# New Modifications of the Exponential Moving Average Algorithm for Bandwidth Estimation

Lars Burgstahler, Martin Neubauer
Institute of Communication Networks and Computer Engineering, University of Stuttgart
Pfaffenwaldring 47, D-70569 Stuttgart
phone: +49 711 685 7966, fax: +49 711 685 7983
email:{burgstahler, neubauer}@ind.uni-stuttgart.de

## Abstract

In this paper, we describe three different modifications of the Exponential Moving Average (EMA) algorithm that can be used for bandwidth estimation. Bandwidth estimation algorithms have to be deployed by nodes in connectionless networks (e.g. IP networks) that perform *Quality of Service (QoS)* routing based on the available bandwidth. Since applications do not signal their bandwidth requirements in such networks, estimation is the nodes' only way to gain knowledge about the status of the links. To use the results of the estimation for routing purposes, several constraints have to be fulfilled: The algorithm should react fast to changes in bandwidth, should converge sufficiently fast to the actual used bandwidth, but should show a certain independence of short-term peaks. Furthermore the algorithm should not be overly complex in terms of computing time and storage. We will compare the modified algorithms to the basic EMA algorithm.

**Keywords: Bandwidth estimation, Exponential Moving Average, QoS Routing**

## 1 Introduction

Bandwidth estimation is essential when link occupancy should be used for QoS routing in connectionless networks. Since these networks have generally no signalling capabilities to announce the resource requirements of end-to-end relationships, the intermediate nodes cannot know how much bandwidth is still available, and which link is therefore best suited for the forwarding of a packet. To measure the traffic and to process these samples is the only way to get meaningful information. This is were bandwidth estimation algorithms come into action.

In connection-oriented networks, estimation algorithms are not so important for routing purposes. Applications running over ATM networks can easily reserve the necessary bandwidth or get a guarantee on maximum delay, provided they know the respective parameters. This is due to ATM's signalling capability. Depending on the service category various traffic parameters can be used in signalling messages, e.g. peak cell rate (PCR), maximum burst size (MBS), maximum cell transfer delay (maxCTD), and others [1]. Connection admission control (CAC) decides whether the network can accept a new connection and guarantee the required service quality. As long as the application complies to its traffic contract, the negotiated QoS is guaranteed. Another important aspect of signalling is that the nodes always know the residual (average) bandwidth on their attached links. Thus, bandwidth-based routing information is easily available and can be forwarded to other nodes or be used for route computation in the respective node. However, estimation algorithms can be used to monitor traffic and to ensure that the senders comply to their agreed traffic contract. This is called *Usage Parameter Control (UPC)*.

Classical IP networks are connectionless and do not have any signalling mechanisms by which the applications can make reservations or announce requirements about packet delay. Each packet is routed independently, i.e. that it might travel along a different path than other packets belonging to the same end-to-end relationship. RFC 2205 describes the Resource ReSerVation Protocol (RSVP) that can be used by applications to signal traffic characteristics to RSVP-capable IP routers which can reserve bandwidth [2]. In principle, RSVP could help the nodes to gain knowledge about link status. However, even if nodes knew about an application's bandwidth consumption — via signalling — they cannot predict which path a packet will take and they do not know the residual bandwidth of the attached links. As a solution to this problem, route pinning in conjunction with RSVP was proposed [3]. As for all approaches of this kind, this only works when all nodes within a network support the mechanism.

To get information on the link status in IP networks without the methods described above, estimation algorithms can be employed. Thereby, each node measures the amount of traffic on its (outgoing) ports and thus knows about the occupancy of the links at any time. Using a routing protocol like OSPF [4] – possibly with the QoS extensions described in [5] – the informations can be exchanged between the routers and consequently be used for route calculations. The estimation of occupied bandwidth is based on the size of the passing packets and time measurement and can involve several computational steps.

The results of the estimation can be used to distinctively route certain flows, e.g. flows belonging to a higher prioritized DiffServ class, to keep adaptive routing tables to balance load equally in a network, or for connection admission control as described in [6] (e.g. for RSVP).

In chapter 2 an overview on bandwidth estimation is given, including the exponential moving average algorithm. Chapter 3 explains our modifications on the algorithm and chapter 4 shows the impact of the modifications by simulation results of the modifications. In chapter 5 conclusions are made and future work is discussed.

# 2 Bandwidth estimation

In reality, links are always either completely occupied, i.e. a packet is currently being transmitted, or completely empty, i.e. no packet is currently being transmitted. However, this information does not help for routing. Instead, we need to calculate mean values over a certain interval.

This mean value calculation is not as monolithic a task as it might seem. Actually it consists of different tasks, some of which are only optional. The first and always necessary task is to measure the currently occupied bandwidth. Depending on the method, this leads either to a very volatile curve or a more steady, but not accurate one. Neither of them can be used well for routing. The second (optional) step is to smooth the measurement, i.e. levelling out short peaks. Smoothing can take into account the past *and* the future of a given measurement sample (we will show later how this "non-causality" can make sense). The third step is the real estimation, where we try to derive a bandwidth value which is a meaningful representation of the current occupied bandwidth.

It would also be possible to finally use a prediction mechanism, i.e. an algorithm that tries to predict the trend of the occupied bandwidth. This would help the routing algorithm to work pro-active. However, in this paper we will not discuss this method.

In the next sections we will give a brief classification of methods that can be used for these three steps.

## 2.1 Measurement

Measurement can be classified by the place where the sample is taken. Routers take samples at the input or output port. Measurement at the output port has the advantage that it gives information about the router's own outgoing resources, i.e. information that can directly be used for local routing decisions. We will not discuss this criterion further at this time, as it has no influence on the results of the estimation. Instead the distinction between packet-based and time-window-based measurement is much more relevant for bandwidth estimations approaches.

