QNA Inverse Model for Capacity Provisioning in Delay Constrained IP Networks

Irena Atov

Centre for Advanced Internet Architectures. Technical Report 040611A
Swinburne University of Technology
Melbourne, Australia
iatov@swin.edu.au

***Abstract*—This technical report is an extended version of a paper [1], which addresses the problem of dimensioning links in a multiservice IP network subject to satisfying varying performance requirements for different traffic classes. These performance requirements are viewed in terms of mean end-to-end delays required for the various traffic classes or in terms of random variations of their delays (i.e., jitter) or a combination of both.**

In this report we provide a detailed presentation and analysis of the recursive methods for inversion of the well-known QNA performance models, which [1] employs. Also, the accuracy of the model is investigated by means of a simulation study over an extended range of test cases. The results demonstate the capability of the model in guaranteeing the end-to-end delay requirements for the traffic classes.

I. Introduction

Generally defined, the capacity allocation (CA) problem considered in [1] is the problem of determining bandwidth allocations for the delay-sensitive traffic classes, as well as total bandwidth (capacities) of the links in the network so that multiple delay constraints for the traffic classes can be satisfied. Determining bandwidth allocations for the various traffic classes on the links is an important design issue as standard Weighted Fair Queueing (WFQ) service disciplines [2], [3] can only provide *tight* end-to-end delay guarantees for the classes if an adequate level of resources (in terms of bandwidth and buffer space) is allocated along their respective data paths through the network.

The CA problem poses the challenge of performing appropriate link dimensioning in order to balance quality of service against costly overprovisioning. For this problem, an accurate model for describing the characteristics of both the external IP traffic flows and the internal flows (on the links) is required. However, there is a tradeoff between the modelling power of the traffic descriptors used to describe the real traffic and the complexity that they impose when used in network planning functions. In this report, we present an extended version of a dimensioning model presented in [1] for the solution of the CA problem, which incorporates traffic characterisation procedures that allow burstiness of multi-class IP traffic to be effectively modelled. The selection of renewal traffic model i.e., GI arrival process, for this purpose, represents a reasonable balance between the accuracy (modelling power) and the efficiency required by a network planning tool.

Based on analysis of the Quality of Service (QoS) mechanisms employed in multiservice IP networks and their implications for network planning, an approximate model of QoS-based queueing mechanisms used at the routers was discussed in [1]. This approximate model assumes fixed bandwidth partitioning to split the link capacity between different traffic classes. Such a model allows us to consider the different traffic classes separately and determine their bandwidth allocations on the links independently. Figure 1 shows the network dimensioning process that we consider for the solution of the problem CA. The information required for this process includes:

- Traffic descriptors for the demands of the delay-sensitive classes,
- Routing information for the delay-sensitive classes,
- Local (link) partitions of the global (end-to-end) QoS constraints for the delay-sensitive classes,
- Network topology

The algorithms for the solution of Problem OPQR-G (Optimal QoS Partition and Routing in multicommodity flow network), discussed in [4], [5], when applied for each delay-sensitive class of traffic independently, will determine the required input for the dimensioning process; that is, the primary routes between the origin-destination (OD) pairs and the QoS partitions on the links for each class of traffic, respectively. We define a *class flow* as a single class of traffic between an OD pair. We model the class flow as a general (GI) arrival process characterised by a mean packet arrival rate and a squared coefficient of variation (SQV) of the packet inter-arrival time. For this renewal process, the coefficient of variation is used to characterise traffic burstiness, i.e., the variablity of the arrival stream. IP traffic is bursty in the sense that its squared coefficient of variation is always greater than or equal to unity. In [6], we have presented models, which can be used to translate the OD pair traffic demands for each class of traffic as obtained from traffic measurement data into equivalent class flow traffic descriptors (i.e., GI arrival process parameters) for direct application into the dimensioning procedure.

The dimensioning process can be summarised into three major stages (depicted in Fig. 1 as shaded boxes):

1) *Traffic-based decomposition model*. From the offered class flows to the network and the given routing information, obtain a characterisation of the internal (or internode) class flows by applying the methods for superposi-

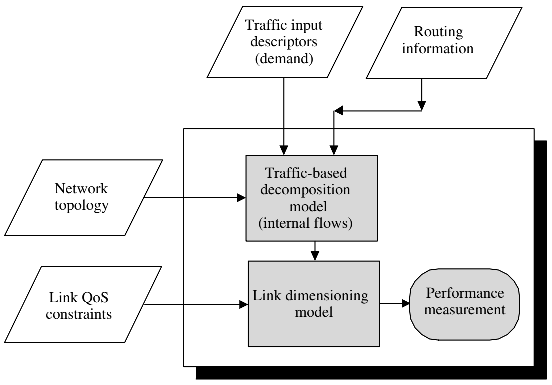

Fig. 1. Multiservice IP network dimensioning process

tion, departure and splitting of GI traffic arrival processes.

2) *Link dimensioning model*. On a link by link basis, given the internal class flows and their link QoS constraints, determine the bandwidth allocations required for the delay-sensitive classes, as well as, the total link capacities by using a link dimensining model.
3) *Dimensioning model validation*. Examine the end-to-end QoS performance of the class flows in the capacitated network by using a queueing network simulator.

It is important to emphasize here that, the solution of the CA problem, as obtained from the above dimensioning process, will determine the bandwidth allocations on the links for the delay-sensitive traffic classes. The so-called total link capacities obtained through this process, shall be used as *setup* link capacities in a further optimisation process that the network planner has to consider, as in practise link capacities are limited to a discrete set of values [7].

The rest of this report is organized as follows. In Section II, renewal-based traffic decomposition models are reviewed. A detailed description of the procedures that combine these models in order to achieve characterisation of the internal class flows for the specific network model considered in [1] is given in Section III. The algorithm for determining the class-based bandwidth allocations on the links is decribed in Section IV along with the recursive methods that it employs for inverting an appropriate link performance model. Finally, this report concludes with a simulation study that validates the dimensioning model used in solving the CA problem.

II. Renewal-based Traffic Decomposition Model

Traffic-based decomposition models encompass procedures required for modelling of the basic network operations of merging, departure and splitting, arising due to the common sharing of the resources and routing decisions in the network. In addition, they deliver approximations for performance measures (i.e., mean queue lengths, mean waiting times, the corresponding second moments, etc.) on both a per-queue and a per-network basis. In any case, only the steady-state of the network is considered.

For the traffic-based decomposition models that employ GI arrival processes as traffic descriptors, two different solution approaches can be identified: the successive/iterative approach and linearization. For a comprehensive study on traffic-based decomposition models see [8]. In the first case, starting from the external inputs, the basic network operations are applied successively one at a time for each node of the network in isolation. For each node, the departure (or output) process is approximated. According to the probabilistic routing after the node, the output traffic process is split and subsequently merged (or superposed) with other traffic processes in order to obtain the inputs for the nodes further downstream in the network. The internal flows and performance measures can be obtained in a single iteration step i.e., after single analysis of each node, if there are no feedback loops present in the network. Otherwise, additional iterations are required until convergence at the nodes within the feedback loop is reached. In the linearization solution approach, approximating linear equations are needed to describe the transformations for the parameters of the internal flows, arising from the basic network operations. The parameters of the internal flows are then obtained by solving a system of linear equations.

1) *Kühn's and Whitt's decomposition models:*

In 1979, Kühn published a traffic-based decomposition model for open networks of GI / G /1 queues [9]. The arrival processes and the service-time distributions are partially characterised by their first two moments or, equivalently, by λ_A (the mean arrival rate), c_A^2 (the SQV of the interarrival time), τ_S (the mean service time), and c_S^2 (the SQV of the service time). The performance at each node is described by approximate formulae that depend only on these parameters. This model employs a successive/iterative approach to solve for the internal flows and performance measures at the nodes in the network. Later, in 1983, Whitt extended Kühn's model by adding several new features and he also implemented his decomposition model into a software tool widely known as QNA (Queueing Network Analyzer) [10], [11]. Specifically, he improved and modified the merging operation for GI traffic arrival processes and applied a linearization solution approach for the network decomposition. He developed second-order traffic equations, which when combined with Jackson's first-order traffic equations [12] enable the two characteristic parameters of the internal flows to be derived from a system of linear equations.

In view of the two-parameter descriptions of the arrival process (λ_A, c_A^2) and service time (τ_S, c_S^2) at a considered node, the elementary calculus that transforms the two parameters of the internal flows for each of the three basic network operations, as given in [10], is described in the following:

a) *Superposition of GI traffic flows:*

The superposed process of n individual GI flows, each characterised by λ_j and c_j^2 $(j = 1, \ldots, n)$, as it enters the considered node is approximated by a GI traffic flow with parameters λ_A and c_A^2, representing the mean arrival rate and SQV of the inter-arrival time of the superposed flow, respectively. By conservation of flow, the mean arrival rate of the superposed flow is given by:

$$\lambda_\text{A} = \sum_{j=1}^{n} \lambda_j \; . \qquad (1)$$

For the derivation of the SQV of the inter-arrival time of the superposed flow, Whitt combines the stationary-interval method implemented by Kühn with the asymptotic method [13] into a hybrid procedure:

$$c_{\mathrm{A}}^2 = w \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_{\mathrm{A}}} c_j^2 + 1 - w \,, \tag{2}$$

with:

$$w = \frac{1}{1 + 4(1-\rho)^2(\nu - 1)} \,, \qquad \nu = \frac{1}{\sum_{j=1}^{n} \left(\frac{\lambda_j}{\lambda_{\mathrm{A}}}\right)^2}$$

and ρ is the *traffic intensity* or utilisation at the node, defined by $\rho = \lambda_{\mathrm{A}} \tau_S$. The superposition of renewal processes represents a nonrenewal point process as the inter-departure times are correlated. In this regard, the variability parameter represents the SQV of the interarrival time in the approximating superposed renewal process, but this does not mean that the dependence in the original process is completely ignored. Whitt conveys this dependence by making the variability parameter depend on the traffic intensity of the queue (even though the arrival process is independent of the service time in the queue), as he showed earlier [14] that the relevant covariances tend to depend on the traffic intensity in the queue.

b) Departure flow from a queue: Except for a very few specialized cases (e.g., $\mathrm{M/M}/m$ and $\mathrm{M/G}/\infty$ whose departure process is Poisson), the departure process from a queue represents a nonrenewal point process as the inter-departure times are correlated. The departure flow from a queue is approximated as a GI traffic flow, characterised by λ_{D} and c_{D}^2, representing the mean arrival rate and SQV of the inter-arrival time of the departure flow, respectively. Under equilibrium conditions, the mean flow entering a queue is always equal to the mean flow exiting the queue (by conservation of flow) and consequently $\lambda_{\mathrm{D}} = \lambda_{\mathrm{A}}$. The SQV of inter-arrival time of the departure flow is given by:

$$c_{\mathrm{D}}^2 = \rho^2 c_s^2 + (1 - \rho^2) c_{\mathrm{A}}^2 \,. \tag{3}$$

Whitt obtains the above result by applying a linear approximation to the stationary-interval method implemented by Kühn.

c) Splitting a GI flow Probabilistically: If a GI flow with parameters λ, and c^2 is split into n flows, each selected independently with probability q_i, the parameters for the i-th flow will be given by:

$$\lambda_i = q_i \lambda \,, \tag{4}$$

$$c_i^2 = q_i c^2 + (1 - q_i) \,. \tag{5}$$

These basic network operations are illustrated in Fig. 2. In addition to these basic operations, traffic based-decomposition models also provide a method for eliminating immediate feedback, which we do not discuss here, as such operation becomes redundant in the modelling of practical communication networks.

