had to have the same value of i. In this paper, the formulation of (8) allows the flows to assume any QoS class at any link. Constraints (9) and (10) were added to the problem formulation to represent the general work-conserving scheduler. The equality constraint in (9) represents the inter-relation between the average delays of different classes, which is introduced by the work conserving scheduler. It ensures that the class delays are chosen from the linearly related feasible set characterized by (9). Constraint (10) ensures that any chosen delay is larger than the received delay when this class has priority over all other classes. Without constraints (9) and (10), the problem can be treated as an unconstrained nonlinear optimization. When (9) and (10) are added, the problem is transformed into a constrained nonlinear optimization. Investigation of the function convexity in the presence of the delay operating point adjustment capability in Appendix A shows that the function represents convex hull that is interrupted by concave areas, and these concave areas are the result of a trunk splitting its load between two classes at the same link. In this case, the minimum of U falls on the side of this concave area, corresponding to the case when traffic of all the routes is assigned to only one class.

III. PROPOSED TE SCHEMES

To solve for MC-DMPM, we propose to decompose the optimization problem into three sub-problems. The first one (to be discussed in Section IV-A) focuses on *route configuration*, aiming to find the best set of *routes* for each *trunk* and the corresponding traffic loading for each *route*. Based on this resulting route configuration, the second sub-problem (to be discussed in Section IV-B) deals with the *link class delay assignment* to search for the optimal set of *delay operating points* for all the *links*. The third sub-problem (to be discussed in Section IV-C) aims to overcome the local minima caused by concave areas that may arise as discussed in Appendix A. In *single-class* networks, centralized offline route configuration is the only required algorithm.

A. Centralized Off-Line Route Configuration

The centralized offline route configuration is a gradient-based multi-class nonlinear optimization that operates on the convex region of the objective function. The algorithm, summarized in Fig. 2(a), moves the trunk traffic from routes/classes with high objective function increase rates to routes/classes with lower increase rates, and hence gradually descending to the optimal routing configuration. When there is change in traffic of *trunk* v assigned to *route* $E_{v,q}$, the bandwidth sharing between *trunks* over different *links* will affect the delay-margin penalty of traffic $\gamma_{m,p}$ (of trunk m on *route* $E_{m,p}$) if both $E_{m,p}$ and $E_{v,q}$ share at least one link-class pair $\{e,i\}$. The impact of this change can be represented by the overall rate of change in the objective function,

$$L_{v,q} = \frac{\partial U}{\partial \gamma_{v,q}} = \sum_m \left(\frac{1}{\gamma_m} \sum_p \left(U_{m,p} \frac{\partial \gamma_{m,p}}{\partial \gamma_{v,q}} + \gamma_{m,p} \frac{\partial U_{m,p}}{\partial \gamma_{v,q}} \right) \right)$$

where $\frac{\partial U_{m,p}}{\partial \gamma_{v,q}} = \frac{U^2_{m,p}}{\hat{D}_m} \frac{\partial \hat{d}_{m,p}}{\partial \gamma_{v,q}}$ and $\frac{\partial \hat{d}_{m,p}}{\partial \gamma_{v,q}} = \sum_{e \in (E_{m,p} \cap E_{v,q})} \frac{\partial d_{e,i}}{\partial \gamma_{v,q}}$

where $\partial \gamma_{m,p} / \partial \gamma_{v,q}$ is 1 if $E_{m,p} = E_{q,v}$ and zero otherwise. Similarly $\partial d_{e,i} / \partial \gamma_{v,q}$ is non-zero only when $e \in E_{v,q}$. Rearranging terms and replacing $\partial \gamma_{v,q}$ by $\partial \lambda_{e,i}$ (since at any node $\lambda_{e,i} = \sum_v \sum_q (\gamma_{v,q} | \{e,i\} \in E_{v,q})$), $L_{v,q}$ can be rewritten as

$$L_{v,q} = \frac{1}{\gamma_m} [U_{v,q} + \sum_{\{e,j\} \in E_{v,q}} l_{e,i}] \tag{11}$$

where

$$l_{e,i} = \sum_i \frac{\partial d_{e,i}}{\partial \lambda_{e,i}} \psi_{e,i} \tag{12}$$

and

$$\psi_{e,i} = \sum_m \sum_{(p \mid \{e,i\} \in (E_{m,p}))} \gamma_{m,p} \frac{U^2_{m,p}}{\hat{D}_m}. \tag{13}$$

(a) Route Configuration:

while $(\Delta < \eta)$

for $(\forall m \in M)$

for $(\forall p \in R_m)$ $L_{m,p,i} = \partial U / \partial \gamma_{m,p,i}$

end for

$E^*_{m,p} = (E_{m,p} | L_{m,p} = \min_{p \in R_m} (L_{m,p}))$

$P_m = P_m \cap p$

$\alpha^* = \arg\min_{\alpha} (\sum_m \sum_p U_{m,p}(\alpha))$

where

$$\gamma^{(k+1)}_{m,p} = \begin{cases} \gamma_{m,p} + \alpha(\gamma^{(k)}_m - \gamma^{(k)}_{m,p}), & \text{if } E_{m,p} = E^*_{m,p}, \\ (1-\alpha)\gamma^{(k)}_{m,p}, & \text{otherwise,} \end{cases}$$

$U^{(k+1)} = \sum_m \sum_p U_{m,p}(\alpha^*)$

end for

Link Class Delay Assignment

$\Delta = 1 - U^{(k+1)}/U^{(k-m)}$

end while

(b) Link Class Delay Assignment:

while $\eta > \varepsilon$

$\tilde{U} = U^{(k)} + \sum \sum (\tilde{d}_{e,j} - d^{(k)}_{e,j}) R_{e,j}$

$$\tilde{\mathbf{d}} = \arg\min_{\mathbf{d}^{(k)}} \left(\sum_e \sum_j \tilde{d}_{e,j} \; R_{e,j}|_{\mathbf{d}^{(k)}} \right)$$

$\mathbf{d}^{(k+1)} = \alpha \tilde{\mathbf{d}} + (1-\alpha)\mathbf{d}^{(k)}$

where

$$\alpha = \arg\min_{\alpha} \left[U(\alpha \tilde{\mathbf{d}} + (1-\alpha)\mathbf{d}^{(k)}) \right]$$