*Packet-based* means, that for each packet arrival, the time difference between the last packet arrival and the current time is calculated. Then the rate is defined as packet length divided by the respective time interval. Although this method is quite accurate, it has several drawbacks. First, if no packet arrives for some time, the router does not get any new information on bandwidth occupancy. Thus, it is impossible to ever discover a completely empty link. To overcome this problem, a timer can be used. If during the timer's interval no packet arrives, the timer triggers the calculation as described above (packet length equals zero). Another disadvantage is the large amount of samples that is created and that has to be processed by the router.

*Time-window-based* means that for a fixed time window, packet lengths are summed up. After that time, the ratio of total data length and time gives the mean bandwidth over that interval. The time window will then be restarted (*jumping window*). Alternatively, the window can be moved for only a fraction of the interval after either a packet arrival or a fixed time. Yet its size remains constant (*sliding window*). This leads to a finer granularity of the results. Both methods are very stable and produce no direct problems. However, the quality of the results depends largely on the size of the time window.

## 2.2 Smoothing

Smoothing is a way to remove short peaks in a series of measurements. For the following, we have to remember, that if a series of measurement samples $m_1 \dots m_n$ is given, a smoothing algorithm produces a new *series* of values $s_1 \dots s_n$, which describes a smoother curve. The deviation from the original curve (fitting) depends on the parametrisation of the smoothing algorithm. Possible smoothing algorithms are *Least Squares Fitting* or *Penalized Least Square*.

## 2.3 Estimation

Estimation creates a *single* new value $e_i$ which is based on a series of values $b_n$ (e.g. measured or smoothed bandwidth). We can distinguish between two different classes of estimation algorithms: *stateless*, i.e. new estimations do not take into account history and *state-dependent*, i.e. the history is taken into account. In the following, only state-dependent algorithms are considered.

The most simple algorithm is the *arithmetic mean*

$$e_i = \frac{\sum_{k=0}^{n-1} b_{i-k}}{n} \tag{1}$$

where $b_{i-k}$ is the occupied bandwidth (raw measurement or smoothed) at time $i-k$. It requires the existence of a set $B$ of samples $b_j$. The main problem is to find the right size $n$ for the set. If $n$ is too big, real changes are levelled out, if $n$ is too small, brief peaks are not suppressed.

For the *exponential moving average* in its simplest form (of order 1), only one old value has to be remembered. Still the results are satisfying. A new estimation $e_i$

is calculated as follows:

$$e_i = (1 - \alpha)e_{i-1} + \alpha b_i \quad (2)$$

The difficulty lies in the proper choice of $\alpha$, the exponential weight. With a large $\alpha$ the estimation follows the measurement truly, but does not suppress peaks, whereas with a small $\alpha$ peaks are suppressed but the estimation follows real changes too slowly.

In [7], an EMA algorithm with dynamic $\alpha$ is presented. The *distance adaptive EMA* algorithm uses the inter-arrival time between packets to modify the weight. For small time differences, the weight of the new sample is smaller, for larger time differences its weight is higher. This helps to bring down the estimate when a packet arrives only after a long time interval. On the other hand, a short inter-arrival time, that might only indicate a peak is not over-valued.

### 2.4 Influence of estimation on routing

Different bandwidth estimation algorithms have different characteristics, that make them better or worse suited to create input for a routing algorithm. In the following classification we show these characteristics and describe the desired behavior of the algorithm.

- Reaction: The algorithm should quickly discover an increase or decrease in occupied bandwidth. However, it should only include this change into its result, when a long-term change is about to happen. Brief peaks should influence the result as little as possible. In practice, this leads often to the fact, that the estimation lags behind the real trend (in terms of time), because current values are only moderately taken into account to avoid over-reaction.
- Stability: The algorithm should change the result of the estimation as rarely as possible to avoid updates for the routing. On the other hand, the results should still reflect reality as close as possible (which leads to inevitable changes).
- Symmetry: Some algorithms are fast in discovering an increase and slow in discovering a decrease in occupied bandwidth, they tend to overestimate occupied bandwidth. Others behave just the other way round, they tend to underestimate the occupied bandwidth. While the first group wastes resources, the second one might lead to an overloaded network. Algorithms that are slow on both sides are worst. So the goal is to have a symmetrically fast algorithm.
- Convergence: No matter how fast the algorithm is, it should eventually reach the true current mean value of the occupied bandwidth. While most algorithms approach this value from below after an increase and from above after an decrease, there are also some algorithms that tend to overshoot and converge only afterwards.
- Cost: Costs can be expressed in various terms: computational complexity, memory, and time for calculation. The optimum is of course a simple, fast algorithm that need not store much data.

Note that although the first two characteristics are the most important, each of them has to satisfy contradictory requirements. The goal is to find the most effective compromise, while achieving also good results for the three other characteristics.

## 3 Modifications of the EMA algorithm

The basic EMA algorithm shown in Equation 2 is statically configured, which means that $\alpha$ is possibly optimized for special traffic conditions only. However, the composition of traffic can not always be predicted which complicates the selection of a suitable parameter.

It is desirable that the estimation algorithm adapts to the traffic: reacting fast on real load changes, especially under heavy load conditions, but staying on an average level when brief oscillations happen. The ideal algorithm would not need any preconfigured parameters — which might be suitable only for certain conditions — but would calculate any parameters by itself, based on monitoring of the traffic. Nevertheless, a very basic parametrisation is almost always inevitable.

The following sections show first general considerations on how to dynamically modify the weight $\alpha$ of an EMA algorithm. Then we will present two modifications that we derived from these ideas. Finally we will present a new EMA algorithm which is not based on dynamic adaptation, but on smoothing the measurements first. Still, our goal was to match the characteristics presented in section 2.4.