Despite the comprehensivness of QNA, the algoritmic procedure that we adopt for the solution of the internal flows in the network, which we discuss next, is based on the successive solution approach introduced by Kühn. Whitt's improved results for the basic network operations (as described above) are, however, considered in the procedure. In doing so, our intention is to develop a unique framework for implementation of the traffic-based decomposition (i.e., for the computation of the internal flows) that would also be employed when more complex traffic descriptors are considered.

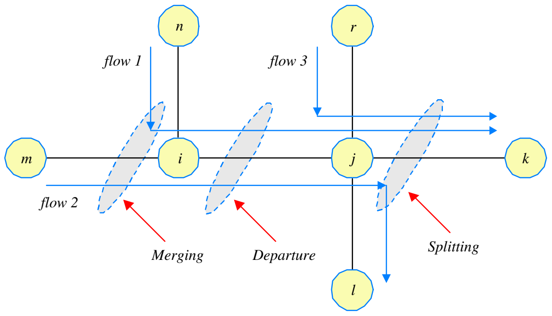

Fig. 2. Basic network operations

III. Modelling Internal Class Flows

After the preliminaries discussed in the previous section, we devote our attention to the actual algorithmic procedure used in the computation of the internal flows. The approximate network model discussed in [1] enables us to confine the CA problem to a network of single-server queues for each class of traffic, respectively, and determine the class-based flows and bandwidth allocations on the links independently. This network model is based on the assumption that a queue is associated with each link in the network. Accordingly, the departure process from a queue is the actual process (from a specific class) observed on a link. The departure processes from queues may be split and subsequently merged (also with external input traffic) in order to serve as inputs in the downstream queues. Note that, in reality, in this type of network, probabilistic splitting does not occur. Instead, the splitting is based on predetermined static routing matrix (resulting from the fixed routing assumption) and therefore is deterministic. Whitt discusses this in the context of "deterministic flows" and suggests an equivalent probabilistic routing interpretation for deterministic (fixed) routing. We shall use this probabilistic routing approximation subsequently.

From the input class flows and the given routing information, one can obtain a characterisation of the total traffic flows for each class $c\,(c = 1, \ldots, C)$ on every link in the network, which are also referred to as *internal* class flows. Each input class c flow between an OD pair $(u, v) \in \Pi$ is assigned to a fixed route r_c^{uv} $(r_c^{uv} \in \boldsymbol{r_c})$, where $\boldsymbol{r_c}$ is the set of routes for class c. We characterise an input class flow with the following set of parameters $\{\lambda_c, c_c^2, \overline{X_c}, \overline{X_c^2}\}_{r_c^{uv} \in \boldsymbol{r_c}}$. The first two parameters denote the mean packet arrival rate and the SQV of the packet inter-arrival times of the class c flow $(c = 1, 2, \ldots C)$, which has been assigned a route r_c^{uv} [1]. The third and fourth parameter denote the first two moments of the packet size of class c flow, respectively. We shall use the following notation:

- λ_{cl} - mean packet arrival rate of class c flow into the queue (reserved for class c) associated with the link l,

[1]Where required we will use superscript uv for the class flow parameters (λ_c, c_c^2) to designate to which OD pair $(u, v) \in \Pi$ the flow is assigned.

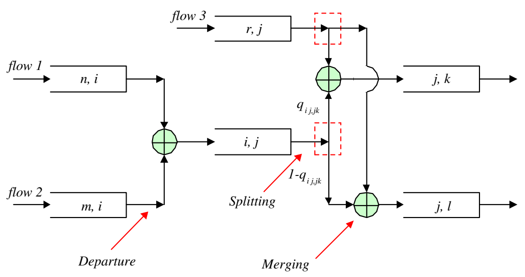

Fig. 3. Network queueing model

- c_{cl}^2 - SQV of the packet interarrival time of class c flow entering the queue (for class c) on the link l,
- $\rho_{cl} = \lambda_{cl}\tau_{cl}$ - traffic intensity (utilisation) of class c on link l.

By applying the method for superposition (1), the mean packet arrival rate of class c flow on a link l, can be determined as the sum of all class c flows, whose routes traverse the link l:

$$\lambda_{cl} = \sum_{(u,v)\in\Pi} \lambda_c^{uv} a_{cl}^{uv} \tag{6}$$

where a_{cl}^{uv} is an indicator function with value 1 if link l lies on route r_c^{uv} and 0 otherwise.

In order to obtain the variability parameter of the internal class flows i.e., c_{cl}^2 for $(l \in E, c \in C)$, we apply the approximate methods for merging, departure and splitting of GI traffic flows. We illustrate the procedure for the derivation of the variability parameter of the internal flows by means of the following example. Consider the problem of finding the variability parameter of a flow on a link l connecting nodes (j,k), denoted by c_{jk}^2, in the example network shown in Fig. 2 (for this illustration, we adopt the convention of representing a link with the pair of nodes that it connects). The two characteristic parameters for the three input flows depicted in the figure are considered given. The equivalent queueing network model is shown in Fig. 3.

The procedure for the derivation of the variability parameter of the internal flow on link (j,k) consists of the following three steps:

1) *Splitting* First, find all incoming links to node j. For the flow on each incoming link (i,j) $\forall i, i \neq j$, find the proportion of it, denoted by $q_{ij,jk}$, which enters the link (j,k) and subsequently find the mean arrival rate $\lambda_{ij,jk}$ and SQV of inter-arrival time $c_{ij,jk}^2$ of this fractional flow. The mean arrival rate $\lambda_{ij,jk}$ is readily determined from the routing information for the flows, as the sum of all flows whose routes traverse the links (i,j) and (j,k) in sequence. Similarly, from (6), the mean arival rate of the flow on each incoming link (i,j), λ_{ij}, is determined as the sum of all flows whose routes traverse that link. The proportion $q_{ij,jk}$ is then readily computed as:

$$q_{ij,jk} = \frac{\lambda_{ij,jk}}{\lambda_{ij}} \tag{7}$$

and this is subsequently used for the derivation of $c_{ij,jk}^2$ by applying (5).

2) *Merging* By applying (1) and (2) merge all the fractional flows $\lambda_{ij,jk}$, $c_{ij,jk}^2$ $(\forall i, i \neq j)$ to derive the mean rate and the SQV of the inter-arrival time of the flow entering link (j,k), represented by λ_{jk}^*, c_{jk}^{*2}.
3) *Departure* From the input flow to the queue associated with link (j,k) λ_{jk}^*, c_{jk}^{*2}, apply (3) to determine the departure flow (i.e., the actual flow on link (j,k), denoted by λ_{jk}, c_{jk}^2.

In step 1, the SQV of the fractional flow $c_{ij,jk}^2$ is found by applying the splitting method (5) to the SQV of the flow on link (i,j) as: $c_{ij,jk}^2 = q_{ij,jk}c_{ij}^2 + (1 - q_{ij,jk})$. Subsequently, we use the values of $c_{ij,jk}^2$ to determine the values of c_{jk}^{*2} and c_{jk}^2 in step 2 and step 3, respectively. Note that, the values of $c_{ij,jk}^2$, c_{jk}^{*2}, c_{jk}^2 will not be initially known. According to the successive approach for network decomposition, the computation of these values starts with the queues associated with the links that are incident to the nodes where the external flows enter the network, as the mean rate and the SQV parameter for the external flows are given. Then, by performing the calculus for split, merge and departure operations for each flow, the algorithm proceeds until the destination for each of the flows is reached. For the example in Fig 3, we start by computing the departure processes from queues (n,i) and (m,i) and subsequently merge these two processes to determine the input flow into queue (i,j). After the departure process from link (i,j) is derived, this flow is split according to deterministic routing into two fractional flows. The fraction of the flow on link (i,j), which enters the downstream queue (j,k) is derived as $q_{ij,jk} = \frac{\lambda_{\text{flow1}}}{\lambda_{\text{flow1+flow2}}}$. Finally, this fractional flow is merged with flow 3 to determine the input flow into the queue (j,k). Note that the fraction of flow 3 that goes next to queue (j,l) is actually equal to zero. As illustrated in this example, the computation of the variability parameters may be done directly by tracing each flow from its origin to destination. In this case, the ordering in which the queues are selected for processing plays a crucial role. This may be difficult to implement, and therefore, we employ the following iterative procedure.

