RAEM: An Asynchronous & Randomized Bandwidth Adjustment
Algorithm

Jeff Edmonds, Suprakash Datta, and Patrick Dymond
Computer Science Department
York University, Toronto, Canada
jeff,datta,patrick@cs.yorku.ca

July 1, 2003

Abstract

We study a congestion control mechanism which generalizes the well-known RED (Random
Early Detection) algorithm that achieves smoother transmission rates and higher goodput by
randomly dropping packets. We prove that the system converges quickly.

While TCP works well in practice in general, it suffers from a few drawbacks. In this section,
we address two of these: adding asynchrony in the adjustment times of the jobs and minimizing
the number of packets that are actually dropped.

In our TCP model all jobs adjust their transmission bandwidth at exactly the same time.
The problem with this is that the total bandwidth being utilized then continually decreases and
increases again. Though in practice delays in transmission and the like are likely to introduce
considerable asynchrony to the adjustment times of the jobs, it is still likely that this effect occurs.
Our new model, explicitly adds asynchrony. Inadvertently, this has the additional benefit that is
automatically hits jobs with more bandwidth harder, bringing the system to EQUI faster.

The second drawback addressed in this section is packet loss. One obvious way of minimizing the
number of packets being dropped is to introduce a smaller virtual bottleneck. Instead of actually
dropping packets that are transmitted beyond this smaller bottleneck, the packets are only marked.
The sender is then to react the same as if the packet had actually been dropped, except for the
fact that it would not have to retransmit the packet.

One strategy that has been proposed is called Random Early Detection (RED) [?] marks a
constant fraction of the packets as “dropped”. If there are lots of packets, this still amounts to
both a large number of packets being marked per job and a large number jobs having at least one
packet being marked. This section takes this idea a step further. We design an algorithm RAEM
(random asynchronous early marking), in which the bottleneck will mark only one packet at a time
causing only one job to adjust at a time.

The new model is as follows. Instead of the bottleneck simultaneously signaling all the jobs to
adjust their bandwidth, the bottleneck periodically marks one randomly chosen packet as being
“dropped”. This causes the sender of this packet to adjust, i.e. decrease his bandwidth $b_{i,t}$ by a
multiplicative factor of $\beta$. The probability that job $J_i$ with bandwidth $b_{i,t}$ is selected is $b_{i,t}/\sum_i b_{i,t}$.  

The goal of the bottleneck is to determine how often to signal these adjustments in order to maintain
a total bandwidth $\sum_i b_{i,t}$ utilization that is just slightly under the bottleneck’s capacity $B$ and to
ensure that all jobs are allocated the same bandwidth. The difficulty for the bottleneck is that it
has access to very little information about the state of the system. We assume that it does know
the current total $\sum_i b_{i,t}$ bandwidth through it, denoted $b_t$. However, it does not know which job
a given packet belongs to and hence it is unable to select a particular job for an adjustment. It
does not know the number $n_t$ of jobs active and hence does not know the fair amount of bandwidth
$b_{i,t} = \frac{B_i}{n_t}$ that each job should be operating at. Finally, it does not know the individual bandwidths,
$b_{i,t}$, currently being used by the jobs and hence it does not know the amount $(1 - \beta)b_{i,t}$ that the
bandwidth will drop when a single adjustment occurs. Within this model, we devise an algorithm
for the bottleneck which ensures that all the job’s bandwidths converge to the desired level at least
as quickly, $O(\frac{B}{\alpha n}(\ln(n) + q))$, as they do in the TCP model.

The reason why this bottleneck model inadvertently bringing the system to EQUI faster than
TCP is that there are now two separate forces in this direction. The first such force is that as with
TCP, jobs with more bandwidth $b_{i,t}$ decrease their bandwidth by more, namely by $(1 - \beta)b_{i,t}$. The
second such force is that jobs with more bandwidth $b_{i,t}$ are more likely to hit with an adjustment
because they are sending more packets that might get marked. Recall that the probability is $b_{i,t}$. The sum effect is that the difference between the individual bandwidths changes, not linearly, but
quadratically with the current difference.