### 3.1 Modifiers for dynamic weights

In the following we will restrict ourselves to the time-window measurement as a base for getting the measurement samples. In contrast to the distance adaptive EMA [7], our approaches can not use the time distance between measurement points since the time window and thereby this distance is constant.

The basic assumption is, that sharp increases or decreases can first be treated as peaks. Only if the change persists, it should be taken into account (but then very quickly). The goal is to consider these changes with lower weight. Smaller changes or stagnancy indicate a stable trend, so these measurements can be considered with a higher weight. As a consequence the main problem is to distinguish different situations, and to identify the correct reaction.

Changes in the amount of traffic generally have two different reasons:

- Activation or deactivation (or rerouting of the traffic) of one or several sources at roughly the same time leads to a permanent increase or decrease of occupied bandwidth. These changes should be processed.
- Superposition of different traffic sources with different packet rate and rate variation leads to many brief and small (although sometimes considerably large) changes in occupied bandwidth that result in a very unsteady curve with many short peaks or gaps. These changes should be neglected.

The problem is shown in Figure 1: During intervals 1 and

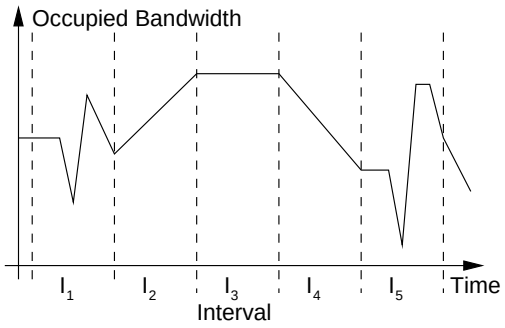


Figure 1: Short peaks and long-term changes

5 the changes are very brief and only temporary, caused by variations in single flows. In contrast, changes in interval 2 and 4 are more permanent, caused by flows beginning or ending respectively. The difference between the two situations is clear: although the changes are approximately equally large, the duration of them differs significantly. We could easily suppress the short-term variations by using long measurement intervals (like in the figure), however this might lead to a delayed reaction to real changes. As a consequence samples are not only taken at the beginning of each interval but also several times in between, i.e. in relatively short intervals.

Based on these considerations, we have identified two different approaches using the gradient of occupied bandwidth and the change of the gradient of the occupied bandwidth for the generation of a dynamic weight.

### 3.1.1 Gradient of occupied bandwidth

For the first approach, consider the example in Figure 2. Looking at times $t_1$ and $t_2$, we see that the occupied bandwidth is identical, but the occupancy trend is different which can be seen from the gradient of the curve. A good estimation not only depends on the currently occupied bandwidth, but also on the change of the occupied bandwidth over time. To enable this, we can take into account the gradient $m_i$ between two points $t_{i-1}$ and $t_i$:

$$m_i = \frac{b_i - b_{i-1}}{t_i - t_{i-1}} \tag{3}$$

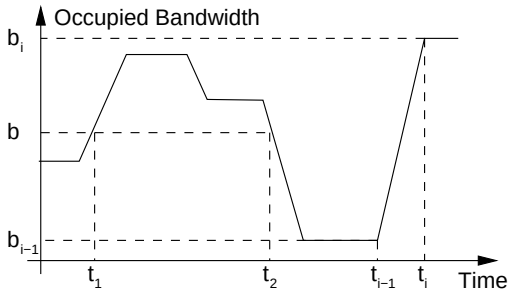


Figure 2: Change of occupied bandwidth

We will show in chapter 3.2 how this refined information of the bandwidth status of a link can be used for a better estimation.

While the gradient of the occupied bandwidth already can indicate a certain trend, the variations of this change over time give further information. The next approach will take this into account.

### 3.1.2 Change of gradient of occupied bandwidth

For the second approach we will compare consecutive measurements to find out more about the general trend. This comparison should be independent of the absolute value of occupied bandwidth. Therefore, we will use the difference of two consecutive gradients (see Figure 3).

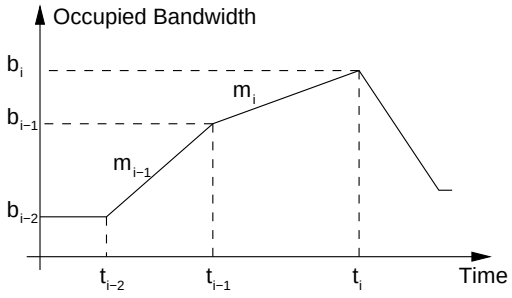


Figure 3: Determination of the difference of gradients

This difference of the gradients is calculated as follows:

$$\Delta m_i = m_i - m_{i-1} = \frac{b_i - b_{i-1}}{t_i - t_{i-1}} - \frac{b_{i-1} - b_{i-2}}{t_{i-1} - t_{i-2}} \tag{4}$$

To evaluate the result, different conditions must be considered:

- When the load is constant for some time, the gradient is always zero and so is the difference between two gradients.

- Constant long-term increases or decreases also show the same gradient for some time and therefore the difference between successive gradients is also zero.

- Due to oscillations, the gradient changes frequently and the change is very sharp. Although this could also happen for long-term increases or decreases, chances are much higher that a short-term peak arrived.

The first two cases show the difference of this approach: Long-term tendencies are detected, since the difference between the successive gradients is zero or at least very low. There is no need to differentiate these two cases for the adaptation of the weight. In the first case, the weight of new samples has no influence on an already steady estimation. In the second case, it is not important that the samples change, but that the change is constant. So the weight of the new sample can remain the same. The third condition can be easily detected and a suitable metric (e.g. derived from the change of gradients) can be used to adapt $\alpha$.