Figure 4 depicts the iterative procedure that we employ in determining the parameters of the internal flows for each service class. The algorithm executes only if the condition that all queues in the network are stable is satisfied. The queue is stable if the traffic intensity (utilisation) is less than unity. The algorithm starts by initializing the variablity parameters of all flows before and after entering the queues associated with the links in the network, as well as the variablity parameters of all fractional flows. Specifically, the initialization process first sets the values of the variability parameters of all fractional flows to zero. Further, the variability parameters of all flows on the links (i.e., after the queues on the links) are set to zero. Finally, the variability parameters of all flows before entering the queues on the links are also set to zero, except for the links which represent first traversing links in the paths through the network of the external flows. The variability parameters of such flows are computed from the variability parameters of the external flows by applying the calculus for the merge operation (2) if more than one external flow traverses the given link. For clarity, in the flow chart (Fig 4) the queues (associated with the links in the network) are indexed by numbers from 0 to $E-1$, with E

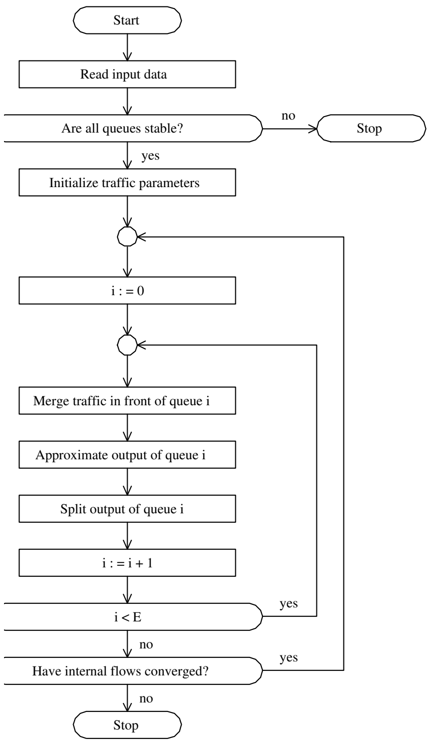
Start
Read input data
Are all queues stable?
no
Stop
yes
Initialize traffic parameters
i : = 0
Merge traffic in front of queue i
Approximate output of queue i
Split output of queue i
i : = i + 1
i < E
yes
no
Have internal flows converged?
yes
no
Stop

Fig. 4. Algorithm for determining internal flows

being the total number of links in the network. The algorithm proceeds by performing the calculus for the three basic operations for each queue independently. The order in which the algorithm selects the queues for processing, in general, may be arbitrary. After each queue in the network has been considered the algorithm will execute another iteration if a suitable convergence criterion has not been met. Convergence is reached when the maximum relative error of the variability parameters of the flows on all links in the network between two consecutive iterations becomes less than or equal to a predefined sufficiently small value. That is, for a given class c the stopping criteria is defined by $\max\left\{\left|\frac{c_{cl}^2-\widehat{c_{cl}^2}}{\widehat{c_{cl}^2}}\right|\right\}_{l\in E} \leq \epsilon$ with c_{cl}^2 and $\widehat{c_{cl}^2}$ denoting the variablity parameters of the internal class c flow on link l, as computed in the last and second last iteration, respectively, and $\epsilon = 0.0001$ (for example).

From the models for superposition of GI processes as they enter the queue and for the departure from a queue it should be noted that the respective input and output processes depend on the traffic intensity at the queue. Specifically, the variability parameter of class c flow on each link l in the network, c_{cl}^2, depends on the traffic intensity of the queue associated with class c on each link, ρ_{cl}. The traffic intensity ρ_{cl}, on the other hand, is a function of the portion of the capacity of link l allocated to class c, which we actually need to determine. Therefore, we shall calculate the internal flows (e.g., the variability parameters of the flows) iteratively, as part of the dimensioning procedure, as discussed in the next section.

IV. Determining Class-based Bandwidth Allocations on Links

The bandwidth required for each delay-sensitive class c on a link l, which we denote by b_{cl}, can be determined from the traffic characteristics of the total amount of class c traffic on a link l and its delay (QoS) constraint for that link q_{cl}, by inverting an approximate delay formula F; that is, find b_{cl} where:

$$q_{c\_l} = F(\lambda_{cl}, c_{cl}^2, b_{cl}) \qquad c = 1, 2, \ldots, \widehat{C}\ .$$

For this purpose, we analyse the GI / G /1 delay performance model. The delay (QoS) constraint for a traffic class on a link, q_{cl}, can be expressed in terms of the mean delay and the variance of the delay, which we denote as d_{cl}, and $\sigma_{d\_cl}^2$, respectively.

The mean delay of a packet from class c on a link l represents the sum of the waiting time of the packet in the queue until it is being serviced, W_{cl}, and the service time of the packet, τ_{cl}:

$$d_{cl} = W_{cl} + \tau_{cl}\ . \tag{8}$$

In a similar way, the variance of the delay for a packet of class c on a link l, is the sum of the variance of the waiting time of the packet in the queue, $\sigma_{W\_cl}^2$, and the variance of the service time of a packet, $\sigma_{\tau\_cl}^2$:

$$\sigma_{d\_cl}^2 = \sigma_{W\_cl}^2 + \sigma_{\tau\_cl}^2\ . \tag{9}$$

The mean delay for a packet of class c in the queue associated with class c on a link l may be approximated using the QNA approximation (Kramer and Langenbach-Belz approximation) [10] as follows:

$$W_{cl} = \frac{\tau_{cl}\rho_{cl}(c_{cl}^2 + c_{s\_cl}^2)g}{2(1-\rho_{cl})} \tag{10}$$

where $g \equiv g(\rho_{cl}, c_{cl}^2, c_{s\_cl}^2)$ is defined as

$$g(\rho_{cl}, c_{cl}^2, c_{s\_cl}^2) = \begin{cases} \exp\left[-\frac{2(1-\rho_{cl})(1-c_{cl}^2)^2}{3\rho_{cl}(c_{cl}^2+c_{s\_cl}^2)}\right] & \text{for} \quad \mathrm{c}_{\mathrm{cl}}^2 < 1\ , \\ 1 & \text{for} \quad \mathrm{c}_{\mathrm{cl}}^2 \geq 1\ , \end{cases} \tag{11}$$

and:

- τ_{cl} is the mean service time of a packet of class c on link l i.e, the time it takes for the mean sized packet of class c to be transmitted on the link which has capacity b_{cl}:

$$\tau_{cl} = \frac{\overline{X_c}s}{b_{cl}}\ , \tag{12}$$

- $c_{s\_cl}^2$ is the SQV of the service time for a packet of class c, which is the variance of the service time divided by the square of its mean and after some algebraic manipulation it can be shown that it is equal to the SQV of the packet size, $\sigma_{X\_c}^2$:

$$c_{s\_cl}^2 = \frac{\overline{X_c^2} - \overline{X_c}^2}{\overline{X_c}^2} = \sigma_{X\_c}^2\ , \tag{13}$$

- b_{cl} is the service rate or bandwidth allocated to class c (in bps) on link l, and s is a scaling factor ($s = 8$, as 1 byte = 8 bits).

The variance of the queueing delay for packets of class c in the queue may be approximated by using the QNA model result [10] as follows:

$$\sigma^2_{W\_cl} = W^2_{cl} c^2_{W\_cl} \tag{14}$$

where $c^2_{W\_cl}$ is the SQV of the queueing delay defined by

$$c^2_{W\_cl} = \frac{c^2_D + 1 - \chi}{\chi} \tag{15}$$

with χ denoting the probability of delay $\chi = P\{W_q > 0\}$, which is given by

$$\chi = \rho_{cl} + (c^2_{cl} - 1)\rho_{cl}(1 - \rho_{cl})h \, , \tag{16}$$

$$h = \begin{cases} \frac{1 + c^2_{cl} + \rho_{cl} c^2_{s\_cl}}{1 + \rho_{cl}(c^2_{s\_cl} - 1) + \rho^2_{cl}(4c^2_{cl} + c^2_{s\_cl})} & \text{for} \quad \mathrm{c}^2_{\mathrm{cl}} < 1 \, , \\ \frac{4\rho_{cl}}{c^2_{cl} + \rho^2_{cl}(4c^2_{cl} + c^2_{s\_cl})} & \text{for} \quad \mathrm{c}^2_{\mathrm{cl}} \geq 1 \, , \end{cases} \tag{17}$$

and

$$C^2_D = 2\rho_{cl} - 1 + \frac{4(1 - \rho_{cl})\alpha}{3(c^2_{s\_cl} + 1)^2} \, , \tag{18}$$

$$\alpha = \begin{cases} 3c^2_{s\_cl}(1 + c^2_{s\_cl}) & \text{for} \quad \mathrm{c}^2_{\mathrm{s\_cl}} \geq 1 \, , \\ (2c^2_{s\_cl} + 1)(c^2_{s\_cl} + 1) & \text{for} \quad \mathrm{c}^2_{\mathrm{s\_cl}} < 1 \, . \end{cases} \tag{19}$$

The variance of the service time of a packet, $\sigma^2_{\tau\_cl}$, is given by: $\sigma^2_{\tau\_cl} = \tau^2_{cl} c^2_{s\_cl}$.

The algorithm for determining the bandwidth allocations for the delay-sensitive classes is described in figure 5. Based on the given topology, offered class flows, routing information and link delay QoS constraints for each class of traffic, the algorithm returns class-based bandwidth allocations for all links in the network. Initialization of the algorithm involves assigning starting values for the service times at the nodes for each traffic class, respectively. The service times for the classes are specified by the two parameters $(\tau_{cl}, c^2_{s\_cl})$ and their initial values are computed from the first two moments of the packet size for the classes and their initial fixed capacity shares on the links, $\widehat{b_{cl}}$, by applying (12) and (13), respectively. The fixed capacity shares of each class on the links, $\widehat{b_{cl}}$, are initially set to sufficiently large values in the following way: $\widehat{b_{cl}} \geq \frac{\lambda_{cl}\overline{X_c s}}{\rho_{\text{init}}}$, where ρ_{init} is the value for the initial traffic intensities of all link queues in the network i.e., $\rho_{cl} = \rho_{\text{init}}$ $(l \in E)$, which we set to an appropriately small value e.g., 10^{-3}. The mean arrival rates of the internal class flows λ_{cl} $(l \in E, c \in \widehat{C})$ are computed from (6).

Once the initialisation step is completed the variability parameters of the internal class flows are computed c^2_{cl} $(l \in E, c \in \widehat{C})$, by applying the algorithm described in figure 4. After the internal flows have been determined, the algorithm cycles through the classes and, on a link by link basis, determines their required bandwidth allocations b_{cl} by numerically inverting a delay formula, a variance of delay formula or both depending on the type of class considered. For *delay-sensitive* type service class we need to find the value of its bandwidth allocation on a link for which the resulting link delay would be less than or equal to its link delay constraint. Similarly, for *jitter-sensitive* type service class we need to find the value of its bandwidth allocation on a link such that its jitter link constraint is satisfied. In the case of *delay- and jitter-sensitive* service class we are given two constraints for each link, one for the delay and one for the jitter, respectively. As before, we need to find the bandwidth allocations which are required to satisfy these constraints individually. The bandwidth allocation for this class would then be the maximum of these two values.

```
Input:  $G(V,E), \boldsymbol{r_c} = \{r^{uv}_c\}_{(u,v)\in\Pi}, \boldsymbol{q_c} = \{q_{cl}\}_{l\in E}$,

$\boldsymbol{\Lambda_c} = \{(\lambda_c, c^2_c, \overline{X_c}, \overline{X^2_c})\}_{r^{uv}_c \in \boldsymbol{r_c}}$ for $c \in \widehat{C}$.
Output: $\boldsymbol{b_{cl}} = \{b_{cl}\}_{l\in E}$ for all QoS sensitive classes.

1    for class $c = 1$ to $C - 1$
2        for all link $(l \in E)$
3            $\tau_{cl} \leftarrow \frac{\overline{X_c s}}{\widehat{b_{cl}}}$, $c^2_{s\_cl} \leftarrow c^2_{X\_c}$
4    for class $c = 1$ to $C - 1$
5    switch $(\boldsymbol{q_c})$
6        case $\boldsymbol{q_c} = \{d_{cl}\}_{l\in E}$
6a           for all link $(l \in E)$
6b               Calculate internal flows: $\{(\lambda_{cl}, c^2_{cl})\}_{l\in E}$,
6c               Invert the delay formula (8) to derive $b_{cl}$,
6d               if $\max\left\{\left|\frac{b_{cl} - \widehat{b_{cl}}}{\widehat{b_{cl}}}\right|\right\}_{l\in E} > \epsilon$
6e               $\widehat{b_{cl}} \leftarrow b_{cl}$ for $\forall l$ and repeat steps 3,6.
7        case $\boldsymbol{q_c} = \{\sigma^2_{d\_cl}\}_{l\in E}$
7a           The same as step 6, except in 6c, invert the
             variance formula (9) to derive $b_{cl}$.
8        case $\boldsymbol{q_c} = \{d_{cl}, \sigma^2_{d\_cl}\}_{l\in E}$
8a           do steps 6 and 7 to get $(b_{cl})_{\text{step 6}}$ and $(b_{cl})_{\text{step 7}}$ for $\forall l$,
8b           $b_{cl} \leftarrow \max\{(b_{cl})_{\text{step 6}}, (b_{cl})_{\text{step 7}}\}$ for $\forall l$.
9    return the set of class-based link capacities $\boldsymbol{b_{cl}}$.
```

