A Comparative Analysis of On-line Measurement-based Capacity Allocation Schemes

Fatih Hacıömeroğlu* and Michael Devetsikiotis†
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27606-7911, USA
†mdevets@eos.ncsu.edu
*fatih61@ieee.org

Submitted to ITC 18, Berlin, August 2003

Abstract—Today’s high-speed packet-switched networks are faced with the task of handling an increasing amount and variety of services, requiring different QoS constraints. Within the context of such traffic demands the choice of appropriately accurate but also practically implementable measurement algorithms becomes crucial.

In this paper, we perform a comparative study of alternative on-line bandwidth allocation algorithms, we analyze their complexity, and perform comparisons via simulation experiments. Our motivation is to use these algorithms in the data plane of “self-sizing” frameworks, and make use of their output in taking control plane decisions either locally or globally, in an on-line fashion. Previously, no such comprehensive comparison of relevant methods has been carried out, especially from a combined accuracy versus implementation complexity point of view.

Due to the dynamic characteristics of the algorithms, we encounter the choice of time resolution, i.e., measurement time scale and window. After numerous simulations, we gain insight on the critical effect of these choices on the performance of the algorithms. We deduce that the time scale parameter itself can be determined dynamically so that measurement-based algorithms perform successfully independent from the varying traffic conditions. Finally, we demonstrate the effectiveness of this new approach over the static one.

I. MOTIVATION AND INTRODUCTION

The demand on high-speed networks is getting higher and tougher to satisfy everyday, with the invention and commercialization of new bandwidth-hungry applications. Network resources are to be increased accordingly, to sustain an acceptable level of service. However, it is not always feasible to increase resources at the same pace of the increase in the traffic demand. In this regard, the bandwidth allocation in high-speed QoS-oriented networks is critical and needs to be made dynamic, adaptive and measurement-based, rather than static, to attain a more efficient use of resources. Especially for network links shared through statistical multiplexing, adaptive bandwidth allocation algorithms based on traffic measurements can achieve important gains.

For these reasons, we performed a comparison of on-line, dynamic measurement-based bandwidth allocation algorithms, which have low time complexity, and are based only on measurements instead of unreasonable assumptions about the incoming traffic. To the best of our knowledge, no such comprehensive comparison of relevant methods has been previously carried out, especially from a combined accuracy versus implementation complexity point of view.

We are also motivated by the fact that such algorithms can be used in the data plane of “self-sizing” network frameworks such as [1], [2] and in which every node in the network runs a measurement-based bandwidth allocation algorithm for every traf-
tic class or “band”. Periodically, the output of the algorithms, which give the required capacity demands of the traffic types, are collected, and either locally or globally, a control plane action is taken (i.e., the virtual links or scheduling allocations are recalculated) so as to minimize a predefined objective such as bandwidth cost or maximize revenue.

In this context, it is very important to choose measurement methods that satisfy stringent constraints in terms of both accuracy and complexity. Most of the algorithms we used in this paper naturally originated from the effective bandwidth concept, since effective bandwidth is the amount of the required bandwidth to be allocated for the satisfaction of a QoS constraint. Furthermore, in the literature, we identified two research areas which are related to our aim. These are network traffic prediction [3]–[6] and measurement-based admission control (MBAC) [7]–[9].

MBAC algorithms are composed of separate measurement procedure and admission criterion. We investigated the applicability of these separable measurement procedures for our purposes. However, other than the trivial Gaussian Approximation bandwidth estimator, such methods are not suitable for on-line re-sizing due to the fact that either they rely on unreasonable information (i.e., they presume that a priori traffic descriptors, such as present number of connections are known) or they have unacceptable computational complexity.

These incompatibilities stem from the different design considerations between measurement-based estimators in MBAC and self-sizing frameworks. First, MBACs are designed to operate only in the ingress nodes, where admission decisions are taken. Second, the period of execution of MBAC algorithms is at the connection level time scales. Our aim is to obtain on-line algorithms, working on every node in the network. Therefore their timescale and computational complexity are smaller than connection level timescales. Similar to MBAC algorithms, traffic predictors do not suit our consideration either. The reason this time is not because they are centralized and computationally complex as in MBAC algorithms, but that they do not target a QoS constraint.