However, there is a slight disadvantage in this method: To explain this, assume that only one source is sending packets of equal size at a constant rate. The result of the measurement should become a straight line. However, if the time window is not synchronized with the source – which is generally the case – there is a jitter in the results, depending on how the time window border matches with the packet borders. We will later show that using the difference of gradients as an exponent to $\alpha$ helps to avoid this problem to a large degree. This is because these kind of changes are very small. With an exponent close to zero, the impact is even smaller. Note that the same is true for a constant increase or decrease.

In the next two sections we will show how to compute the weights for the EMA algorithm using the approaches explained so far.

## 3.2 Low pass EMA algorithm

The *low pass EMA (LpEMA) algorithm* is based on the first approach and uses the gradient to modify the weight of a $1^{st}$ order EMA algorithm. The weight is calculated with the help of a low pass filter of $1^{st}$ order. Using the classical formula for a low pass filter

$$F(f) = \frac{1}{1 + j\frac{f}{f_g}} \tag{5}$$

we replace the frequency $f$ by the gradient $m_i$ and the limit $f_g$ by a normalizing gradient $m_{\text{norm}}$. To control the maximum adaptation, we introduce a maximum weight $\alpha_{\max}$. This leads to the following equation for the weight $\alpha_i$:

$$\alpha_i = \alpha_{\max}\frac{1}{1 + \frac{|m_i|}{m_{\text{norm}}}} \tag{6}$$

The larger the gradient is, compared to $m_{\text{norm}}$, the smaller the fraction becomes, and the smaller $\alpha_i$ becomes. The result is, that short peaks are hardly noticed. In case that the absolute of the current gradient $|m_i|$ is equal to the normative gradient $m_{\text{norm}}$, we get

$$\alpha_i = \frac{1}{2}\alpha_{\max} \tag{7}$$

This leads us to the problem of selecting $m_{\text{norm}}$ and $\alpha_{\max}$. First we define a weight $\alpha_{\text{opt}}$, which gives the optimal result in case the bandwidth change happens to be equal to the mean gradient. Next we define $m_{\text{norm}}$ to be equal to the mean gradient. Thereby $\alpha_{\max}$ can be determined as:

$$\alpha_{\max} = 2\alpha_{\text{opt}} \tag{8}$$

Note that the determination of $\alpha_{\text{opt}}$ is not part of this paper. This leaves only the determination of the mean gradient. In principle, this value can be chosen arbitrarily to provoke a certain behaviour. We will present two specific values, which have a meaning:

- *Traffic-dependent mean*: This $m_{\text{norm}}$ is the real mean gradient of the traffic envelope. The value can be obtained by observation and continual averaging. It can either be calculated over the entire time since the start of the node or by using an interval over which it is calculated. The second approach values the current traffic situation higher.

- *Link-dependent mean*: This $m_{\text{norm}}$ is calculated by simply using the half of the maximum gradient during a time window:

  $$m_{\text{norm}} = \frac{1}{2}\frac{C}{t_i - i_{i-1}} \tag{9}$$

  where $C$ is the capacity of the link. Although this might not be the real mean value, it gives a rough estimation, based on the minimum and maximum gradient. The big advantage is, that this value is independent of the real traffic. As a result, the algorithm will weight deviations from the mean much stronger or weaker, since this alternative generally gives a larger $m_{\text{norm}}$ than the first one.

## 3.3 Gradient adaptive EMA algorithm

The *gradient adaptive EMA (GaEMA) algorithm* is based on the second approach and uses the difference of two consecutive gradients to modify the base weight $\alpha_b$. The modification is done by using dynamically calculated exponents for the base weight of a $1^{st}$ order EMA algorithm. Following the dynamic adaptive EMA algorithm of [7], the difference of the gradients is used as an exponent to the base weight $\alpha_b$. Therefore the actual weight for a new rate measurement results as

$$\alpha_i = \alpha_b^{x_i} \tag{10}$$

So, the weight is more sensitive to extreme values and more robust to slight deviations from mean changes.

The exponent is calculated as:

$$x_i = \begin{cases} \frac{|\Delta m_i|}{\Delta m_{\text{norm}}} & : \quad |\Delta m_i| \neq 0 \\ 1 & : \quad |\Delta m_i| = 0 \end{cases} \tag{11}$$

$x_i = 1$ for $|\Delta m_i| = 0$ was introduced to ensure a proper smoothness. This makes sense, as this case can only happen in case of constant load or constant change of load. Then the maximum adaptation rate $\alpha_b$ must be used.

The normalization difference of the gradients defines at which change of gradients the base weight will be used since for

$$|\Delta m_i| = \Delta m_{\text{norm}} \Rightarrow x_i = 1 \tag{12}$$

Note, that in contrast to the low pass EMA algorithm the base weight $\alpha_b$ needs not be modified to compare the results to a simple EMA algorithm (see Equation 8).

### 3.4 Retrospective EMA algorithm

In section 2.2 we have shown that we can already smooth the curve before making the estimation. We will explain here what problems occur when the estimation is made with an EMA algorithm. For reasons of simplicity, the EMA algorithm used here does not have a dynamic component.