Calculate $U^{(k+1)}$ *using* (2) *and*

$\eta = \left| (U^{(k)} - U^{(k+1)})/U^{(k)} \right|$

end while

(c) Route Traffic Promotion/Demotion:

for $(\forall e \in E)$ & $(\forall i \in I)$

for $(\forall \{m,p\} | \{e,i\} \in E_{m,p})$

for $(\forall j \in I | i \neq j)$

$\tilde{E}_{m,p} = E_{m,p} \cup \{e,j\} / \{e.i\}$

$\tilde{\lambda}_{e,i} = \tilde{\lambda}_{e,i} - \gamma_{mp}$

$\tilde{\lambda}_{e,j} = \lambda_{e,j} + \gamma_{mp}$

if

$U(\{\tilde{\lambda}_{e,j}, \tilde{\lambda}_{e,i}\}) < U(\{\lambda_{e,j}, \lambda_{e,i}\})$

$E_{m,p} = \tilde{E}_{m,p}$

end if

end for

end for

end for

Fig. 2. Traffic engineering algorithms.

The term $l_{e,j}$ in (11) is the change in the network objective function with respect to the class j traffic on link e. It can be used to represent the length of a given class on a given link and hence provides a partial decomposition of the problem on a per-class per-link basis. In calculating the route length using (11), each class can be considered as a different link with its own length in the single-class case. The form of $l_{e,j}$ has its advantages in real-life implementation as indicated in [22] for the single-class case. The main advantage in the multi-class case is that each link can introduce the load dependencies between its classes through the calculation of its own $\partial d_{e,j}/\partial \lambda_{e,i}$, based on the monitoring and characterization of its own traffic.

The complexity of this algorithm is similar to that of a gradient-based algorithm, e.g., flow deviation. Flow deviation converges in $O(\rho^2 e^5 \varepsilon^{-3} m)$ of route calculations [23], where ρ is the width, ε is the performance, and we abuse the notation and use e to represent the number of links and m to represent the number of trunks. Because at each link, the number of choices equals the number of classes i, this algorithm is expected to converge in $O(\rho^2 [ei]^5 \varepsilon^{-3} m)$. The main complexity of this algorithm comes from route calculation. Although a minimum value of each term in $L_{v,q}$ can be calculated using a shortest path approach, their sum requires the use of min-cost routing that is constrained by the value of $\hat{D}_{v,q}$. In the centralized approach in this paper we assume the use of pre-computed routes [24] to speed up this evaluation. We also present an approximation in Section III-D.

B. Link Class Delay Assignment

Based on the allocation of routes, the *route configuration* algorithm searches for an optimal set of *delay operating points* for all the *links* to provide route loads and route class assignment. The difficulty of the problem arises from the fact that constraints (9) and (10) are not implied by the objective function and hence the problem has to be solved as a constrained non-linear optimization problem. Fig. 2(b) summarizes the proposed solution that uses a linear approximation of the function at each iteration to find a solution to the resulting constrained linear optimization problem. Line search is then used to find a lower-cost point on the nonlinear function using the linear approximation based on the function gradient represented by

$$R_{e,j} = \frac{\partial U}{\partial d_{e,j}} = \sum_m \sum_p \frac{\gamma_{m,p}}{\gamma_m} \frac{\partial U_{m,p}}{\partial d_{e,j}} = \sum_i \psi_{e,i} \frac{\partial d_{e,i}}{\partial d_{e,j}}. \quad (14)$$

To calculate $R_{e,j}$ at the link level, each link needs to keep track of the value of $\psi_{e,i}$ for each class, and each class needs to keep track of $\partial d_{e,i}/\partial d_{e,j}$ with respect to all other classes j on the same link. The value $\partial d_{e,i}/\partial d_{e,j}$ depends on the multi-class queuing approach in use and can be calculated by the link traffic engineering module based on its own characterization of its input traffic and its current queue configuration. These values can be periodically conveyed to the traffic-engineering server to be used in the optimization, or they can be used locally in between optimization to provide a local approximation to the centralized link class assignment. In the simple two-queue case, $\partial d_{e,i}/\partial d_{e,j}$ can be obtained from (9). When priority queuing is used, such as the case in [3], there is no way to change the delays for a fixed load and hence $\partial d_{e,i}/\partial d_{e,j} = 0$ and $d^*_{e,i} = d_{e,i}$. The optimal values of link class $d^*_{e,i}$ delays can be obtained by iteratively using line search techniques to solve the nonlinear optimization for all combination of e and j and constrained by (9) and (10). The calculation of $R_{e,j}$ requires a maximum m trunk updates and $(i-1)$ calculation of $\partial d_{e,i}/\partial d_{e,j}$ where i is the number of classes. The calculation of $\tilde{\mathbf{d}}$ is a linear optimization problem that has delay constraint equalities up to $m\bar{p}$, where m is the number of trunks and $\bar{p}$ is the maximum number of routes per trunk. Because of the work conserving properties of the link schedulers, the system has e equalities, where e is the number of links. Each equality describes the relation between the delays of each link. For a simplex method, the convergence time for the

calculation of $\tilde{\mathbf{d}}$ is an exponential function of $(m\bar{p}+e)$.

C. Route Traffic Promotion/Demotion

This algorithm aims to move the optimization process from one convex area to the other by examining the routes (of different trunks) one by one as shown in Fig. 2(c). For each link on each route, it checks if further reduction in the objective function can be achieved by moving the route traffic on this link to a higher-or lower-delay class. If a better class is found, the trunk route is promoted or demoted and the objective function is updated accordingly.

The traffic promotion/demotion step in the algorithm has two features: (i) The move of traffic between classes is done per flow, which provides an opportunity for distributed end-to-end implementation. (ii) The move between classes is done over the same established path, which provides a means for collecting information about the cost of other classes along the route links using protocols such as RSVP-TE [25]. Because classes can be switched with the switching of data, promotion/demotion can be done at the link level.