The algorithms in this paper take a window of traffic measurements as input, estimate the parameters they need in a bandwidth allocation calculation formula and output the required amount of capacity to be reallocated. The measurements correspond to the amount of incoming traffic during a slot duration, $t_{slot}$. Consequently, the algorithm is called periodically every $N \times t_{slot}$ seconds (the reallocation period), where $N$ is the window size. The choices of $t_{slot}$ and $N$ are critical [10] and affect significantly the performance of algorithms.

In our simulations, the performance metric is related to a QoS constraint, in our case packet loss probability. To quantify the amount of expanded resources, we use two cost metrics, namely,

- **Allocation Ratio** (Average Capacity Allocation divided by Average Traffic Rate)
- **Average Queue Occupancy**

Our purpose is to obtain a feasible algorithm which is able to use bandwidth optimally (i.e., dynamically as in Figure 2) while still obtaining a performance close to the QoS target. The ideal algorithm should not require any knowledge or
unreasonable assumptions on the traffic, and be based completely on measurements.

The remainder of the paper is structured as follows. After describing the algorithms in Section II, we compare their response to different scenarios (i.e., variable buffer size, level of aggregation of sources, long-range dependence of traffic, computational and memory complexity requirements) and choose promising ones for further simulation study in Section III. Then we explain our simulation methodology in Section IV. Later, simulation results highlighting the importance of measurement time scales in algorithm performance are presented in section V. Section VI introduces the use of a measurement-based time scale and shows its effectiveness over the static time scale approach. Finally, we conclude with a discussion of the results and outline prospects for further enhancements.

II. ALGORITHMS
A. Direct EB Allocation (DEB)

DEB algorithm relies on a direct analytical evaluation, using the definition in [11].

\[ eb(s, t) = \frac{\ln(E(e^{X(0,t)})^{s/t})}{s} \] \hspace{1cm} (1)

\( X(0,t) \) is the amount of incoming work during a duration of \( t \). The \( (s, t) \) parameters are the so-called space and time parameters. Parameter \( s \) is calculated by using Large Deviations Theory (LDT) and by making a large buffer assumption. The overflow probability is calculated from an asymptotically exponential decrease assumption.

\[ P(B < Q) = e^{-s(C)B} \] \hspace{1cm} (2)

The time parameter \( t \) is related to time scales which are responsible for buffer overflow. It should be chosen small enough so that traffic is observed for buffer overflow analysis. Finally, the expectation in (1) is approximated by a time average, as suggested in [12].

The empirical evaluation of (1) is simulated and compared with analytical effective bandwidth of known Poisson and ON-OFF source types in [13].

B. Courcoubetis EB Allocation (CEB)

This method [14] is based on LDT and a large buffer assumption, similar to the DEB algorithm:

\[ eb = m + \frac{ID s}{2B} \] \hspace{1cm} (3)

The parameters \( m, B, s \) and \( ID \) are the mean rate, buffer size, space parameter and index of dispersion of \( X[0,t] \). The space parameter \( s \) is calculated using (2).

ID estimation process has a computational complexity proportional to \( N^2 \).

\[ ID = Var(X(0,t)) \left( 1 + 2 \sum_{k=1}^{1/4N} \left( 1 - 4 \frac{k-1}{N} \right) AC(k) \right) \]

\[ \left( \sum_{k=1}^{N} \frac{X(0,t)}{N} \right)^{-1} \] \hspace{1cm} (4)

C. Many Sources Asymptotic EB Allocation (MSAEB)

This approach [15] is also based on the effective bandwidth approach similar to the first two algorithms, but unlike them, this algorithm uses a different way of estimating the time and space parameters in (1). An assumption of many sources is
made, instead of a Large Buffer assumption, while using LDT to solve the problem of estimation of the space and time parameters \((s, t)\). As described in [15], if \(M\) sources are multiplexed in buffer \(B\), \(r_j\) is the percentage of streams of type \(j\), and maximum allowed buffer overflow probability to be guaranteed is \(e^{-a}\), then minimum required bandwidth can be calculated by solving

\[
C = \sup_{s} (\inf_{s} (R(s, t))) 
\]

where

\[
R(s, t) = \frac{stM \sum_j r_j eb(s, t) + a}{st} - \frac{B}{t} 
\]

In (6), \(eb(s, t)\) term is found from (1).