Figure 4 depicts how an estimation based on the smoothed curve (solid line) is calculated. $t_i$ denotes the

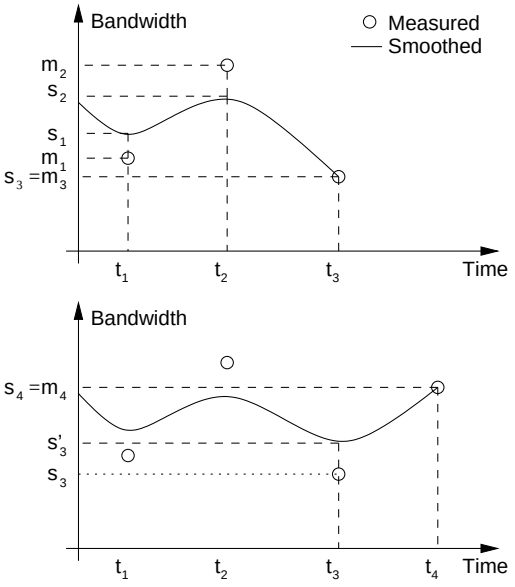


Figure 4: Retrospective change of smoothed value

times at which samples are taken, $m_i$ denotes the measured sample, $s_i$ denotes the smoothed value of the sample, and $e_i$ is the estimation. So in the first graph we get

$$e_3 = (1 - \alpha)e_2 + \alpha s_3 \tag{13}$$

In the next graph, at $t_4$ the same calculation is done again. However, we can see that due to smoothing, $s_3$ has changed to $s'_3$. From the current point of view, it seems that using $s_3$ as a base for the estimation of $e_3$ was not optimal. Thus using $e_3$ as a base for the calculation of $e_4$ is also not optimal.

The solution is "to go back in time" and recalculate $e_3$. However, the smoothing algorithm might not only have changed $s_3$, but also other values $s_j, j < 3$. This means that $e_2$ and other estimations before are also not optimal. Actually recalculation has to start at the last value that is not affected by smoothing anymore, i.e. that has already reached a steady state.

In general terms, to get $e_i$ at time $t_i$ with a smoothing window size of $n$, we have to rely on value $e_{i-n}$, the last trustworthy estimation. Then — step by step — all estimations $e_{i-n+1} \ldots e_i$ have to be recalculated by use of the EMA algorithm.

As a result, we get an estimation, that is based on values that are smoothed by future information (from the point of view of these values). Of course, this approach cannot predict the future for the current estimation.

## 4 Simulation results

In this section, we will present simulation results, that show the behavior of the proposed variations.

For the simulative evaluation we have used the IND SimLib, an object-oriented, event-driven library developed at our institute [8]. Figure 5 shows the simulation model. The sources are multiplexed by the multiplexer in

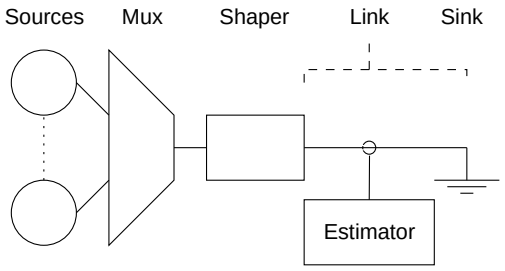


Figure 5: Simulation model

a round-robin fashion. The shaper ensures that the traffic of the sources does not exceed the bandwidth of the link in bursty situations. The shaper works without loss, so the total amount of data at the sink equals the input by the sources. The estimator monitors the by-passing packets between shaper and sink and calculates the estimates.

In this paper, we use three different traffic scenarios to evaluate the algorithms:

- *IP telephony (short)*: the sources send in on-off mode, where packet inter-arrival times during the

very short on-phase (mean of 5 $s$) are constant ($\Delta t = 60\,ms$). Furthermore, all packets have the same size of 160 $Bytes$, leading to a mean rate of approximately 21 $kbit/s$. This profile was derived from measurements of real IP-Telephony traffic [9]. The sources offer a 50% load on a link with a capacity of $C = 384\,kbit/s$. The measured mean gradient is $m_{\text{norm}} = 23.8\,kbit/s^2$, the simple mean gradient is $m_{\text{norm}} = 192\,kbit/s^2$.

- *IP telephony (long)*: the parameters for the sources and the algorithms are the same as for the first series of simulations but the duration of the calls was increased (mean of 180 $s$). The offered load was kept roughly the same (approximately 50%). The measured mean gradient is $m_{\text{norm}} = 3.8\,kbit/s^2$, the simple mean gradient is again $m_{\text{norm}} = 192\,kbit/s^2$. From Equation 6 we can see, that this leads to much larger $\alpha_i$ and thus to a stronger reaction.

- *Internet traffic*: the traffic consists of Pareto distribution-sized blocks that were segmented into IP packets ($\text{MaxSize} = 1500\,Byte$). The parameters for the Pareto are $\alpha = 1.6$ and $\text{MinBlockSize} = 3750$. The generators produce on-off traffic, offering 70% of load to the network, a $10Mbit/s$ link.

The parameters for the different algorithms are:

- Simple EMA: The base weight is set to $\alpha = 0.3$.
- Low pass EMA: The base weight is set to $\alpha_b = 0.6$ — see Equation 8 for the explanation why we have chosen $\alpha_b$ to be twice as large as $\alpha$.
- Gradient adaptive EMA: The base weight is set to $\alpha = 0.3$.
- Retrospective EMA: The base weight is set to $\alpha = 0.3$. For the smoothed and the retrospective EMA algorithm, the penalized least squares algorithm was used before doing the estimation. The parameters are: $\text{NumberOfMeasurements} = 5$, $\text{Smoothness} = 10000$, $\text{DataWeight} = 1$, and $\text{DifferenceDegree} = 1$.

For all simulations, we have chosen the time-window measurement approach with a jumping window with a size of 1 $s$. The choice of the interval size was based on the assumption that most routing protocols have fixed restrictions on the minimum time between routing information updates. So, measurements with too fine a granularity are not necessary. The result of the time-window measurement is always shown as reference. Further the results of a comparable simple EMA algorithm are always shown as a dashed line.

We will show only a cutout of the simulation run. On a larger time scale, the peaks are too close to each other, so that details cannot be recognized anymore. However, the result in the cutout represents the overall behavior very well. We will rate the result of the algorithms by visually comparing the estimation curve with the original measurement to see if the defined requirements are met (see 2.4). This is because so far, we have not identified a good method to make an analytical comparison. Methods like the *root-mean-square (RMS)* are not suitable since

1. if our estimation does not follow a short peak, we get a large deviation which is desired,
2. if our estimation does not follow a long-term trend, we get a large deviation which is not desired.