Fig. 5. Algorithm to determine class-based bandwidth allocations

The algorithm calculates the internal flows and the class-based bandwidth allocations iteratively, until the class-based bandwidth allocations on the links converge. The iterative process stops when the maximum relative error of the class-based bandwidth allocations on all links between two consecutive iterations becomes less than or equal to a specified sufficiently small value e.g., $\max\left\{\left|\frac{b_{cl} - \widehat{b_{cl}}}{\widehat{b_{cl}}}\right|\right\}_{l\in E} \leq \epsilon$ where $\epsilon = 0.0001$.

For inverting the delay and the variance formulae in steps 7c. and 8c., respectively, we have used two different iterative methods, which we discuss in the next section. One is based on Newton's method and the other one is based on an algebraic transformation of the given formula. In practice, we have found both methods converge very rapidly, so that either method can be used.

Having obtained the bandwidth required for each delay-sensitive traffic class on a link, b_{cl} $(c = 1, 2, .., \widehat{C})$, the total capacity of the actual link (i.e., link setup capacity), can be determined as the minimum capacity of all available capacities on the link that is higher than the linear sum of the individual bandwidth allocations for the classes i.e.:

$$\theta_l^{\text{setup}} = mod \left\lceil \sum_{c=1}^{\widehat{C}} b_{cl} \right\rceil \tag{20}$$

where $mod\lceil\cdot\rceil$ is a ceiling function. The set of these link setup capacities θ_l^{setup} ($l \in E$) will determine the required input for the solution of the optimisation problem considered in [7].

A. Iterative Methods for Class-based Bandwidth Allocations

As discussed in the previous section, the bandwidth required for a delay-sensitive class on a link can be determined from the traffic characteristics of its total flow on the link and its delay (QoS) constraint for that link by inverting an approximate delay performance formula. Based on the type of the delay (QoS) constraint for the class, an inversion of a delay and/or variance of the delay formula is performed for this purpose. In the following, we outline the iterative procedures employed for the inversion of a link delay formula. The details of the respective procedures for the variance of the delay formula can be found in Appendix B.

The total time of a packet spent at a queue associated with a link, represents the sum of the waiting time in the queue and the time it takes for the packet to be transmitted on the link. Thus, the total delay for a packet of class c at link l is given by:

$$d_{cl} = \frac{\rho_{cl}}{\lambda_{cl}} \left[1 + \frac{\rho_{cl}(c_{cl}^2 + c_{s\_cl}^2)g}{2(1-\rho_{cl})} \right] . \tag{21}$$

Considering the above link delay formula, we want for a given link delay constraint, d_{cl}, traffic arrival and service characteristics, $\lambda_{cl}, c_{cl}^2, c_{s\_cl}^2$, to obtain a solution for the link queue utilisation ρ_{cl}. Having determined the link queue utilisation, ρ_{cl}, we can directly derive the mean service time $\tau_{cl} = \frac{\rho_{cl}}{\lambda_{cl}}$, as well as the service capacity on the link l i.e., the bandwidth allocation for class c on a link l, $b_{cl} = \frac{\lambda_{cl}\overline{X_c}s}{\rho_{cl}}$, ($\lambda_{cl}\overline{X_c}s$ is the mean flow rate of class c in bit/sec entering the queue at link l). Note that, in the delay formula given above the parameter g can take one of two possible values, depending on the SQV of the arrival flow in the queue, c_{cl}^2 (see equation (11)) and, thus, two iteration procedures are required for the computation of ρ_{cl} (one for each case).

For the solution of ρ_{cl} in the above delay formula we employ two standard iterative methods for solving equations of the form $f(x) = 0$. Thus, we first rewrite (21) in this form for both cases (the subscript indexing the link queue cl is suppressed in the following for clarity):

Case 1: $c^2 \geq 1$

$$f_1(\rho) = \rho^2(c^2 + c_s^2 - 2) + 2\rho(1+\lambda d) - 2\lambda d = 0 \ , \tag{22}$$

Case 2: $c^2 < 1$

$$f_2(\rho) = \rho^2 \left[(c^2 + c_s^2)e^{-A} - 2 \right] + 2\rho(1+\lambda d) - 2\lambda d = 0 \tag{23}$$

where:

$$A = \frac{2(1-\rho)(1-c^2)^2}{3\rho(c^2 + c_s^2)} .$$

1) Algebraic Transformation Iterative Method: In the first iterative method, the equations (22) and (23) are transformed algebraically into the form $\rho = g(\rho)$. Then the corresponding iteration procedure is given by $\rho_{n+1} = g(\rho_n)$. The iteration process defined in this way, is said to be convergent for ρ_0 if the corresponding sequence $\rho_0, \rho_1, \rho_2, \ldots$ is convergent. A sufficient condition for convergence is $|g'(\rho) < 1|$. In order to apply this method, (22) is transformed in the following way:

$$\rho = g_1(\rho) = \frac{2\lambda d}{\rho(c^2 + c_s^2 - 2) + 2(1+\lambda d)} \ ,$$

which results in the following iteration procedure for case 1 (i.e., $c^2 \geq 1$):

$$\rho_{n+1} = \frac{2\lambda d}{\rho_n(c^2 + c_s^2 - 2) + 2(1+\lambda d)} \quad (n = 0, 1, 2, \ldots) \ . \tag{24}$$

Similarly, for case 2 (i.e., $c^2 \leq 1$), (23) can be transformed in the following way:

$$\rho = g_2(\rho) = \frac{2\lambda d}{\rho\left[(c^2 + c_s^2)e^{-A} - 2\right] + 2(1+\lambda d)} \ ,$$

and consequently the iteration procedure for case 2 (i.e., $c^2 < 1$) is given by:

$$\rho_{n+1} = \frac{2\lambda d}{\rho_n\left[(c^2 + c_s^2)e^{-A} - 2\right] + 2(1+\lambda d)} \quad (n = 0, 1, 2, \ldots) \ . \tag{25}$$

2) Newton's Iterative Method: Newton's method (also called the Newton-Raphson method) can also be used for solution of equations of the form $f(x) = 0$, where f is differentiable. In some cases, it produces faster convergence than the previous method. According to this method, the general formula for the iteration procedure in our case is given by:

$$\rho_{n+1} = \rho_n - \frac{f(\rho_n)}{f'(\rho_n)} \quad (n = 0, 1, 2, \ldots) \ . \tag{26}$$

Specifically, the iteration procedure for case 1 (i.e., $c^2 \geq 1$), where $f(\rho)$ is given by (22), after some algebraic transformations can be written as follows:

$$\rho_{n+1} = \frac{\rho_n^2(c^2 + c_s^2 - 2) + 2\lambda d}{2\rho_n(c^2 + c_s^2 - 2) + 2(1+\lambda d)} \ . \tag{27}$$

Similarly, the iteration procedure for case 2 (i.e., $c^2 < 1$), where $f(\rho)$ is given by (23), after some algebraic transformations can be written as follows:

$$\rho_{n+1} = \frac{\rho_n^2 \left[\gamma \left(1 - \rho_n A \frac{\partial A}{\partial \rho_n} \right) - 2 \right] + 2\lambda d}{\rho_n \left[\gamma \left(2 - \rho_n A \frac{\partial A}{\partial \rho_n} \right) - 4 \right] + 2(1+\lambda d)} \tag{28}$$

with

$$A\frac{\partial A}{\partial \rho_n} = -\frac{4(1-\rho_n)(1-c^2)^4}{9\rho_n^3(c^2 + c_s^2)^2} \qquad \text{and} \qquad \gamma = (\mathrm{c}^2 + \mathrm{c}_\mathrm{s}^2)\mathrm{e}^{-\mathrm{A}} \ .$$

For the inversion of the variance of the delay formula, given in (9), four iteration procedures are required for the computation of ρ_{cl}, as there are four distinctive cases to be considered.

Depending on the SQV of the arrival flow in the queue, h in (16) can take two possible values. Similarly, α in (18) can take one of two possible values, depending on the SQV of the service time for a packet, thus, resulting in altogether four possible cases for the variance formula, which need to be considered independently in the iterative procedures. For details on the iterative procedures employed for the variance of the delay formula, see Appendix A.

V. Simulation Results

The most critical components of the dimensioning methodology presented in this report with respect to accuracy are the approximate models comprising the renewal-based traffic decomposition. In this traffic decomposition model, individual queues are analysed separately after approximately characterising the external and internal flows as renewal processes. The renewal processes are characterised by two parameters representing the arrival rate and SQV of interarrival time, respectively. The arrival rates of the internal flows are computed exactly, whereas the variability parameters are obtained approximately and this represents the major approximation of this model. Computing performance measures for each queue, given its arrival and service parameters, also involves an approximation, but the quality of this approximation is relatively well understood, and is pretty good, as reported in [16]. The greatest difficulty with this decomposition model is determining appropriate variability parameters for the arrival processes to the queues.

In the following, we present a simulation study to assess the capability of the proposed dimensioning methodology in delivering the end-to-end delay QoS guarantees for the traffic classes. The test scenarios used in our simulation study are simple, but sufficiently illustrative, to indicate the quality that can be expected from the proposed dimensioning method across a wide range of traffic input parameters and traffic intensities at the nodes.