For a given \(t\), the \(R(s, t)\) is a unimodal function of \(s\), having a unique minimizer. Then, \(R(s, t) = R_t(s)\) is solved by using a golden section search method as described in [15]. This process is repeated for a range of \(t\) values smaller than the measurement window time, and the maximum among them is taken as the required capacity to be allocated.

**D. ON-OFF EB Allocation (OOEB)**

The idea is to obtain estimation values of an equivalent ON-OFF traffic model from measurements, and substitute them in the specific analytical effective bandwidth formula (7) for ON-OFF sources [16].

\[
eb(s, t) = \frac{-sr + a + b - 1/2 (-sr + a - b)^2}{2s} - 2ba 
\]

Parameters \(a\), \(b\) and \(r\) denote a ON-OFF traffic model where ON and OFF periods are exponentially distributed with parameters \(a\) and \(b\) respectively, and \(r\) is the constant traffic generation rate in the ON state. These parameters can be estimated by matching the first three moments of \(N\) data measurements falling into the window. These estimations have difficulties and require search algorithms, since direct solution of high order equations is not trivial. Moreover, this method is weak because of the limited fitting spectrum of ON-OFF model [17].

**E. Norros EB Allocation (NEB)**

In [18], besides introducing modeling of real traffic by fractional Brownian motion (FBM), an effective bandwidth formula (8) for FBM is also given.

\[
eb = m + \frac{K(H)\sqrt{-2 \ln(P_{loss})}}{H} * \frac{a}{2H} * B - \frac{1 - H}{H} * \frac{m}{2H} 
\]

where \(K(H) = H^H (1 - H)^{1-H}\) and \(m\), \(H\), \(P_{loss}\), \(x\) and \(a\) are the mean, Hurst parameter, buffer overflow probability, buffer size and coefficient of variation respectively. Parameter \(a\), is approximated by the index of dispersion. In fact, this is a valid assumption only when the traffic is short range dependent.

The Hurst parameter can be set from a priori measurements. However, to react to unexpected traffic changes, a measurement-based on-line algorithm is favored. Difficulties of \(H\) estimation methods are analyzed in [19], where the comparison of several \(H\) estimation algorithms revealed that Abry-Veitch estimator (AV estimator) based on Wavelet theory is the best approach [20].
F. DRDMW (Improved Empirical EB Allocation)

This method in [21] is an improved version of empirical effective bandwidth methods. A unified phenomenological framework to estimate overflow probability of both long range dependence (LRD) and short-range dependence (SRD) is put forward by including the Hurst parameter in traditional analytical effective bandwidth methods.

\[ P(B < Q) = e^{-s(C)B^2 - 2H} \]  (9)

Second, the difficulty of measuring the effective bandwidth of real-time traffic online by using direct estimator [21] is alleviated by using an approach based on dual recursive algorithm with double moving windows (DRDMW), which is introduced in an empirical calculation of analytical effective bandwidth formula instead of using the direct estimator.

G. Gaussian Approximation Allocation (GA)

The simplest resource allocation method existing in literature is the GA method [22], where link buffer is ignored and server capacity is set according to Gaussian arrival rate distribution:

\[ C = m + \sigma \sqrt{-2 \ln(P_{\text{loss}}) - \ln(2 \pi)} \]  (10)

where \( m \) and \( \sigma \) are the mean and standard deviation of the arrival rate distribution.

III. COMPARISON OF THE ALGORITHMS

The first algorithm, namely Direct Effective Bandwidth Allocation algorithm, relies on the effective bandwidth formula, and possesses the problem of finding appropriate values for \( s \) and \( t \), which depend on QoS requirements and the system parameters. The space parameter is estimated using the Large Buffer Assumption. The time parameter estimation is left somewhat arbitrary, for the time being.

The second algorithm uses (3, which is an alternative generic effective bandwidth definition in terms of the mean rate, index of dispersion, QoS parameter and buffer size. It is simpler, but it still doesn’t address long range dependent traffic.

The Many Sources Asymptotic Effective Bandwidth algorithm relies on the effective bandwidth formula (1) and encounters the problem of estimation of \((s, t)\). This method accomplishes it by solving a functional optimization problem. Although it is a very innovative approach, this may be too slow for our motivational self-sizing scenario where every node takes on-line measurements of every traffic type.