Hence, the difference between the measurement and the estimation does not directly express the quality of the estimation. Only in the context of the current trend, we can make a statement.

## 4.1 Short duration IP telephony

Figures 6–7 show the results of the low pass EMA algorithm. When using the traffic-dependent mean gradient,

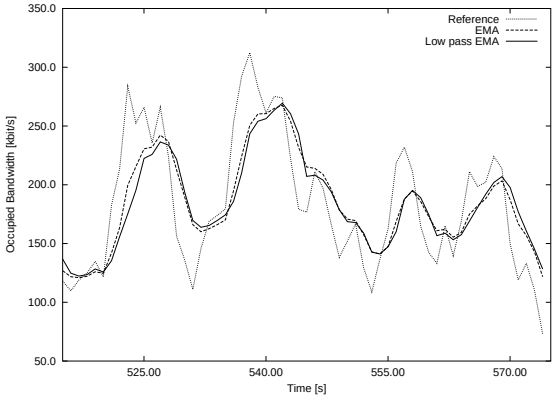


Figure 6: LpEMA – traffic-dep. $m_{\text{norm}}$, short duration

we see, that the results of the simple EMA algorithm and the low pass EMA algorithm hardly differ. As long as the changes of occupied bandwidth happen with approximately the mean gradient, the low pass EMA algorithm behaves much like the simple EMA algorithm. Only for strong deviations from the mean gradient (e.g. between 520 $s$ and 530 $s$), real differences can be observed.

Using a link-dependent $m_{\text{norm}}$ results in a different behavior. The algorithm reacts faster both in increases and decreases of the occupied bandwidth. Naturally, this leads to a higher roughness where peaks are not levelled out as well as expected. However, as long as the peaks are small enough and only disturb a solid trend (as e.g. around 570 $s$), the smoothing is still satisfying.

Figure 8 shows the results of the gradient adaptive EMA algorithm. We have only shown the traffic-dependent result since the link-dependent result follows the measurement too close. This happens, because our

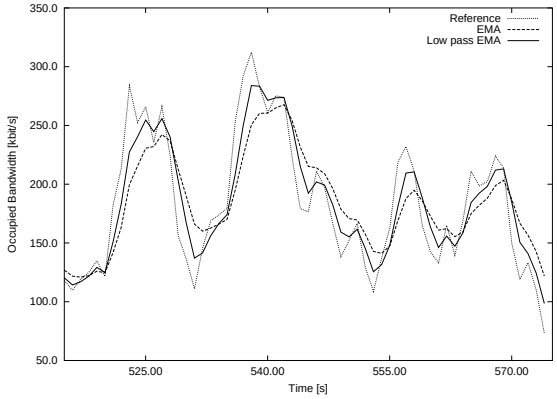


Figure 7: LpEMA – link-dep. $m_{\mathrm{norm}}$, short duration

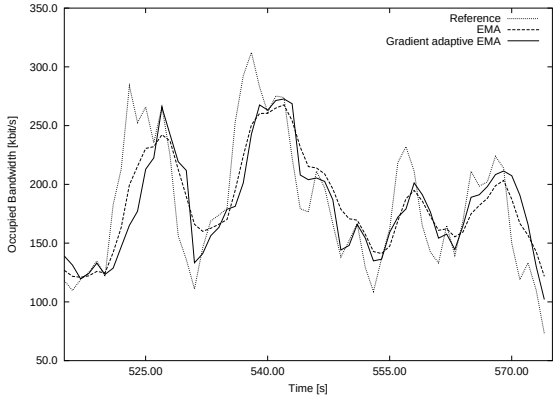


Figure 8: GaEMA – traffic-dep. $m_{\mathrm{norm}}$, short duration

normalizing gradient is larger than the measured one ($192\,kbit/s^2 \gg 23\,kbit/s^2$). Thus, the exponent of $\alpha_i$ is most of the time relatively small and therefore $\alpha_i$ is close to 1. The traffic sources are the same as for the low pass EMA algorithm. The traffic-dependent alternative shows almost the same reaction like the low pass EMA algorithm. However, the estimation follows the measurement much closer. Sometimes this leads to undesired peaks.

Figure 9 shows the result of the retrospective EMA algorithm. For comparison, the reference measurement, the simple EMA algorithm and a smoothed EMA algorithm are included. First of all, we can see, that there is indeed a remarkable difference between the smoothed EMA algorithm and the retrospective EMA algorithm:

- The retrospective EMA algorithm runs almost synchronously with the measurement, whereas the smoothed EMA algorithm is delayed, i.e. it recognizes increases and decreases later.

- The retrospective EMA algorithm follows the trend of the measured traffic — especially minimum and maximum values — much better than the smoothed EMA algorithm and the EMA algorithm. Thereby the result gives a better estimation of the real occupied and free bandwidth respectively.

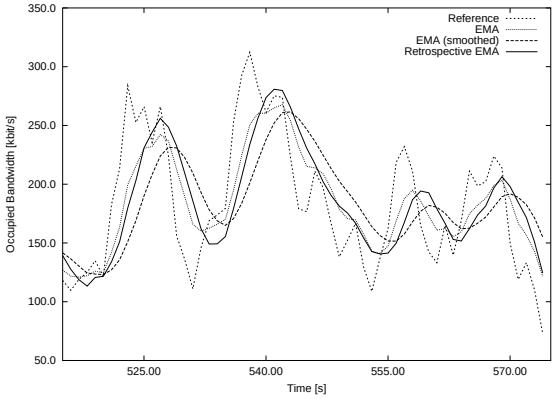


Figure 9: Retrospective EMA, short duration

- The smoothness of the retrospective EMA algorithm and the smoothed EMA algorithm are almost equal. It is much better than that of the simple EMA algorithm. All short peaks are completely levelled out, which means that the true trend is better recognized.