For this study, we have used the ns-2 simulation tool [17]. It is a stochastic discrete-event network simulation tool which, since its development in 1989, has become very popular within the research community. As ns-2 is an open-source project, it is possible to integrate new components or modify existing implementations of components within the object-oriented architecture of ns-2. This enabled us to construct specific simulation scenarios relevant for our study. In addition to ensuring that a valid simulation model is used, two other main issues that have to be addressed for ensuring validity of a simulation study are [18]: (1) application of appropriate pseudo-random generators of independent uniformly distributed numbers, and (2) appropriate analysis of simulation output data.

The source of randomness in ns-2 is provided by a random number generator (RNG), which implements the well-known "minimal standard" generator, originally designed for the IBM System/360 [19] and later improved by Park and Miller in [20]. However, a main shortcoming of this RNG is the length of its cycle $2^{32} - 1$, which with the recent advances in computing power becomes obsolete in all but very short lasting simulation studies. In order to overcome this pitfall, we have integrated the current state-of-the-art RNG, known as the *Mersenne Twister* into the ns-2 RNG component. The Mersenne Twister RNG, proposed by Matsumoto and Nishimura in [21], has an astronomical cycle length of $2^{19937} - 1$, and good virtual randomness in up to 623 dimensions, for up to 32-bit accuracy. The source code for this RNG is freely available at the Mersenne Twister home page [22].

In a simulation study, a sampling experiment is performed on a set of random variables, and, as in traffic measurements, the output is subject to statistical errors. Statistical error associated with a result obtained from a statistical experiment is measured by the corresponding confidence interval at a given confidence level. As the width of a confidence interval (in any correctly implemented simulation) decreases with the number of samples taken, we have performed sufficiently long simulation runs to permit useful interpretation of the results.

There are a number of possible methods for recording the statistics from a simulation study in order to determine confidence intervals for the performance indicators that are under study. Our approach throughout this thesis is to use the replication method with individual warm-up periods prior to each simulation. A simulation study consists of a number of runs where, after each run, statistics on the performance indicators (i.e., end-to-end delay) are taken and accumulated for each traffic stream in the network. As we are interested in studying the behaviour of the network in the steady-state, data collected during the warm-up periods are not used to calculate the estimates of the performance measures. At the conclusion of the simulation, the accumulated statistics are used to find the average performance of the traffic streams and confidence limits are obtained using standard statistical theory. For calculating steady-state confidence limits of mean values, see for example [23].

The external arrival processes in the proposed dimensioning model represent class-based aggregates offered to the network and along with their associated static routes they define the set of traffic streams (or class flows) in the network. In our simulation experiments each traffic stream (TS) is modelled as a hyperexponential process, which represents a suitable model for a bursty renewal traffic since its squared coefficient of variation of inter-arrival times is always greater than unity. Specifically, the interarrival times for the packets are generated according to hyperexponential distribution with two transient states H_2^b, i.e., the mixture of two exponential distributions with balanced means: one with mean m_1 realized with probability p and the other with mean m_2 realized with probability $1 - p$, where $pm_1 = (1 - p)m_2$.

Analysis of the results for the performance of the dimensioning model is performed by applying the following three-step procedure:

1) First, for a given network model, set of traffic streams and their corresponing end-to-end delay, as well as, link delay requirements, run the CA algorithm to determine the required bandwidth allocations on the links in the network.
2) Run ns-2 for the same network model and set of traffic streams by setting the link capacities in the network according to the output results of the CA algorithm.
3) Compare the delay statistics for the OD pairs in the network obtained from the simulator with the target values specified at input, in order to examine the capability of the dimensioning model to deliver the end-to-end delay

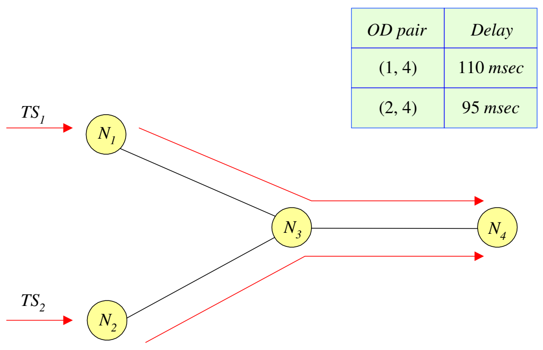

Fig. 6. Simple test network

TABLE I
INPUT TRAFFIC STREAMS

Number	*Mean* - λ	*SQV* - c^2
1	0.5	1.5
2	1.0	2.0
3	1.5	3.5
4	2.0	6.0
5	3.0	8.0

guarantees.

Since, the dimensioning model considers the different traffic classes separately and determines their bandwidth allocations on the links independently, in the experiments we have considered the case of a single delay-sensitive service class only. The purpose is to assess the capability of the proposed dimensioning methodology in determining the bandwidth allocations on the links in the network for a given delay-sensitive service class so that the end-to-end delay QoS requirements for that class are guaranteed. Since the network dimensioning model effectively does not take account of the statistical multiplexing at the nodes, the capability of the designed network in delivering the end-to-end delay QoS guarantees for a given service class will always be better than the performance of the individual service class case.

The specific network model that we analyse consists of 4 nodes and 3 links (see Fig 6). A single service class is considered and the traffic offered to the network is comprised of a number of traffic streams n, assigned between each OD pair i.e., OD pair $(1, 4)$ and $(2, 4)$, respectively. We consider twenty different cases for the traffic load in the network involving two values of n, $n = 10$ and 20 (which result in total of 20 and 40 traffic streams in the network, respectively) and five different types of traffic streams, each having different values for the mean packet arrival rate (λ) and SQV of interarrival times (c^2) as defined in Table I. The packet size for the traffic streams is set to a constant value of 1000 bytes, thus, our model results in a network of GI/D/1 queues. Note that we are testing our model with very bursty traffic streams whose arrival rates vary significantly, as for example, TS 5 has a packet arrival rate that is six times higher than the rate of TS 1. Whitt reported in [11], [15] that the reliability of the approximations used in the QNA decreases when the variability of the external arrival processes

TABLE II
END-TO-END PACKET DELAY RE FOR VARIOUS TRAFFIC INPUTS AND TRAFFIC INTENSITIES AT THE NODES FROM 0.30 TO 0.65

Simulation $d_{(1,4)}$ *[msec]*	*Simulation* $d_{(2,4)}$ *[msec]*	*CA model RE for* $d_{(1,4)}$	*CA model RE for* $d_{(2,4)}$
107.75 ($\pm$ 3.2e-5)	92.51 ($\pm$ 2.0e-5)	-2.04 −−	-2.62 −−
107.04 ($\pm$ 2.9e-4)	89.88 ($\pm$ 1.3e-5)	-2.69 −−	-5.38 −−
106.05 ($\pm$ 2.3e-5)	87.31 ($\pm$ 1.1e-5)	-3.59 −−	-8.08 −−
104.82 ($\pm$ 4.6e-5)	83.12 ($\pm$ 2.4e-5)	-4.70 −−	-12.50 −−
105.80 ($\pm$ 7.5e-5)	88.82 ($\pm$ 4.7e-5)	-3.81 −−	-6.50 −−
105.01 ($\pm$ 3.3e-5)	86.89 ($\pm$ 4.2e-5)	-4.53 −−	-8.53 −−
104.04 ($\pm$ 6.3e-5)	82.38 ($\pm$ 5.4e-5)	-5.41 −−	-13.28 −−
102.58 ($\pm$ 2.8e-5)	85.91 ($\pm$ 1.9e-4)	-6.74 −−	-9.56 −−
101.99 ($\pm$ 1.3e-4)	82.27 ($\pm$ 9.2e-5)	-7.28 −−	-13.40 −−
99.35 ($\pm$ 4.5e-4)	81.73 ($\pm$ 5.3e-4)	-9.68 −−	-13.96 −−
−−	−−	5.04	9.37

increases and when the arrival processes are not in the same time scale (i.e., do not have similar rates). Thus, our intention is to test the dimensioning model with traffic loads of such characteristics that can potentially expose its worst performance.

The packet delay requirements for the traffic aggregates between the OD pairs is 110 msec and 95 msec, respectively (see Fig 6). The link delay requirements are as follows: $d_{1-3} = 85$ msec, $d_{2-3} = 70$ and $d_{3-4} = 25$ msec, where the subscript denotes the connecting nodes of the link.

The first test we performed considered $n = 10$ traffic streams of type i between the OD pair $(1, 4)$ and 10 traffic streams of type j between the OD pair $(2, 4)$, where $i, j = \{1, 2, 3, 4, 5\}$ and $i \neq j$. In this case, all combinations for (i, j) were considered, which results in 10 different traffic load scenarios. The link bandwidth allocations obtained from the CA algorithm for all traffic load scenarios considered in this test resulted in traffic intensities at the nodes (links) $\rho_i (i = 1, 2, 3)$ that fall in the range from 0.30 to 0.65 (i.e., $\rho_i = \frac{\lambda_i 1000 \cdot 8}{b_{li}}$ where b_{li} is the link bandwidth in bps). Thus, this simulation test assesses the accuracy of the dimensioning model for the $(0.30 - 0.65)$ range of traffic intensities at the nodes in the network.

For each test scenario (i.e., (i, j) pair) network simulation experiment was set up based on the link capacities obtained from the dimensioning model. For each experiment we performed nine simulations with different seeds for the random number generator. The estimates of the mean packet delays were computed based on the replication method and 95-percent confidence intervals were obtained assuming a Student-t distribution. The results for all combinations of (i, j) traffic streams and $n = 10$ are summarised in Table II. Each row in the table indicates the traffic load scenario considered in the test. For example, the test scenario in the first row consists of 10 traffic streams of type 1 assigned between the OD pair $(1, 4)$ and 10 traffic streams of type 2 between the OD pair $(2, 4)$. The first and second column show the results obtained from the simulations for the mean packet delay of the traffic aggregates between the node pairs $(1, 4)$ and $(2, 4)$, respectively, in the capacitated network. The confidence limits are given in parentheses below the simulation estimates. Columns three and four show the relative percentage errors for the mean packet delays between the two node pairs with respect to the target delays, respectively, which are defined as

$$\mathrm{RE} = \frac{\text{Simulation delay} - \text{Target delay}}{\text{Target delay}} \cdot 100\% \,. \qquad (29)$$

At the bottom of the table are the average absolute relative percent errors (ARE) for each OD pair, respectively, which are defined as $\mathrm{ARE} = \sum_{\mathrm{i}=1}^{10} |\mathrm{RE_i}|/10$.