The ON-OFF Effective Bandwidth formula (7) for our fourth method is obtained by substituting an ON-OFF arrival process instead of \( X(0, t) \) in the analytical effective formula. With regard to a practical usage of such expressions, we encountered other problems than estimation of \((s, t)\) parameters, such as model parameter estimation, and goodness of fit of the model.

The Norros Effective Bandwidth Allocation and Gaussian Approximation methods are alternatives which do not include non-trivial \((s, t)\) parameter estimations. They are approximate expressions, which are derived independently of the effective bandwidth formula. The Gaussian Approximation algorithm assumes a bufferless link. This will overestimate required capacity. Moreover, the gaussian assumption
is not valid for traffic formed by small number of sources. This places a constraint on the source type, however our aim is to have an algorithm capable of functioning without unreasonable assumptions.

The self-similarity is addressed only in the NEB and in the DRDMW method. Others do not discriminate between short range dependence and long range dependence. Although the index of dispersion in the Courcoubetis formula of the second algorithm stands for burstiness of the source, the formula is not for long range dependent traffic. Thus the effective bandwidth approximation on which Courcoubetis formula is based, (i.e., exponential decay of buffer overflow probability with increasing buffer size) is not valid for self-similar traffic (the decay is hyperbolic and slower than exponential). We provide a summary of our performance comparisons with respect to various network scenarios in Table I.

The Gaussian Approximation algorithm is the easiest to implement, and suitable to be used as an algorithm setting an upper bound, since it does not consider buffer size. The Courcoubetis Effective Bandwidth Allocation is also another easy, and promising one, since this one takes into account buffer also. But neither of the previous two algorithms is designed with long range dependent traffic in mind. Norros effective bandwidth and DRDWM algorithms are the only ones incorporating the Hurst parameter, therefore addressing to long range dependent traffic. Although DRDMW is designed to alleviate the numerical overflows in the direct effective bandwidth allocation, that problem can not be completely alleviated due to the structure of (1). Therefore, we picked the following three algorithms for further simulation analysis:

- Gaussian Approximation (GA)
- Courcoubetis Effective Bandwidth Allocation (CEB)
- Norros Effective Bandwidth Allocation (NEB)

IV. SIMULATION METHODOLOGY

The selected on-line measurement-based resource allocation algorithms are implemented and tested in a simulation scenario as shown in Figure 1. This is a single server queue simulation where the service rate is changed, in an on-line fashion, periodically based on recent traffic measurements. We used the Sup-FRP traffic model [23]. The simulation flow slides packet by packet, emulating a real case scenario as in an Ethernet card passing packets to upper network layers.

![Simulation Scenario](image)

Figure 1. Simulation scenario.

Figure 2 gives a visual representation of how algorithms adjust service rates, tracking fluctuations in the incoming traffic rates, so as not to waste resources.

We performed simulations with 5 different \( t_{\text{slot}} \) values (0.01, 0.05, 0.1, 0.5, 1 s) and 5 window sizes \( (N) \) values (3, 6, 30, 60, 300 slots) in every method. Therefore, we had 25 simulations per method. As a total, we present here results of 75 simulations. Also note that the measured statistics in this paper
resulted after 30 simulation replications and confidence intervals are insignificant.

In all of the simulations in this section, we generated traffic with the same mean value of 20 Kbytes/s, the same Hurst parameter of 0.7 and the same buffer size of 5 Kbytes. We set the QoS target to packet loss probability of $10^{-3}$, so as to have a common ground for the performance comparisons of algorithms in the simulation scenario (Figure 1).

Note that the average traffic rate of 20 Kbytes/s is the product of an average packet size of 200 bytes and average packet arrival rate of 100 packets/s. Therefore, the average time between two consecutive packets is 0.01 seconds and the $t_{slot}$ values chosen in the simulations, which are (0.01, 0.05, 0.1, 0.5, 1s), correspond to cases where 1, 5, 10, 50 and 100 packet arrivals take place on average in a slot time duration, respectively. Also note that the choices for $t_{slot}$ and $N$ are made deliberately to have simulations where reallocation takes place in every 3 seconds, but with different measurement resolution in the recent history of the measurement data. For example, a simulation with (0.01s, 300slots) includes 300 measurements, whereas the one with (1s, 3 slots) includes three measurements in the same recent 3 seconds history.