D. Distributed On-Line Routing Approximation

The distributed version of the algorithm makes use of $L_{m,p}$ to find $\tilde{E}^*_{m,p}$, the optimal path for an arriving call for trunk m at its time of arrival. To enable a more distributed implementation of the algorithm that can make use of vector routing tables, we use an approximation for the calculation of $L_{m,p}$. The approximation is based on the observation that the smallest value of $L_{m,p}$ occurs either at the path with the smallest value of $\hat{d}_{m,p}$ or with the smallest values of $l_{e,i}$. The approximation is based on calculating two shortest paths. The first one uses link delay as length; if several paths with the same smallest delay exist, it picks the one with the smallest values of $l_{e,i}$. The second one uses $l_{e,i}$ as the length; if several shortest paths with the length $l_{e,i}$ are found, then the one with the smallest delay is chosen. Shortest path algorithms such as the Dijkstra [26] algorithm can be used in both calculations. In both cases, a route selection message such as the Path message in RSVP-TE [25] records $d_{e,i}$ and $l_{e,i}$ along both paths. The destination node receives both messages for the two chosen shortest paths and uses them to calculate the exact value of $L_{m,p}$ for both paths. It then chooses the one with the smaller value of $L_{m,p}$ for the call to send traffic along that route. The distributed implementation of this approximation requires each node to keep and update 2 vector routing tables per priority class, one recording the shortest path to any node in the network in terms of delay and the other recording the shortest path in terms $l_{e,i}$. These vector routing tables can be built and maintained using OSPF. It also requires the presence of path-established protocol such as RSVP-TE or CR-LDP. This reduces the complexity of the routing approach to two times that of shortest path routing.

E. On-Line Class Delay Assignment

The online delay class assignment aims to minimize the cost function by optimizing the delays at each link such as to set the gradient $R_{e,j}$ as close to zero as possible for all classes j on each link. Because of the work conserving properties of the link schedulers, only the first term $\psi_{e,j}\partial d_{e,j}/\partial d_{e,j}$ in the $R_{e,j}$ will be positive and all other terms will be negative. Increasing the delay of given class increases the objective function of flows going through this class and decreases the objective function of flows going through other classes. A negative value of $R_{e,j}$ means that increasing the delay of class j will decrease the overall network cost. The algorithm is triggered by periodic updates to $\psi_{e,i}$. It works by calculating the value of $R_{e,j}$ after each update, and the delay of the class with the smallest negative $R_{e,j}$ class is increased by small value α.

IV. PROPOSED END-TO-END TE ARCHITECTURES

The proposed TE provides the platform and functionality needed to integrate the solution approaches discussed in the previous section to give the DiffServ QoS provisioning the capability to support delay guarantees, and to increase the number of accepted users in the network. The core of the proposed TE approach is based on allowing LSPs with the same DiffServ class to share queues and their ability to switch this class at different links. This feature is enabled by allowing the switching of DiffServ PHB with the switching of MPLS labels. Details and possible implementation of this proposed variable QoS class assignment is described in [20]. As shown in Fig. 3, each delay category needs a separate queue. Either WFQ or PQ is used for the scheduling of traffic. The queue weights or priority are adjusted to enable each queue to operate at a given delay point.

The proposed TE approach operates on three time-scales as shown in Fig. 4. Each time-scale is associated with control over a topology scale. The longer time scale (hours to days) is associated with network-wide TE. A medium time scale operating on a range of minutes to hours deals with the LSP-level TE. The finest time scale TE operates on a range of seconds to minutes and deals with queue configuration and scheduling issues.

A. Long-Term Network TE

On the long term TE, routes and their QoS class assignments are off-line computed using the centralized offline route configuration. The optimization process is based on long-term traffic characterization of the input traffic. The approach assumes the presents of a centralized server to make the calculation and distribute the resulting configuration on network routers. Once the optimal route configuration is calculated, the routing tables corresponding to these configurations are downloaded to different routers and are used for forwarding the flows. Flows at the edge of the network will be mapped to these pre-calculated routes, and do not need to communicate with the central TE server to determine how to get to their destination. The network will continue to use this routing configuration until a new update, and hence can operate for hours without the centralized TE server.

B. Medium-Term Flow Level TE

The central offline network configuration is calculated only for a long time interval. To take care of new flows that arrive in the network between two long-term network configurations, this paper uses a medium-term flow-level TE. The flow-level TE handles the actual forwarding, establishment, and resource assignment at the edges of the network to deal with functions such as accepting end-user connections inside the flow and estimating the statistical characteristics of the flow traffic from its con-

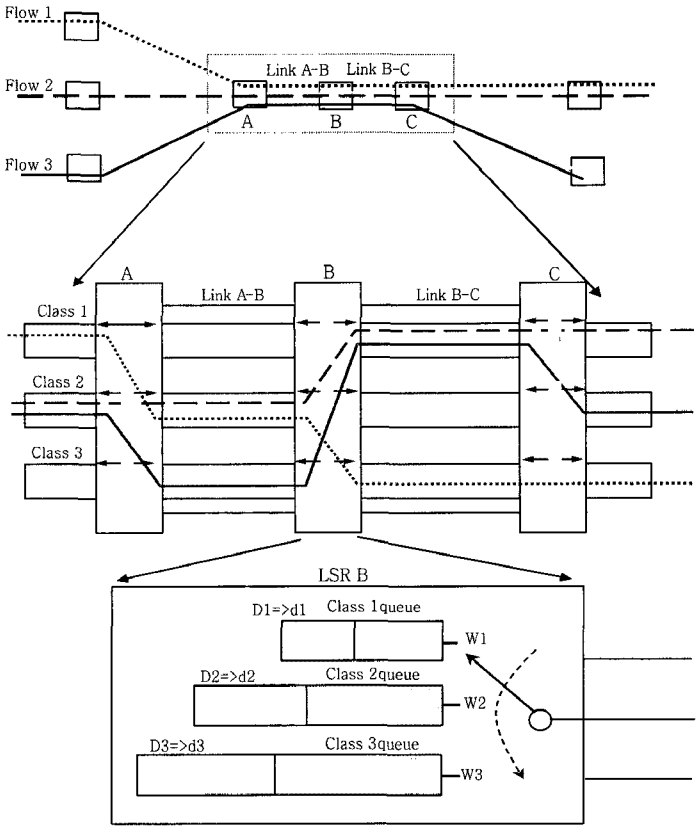


Fig. 3. Example of operations in the proposed TE architecture.