The table shows the difference between the end-to-end packet delays in the capacitated network and the delay requirements given at input for various traffic loads for which the resulting traffic intensities at the nodes fall in the range $(0.30 - 0.65)$. It can be seen that, for this range of traffic intensities, the dimensioning model performs well with ARE for the two OD pairs of 7.20%.

The second test we performed considered $n = 20$ traffic streams of type i between the OD pair $(1, 4)$ and 20 traffic streams of type j between the OD pair $(2, 4)$, where $i, j = \{1, 2, 3, 4, 5\}$ and $i \neq j$. Similarly to the previous test, all combinations for (i, j) were considered, which results in 10 different traffic load scenarios. Also, the same packet delay requirements for the traffic aggregates between the two OD pairs, as well as, link delay requirements are considered. Based on the given traffic demands for this set of test scenarios and the delay requirements, the link bandwidths computed by the CA algorithm resulted in traffic intensities at the nodes, which fall in the range from $\rho_i = 0.50$ to $\rho_i = 0.90$ $(i = 1, 2, 3)$. The results for all combinations of (i, j) traffic streams and $n = 20$ are summarised in Table III.

The results in Table III show that for the case of various traffic loads, which result in traffic intensities at the nodes in the range $(0.50 - 0.90)$, the dimensioning model performs well with ARE for the node pairs delay requirements of 8.97%. These results together with the results from Table II lead us to conclude that the dimensioning model performs well with repect to guaranteeing the end-to-end delay requirements for the traffic demands in the case of various tarffic loads and traffic intensities at the nodes in the range of $(0.30 - 0.90)$. In our analysis we did not consider the case where traffic intensities at the nodes fall out of this range as such cases are not of practical interest when considering a well planned functional network.

In all cases, the dimensioning model slightly overprovisions the network, as the mean delays obtained in the capacitated network are always less than the specified end-to-end delay constraints. Thus, the performance of the network is guaranted to be better than what is required by the delay-sensitive traffic classes. The results also indicate, as expected, that the reliability of the approximations used in QNA and, thus the accuracy of the dimensioning model, decreases when the variability of the external arrival processes increases (e.g., $c^2 > 6$) and when the arrival processes have rates that vary significantly.

TABLE III
END-TO-END PACKET DELAY RE FOR VARIOUS TRAFFIC INPUTS AND TRAFFIC INTENSITIES AT THE NODES FROM 0.50 TO 0.90

Simulation $d_{(1,4)}$ *[msec]*	*Simulation* $d_{(2,4)}$ *[msec]*	*CA model RE for* $d_{(1,4)}$	*CA model RE for* $d_{(2,4)}$
105.95 (± 8.1e-5)	90.46 (± 9.3e-5)	-3.68 $--$	-4.77 $--$
104.20 (± 8.7e-5)	87.15 (± 8.2e-5)	-5.27 $--$	-8.26 $--$
102.88 (± 1.0e-4)	85.95 (± 1.2e-4)	-6.47 $--$	-9.52 $--$
100.61 (± 2.2e-4)	81.79 (± 2.5e-4)	-8.53 $--$	-13.90 $--$
103.40 (± 3.0e-5)	86.04 (± 3.6e-4)	-6.00 $--$	-9.43 $--$
101.73 (± 7.3e-4)	85.18 (± 6.2e-4)	-7.51 $--$	-10.33 $--$
101.08 (± 4.9e-5)	81.34 (± 5.3e-4)	-8.10 $--$	-14.37 $--$
101.12 (± 3.7e-4)	84.59 (± 4.2e-4)	-8.07 $--$	-10.95 $--$
99.58 (± 5.0e-4)	82.60 (± 5.2e-4)	-9.47 $--$	-13.05 $--$
99.95 (± 6.3e-4)	83.01 (± 6.8e-4)	-9.13 $--$	-12.62 $--$
$--$	$--$	7.22	10.72

VI. CONCLUSIONS

In this report, we presented a dimensioning model for the solution of the CA problem. Specifiicaly, it determines bandwidth allocations for the delay-sensitive traffic classes, as well as the total continuous bandwidth of the links by taking into account the varying delay requirements (QoS) of the traffic classes. This model allows the burstiness of IP traffic to some extent to be modelled, however, it cannot handle correlations often observed in real input traffic. Although it lacks modelling accuracy in describing the characteristics of the external and internal IP traffic flows, this model is very computationaly efficient and, therefore, is appropriate to be used as a solution method for problem CA when large size networks are being considered. Not only should this model be used in this case, due to its efficiency, but also because it provides a reasonable solution method when large networks, as well as large aggregates of individual flows into service classes are considered. This is due to the multiplexing which occurs on a very large scale in this case and, as a result, the correlations significantly reduce due to the intermixing of packets from different traffic streams.

The proposed algorithm for the solution of the CA problem consists of two main algorithmic steps. In the first step, from the offered class flows to the network and the given routing information characterisation of the internal class flows is achieved by applying the methods for superposition, departure and splitting of GI traffic arrival processes as provided by the famous QNA approach. For this purpose we have incorporated the QNA procedures in an efficient algorithm for computation of the variability parameters of the internal flows. Having determined the internal class flows, the second step employs an algorithm, which on a link by link basis determines the class-based bandwidth allocations by numerically inverting a delay formula, a variance of delay formula or both depending on the type of class considered. For the inversion of the delay and variance of the delay formulae we have devised two iterative procedures.

Although, the required building blocks for consideration of a service class which is sensitive to the variations of the delay (i.e., jitter) are developed, they havden not been implemented into the dimensioning tool. Therefore, in this report we have evaluated the capability of the dimensioning model in guranteeing the end-to-end delays only for the service classes. The simulation study validated the modelling approach and confirmed that in the case of renewal traffic inputs the dimensioning model performs well with respect to guaranteeing the end-to-end delay requirements for the traffic demands. The accuracy of the model has been demonstrated for various burstiness characteristics of the traffic load and for a wide range of the traffic intensities at the nodes.

Finally, we recognize that the dependence (i.e., correlation of inter-arrival times) in both external and internal arrival processes deserves further emphasis, especially when smaller sized networks are considered and, therefore, in [24] we have considered development of a dimensioning model that takes account of this dependence.

Appendix A

The following four distinctive cases have to be considered in the iteration procedures for inversion of the variance of the delay formula, given in (9) (the subscript indexing the link queue c_l is suppressed for clarity):

- Case 1: $c^2 \geq 1$ and $c_s^2 \geq 1$,
- Case 2: $c^2 \geq 1$ and $c_s^2 < 1$,
- Case 3: $c^2 < 1$ and $c_s^2 \geq 1$,
- Case 4: $c^2 < 1$ and $c_s^2 < 1$.

For the solution of ρ in the variance of the delay formula (from which we can directly obtain the service capacity or bandwidth required on a link) we employ the two iterative methods for solving equations of the form $f(x) = 0$, as discussed in Section IV-A. For this purpose, we first rewrite (9) in this form in a general way that is applicable to all four cases, where the subscript i indicates the case under consideration:

Case i:

$$f_{(i)}(\rho) = \frac{\rho^2}{\lambda^2}c_s^2 + \left(\frac{\rho^4(c^2+c_s^2)^2 g_{(i)}^2}{4\lambda^2(1-\rho)^2}\right)\left(\frac{c_{D(i)}^2+1}{\chi_{(i)}} - 1\right) - \sigma_d^2 = 0 \tag{A.1}$$

where both $g_{(i)}$ and $\chi_{(i)}$ depend only on the variability parameter of the arrival process, c^2, and $c_{D(i)}^2$ depends only on the variability parameter of the service time c_s^2. Their respective values for all four cases are as follows:

$$c_{D(i)}^2 = \begin{cases} 2\rho - 1 + \frac{4(1-\rho)c_s^2}{c_s^2+1} & i = 1,3\,(c_s^2 \geq 1) \\ 2\rho - 1 + \frac{4(1-\rho)(2c_s^2+1)}{3(c_s^2+1)} & i = 2,4\,(c_s^2 < 1)\,, \end{cases}$$

$$\chi_{(i)} = \begin{cases} \rho + \frac{4\rho^2(1-\rho)(c^2-1)}{c^2+\rho^2(4c^2+c_s^2)} & i = 1,2\,(c^2 \geq 1) \\ \rho + \frac{\rho(1-\rho)(c^2-1)(1+c^2+\rho c_s^2)}{1+\rho(c_s^2-1)+\rho^2(4c^2+c_s^2)} & i = 3,4\,(c^2 < 1)\,, \end{cases}$$

$$g_{(i)} = \begin{cases} 1 & i = 1,2\,(c^2 \geq 1) \\ e^{-\frac{2(1-\rho)(1-c^2)^2}{3\rho(c^2+c_s^2)}} & i = 3,4\,(c^2 < 1)\,. \end{cases}$$

After some algebraic manipulations we rewrite (A.1) in the following way:

$$f_{(i)}(\rho) = a_{6(i)}\rho^6 + a_{5(i)}\rho^5 a_{4(i)}\rho^4 + a_{3(i)}\rho^3 + a_{2(i)}\rho^2 + a_{1(i)}\rho + a_{0(i)} = 0 \tag{A.2}$$

where the parameters $a_{6(i)}, a_{5(i)}, a_{4(i)}, a_{3(i)}, a_{2(i)}, a_{1(i)}, a_{0(i)}$ for each case $i = 1, 2, 3, 4$ are given below:

Case 1: $c^2 \geq 1$ and $c_s^2 \geq 1$

$$\begin{aligned}
a_6 &= 8c^2c_s^2 + 3(c_s^2)^4 - 16(c_s^2)^2 + 4(c^2)^2 + 19(c^2)^2(c_s^2)^2 - \\
&\quad 2c^2(c_s^2)^2 - (c_s^2)^3 + 8(c^2)^3c_s^2 - 8(c^2)^3 - 13(c^2)^2c_s^2 + \\
&\quad 14c^2(c_s^2)^3 - 16c_s^2 \\
a_5 &= -20c^2(c_s^2)^2 + 4(c_s^2)^3 + 52(c_s^2)^2 - 28(c_s^2)^2(c^2)^2 - \\
&\quad 24c^2c_s^2 + 4c_s^2(c^2)^2 + 48c_s^2 - 20(c_s^2)^3c^2 - 4(c_s^2)^4 - \\
&\quad 12(c^2)^3c_s^2 + 4(c^2)^3 - 4(c^2)^2 \\
a_4 &= 20\sigma_d^2\lambda^2c_s^2 + 28c^2c_s^2 - 2c_s^2(c^2)^2 + 3(c_s^2)^3c^2 - (c^2)^3 + \\
&\quad 3(c^2)^3c_s^2 + 16\sigma_d^2\lambda^2 + 4\sigma_d^2\lambda^2(c_s^2)^2 + 27(c_s^2)^2c^2 - \\
&\quad 48c_s^2 - 4(c_s^2)^3 - 52(c_s^2)^2 + 6(c_s^2)^2(c^2)^2 \\
a_3 &= -8\sigma_d^2\lambda^2(c_s^2)^2 - 48\sigma_d^2\lambda^2 - 8(c_s^2)^2c^2 - 56\sigma_d^2\lambda^2c_s^2 + \\
&\quad 16c_s^2 - 8(c_s^2)^2(c^2)^2 - 16(c_s^2)^2 + 16\sigma_d^2\lambda^2c^2c_s^2 - \\
&\quad 4(c^2)^3c_s^2 - 4(c_s^2)^3c^2 + 16\sigma_d^2\lambda^2c^2 \\
a_2 &= 4(c_s^2+1)(\sigma_d^2\lambda^2c_s^2 - c^2c_s^2 + 12\sigma_d^2\lambda^2 - 7\sigma_d^2\lambda^2c^2) \\
a_1 &= 8\sigma_d^2\lambda^2(c^2-2)(c_s^2+1) \\
a_0 &= 4\sigma_d^2\lambda^2c^2(c_s^2+1)
\end{aligned}$$