We did not implement an on-line $H$ estimation [24]. We provided the value of $H$ (i.e., 0.7) to the algorithms beforehand, so that we can examine the performance of the bandwidth allocator, independent from the performance of the $H$ estimator.
The combined, on-line Hurst parameter and EB estimation is beyond the scope of this paper and is left as future work.

V. SIMULATION RESULTS

In this section, we first present performance and cost plots. We demonstrate and observe the importance of time scale choice in measurement-based algorithms.

The plots in this section are in the form of three dimensional graphs. The \( xy \) plane is composed of slot duration and window size pair, i.e., \((t_{\text{slot}}, N)\). There are 25 \( z \) coordinate points coming from the convolution of 5 values of \( t_{\text{slot}} \) and 5 values of \( N \), which are \((0.01, 0.05, 0.1, 0.5, 1\text{s})\) and \((3, 6, 30, 60, 300\text{slots})\), respectively. We chose to present simulation results in this form, rather than listing the numbers in a table form, because it would be difficult in the table form to observe the trends of the data when the time slot length increases while the window size is kept constant and vice a versa. At the end of this section, we provide the processing time plots with respect to the window sizes.

A. Performance Plots

The following three plots show the performance results for different \( t_{\text{slot}} \) and \( N \) values in the simulation scenario given in Figure 1. Figure 3 shows the loss probability results when GA is used for link dimensioning.

As seen from Figure 3, the loss probability increases when \( t_{\text{slot}} \) is increased while \( N \) is kept constant. Furthermore, it can be deduced that the loss probability decreases when \( N \) is increased while \( t_{\text{slot}} \) is kept constant.

The QoS target, which is \(10^{-3}\), is satisfied in every \((t_{\text{slot}}, N)\) combination with the exception of \((t_{\text{slot}} = 1\text{s}, N = 3\text{slots})\). Note that GA is used as an upper band of resource allocation for comparison purposes. It does not consider buffer size. In fact, it assumes there is no buffer. This is why, it is expected to be more generous than other algorithms. The fact that a violation of QoS took place in this method implies trouble for other methods.

Figure 4 tells us that the CEB’s performance changes similar to GA against \( t_{\text{slot}} \) and \( N \) variations, but \( P_{\text{loss}} \) values are relatively about an order of magnitude higher. The QoS target is violated for the following \((t_{\text{slot}}, N)\) pairs: \((0.5\text{s}, 3\text{slots})\), \((1\text{s}, 3\text{slots})\), \((0.5\text{s}, 6\text{slots})\) and \((1\text{s}, 6\text{slots})\). This algorithm, considering the presence of buffer, theoretically permits lesser resource usage than GA.

The loss probability results of NEB are provided in Figure 5. For \( N \) values of 3 and 6, the \( P_{\text{loss}} \) values
are between the ones of GA and CEB. However, when $N$ is either 30, 60 or 300, $P_{loss}$ is smaller than other algorithms, which implies an over-allocation of bandwidth.

### B. Cost Function Plots

The plots in this section show how much resources are used while getting the performance results introduced in the previous section.

The following two plots are cost plots for GA. Figure 6 agrees with Figure 3 and shows that as $t_{slot}$ is increased for a constant $N$, the allocation ratio decreases towards 1. We also observe that for constant $t_{slot}$, increasing $N$ results in a larger capacity allocation. But this rate of increase in capacity allocation depends on the $t_{slot}$ value. For instance, when $t_{slot}$ length is fixed to 1s, the allocation ratios for $N = 3, 6, 30, 60$ and 300 are 1.34, 1.40, 1.46, 1.47 and 1.51, respectively. However, when $t_{slot}$ is set to 0.01s, the corresponding ratio values are 4.21, 5.02, 5.89, 6.23 and 9.28. So, once $t_{slot}$ is properly chosen, choosing $N$ looses its importance, since the change in the ratio values are much smaller.
Figure 7 is in accordance with Figure 6 and Figure 3, and gives an idea about how much buffer is occupied. Figures 3, 7 and 6 show the importance of time scale choice. We observe that even if GA does not consider buffer, it can still be an acceptably good resource allocator, given that $t_{\text{slot}}$ and $N$ values are chosen properly.