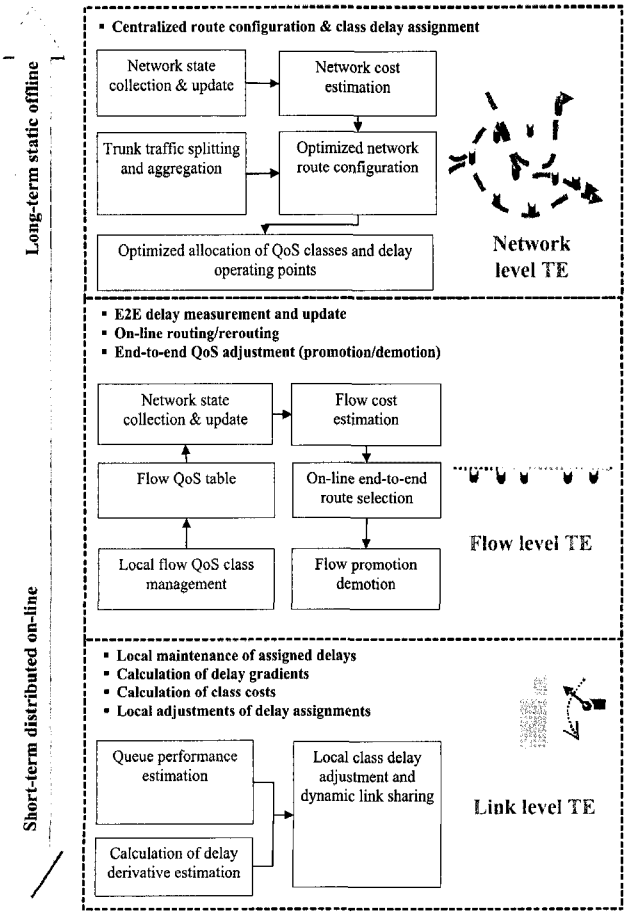


Fig. 4. Hierarchy and functions of the proposed TE approach.

stituent end-user statistical characteristics. To keep the network operating close to the optimal point previously derived by the *long-term* TE configuration, the flow-level TE monitors end-to-end traffic on each route and its received QoS and reacts in order to calculate the delay margin. Monitoring the delay margin of each flow allows for implicitly monitor the reservation status on its links. The flow-level TE aims to keep online routing and class assignment having the least deviation from the configuration established by the centralized offline traffic engineering. The objective is to make the network more stable by minimizing the need for offline configurations by diverting the traffic to another route.

The operation of the flow-level TE requires cooperation at the link level. The flow-level TE requires the ability of the link to change its scheduling parameters and queuing configuration to adjust the delay operating points. The flow level TE assumes the use of an RSVP like protocol for the establishment, maintenance, and release of the flow routes. For each established flow, the TE keeps a record of its QoS requirements in the *flow QoS table* at the edge of the network. However, it is worth noting that this information is not kept at the link level. In line with the DiffServ concept, each class on a link is rather required to keep an aggregate state information about all flows going through it. This aggregate state is a function of the end-to-end delay margins of all flows using the link QoS class. As shown in Fig. 4, the *flow state collection and update* operates at the edge of the network and measures the QoS received.

The flow end-to-end QoS is measured using either piggyback end-to-end network monitoring messages, or is collected using state collection messages such as PATH message in RSVP. The measured delay is used to calculate the delay margin and the same set of messages are used to report this margin to all the links used by the flow. Note that a flow end-to-end state collection is not limited to its QoS class, but it can collect QoS measures about other QoS classes across its route. The *flow cost estimation* module estimates the expected cost increase/decrease associated with promoting/demoting a flow using the collected data. The online approximation of the route configuration evaluates the possible route choices for an arriving flow and chooses a route using link costs from the *flow cost estimation* module. The *flow promotion/demotion* module accepts the promotion/demotion decisions taken by links.

C. Short-Term Link Level TE

The short-term TE has three main roles. The first role is to locally estimate delay and calculate the local link cost. The second role is to adjust the scheduling parameters in such a way that enables achieving the QoS operating points. The third role is to adjust the delay operating points such as to increase the overall delay margins of flows using the network. The link-level TE will do that through the proper assignment of queue configurations (queue length, queue priority, and weight). For QoS performance estimation required in TE algorithms, we propose the use of multi-scale traffic modeling techniques [27] to characterize the aggregation of LSP traffic at each class. The use of multi-scale characterization enables techniques introduced in [28] and [29] to evaluate the queuing performance and calculate the delay gradients.

V. PERFORMANCE EVALUATION

A. Example Scenario

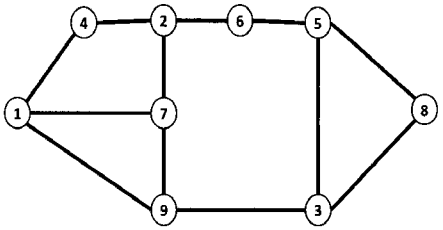
Fig. 5 shows a randomly generated example network with all links set to 2,400 capacity units. The same network was used in [3] to evaluate an earlier version of the centralized offline route configuration that considers the special case of priority queuing. The example transports 21 pairs of trunks, each pair has the same source and destination but different delay requirements (e.g., 3 ms, 6 ms). Calls of equal traffic requirements are randomly generated using a uniform distribution over the trunks. In order to examine the impacts of more realistic bursty traffic with varying loads, each call is modeled as a Poisson traffic source with at an average rate γ_c. The trunk total traffic at any point in time is the sum of the Poisson distributed traffic generated by calls admitted to this trunk. In the simulation we use a two queue WFQs for scheduling traffic at the links, and we consider the independence assumption [30] to allow easy aggregation of the traffic. Because of the absence of an accurate performance analysis for WFQ, we use an empirical approximation of the delay function where

$$d_1 = \begin{cases} (C - w_2\lambda - \lambda_1)^{-1}, & \text{if } d_1 < (C-\lambda)^{-1}, \\ (\frac{\lambda}{\lambda_1})(C-\lambda)^{-1} & \\ -(\frac{\lambda_2}{\lambda_1})(C-(1-w_2)\lambda - \lambda_2), & \text{otherwise.} \end{cases} \tag{15}$$

The admission of a new call with an average traffic rate γ_c is decided as follows. The call is assigned to the route/class with the smallest value of value of $L_{m,p,i}$ (or its approximation). The model of a priority queue with Poisson packet arrivals [31] is used to estimate the average class delay for each link in the network assuming an increase of traffic equal γ_c. Subsequently, the corresponding normalized delay margin, $(1 - \hat{d}_{v,q}/\hat{D}_{v,q})$, is estimated for each trunk. The new call will be admitted to the network if the estimated $(1 - \hat{d}_{v,q}/\hat{D}_{v,q})$ is positive for all trunks. Otherwise, it is rejected.