Case 2: $c^2 \geq 1$ and $c_s^2 < 1$

$$\begin{aligned}
a_6 &= 12(c^2)^2 + 24c^2c_s^2 + 5(c_s^2)^4 - 48(c_s^2)^2 - 48c_s^2 + \\
&\quad 21(c_s^2)^2(c^2)^2 + 18(c_s^2)^2c^2 + (c_s^2)^3 + 8(c^2)^3 - \\
&\quad 8(c^2)^3c_s^2 - 3c_s^2(c^2)^2 + 18(c_s^2)^3c^2 \\
a_5 &= -8(c_s^2)^4 + 8(c_s^2)^3 + 156(c_s^2)^2 - 24(c_s^2)(c^2)^2 - \\
&\quad 48(c_s^2)^2(c^2)^2 - 84(c_s^2)^2c^2 - 4(c^2)^3 - 20(c^2)^3c_s^2 + \\
&\quad 144c_s^2 - 72c^2c_s^2 - 36(c_s^2)^3c^2 - 12(c^2)^2 \\
a_4 &= 5(c^2)^3c_s^2 + 84c^2c_s^2 + 2c_s^2(c^2)^2 + 5(c_s^2)^3c^2 + (c^2)^3 + \\
&\quad 48\sigma_d^2\lambda^2 - 144c_s^2 + 60\sigma_d^2\lambda^2c_s^2 + 85(c_s^2)^2c^2 - 12(c_s^2)^3 - \\
&\quad 156(c_s^2)^2 + 12\sigma_d^2\lambda^2(c_s^2)^2 + 10(c_s^2)^2(c^2)^2 \\
a_3 &= -24\sigma_d^2\lambda^2(c_s^2)^2 - 168\sigma_d^2\lambda^2c_s^2 - 4(c^2)^3 - 144\sigma_d^2\lambda^2 - \\
&\quad 8c_s^2(c^2)^2 + 48\sigma_d^2\lambda^2c^2 + 48c_s^2 - 16(c_s^2)^2(c^2)^2 - \\
&\quad 24c^2c_s^2 - 28(c_s^2)^2c^2 - 8(c^2)^3c_s^2 - 8(c_s^2)^3c^2 + \\
&\quad 48\sigma_d^2\lambda^2c^2c_s^2 + 48(c_s^2)^2 \\
a_2 &= 12(c_s^2+1)(\sigma_d^2\lambda^2c_s^2 - c_s^2c^2 + 12\sigma_d^2\lambda^2 - 7\sigma_d^2\lambda^2c^2) \\
a_1 &= 24\sigma_d^2\lambda^2(c^2-2)(c_s^2+1) \\
a_0 &= 12\sigma_d^2\lambda^2c^2(c_s^2+1)
\end{aligned}$$

Case 3: $c^2 < 1$ and $c_s^2 \geq 1$

$$\begin{aligned}
a_6 &= 8(c_s^2)^3 + 8(c_s^2)^2 + 16c^2c_s^2 + 12(c_s^2)^2c^2 - 4(c_s^2)^3c^2 + \\
&\quad 8e^{-2A}(c^2)^2c_s^2 + 2e^{-2A}(c_s^2)^3(c^2)^2 + e^{-2A}(c_s^2)^4c^2 + \\
&\quad e^{-2A}(c^2)^3(c_s^2)^2 - 11e^{-2A}(c^2)^3c_s^2 - 4e^{-2A}(c_s^2)^4 - \\
&\quad 19e^{-2A}(c_s^2)^3c^2 - 26e^{-2A}(c^2)^2(c_s^2)^2 + \\
&\quad 4e^{-2A}(c^2)^3 + 4e^{-2A}(c_s^2)^2c^2 \\
a_5 &= -16(c_s^2)^3 - 16(c_s^2)^2 - 32c^2c_s^2 - 20(c_s^2)^2c^2 - \\
&\quad 4(c_s^2)^2(c^2)^2 - 4c_s^2(c^2)^2 + 12(c_s^2)^3c^2 + 4e^{-2A}(c^2)^2c_s^2 - \\
&\quad e^{-2A}(c_s^2)^3(c^2)^2 - e^{-2A}(c_s^2)^4c^2 + e^{-2A}(c^2)^3(c_s^2)^2 + \\
&\quad 8e^{-2A}(c_s^2)^2c^2 + 17e^{-2A}(c^2)^3c_s^2 + 33e^{-2A}(c^2)^2(c_s^2)^2 + \\
&\quad 19e^{-2A}(c_s^2)^3c^2 + 2e^{-2A}(c_s^2)^4 + 4e^{-2A}(c_s^2)^3 + \\
&\quad e^{-2A}(c^2)^4 + e^{-2A}(c^2)^4c_s^2 - 4e^{-2A}c^2c_s^2 - \\
&\quad 2e^{-2A}(c_s^2)^2 - 2e^{-2A}(c^2)^2 \\
a_4 &= 4\sigma_d^2\lambda^2c^2(c_s^2)^2 + 12(c_s^2)^2(c^2)^2 - 12\sigma_d^2\lambda^2c^2c_s^2 + \\
&\quad 16c^2c_s^2 + 8(c_s^2)^2 - 16\sigma_d^2\lambda^2c^2 - 8\sigma_d^2\lambda^2(c_s^2)^2 - \\
&\quad 8\sigma_d^2\lambda^2c_s^2 - 12(c_s^2)^3c^2 + 4(c_s^2)^2c^2 + 12c_s^2(c^2)^2 + \\
&\quad 4e^{-2A}(c_s^2)^4 - 6e^{-2A}(c^2)^2c_s^2 - e^{-2A}(c_s^2)^3(c^2)^2 - \\
&\quad 2e^{-2A}(c^2)^3(c_s^2)^2 + 8(c_s^2)^3 - 12e^{-2A}(c_s^2)^2c^2 - \\
&\quad 2e^{-2A}(c^2)^3c_s^2 + 3e^{-2A}(c^2)^2(c_s^2)^2 + 8e^{-2A}(c_s^2)^3c^2 - \\
&\quad 6e^{-2A}(c_s^2)^3 - e^{-2A}(c^2)^4 + 2e^{-2A}(c_s^2)^2 + \\
&\quad 2e^{-2A}(c^2)^2 - e^{-2A}(c^2)^4c_s^2 + 4e^{-2A}c^2c_s^2 \\
a_3 &= -12\sigma_d^2\lambda^2(c_s^2)^2c^2 + 4\sigma_d^2\lambda^2c_s^2(c^2)^2 + 4\sigma_d^2\lambda^2(c^2)^2 + \\
&\quad 20\sigma_d^2\lambda^2c^2c_s^2 + 4(c_s^2)^2c^2 + 4c^2(c_s^2)^3 - 12(c_s^2)^2(c^2)^2 - \\
&\quad 12c_s^2(c^2)^2 + 16\sigma_d^2\lambda^2(c_s^2)^2 + 16\sigma_d^2\lambda^2c_s^2 + 32\sigma_d^2\lambda^2c^2 + \\
&\quad 4e^{-2A}(c^2)^2c_s^2 + 8e^{-2A}(c_s^2)^2c^2 + 4e^{-2A}(c_s^2)^3 \\
a_2 &= 4(c_s^2+1)(3\sigma_d^2\lambda^2c^2c_s^2 - 2\sigma_d^2\lambda^2c_s^2 + c_s^2(c^2)^2 - \\
&\quad 4\sigma_d^2\lambda^2c^2 - 3\sigma_d^2\lambda^2(c^2)^2) \\
a_1 &= -4\sigma_d^2\lambda^2c^2(c_s^2+1)(c_s^2-3c^2) \\
a_0 &= -4\sigma_d^2\lambda^2(c^2)^2(c_s^2+1)
\end{aligned}$$