Figures 8 and 9 are cost plots when CEB is used as the dynamic resource allocation method. Figure 9 shows that the buffer occupancy changes drastically with the choice of $N$. For small $N$, the occupancy is bigger than the one for GA, but for big window size values (60 and 300), the buffer is less occupied than when GA is used as resource allocator. These implications are justified in Figure 8. As a result, this shows that if $N$ is chosen poorly, the resource allocation can be even worse than GA’s allocation.

Compared to GA, we observe that bigger $t_{\text{slot}}$ values lead to better allocation ratios (i.e., ratios closer to 1) and the choice of $N$ has a greater effect on CEB. With proper choice of $t_{\text{slot}}$ and $N$, the same performance can be achieved with lesser resource usage. To illustrate, when $t_{\text{slot}}$ is fixed to 1s, the allocation ratios for $N$ 3, 6, 30, 60 and 300 are 1.04, 1.08, 1.29, 1.54 and 3.88 respectively, and when $t_{\text{slot}}$ is set to 0.01s, the corresponding ratio values are 4.86, 8.45, 30.66, 58.25 and 540.13. Here, similar to GA, we see the importance of choosing $t_{\text{slot}}$ properly. But unlike for GA, here choosing $N$ is also important. This is because the rate of increase of ratio values when $N$ is increased is much more significant. Choosing a large $N$ leads to serious over-allocation. Also note that in the above cases where allocation ratios are 1.04 and 1.08, the $P_{\text{loss}}$ values are 0.0134 and 0.0087 respectively, and the QoS criterion of $10^{-3}$ is not satisfied by one order of magnitude.
Figures 10 and 11 are cost metrics plots when NEB is used as the dynamic resource allocation method. Figure 10 shows that there exists over-allocation of resources when the window size is 30, 60 and 300, similar to the behavior observed in CEB.

When it comes to the choice of $t_{slot}$, as in the previous methods, a bigger $t_{slot}$ resulted in better resource allocation. This method is the only one which allocates more capacity to the traffic possessing higher long-range dependence. Overall, it can be said that NEB includes similar performance and cost changes as the ones of CEB, but performance values are around one order of magnitude better, and consequently, cost values are higher. For example, for $t_{slot}$ and $N$ pairs of (1s, 3slots) and (1s, 6slots), the $P_{loss}$ and allocation ratio values were 0.0134, 0.0087 and 1.04, 1.08, respectively, for CEB, but the corresponding values for NEB are 0.0022, 0.0003 and 1.38, 1.61 respectively. This shows that NEB has a tendency of allocating more resources than CEB, and results in better QoS constraint satisfaction.

C. Complexity

Figure 12 shows the processing times of the algorithms as a function of window size values. The processing time is the time required for the algorithm to re-calculate capacity. The processing time is seen to be related to $N$ for CEB and NEB, with a complexity of $O(N^2)$. This result is in parallel with our expectations. CEB and NEB calculate autocorrelations of measurements falling into the measurement window of size $N$, and this requires a processing time proportional to $N^2$. Whereas, GA uses only the mean and variance of the measurements, whose calculations are fully online. As a result, GA has complexity of $O(1)$ and

![Fig. 9. Average Queue Occupancy against different time slot length and window size values, when the CEB is used for dynamic resource allocation.](image1)

![Fig. 10. Ratio of the average capacity allocation to the average traffic rate against different time slot length and window size values, when NEB is used for dynamic resource allocation.](image2)
VI. Dynamic vs. Static Time Scale

Sections V-A and V-B show how drastically the performances of the measurement-based capacity allocation algorithms change depending on the time scale choice.

Mainly, we observed that increasing the measurement slot, $t_{\text{slot}}$, results in a decrease in the capacity allocation and consequently an increase in $P_{\text{loss}}$ in all of the algorithms. This can be explained intuitively from the structure of the formulas used in the capacity allocation algorithms. To illustrate, consider the Gaussian Approximation Algorithm’s formula (10), where $m$ and $\sigma$ are the mean and standard deviation of the most recent $N$ measurements, $[X_1, X_2, ..., X_N]$. Each $X_i$ represents incoming traffic load in consecutive $t_{\text{slot}}$ durations. By the law of large numbers, as $t_{\text{slot}}$ increases, say $t_{\text{slot}} \to A$, where $A$ is a time parameter, which is large (dependent on the traffic characteristic), then $X_i$ approaches $m \ast A$ for all $i$. This causes $\sigma$ in (10) to go to zero, and the capacity value, $eb$, to approach to the mean rate, $m$. A similar reasoning can be given for other methods, in which not only standard deviation, but also autocorrelations of measurements, $[X_1, X_2, ..., X_N]$ are used.