The results in this paper use the same approach as the static case in [32]. At each simulation step, a trunk is randomly chosen to generate a call. If the call is admitted to the trunk, the trunk total traffic is increased by γ_c, otherwise the amount of unaccepted traffic is increased by γ_c. Once admitted, the call stays until the end of the simulation. In other words, the traffic offered to the network will increase gradually with the arrival of calls. As the traffic increases, end-to-end delay will increase, causing the end-to-end delay margin of some trunks to approach zero, hence preventing them from accepting more new calls. A trunk with a delay margin close to zero, will also prevent other trunks sharing links with it from accepting calls, as this will cause its own delay violation. When the centralized approach is used, the optimization process is re-run whenever a new call is accepted. The performance of an approach in this simulation is measured by its capability to accept more calls in the network (or, equivalently, to decrease the amount of unaccepted traffic) with the increase in offered traffic.

To examine the algorithms we will gradually introduce them to evaluate the effect of each of them. We will start with a network that uses min-delay algorithms. As a starting point we assume that trunks are assigned to end-to-end QoS classes corresponding to their delay. Link weights are set to 0.6 and 0.4,



m	s	t	m	s	t	m	s	t	m	s	l
1	1	6	7	3	5	13	7	2	19	9	7
2	1	3	8	4	2	14	7	8	20	9	6
3	1	4	9	4	2	15	8	3	21	9	5
4	2	4	10	5	1	16	7	1			
5	2	3	11	5	4	17	8	5			
6	3	5	12	6	1	18	8	7			

Fig. 5. Example of network topologies and trunk requirements.

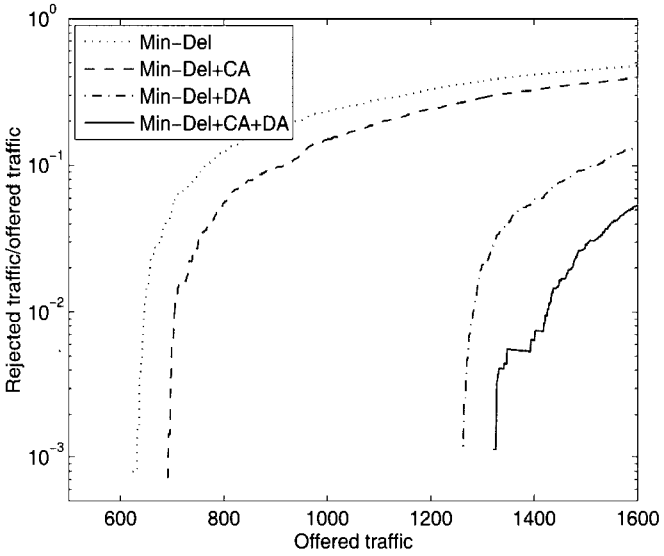


Fig. 6. Performance of *class* and *delay* assignments with *min-delay* routing.

hence creating a higher and lower delay classes. We will then first introduce each algorithm separately and then combine them to show the effect of them working together.

Fig. 6 shows the results for min-delay routing. It shows that when class assignment (CA) is introduced, it increases the amount of accepted traffic for both min-hop and min-delay routing. This is mainly due to the added choices along the route and the ability to distribute resources along the route by promoting the QoS of more LSPs at more congested links and demoting them at highly loaded links. Demotion may also be used to compensate for high QoS that is gained by sharing certain classes with connection that have a stringent QoS requirement. When delay operating point assignment (DA) is introduced, it presents a major increase in the amount of accepted traffic. This choice represents an adaptation to the end-to-end trunk QoS requirements and network topology. As expected, when both are combined, a better performance is achieved.

Fig. 7 shows the effect of allowing flows to vary their class

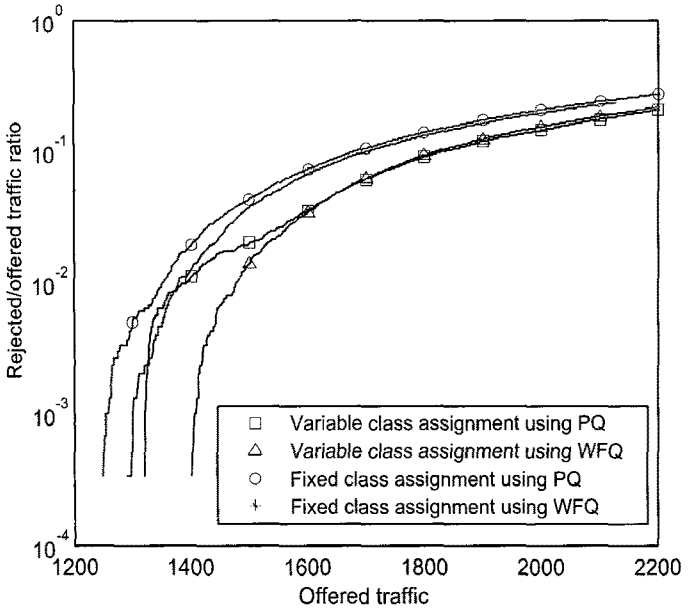


Fig. 7. Performance of *variable* and *fixed* DiffServ class assignment with PQ and WFQ.

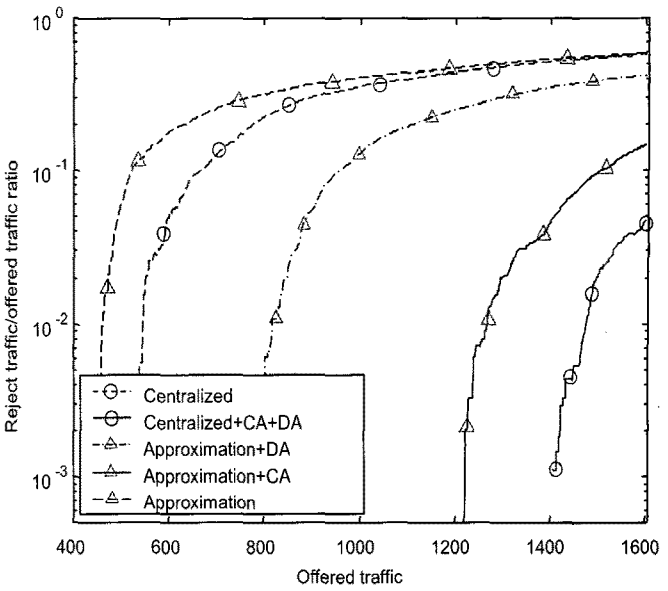


Fig. 8. Performance of *centralized* and *approximation* delay-margin route configuration schemes.