Case 4: $c^2 < 1$ and $c_s^2 < 1$

$$\begin{aligned}
a_6 &= 24(c_s^2)^3 + 24(c_s^2)^2 + 48c^2c_s^2 - 42e^{-2A}(c^2)^2(c_s^2)^2 - \\
&\quad 12(c_s^2)^3c^2 - 12e^{-2A}(c^2)^2c_s^2 + 6e^{-2A}(c_s^2)^3(c^2)^2 + \\
&\quad 3e^{-2A}(c_s^2)^4c^2 + 3e^{-2A}(c^2)^3(c_s^2)^2 - 12e^{-2A}(c_s^2)^2c^2 - \\
&\quad 17e^{-2A}(c^2)^3c_s^2 + 36(c_s^2)^2c^2 - 33e^{-2A}(c_s^2)^3c^2 - \\
&\quad 8e^{-2A}c_s^{2^4} - 4e^{-2A}(c^2)^3 - 4e^{-2A}(c_s^2)^3 \\
a_5 &= -48(c_s^2)^3 - 48(c_s^2)^2 - 96c^2c_s^2 - 60(c_s^2)^2c^2 - \\
&\quad 12(c_s^2)^2(c^2)^2 + 3e^{-2A}(c^2)^3(c_s^2)^2 + 36(c_s^2)^3c^2 + \\
&\quad 40e^{-2A}(c^2)^2c_s^2 - 3e^{-2A}(c_s^2)^3(c^2)^2 - 3e^{-2A}(c_s^2)^4c^2 - \\
&\quad 12c_s^2(c^2)^2 + 32e^{-2A}(c_s^2)^2c^2 + 35e^{-2A}(c^2)^3c_s^2 + \\
&\quad 67e^{-2A}(c^2)^2(c_s^2)^2 + 41e^{-2A}(c_s^2)^3c^2 + 6e^{-2A}(c_s^2)^4 + \\
&\quad 16e^{-2A}(c^2)^3 + 3e^{-2A}(c^2)^4c_s^2 - 4e^{-2A}c^2c_s^2 + \\
&\quad 8e^{-2A}(c_s^2)^3 + 3e^{-2A}(c^2)^4 - 2e^{-2A}(c_s^2)^2 - 2e^{-2A}(c^2)^2 \\
a_4 &= 12\sigma_d^2\lambda^2(c_s^2)^2c^2 + 36(c_s^2)^2(c^2)^2 - 36\sigma_d^2\lambda^2c^2c_s^2 + \\
&\quad 48c^2c_s^2 + 24(c_s^2)^2 - 48\sigma_d^2\lambda^2c^2 - 24\sigma_d^2\lambda^2(c_s^2)^2 - \\
&\quad 24\sigma_d^2\lambda^2c_s^2 - 36(c_s^2)^3c^2 + 12(c_s^2)^2c^2 + 36c_s^2(c^2)^2 + \\
&\quad 24(c_s^2)^3 + 8e^{-2A}(c_s^2)^4 - 6e^{-2A}(c^2)^2c_s^2 - \\
&\quad 3e^{-2A}(c_s^2)^3(c^2)^2 - 6e^{-2A}(c^2)^3(c_s^2)^2 - 4e^{-2A}c^2c_s^2 - \\
&\quad 6e^{-2A}(c^2)^3c_s^2 + 5e^{-2A}(c^2)^2(c_s^2)^2 + 16e^{-2A}(c_s^2)^3c^2 - \\
&\quad 6e^{-2A}(c_s^2)^3 - 3e^{-2A}(c^2)^4 - 2e^{-2A}(c_s^2)^2 - \\
&\quad 2e^{-2A}(c^2)^2 - 3e^{-2A}(c^2)^4c_s^2 - 12e^{-2A}(c_s^2)^2c^2 \\
a_3 &= -36\sigma_d^2\lambda^2(c_s^2)^2c^2 + 12\sigma_d^2\lambda^2c_s^2(c^2)^2 + 12\sigma_d^2\lambda^2(c^2)^2 + \\
&\quad 60\sigma_d^2\lambda^2c^2c_s^2 - 36(c_s^2)^2(c^2)^2 - 36c_s^2(c^2)^2 + 12c^2(c_s^2)^3 + \\
&\quad 48\sigma_d^2\lambda^2(c_s^2)^2 + 48\sigma_d^2\lambda^2c_s^2 + 96\sigma_d^2\lambda^2c^2 + \\
&\quad 8e^{-2A}(c^2)^2c_s^2 + 12c^2(c_s^2)^2 + 16e^{-2A}(c_s^2)^2c^2 + \\
&\quad 8e^{-2A}(c_s^2)^3 + 8e^{-2A}c^2c_s^2 + 4e^{-2A}(c_s^2)^2 + 4e^{-2A}(c^2)^2 \\
a_2 &= 12(c_s^2+1)(3\sigma_d^2\lambda^2c^2c_s^2 - 2\sigma_d^2\lambda^2c_s^2 + c_s^2(c^2)^2 - \\
&\quad 4\sigma_d^2\lambda^2c^2 - 3\sigma_d^2\lambda^2(c^2)^2) \\
a_1 &= -12\sigma_d^2\lambda^2c^2(c_s^2+1)(c_s^2-3c^2) \\
a_0 &= -12\sigma_d^2\lambda^2(c^2)^2(c_s^2+1)
\end{aligned}$$

with $A = \frac{2(1-\rho)(1-c^2)^2}{3\rho(c^2+c_s^2)}$.

In order to apply the first iterative method, as discussed in Section IV-A, we perform an algebraic transformation of (A.2) into the form $\rho = g_{(i)}(\rho)$ for each case i as follows:

$$\rho = g_{(i)}(\rho) = \frac{-a_{0(i)}}{a_{6(i)}\rho^5 + a_{5(i)}\rho^4 a_{4(i)}\rho^3 + a_{3(i)}\rho^2 + a_{2(i)}\rho + a_{1(i)}} \tag{A.3}$$

where $i = \{1, 2, 3, 4\}$ and consequently the iteration procedure for case i is given by:

$$\rho_{n+1} = \frac{-a_{0(i)}}{a_{6(i)}\rho_n^5 + a_{5(i)}\rho_n^4 a_{4(i)}\rho_n^3 + a_{3(i)}\rho_n^2 + a_{2(i)}\rho_n + a_{1(i)}} \tag{A.4}$$

for $n = 0, 1, 2, \ldots$.

According to the second iterative method (Newton's method) the general formula for the iteration procedure for case i is given by:

$$\rho_{n+1} = \rho_n - \frac{f_{(i)}(\rho_n)}{f'_{(i)}(\rho_n)} \quad (n = 0, 1, 2, \ldots) \qquad \text{(A.5)}$$

with $f_{(i)}(\rho)$ given in (A.2) and $f'_{(i)}(\rho) = 6a_{6(i)}\rho^5 + 5a_{5(i)}\rho^4 4a_{4(i)}\rho^3 + 3a_{3(i)}\rho^2 + 2a_{2(i)}\rho + a_{1(i)}$.

The iteration procedure for case i after some algebraic transformations can be rewritten as follows:

$$\rho_{n+1} = \frac{5a_{6(i)}\rho^6 + 4a_{5(i)}\rho^5 + 3a_{4(i)}\rho^4 + 2a_{3(i)}\rho^3 + a_{2(i)}\rho^2 - a_{0(i)}}{6a_{6(i)}\rho^5 + 5a_{5(i)}\rho^4 4a_{4(i)}\rho^3 + 3a_{3(i)}\rho^2 + 2a_{2(i)}\rho + a_{1(i)}}. \qquad \text{(A.6)}$$

ACKNOWLEDGMENTS

This work has been developed as part of my PhD studies at Royal Melbourne Institute of Technology, Australia, under supervision of Prof. Richard J. Harris.

REFERENCES

[1] I. Atov and R. J. Harris, "Dimensioning Method for Multiservice IP Networks to Satisfy Delay QoS Constraints", *Proceedings of IFIP-TC6 Interworking 2002 - The Conference on Converged Networking Data & Real-Time Over IP*, Perth, Australia, October 13-16, 2002, pp. 13-25.

[2] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", *ACM Transactions on Networking*, Vol. 3, 1995.

[3] R. Guérin and V. Peris, "Quality-of-Service in Packet Networks Basic Mechanisms and Directions", *Computer Networks*, Vol. 31, No. 3, February 1999, pp. 169-179.

[4] I. Atov, H.T. Tran and R.J. Harris, "OPQR-G: Algorithm for Efficient QoS Partition and Routing in Multiservice IP Networks", *Elsevier Journal of Computer Communications*. To appear.

[5] I. Atov, H.T. Tran and R.J. Harris, "Efficient QoS Partition and Routing in Multiservice IP Networks", *Proceedings of 22nd IEEE International Performance, Computing and Communications Conference (IPCCC 2003)*, Phoenix, Arizona, April 9-11, 2003, pp. 435-443.

[6] I. Atov and R.J. Harris, "Characterization of Class-Based Traffic Flows in Multiservice IP Networks", *Proceedings of 8th IEEE International Conference on Communication Systems 2002 (ICCS 2002)*, Singapore, November 25-28, 2002.

[7] I. Atov and R.J. Harris, "LP-based Algorithm for Optimization in Multiservice IP Networks", *WSEAS Transactions on Communications*, Vol. 2, No. 4, 2003, pp. 432-438.

[8] A. Heindl, *Traffic-Based Decomposition of General Queueing Networks with Correlated Input Processes*, PhD Thesis, Technische Universität Berlin, 2001.

[9] P. J. Kühn, "Approximate Analysis of General Queueing Networks by Decomposition", *IEEE Transactions on Communications*, Vol. 27, No. 1, pp. 113-126, 1979.

[10] W. Whitt, "The Queueing Network Analyzer", *Bell System Technical Journal*, Vol. 62, No. 9, pp. 2799-2815, 1983.

[11] W. Whitt, "Performance of the Queueing Network Analyzer", *Bell System Technical Journal*, Vol. 62, No. 9, pp. 2817-2843, 1983.

[12] J. R. Jackson, "Networks of Waiting Lines", *Operations Research*, Vol. 5, 1957, pp. 518-521.

[13] W. Whitt, "Approximating a point process by a renewal process, I. Two basic methods", *Operations Research*, Vol. 30, No.1, pp.125-147, 1982.

[14] D. L. Iglehart and W. Whitt, "Multiple Channel Queues in Heavy Traffic, 2: Sequences, Networks and Batches", *Adv. Appl. Prob.*, Vol. 2, No. 2, pp. 355-369, 1970.

[15] K. Shiram and W. Whitt, "Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data", *IEEE Journal on Selected Areas in Communications*, Vol. 4, No. 6, 1986, pp. 833-846.

[16] W. Whitt, "Approximations for the GI / G / m Queue", *Productions and Operations Management*, Vol. 2, 1993, pp. 114-161.

[17] S. McCanne and S. FLoyd, UCB/LBNL/VINT Network Simulator - ns (version 2.26), March 2003. http://www.isi.edu/nsnam/ns/

[18] K. Pawlikowski, H. -D. J. Jeong and J. -S. R. Lee, "On credibility of simulation studies of telecommunication networks", *IEEE Communications Magazine*, Vol. 40, No. 1, January 2002.

[19] P. A. Lewis, A. S. Goodman, J. M. Miller, "A Pseudo-Random Number Generator for the System/360", *IBM Syst. J.*, Vol. 8, pp. 136-146, 1969.

[20] S. K. Park and K. W. Miller, "Random Number Generators: good ones are hard to find", *Comm. ACM*, Vol. 31, pp. 1192-1201, 1988.

[21] M. Matsumoto and T. Nishimura, "Mersenne Twister: a 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator", *ACM Trans. on Modelling and Computer Simulation*, Vol. 8, No. 1, pp. 3-30, 1998.

[22] M. Matsumoto, Mersenne Twister Home Page, 1998. http://www.math.keio.ac.jp/ matsumoto/emt.html/

[23] K. Pawlikowski, "Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions", *ACM Computing Surveys*, Vol. 2, pp. 23-170, 1990.

[24] I. Atov and R. J. Harris, "Capacity Provisioning in DiffServ/MPLS Networks based on Correlated Traffic Inputs", *Proceedings of 23rd IEEE International Performance, Computing and Communications Conference (IPCCC)*, Phoenix, Arizona, USA, April 2004, pp. 187-193.