## 4.2 Long duration IP telephony

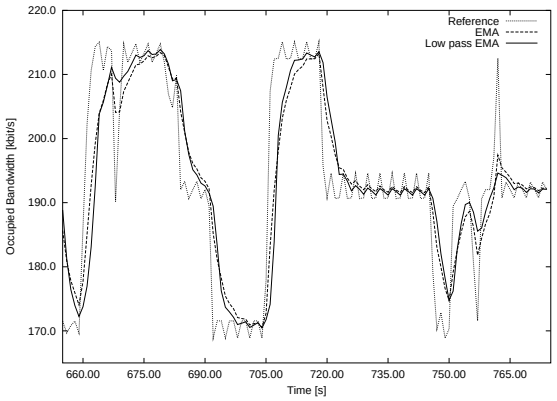


Figure 10: LpEMA – traffic-dep. $m_{\mathrm{norm}}$, long duration

Predictably, the results for the low pass EMA algorithm are much the same as in the first series of simulations (Figure 10). When using the link-dependent mean gradient, the reaction of the low pass EMA algorithm again is definitely faster and converges faster than the simple EMA algorithm. On the other hand, the small oscillations caused by the measurement approach are not smoothed out as well. Figure 11 shows a magnification of one of the ”constant” phases of the low pass EMA algorithm.

For the gradient adaptive EMA algorithm we also get similar results compared to both the low pass EMA algorithm and the scenario from the first series (see Figure 12). However, the traffic-dependent alternative shows an even worse reaction time. As we can see, it reacts later

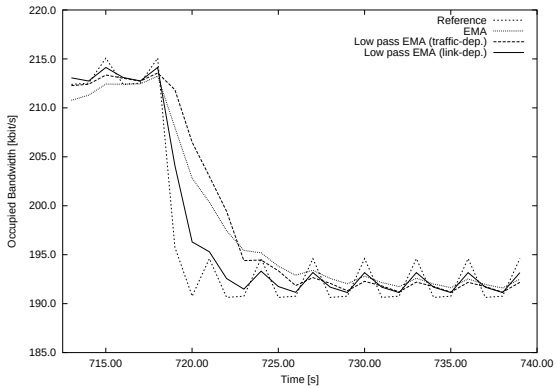


Figure 11: LpEMA – Smoothing out of oscillations

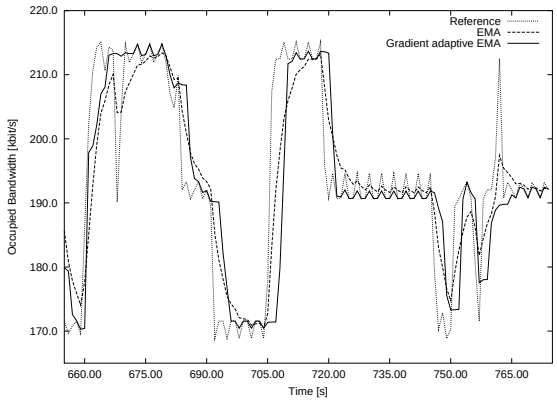


Figure 12: GaEMA – traffic-dep. $m_{\mathrm{norm}}$, long duration

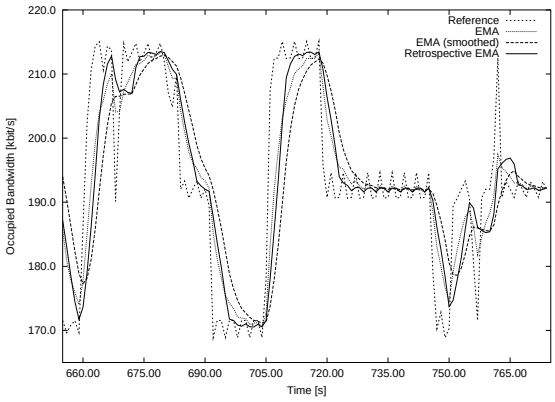


Figure 13: Retrospective EMA, long duration

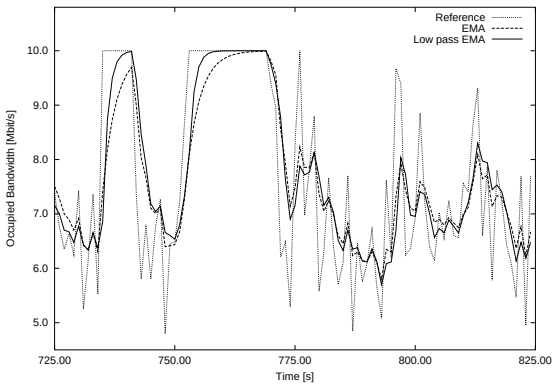


Figure 14: LpEMA – traffic-dep. $m_{\mathrm{norm}}$, Internet traffic

to changes than the simple EMA algorithm. Only due to its sharp increase or decrease it converges at least faster. The link-dependent alternative again shows almost complete identity with the measurement curve. One advantage of the traffic-dependent alternative over its low pass counterpart is, that the oscillations during the "constant" phases are better smoothed out. As for the traffic with short peaks, the retrospective EMA algorithm shows the best results. We can see this in Figure 13. The oscillations are hardly detectable, it reacts as fast as the simple EMA algorithm and converges better.

### 4.3 Internet traffic

The low pass EMA shows good results (Figure 14). It recognizes the fully loaded link faster than the normal EMA. Not only does it react slightly faster, but it also converges much faster. The traffic-Dependant alternative is a little bit smoother. The link-dependent alternative is rarely faster although it uses a much higher normalizing gradient. On the other hand, it shows a rougher curve, following the measurement much closer.