assignment along a route (i.e., *variable* class assignment) versus the traditional DiffServ end-to-end class assignment (i.e., *fixed* class assignment). Fig. 7 shows the results when both centralized routing and class adjustment is used. For each case, we consider the use of both priority and weighted-fair queues. In the configurations with weighted-fair queues, DA is used to adjust each class delay. The configuration using fixed class assignment and priority queue represents the case considered in [3]. As shown in Fig. 7, when the system used in [3] is allowed to adjust the delay operating points through the use of a WFQ, it can decrease the percentage of rejected traffic. A similar result can be obtained using priority scheduling by allowing the system to use variable class assignment. Combining CA with DA presents a slight improvement over combining each assignment with *routing*. However, the results show that this combination also decreases the delay penalty function. This means that although the traffic accepted in the network may be similar, the use of DA increases the delay margin seen by this traffic. This enables traffic to compensate against unexpected short-term traffic increases.

Fig. 8 presents a performance comparison of centralized and approximation approaches. It is interesting to notice that when multi-class delay margin based routing configuration is used alone, it does not present a major enhancement over minimum delay, even when using the centralized off-line version. However, when combined with the other two algorithms, it outperforms minimum delay routing combined with same two algorithms by more than 100 traffic units. Fig. 8 shows that, when combined with CA and DA, the approximation provides reasonable performance in trade for its distributed nature. Compared with min-hop the advantage for using the delay margin approximation is that it is based on the same formulation as the delay and class assignment. This enhances the interaction between these algorithms as they all aim to achieve the same objective. The understanding of the nature of this interaction and conditions where each of the algorithms become more valuable are our current research focus.

VI. CONCLUSIONS

In this paper, we presented a delay-margin based TE approach to provide end-to-end QoS in MPLS networks using DiffServ at the link level. Three traffic engineering algorithms are developed using a nonlinear formulation of the TE problem in the form of end-to-end delay margin. The algorithms provide more control dimensions of the network, while keeping the simple DiffServ service provisioning architecture. Simulations show that the end-to-end class adjustment and the weight adjustment algorithms can be used separately to enhance the performance of existing routing techniques. In conjunction with the route configuration algorithm, they further increase the network performance. Approximations were also proposed for a possible distributed traffic engineering structure.

APPENDIX A: CONVEXITY INVESTIGATION

Convexity of delay-margin penalty function: For (2) to be convex over a set of delay feasible load points $\gamma_{m,p}$, two conditions have to be satisfied, i.e., the domain of feasible delay $\gamma_{m,p}$ has to be convex and $\nabla^2 U$ has to be positive for any value in the feasible domain.

Convexity of feasible delay domain: For domain of feasible delay $\gamma_{m,p}$ to be convex, the load points on the line $\gamma = \alpha\gamma^{(1)} + (1-\alpha)\gamma^{(2)}$ (where $0 \leq \alpha \leq 1$) connecting any two delay feasible load points $\gamma^{(1)}$ and $\gamma^{(2)}$ have to lie within the feasible set. Points on that line can be interpreted as a load configuration (i.e., a combination of both feasible load points reduced by factors α and $1-\alpha$). Since delay is an increasing function of load, then each component is still guaranteed to be within the feasible region, and hence the domain of feasible delay $\gamma_{m,p}$ is convex. From (11) each component in $\nabla^2 U$ can be defined as:

$$\frac{\partial^2 U_{m,p}}{\partial\gamma_{v,q}\partial\gamma_{w,z}} = \frac{2\gamma_{m,p}}{\gamma\hat{D}^2_{m,p}(1-\hat{d}_{m,p}/\hat{D}_{m,p})^3}\frac{\partial\hat{d}_{m,p}}{\partial\gamma_{v,q}}\frac{\partial\hat{d}_{m,p}}{\partial\gamma_{w,z}} + \frac{1}{\gamma\hat{D}_{m,p}(1-\hat{d}_{m,p}/\hat{D}_{m,p})^2}\left(\frac{\partial\hat{d}_{m,p}}{\partial\gamma_{w,z}}\frac{\partial\gamma_{m,p}}{\partial\gamma_{v,q}} + \frac{\partial\hat{d}_{m,p}}{\partial\gamma_{v,q}}\frac{\partial\gamma_{m,p}}{\partial\gamma_{w,z}}\right) + \frac{1}{\gamma\hat{D}_{m,p}(1-\hat{d}_{m,p}/\hat{D}_{m,p})}\left(\frac{\gamma_{m,p}}{\hat{D}_{m,p}}\frac{\partial^2\hat{d}_{m,p}}{\partial\gamma_{v,q}\partial\gamma_{w,z}} + \frac{\partial^2\gamma_{m,p}}{\partial\gamma_{v,q}\partial\gamma_{v,q}}\right).$$

For a *single-class* network $I = \{1\}$, delay is a convex monotonically increasing function of γ [31]. Hence, $\partial\hat{d}_{m,p}/\partial\gamma_{w,z}\partial\hat{d}_{m,p}/\partial\gamma_{v,q}$, $\partial^2\hat{d}_{m,p}/(\partial\gamma_{v,q}\partial\gamma_{w,z})$ are all positive. On the other hand, $\left(1-\hat{d}_{m,p}/\hat{D}_{m,p}\right)$ is positive for all delay feasible load points, hence (14) is always positive within the feasible region. This means $\nabla^2 U$ is always positive, i.e., the penalty function is convex with the domain of feasible delay. On the other hand,