On the other hand, we also observe that taking $t_{\text{slot}}$ arbitrarily small ends up in over-allocation of resources. As $t_{\text{slot}}$ decreases, the measurement history ($t_{\text{slot}} \ast N$) decreases too. This decreases the confidence and increases the randomness in the formula parameter estimations, yielding over-allocations.

We could obtain empirical $t_{\text{slot}}$ values from the performance and cost metrics plots, so that the QoS is satisfied with minimum resource allocation.
However this particular $t_{slot}$ value would be useful only for the traffic that we used in our simulations. Consider using a traffic whose mean is $m \times K$ (that is $K$ times bigger). This time, on average $K$ times more traffic load will fall on average into the slots. Relatively, this is the same experiment as using $K$ times bigger $t_{slot}$ measurement slots, with mean traffic rate $m$. In other words, the measurement time scale is relative to the traffic characteristics.

A static $t_{slot}$ may correspond to cases where we described previously as small or large, depending on the incoming traffic.

As a result, we believe that the measurement time scale $t_{slot}$ should also change dynamically based on measurements $[X_1, X_2, ..., X_N]$, in order to keep the algorithms always working close to their best.

In [25], the Maximum Time-Scale (MaxTS = $t^*$) is used as the time scale of interest for queueing systems feed by a fractal Brownian motion (fBm) process:

$$t^* = \frac{k \sigma H}{(C - m)}^{\frac{1}{\mu}}$$

(11)

where $k = \sqrt{-2 \times \ln (P_{loss})}$, $m$ is the mean traffic rate, $\sigma$ is the standard deviation of the traffic rate and $C$ is the capacity of the server.

The value of $t^*$ is derived from (12), where $\hat{A}_H(t)$ is the probabilistic envelope process of the fBm cumulative arrival process $A_H(t)$ ($A_H(0) = 0$), such that $P(A_H(t) > \hat{A}_H(t)) \approx P_{loss}$:

$$\frac{d\hat{A}_H(t^*)}{dt} = C$$

(12)

On the basis of the law of large numbers, as $t \rightarrow \infty$, $\frac{d\hat{A}_H(t)}{dt}$ converges to the mean arrival rate. $\hat{A}_H(t)$ increases with a decreasing rate after $t^*$. This means that the probability that the average arrival rate exceeds the link capacity decreases for $t > t^*$.

In the remaining of the paper, we test using a dynamic time scale by estimating $t^*$ using the recent $N$ measurements and taking $t_{slot} = t^*$ as the measurement slot duration for the next $N$ measurements. In other words, besides effective capacity, $t_{slot}$ is also recalculated after every $N$ measurements.

Instead of the $(C - m)$ term in (11), we used $L \times m$, where $L$ is taken as a constant $L = (AllocationRatio) - 1$. The reason is that we allocate capacity dynamically and do not have a constant $C$.

Table II shows the improvement of using a dynamic $t_{slot} = t^*$ against static $t_{slot}$ choices (GA is used as the capacity allocation algorithm, and the $P_{loss}$ target is set to $10^{-3}$ as in the previous simulations). As the mean rate increases, the performance of the static $t_{slot}$ cases changes (the ratio decreases and $P_{loss}$ increases), whereas the performance of the dynamic $t_{slot} = t^*$ case remains the same. This shows that on-line measurement-based algorithms with constant measurement intervals are heavily dependent on the incoming traffic’s mean rate, whereas the ones with dynamic measurement intervals are more robust.

To illustrate the benefits visually, we generated a traffic trace of 1000s, where the mean rate of traffic between 200 and 800s is 5 times the mean rate at the remaining intervals. Figure 13 shows the dynamic capacity allocations. Note that the allocation ratios in the static $t_{slot}$ cases change in the region of traffic with high mean rate. But the allocation ratio

1 We tried using the average capacity allocation instead of $C$, but this caused a multiplicative effect, such that, when capacity allocation increases, $C - m$ term in (11) decreases. But decreasing $t_{slot}$ results in increased capacity allocation measurement in the next window, and this loop ends up having $t^* \approx 0$.