The results for the gradient adaptive EMA are bad. Using the traffic-dependent norm gradient, it underestimates the occupancy very often (Figure 15). Secondly, for most of the time it reacts too late. The very good reaction around $750s$ can be explained by the first and smaller step that triggers a reaction. When the curve increases again heavily after, the estimation follows very quickly. However, only in situations like this, the algorithm works well. Using the link-dependent norm gradient, it follows the measurement very close, so that there is no benefit from not using the measurement directly.

The results of the retrospective EMA are shown in Figure 16. It is a little bit slower than the simple EMA, but it converges earlier, when the bandwidth remains constant for some time. Besides, it provides a better envelope curve when the trend remains for some time. Whereas the smoothing generally overestimates the occupancy during decreases, the retrospective EMA is more optimistic without being overly optimistic. The overall behaviour is very similar to the other two scenarios.

## 5 Conclusion

In this paper, we have presented and evaluated by simulation three different modifications on the exponential moving average algorithm. For two of them, the weight was dynamically calculated to adapt to different load situa-

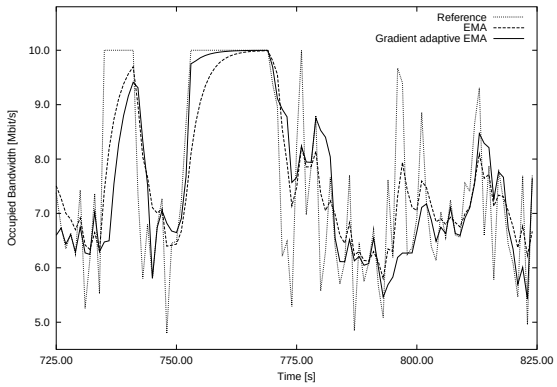


Figure 15: GaEMA – traffic-dep. $m_{\mathrm{norm}}$, Internet traffic

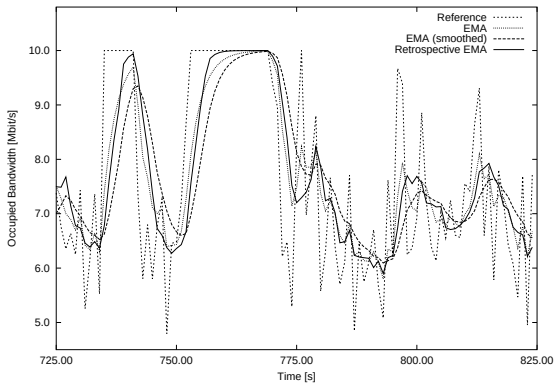


Figure 16: Retrospective EMA, Internet traffic

tions and to filter out short-term effects. For the third one, a smoothing algorithm was used to filter out short-term effects before employing the EMA algorithm. The simple EMA algorithm was then adapted to get the maximum out of the smoothed results, by modifying the history.

The simulations showed that the low pass EMA algorithm works generally better than the gradient adaptive EMA algorithm. The first one does not differ much from a simple EMA algorithm, but the second one stays too close to the measurement and produces numerous unwanted peaks that should have been levelled out. Whether these algorithms could perform better with different parameters has yet to be investigated. It has been shown that they show the same behavior under different traffic conditions. For both approaches, the link-dependent normalizing gradient has shown an unsatisfying performance.

The retrospective EMA algorithm showed a good performance. The curve runs very smooth, levelling out minor peaks. At the same time, it reacts at least as fast as the simple EMA algorithm but converges faster. Using a small series of samples (in our example it was five samples) to smooth the curve, the computational overhead is sufficiently small.

Our further work, will take a closer look at the parametrisation hoping that different weights might lead to better results, independently of the traffic. We will also investigate the dynamic calculation of the traffic-dependent normalizing gradient. This will improve the usability of these algorithms since no arbitrary setting of $m_{\mathrm{norm}}$is necessary. We will also examine the combination of retrospective behavior and dynamic adaptation of the weight could bring further enhancement.

During the evaluation of our modifications, several other ideas for adaptive estimation algorithms were developed, that have to be analyzed. Eventually, we like to investigate the effects of bandwidth estimation on routing decisions. Therefore we have to deploy a larger simulation environment, that uses several nodes where traffic is really routed.

# References

[1] Technical Committee, The ATM Forum, *Traffic Management Specification, Version 4.0*, ATM Forum/95-0013R10. February 1996.

[2] Zhang, L., Berson, S., Herzog, S., Jamin, S., Braden, R., *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification*, RFC 2205, IETF, September 1997.

[3] Kamat, S., Guerin, R., Herzog, S., *QoS Path Management with RSVP*, Proceedings of 2nd Global Internet Miniconference (joint with Globecom'97), Phoenix, AZ, November 1989.

[4] Moy, J., *OSPF Version 2*, RFC 2328, IETF, April 1998.

[5] Williams, D., Kamat, S., Guerin, R., Orda, A., Przygienda, T., Apostolopoulos, G., *QoS Routing Mechanisms and OSPF Extensions*, RFC 2676, IETF, August 1999.

[6] Shenker, S., Danzig, P., Jamin, S., *Comparison of Measurement-based Admission Control Algorithms for Controlled-Load Service*, Proceedings of IEEE INFOCOM '97, pp. 973–980, Kobe, Japan, April 1997.

[7] LeBoudec, J.-Y., Ferrari, T., Almesberger, W., *Scalable resource reservation for the Internet*, Proceedings of IEEE Conference, Protocols for Multimedia Systems – Multimedia Networking (PROMSMmNet), Santiago, Chile, November 1997.

[8] *IND Simulation Library* http://www.ind.uni-stuttgart.de/INDSimLib/

[9] Roth, R. (Ed.), *QUASAR – QoS Reference Model*, http://www.ind.uni-stuttgart.de/quasar/, 2001.