The first step in designing our algorithm is to define the desired level of bandwidth utilization.
This is set by a fixed function $\tilde{b}(n_t)$. The goal for the bottleneck is to maintain a total bandwidth
$b_t = \sum_i b_{i,t}$ that is equal to $\tilde{b}(n_t)$, where $n_t$ is the current number of active jobs. A good candidate
is $\tilde{b}(n) = (1 - \gamma)(1 - e^{-\alpha n})B$. In order to be as general as possible, however, our analysis allows this
function $\tilde{b}(n)$ to be chosen arbitrarily by the implementer subject to the following requirements. Presumably, it will stay relatively close, yet under, the total bandwidth $B$ and is defined for a
large range of values $n$. It needs to be monotonically increasing, i.e. more jobs utilize more of the
bandwidth. The last requirement is that the second derivative $\tilde{b}''(n)$ is negative. (This is mainly
used to insure that $\frac{\Delta \tilde{b}(n)}{\Delta n} \leq \frac{\tilde{b}(n)}{n}$.) This is reasonable because with larger and larger $n$, $\tilde{b}(n)$ will
need to get squeezed closer and closer to the capacity $B$.

One would think that having the desired bandwidth $\tilde{b}(n_t)$ be a function of the number of active
jobs would complicate things because the bottleneck does not know this number. However, doing
so allows the bottleneck to guesstimate the number of active jobs. Define $\tilde{n}(b_t) = \tilde{b}^{-1}(b_t)$ to be
the inverse function. If the algorithm is working correctly and the current total bandwidth is $b_t$
then $\tilde{n}(b_t)$ would be the number of active jobs. In general, the bottleneck uses $\tilde{n}(b_t)$ as its best
approximation of the number of active jobs.

The remaining step in designing the algorithm is to define the function $f(b_t) = \frac{\alpha}{(1-\beta)} \frac{\tilde{n}(b_t)^2}{b_t}$,
which tells the bottleneck the frequency (drops per unit time) at which to mark single packets as
being “dropped”. Note this rate depends only on the total bandwidth $b_t$ that the bottleneck is
receiving, because this is all that the bottleneck knows about the system. Algorithmically, it then
makes the most sense for a packet to be dropped at a steady rate of once every $\frac{1}{f(b_t)}$ time units. The
proof of correctness, however, is simpler if the process is made continuous, i.e. every small interval
$\delta t$ of time the algorithm makes one of the jobs adjust with probability $f(b_t)\delta t$. The analysis then
focuses on the continuous expected change.

The drop frequency function $f(b_t)$ is designed to maintain a steady state at the desired levels,
i.e. that in which each of the $n$ jobs have bandwidth $b_{i,t} = \frac{\tilde{b}(n)}{n}$ for a total of $b_t = \tilde{b}(n)$. This
is done as follows. Each job is increasing its bandwidth at an fixed additive rate of $\alpha$. Hence
the total bandwidth is increasing at a rate of $\alpha n$. A single adjustment decreases the single job’s
bandwidth from $b_{i,t} = \frac{\tilde{b}(n)}{n}$ to $\beta b_{i,t} = \beta \frac{\tilde{b}(n)}{n}$. This decreases the total bandwidth by $(1 - \beta)\frac{\tilde{b}(n)}{n}$.
The frequency of adjustments per time unit is $f(b_t)$. Hence, these adjustments decrease the total
bandwidth at a rate of $f(b_t)(1 - \beta)\frac{\tilde{b}(n)}{n}$. The bottleneck maintains the current total bandwidth by
balancing this increase and this decrease, namely $\alpha n = f(b_t)(1 - \beta)\frac{\tilde{b}(n)}{n}$. This is done by setting the
adjusting frequency to \( f(b_t) = \frac{\alpha}{(1 - \beta) b(n)} \). Not knowing the number of jobs \( n \), the bottleneck uses the approximation \( \tilde{n}(b_t) \). The bottleneck’s algorithm is to adjust at a frequency of \( f(b_t) = \frac{\alpha}{(1 - \beta) \frac{\tilde{n}(b_t)^2}{b_t}} \) when the current total bandwidth is \( b_t \). Standard TCP, as stated in Lemma 6, has all \( n \) jobs adjust every \( \frac{1}{\alpha n} \) time units, giving that the frequency at which some job adjusts is the same.