2 The particular performance figures for dynamic $t_{slot}$ case in Table II are dependent on the value of $L$ (we used $L = 1.5$). However, note that the choice of $L$ does not affect the robustness of the algorithm.
TABLE II
Performance Metrics vs. Time Scale

<table>
<thead>
<tr>
<th>Mean (Kbytes/s)</th>
<th>t_{slot} 0.08s</th>
<th>t_{slot} 0.4s</th>
<th>t_{slot} 2s</th>
<th>t_{slot} t^*</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4.751</td>
<td>2.657</td>
<td>1.745</td>
<td>1.915</td>
</tr>
<tr>
<td>( P_{loss} )</td>
<td>0</td>
<td>0</td>
<td>0.000003</td>
<td>0.000391</td>
</tr>
<tr>
<td>20</td>
<td>2.952</td>
<td>1.739</td>
<td>1.353</td>
<td>1.914</td>
</tr>
<tr>
<td>( P_{loss} )</td>
<td>0</td>
<td>0.000009</td>
<td>0.000382</td>
<td>0.000696</td>
</tr>
<tr>
<td>100</td>
<td>1.742</td>
<td>1.334</td>
<td>1.183</td>
<td>1.913</td>
</tr>
<tr>
<td>( P_{loss} )</td>
<td>0.000010</td>
<td>0.000332</td>
<td>0.002649</td>
<td>0.000910</td>
</tr>
</tbody>
</table>

remains roughly the same in the dynamic time scale case. This is achieved by adjusting \( t_{slot} \) as shown in Figure 15.

The number of packets dropped increases when the mean is increased gradually in Figure 14 (for \( t_{slot} = 0.01s \), no loss occurred, due to over-allocation). The \( t^* \) case performs again in between the static \( t_{slot} \) cases. But note that when \( t_{slot} = t^* \), the algorithm can self-adjust and perform similarly against traffic mean changes, whereas the performance of an algorithm with static \( t_{slot} \) is dependent on the traffic. To illustrate, a method with static \( t_{slot} = 0.01s \) case will over-allocate significantly when the mean rate decreases much below of 4Kbytes/s, and a method with static \( t_{slot} = 2s \) case will suffer significant degradation of the QoS target when the mean rate increases much above 100 Kbytes/s.

VII. Summary and Conclusions

In this paper, we presented and compared measurement-based on-line capacity allocation algorithms and proposed a way to improve their robustness.

We distinguished such algorithms from MBAC and traffic predictors due to their smaller time scales and QoS-oriented use. We observed that their performance is directly dependent on the involved measurement time scales. Mainly, we saw that when the time slot length is increased while the window size is kept constant, due to increasing aggregation of packets in the slot interval, the variations between...
the measurements in the measurement window decrease and the allocated capacity approaches the mean traffic rate. This causes the loss probability to increase.

Since the measurement time scale is directly related to the measured traffic, the result of a measurement-based algorithm using constant time scale is open to the performance degradations due to the changes in traffic trends. However, our aim was to obtain an algorithm which does not require any a priori traffic knowledge, and which is based fully on the measurements. Therefore we incorporated the Maximum Time-Scale (MaxTS) parameter and tested successfully adapting the measurement time scales based on measurements themselves.

To sum up, in this paper, we

- identified on-line measurement-based capacity allocation algorithms,
- compared their performances analytically,
- simulated promising ones,
- observed significant affects of the choice of measurement time scale,
- proposed to vary measurement time scale adaptively,
- through an example, showed the performance robustness of measurement-based algorithms, in which measurement time scale is adaptive (measurement-based).

The outcomes of this study can be used for choosing algorithms to be implemented in real switches, taking into account trade-offs of complexity, accuracy and robustness.

ACKNOWLEDGMENTS

This research was partly supported by the Center for Advanced Computing and Communication - North Carolina State University, and by Alcatel. The authors thank Aziz Mohammed and Chao Kan of Alcatel, Plano, for their comments and assistance.

REFERENCES