$$\frac{\partial U}{\partial\alpha_{m,p,e}} = \frac{\gamma_{m,p}}{\gamma\hat{D}_m\left(1-\frac{d_{m,p1}}{\hat{D}_m}\right)} - \frac{\gamma_{m,p}}{\gamma\hat{D}_m\left(1-\frac{d_{m,p2}}{\hat{D}_m}\right)} + \left(\frac{\alpha_{m,p,e}\gamma_{m,p}}{\gamma\hat{D}_m\left(1-\frac{d_{m,p1}}{\hat{D}_m}\right)^2} + \sum_{\substack{m,p\mid\{e,1\}\in Em,p\\ \{m,p\}\neq\{v,q\}}}\frac{\gamma_{v,q}}{\gamma\hat{D}_v\left(1-\frac{d_{v,q}}{\hat{D}_v}\right)^2}\right)\delta_{d1} + \left(\frac{(1-\alpha_{m,p,e})\gamma_{m,p}}{\gamma\hat{D}_m\left(1-\frac{d_{m,p2}}{\hat{D}_m}\right)^2} + \sum_{\substack{v,q\mid\{e,1\}\in Ev,q\\ \{m,p\}\neq\{v,q\}}}\frac{\gamma_{v,q}}{\gamma\hat{D}_v\left(1-\frac{d_{v,q}}{\hat{D}_v}\right)^2}\right)\delta_{d2}$$

where

$$\delta_{d1} = \frac{\partial d^*_{e,1}}{\partial\alpha_{m,p,e}}, \qquad \delta_{d2} = \frac{\partial d^*_{e,2}}{\partial d^*_{e,1}}\frac{\partial d^*_{e,1}}{\partial\alpha_{m,p,e}},$$

$$\frac{\partial d^*_{e,2}}{\partial d^*_{e,1}} = -\frac{\lambda_{e,1}}{\lambda_{e,2}} = -\frac{\alpha_{m,p,e}\gamma_{m,p}+\lambda^{-\{m,p1\}}_{e,1}}{(1-\alpha_{m,p,e})\gamma_{m,p}+\lambda^{-\{m,p2\}}_{e,2}},$$

$$\lambda^{-\{m,p1\}}_{e,1} = \sum_{\substack{v,q\mid\{e,1\}\in Ev,q\\ \{m,p1\}\neq\{v,q\}}}\gamma_{v,q}, \quad \lambda^{-\{m,p2\}}_{e,2} = \sum_{\substack{v,q\mid\{e,2\}\in Ev,q\\ \{m,p2\}\neq\{v,q\}}}\gamma_{v,q}.$$

Therefore, $\frac{\partial^2 U}{\partial\alpha^2_{m,p,e}} = H1 + H2$, where

$$H1 = \frac{\gamma_{m,p}}{\gamma\hat{D}_m\left(1-\frac{d_{m,p1}}{\hat{D}_m}\right)^2} + \left(\frac{\alpha_{m,p,e}\gamma_{m,p}}{2\gamma\hat{D}_m\left(1-\frac{d_{m,p1}}{\hat{D}_m}\right)^3} + \sum_{\substack{m,p\mid\{e,1\}\in Em,p\\ \{m,p\}\neq\{v,q\}}}\frac{\gamma_{v,q}}{2\gamma\hat{D}_v\left(1-\frac{d_{v,q}}{\hat{D}_v}\right)^3}\right)\delta_{H1,1},$$

$$H2 = -\frac{\gamma_{m,p}}{\gamma\hat{D}_m\left(1-\frac{d_{m,p2}}{\hat{D}_m}\right)^2} + \left(\frac{(1-\alpha_{m,p,e})\gamma_{m,p}}{2\gamma\hat{D}_m\left(1-\frac{d_{m,p2}}{\hat{D}_m}\right)^3} + \sum_{\substack{v,q\mid\{e,1\}\in Ev,q\\ \{m,p\}\neq\{v,q\}}}\frac{\gamma_{v,q}}{2\gamma\hat{D}_v\left(1-\frac{d_{v,q}}{\hat{D}_v}\right)^3}\right)\delta_{H2,1} + \left(\frac{(1-\alpha_{m,p,e})\gamma_{m,p}}{\gamma\hat{D}_m\left(1-\frac{d_{m,p2}}{\hat{D}_m}\right)^2} + \sum_{\substack{v,q\mid\{e,1\}\in Ev,q\\ \{m,p\}\neq\{v,q\}}}\frac{\gamma_{v,q}}{\gamma\hat{D}_v\left(1-\frac{d_{v,q}}{\hat{D}_v}\right)^2}\right)\delta_{H2,2}$$

where $\delta_{1,1} = \frac{\partial^2 d_{e,1}}{\partial\alpha^2_{m,p,e}}$, $\delta_{2,1} = \frac{\partial d^*_{e,2}}{\partial d^*_{e,1}}\frac{\partial d^*_{e,1}}{\partial\alpha_{m,p,e}}$, and $\delta_{2,2} = \left(\frac{\partial d^*_{e,2}}{\partial d_{e,1}\partial\alpha_{m,p,e}}\frac{\partial d^*_{e,1}}{\partial\alpha_{m,p,e}} + \frac{\partial d^*_{e,2}}{\partial d^*_{e,1}}\frac{\partial^2 d^*_{e,1}}{\partial\alpha^2_{m,p,e}}\right)$.

$H1$ is positive, while $H2$ is negative since $\partial d^*_{e,2}/\partial d^*_{e,1}$ and $\partial d^*_{e,2}/(\partial d_{e,1}\partial\alpha_{m,p,e})$ are negative. When $|H2| > |H1|$, $\partial^2 U/\partial\alpha^2_{m,p,e} < 0$ and hence the function relating U to moving a trunk traffic from one class to another is *not* convex in this region. In other words, in this case, for a given route loading of a trunk, the minimum of U is minimum when traffic of all the routes is assigned to only one class, i.e., the minimum value will fall on either side of the function.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC-2475, Dec. 1998.

[2] F. L. Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services," RFC 3270, May 2002.

[3] M. Ashour and T. Le-Ngoc, "End-to-end delay margin balancing approach for routing in multi-class networks," *Wireless Networks*, vol. 13, no. 3, pp. 311–322, June 2007.

[4] H. Wang, C. Shen, and K. G. Shin, "Adaptive-weighted packet scheduling for premium service," in *Proc. IEEE ICC 2001*, June 2001, pp. 1846–1850.

[5] R. F. Liao and A. T. Campbell, "Dynamic core provisioning for quantitative differentiated service," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 429–442, June 2004.

[6] M. Horng, W. Lee, K. Lee, and Y. Kuo, "An adaptive approach to weighted fair queue with QoS enhanced on IP network," in *Proc. IEEE TENCON*, Aug. 2001, pp. 181–186.

[7] C. Li, S. Tsao, M. C. Chen, Y. Sun, and Y. Huang, "Proportional delay differentiation service based on weighted fair queuing," in *Proc. IEEE ICCCN 2000*, Oct. 2000, pp. 418–423.

[8] D. Hang, H. R. Shao, W. Zhu, and Y. Q. Zhang, "TD2FQ: An integrated traffic scheduling and shaping scheme for DiffServ networks," in *Proc. IEEE HPSR*, May 2001, pp. 78–82

[9] V. Firoiu and D. Towsley, "Call admission and resource reservation for multicast sessions," in *Proc. IEEE INFOCOM*, 1996, pp. 94–101.

[10] D. H. Lorenz and A. Orda, "QoS routing in networks with uncertain parameters," *IEEE/ACM Trans. Netw.*, vol. 6, no. 6, pp. 768–778, Dec. 1998.

[11] D. H Lorenz and A. Orda, "Optimal portioning of QoS requirements on unicast and multicast networks," *IEEE/ACM Trans. Netw.*, vol. 1, no. 1, pp. 102–114, Feb. 2002.

[12] T. Ibaraki and N. Katoh, *Resource Allocation Problems. The Foundation Of Computing*. MIT press, Cambridge, Apr. 1998.

[13] D.S. Hochbaum, "Lower and upper bounds for the allocation problem and other nonlinear optimization problems," *Mathematics of Operations Research*, vol. 19, no. 2, pp. 390–409, May 1994.

[14] G. Lapiotis, S. Mao, and S. Panwar, "GPS analysis of multiple sessions with applications to admission control," in *Proc. IEEE ICC*, June 2001, vol. 6, pp. 1829–1833.

[15] F. Wang, P. Mohapatra, S. Mukherjee, and D. Bushmitch, "A random early demotion and promotion marker for assured services," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 12, pp. 2640–2650, Dec. 2000.

[16] L. Atov, H. T. Tran, and R. J. Harris, "Efficient QoS partition and routing in multiservice IP networks," in *Proc. IEEE IPCCC*, Apr. 2003, pp. 435–441.

[17] B. Gavish and I. Neuman, "A system for routing and capacity assignment in computer communication networks," *IEEE Trans. Commun.*, vol. 37, no. 4, pp. 360–366, Apr. 1989.

[18] R. Guerin and A.Orda, "QoS-based routing in networks with inaccurate information: Theory and algorithms," in *Proc. IEEE INFOCOM*, 1997, pp. 75–83.

[19] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A framework for QoS-based routing in the internet," RFC 2386, Aug. 1998.

[20] Y. Lemieux, M. Ashour, and T. Elshabrawy, "Quality of service (QoS) mechanism in an internet protocol (IP) network," US Patent 6,968,374 B2, Issued: Nov. 22, 2005.

[21] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, 2000, pp. 519–528.

[22] M. Ashour and T. Le-Ngoc, "End-to-end delay satisfaction balanced routing," in *Proc. IEEE GLOBECOM*, Nov. 2005, pp. 852–856.

[23] D. Bienstock, and O. Raskina, "Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem," *Math. Programming*, vol. 91, 2002, pp. 479–492.

[24] A. Orda and A. Sprintson, "QoS routing: The precomputation perspective," in *Proc. IEEE INFOCOM*, 2000, pp. 128–136.

[25] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swalow, *RSVP-TE: Extensions to RSVP for LSP tunnels*, RFC 3209, Dec. 2001.

[26] D. P. Bertsekas, *Network Optimization Continuous and Discrete Models*. Athena Scientific, 1998.

[27] M Ashour and T. Le-Ngoc, "Aggregation of long-range dependent traffic streams using multi-fractal wavelet models," in *Proc. IEEE CCECE*, May 2003, pp. 793–796.

[28] M. Ashour and T. Le-Ngoc, "Priority queuing of long-range dependent traffic," in *Proc. IEEE GLOBECOM*. vol. 6, Dec. 2003, pp. 3025–3029.

[29] M. Ashour and T. Le-Ngoc, "Performance of weighted fair queuing systems with long range dependent traffic inputs," in *Proc. IEEE CCECE*, May 2005, pp. 972–775.

[30] M. Geral and L. Kleinrock, "On the topological design of distributed computer networks," *IEEE Trans. Commun.*, vol. 25, no. 1, pp. 48–60, Jan. 1977.

[31] J. F. Hayes and T. G. Babu, *Modeling and Analysis of Telecommunications Networks*. John Wiley & Sons, 2004.

[32] K. Kar, M. Kodialam, and T. V. Lakshman, "Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 12, pp. 2566–2579, Dec. 2000.

Mohamed Ashour obtained his Ph.D. in Electrical and Computer Engineering from Mcgill university, Montreal, Canada, and his M.Sc and B.Sc in Electrical Engineering from Ain Shams University, Cairo, Egypt. He is currently working as an Assistant Professor at the German University in Cairo. His current Area of research is focused on Traffic Engineering and QoS provisioning in wireless mesh networks. He is also interested in long-range traffic modeling and analysis, MAC satellite protocols, and QoS provisioning in satellite networks.

Tho Le-Ngoc obtained his B.Eng. (with Distinction) in Electrical Engineering in 1976, and his M.Eng. in Microprocessor Applications in 1978 from McGill University, Montreal, and his Ph.D. in Digital Communications 1983 from the University of Ottawa, Canada. During 1977–1982, he was with Spar Aerospace Limited as a Design Engineer and then a Senior Design Engineer, involved in the development and design of the microprocessor-based controller of Canadarm (of the Space Shuttle), SCPC/FM, SCPC/PSK, and TDMA satellite communications systems. During 1982–1985, he was an Engineering Manager of the Radio Group in the Department of Development Engineering of SRTelecom Inc., developed the new point-to-multipoint DA-TDMA/TDM Subscriber Radio System SR500. He was the System Architect of this first digital point-to-multipoint wireless TDMA system. During 1985–2000, he was a Professor the Department of Electrical and Computer Engineering of Concordia University. Since 2000, he has been with the Department of Electrical and Computer Engineering of McGill University. His research interest is in the area of broadband digital communications with a special emphasis on modulation, coding, and multiple-access techniques. He is a senior member of the Ordre des Ingenieur du Quebec, a Fellow of the Institute of Electrical and Electronics Engineers (IEEE), a Fellow of the Engineering Institute of Canada (EIC), and a Fellow of the Canadian Academy of Engineering (CAE). He is the recipient of the 2004 Canadian Award in Telecommunications Research and recipient of the IEEE Canada Fessenden Award 2005.