The rest of this section is to prove an analogous version of that in [EDDsingle] for this new asynchronous model of adjusting, namely that the algorithm is competitive with OPT given extra time and bandwidth. This theorem relies only on Theorem 2. Hence, it is sufficient to prove an analogous version of it. To simplify the analysis, we assume that during the period of time during which the algorithm is converging to \( \text{EQUI} \), the set \( \mathcal{J} \) of active jobs remains fixed. The bandwidths \( b_{i,t_0} \) of these jobs, at the beginning of this converging period, however, can be arbitrary. We also assume a sufficiently large number of randomly chosen adjustments so that the frequency that each job adjusts is effectively equal to its expected frequency.

**Theorem 1**: Independent of the initial bandwidths \( b_{i,t_0} \) of the \( n \) jobs, these bandwidths converge in time \( \mathcal{O}(\frac{B}{\alpha n}(\ln(n) + q)) \) to be within a factor of \((1 - 2^{-q})\) of the desired levels \( \tilde{b}_{i,t} = \frac{b(n)}{n} \).

According to Theorem 2, this time is the same as that needed for TCP.

**Proof of Theorem 1**: There are two dynamics that cause this convergence to happen. We will separate them by considering them in separate stages.

The first dynamic is that as with TCP given in [EDDsingle], the job’s bandwidths converge to being equal. Lemma 2 proves that after \( \mathcal{O}(\frac{B}{\alpha n}(\ln(n) + q)) \) time, each job’s bandwidth \( b_{i,t} \) is at most a factor of \((1 - 2^{-q})\) away from the balanced level \( \frac{b(n)}{n} \). Note that at this point we do not know the value of \( b_t \). This completes the first stage.

The second dynamic is that when the individual bandwidths are unbalanced, the total bandwidth \( b_t \) decreases lower than it should. However, Lemmas 3, 4, and 5 prove that when the job’s bandwidths are as close to being equal as they are after the first stage, then the total \( b_t \) converges to being within a factor \((1 - 2^{-q})\) from \( \tilde{b}(n) \) within time \( \mathcal{O}(\frac{B}{\alpha n}) + \mathcal{O}(\frac{g \delta b(n)}{\alpha \delta n}) \). This completes the second stage.

The remaining step is to prove that \( \delta \frac{\tilde{b}(n)}{\delta n} \leq \tilde{b}(n) \leq \frac{B}{n} \). Note that in our candidate function \( \tilde{b}(n) = (1 - \gamma)(1 - e^{-cn})B \), \( \tilde{b}(n) = (1 - \gamma)e^{-cn}B \ll \frac{b(n)}{n} \). We see as follows that this is true for all legal functions \( \tilde{b}(n) \). Having a negative second derivative insures that \( \frac{\delta b(x)}{\delta x} \geq \frac{\delta b(n)}{\delta n} \) for \( x \in [0, n] \). It follows that \( \tilde{b}(n) = \tilde{b}(0) + \int_{x=0}^{n} \frac{\delta b(x)}{\delta x} \geq \left[ n \right] \frac{\delta b(n)}{\delta n} \).

The first step is to determine how the job’s individual bandwidths \( b_{i,t} \) change.

**Lemma 1**: \( \frac{\delta b_{i,t}}{\delta t} = \alpha \left[ 1 - \left( \frac{b_{i,t} \tilde{n}}{b_t} \right)^2 \right] \)

This change moves job \( J_i \)’s bandwidth \( b_{i,t} \) continuously towards \( \frac{b_t}{n(b_t)} \), which is the bottleneck’s best approximation of what each job’s bandwidth should be, i.e. \( b_{i,t} \) increases when it is smaller than this and decreases when it is large.

**Proof of Lemma 1**: Consider job \( J_i \). It increases its bandwidth at an additive rate of \( \alpha \). The bottleneck signals for an adjustment from some job at a frequency of \( f(b_t) = \frac{\alpha}{(1 - \beta) \frac{\tilde{n}(b_t)^2}{b_t}} \) and when it does this job is hit with probability \( \frac{b_{i,t}^2}{b_t^2} \). Hence, job \( J_i \)’s expected frequency of adjustments is \( f(b_i) \frac{b_{i,t}}{b_t} \). Each such adjustment decreases its bandwidth by \((1 - \beta)b_{i,t}\). In conclusion, the expected
rate of change of job \( J_i \)'s bandwidth \( b_{i,t} \) is as follows.

\[
\frac{\delta b_{i,t}}{\delta t} = \alpha - f(b_{i}) \frac{b_{i,t}}{b_e} (1 - \beta) b_{i,t} = \alpha - \left[ \frac{\alpha}{(1 - \beta) b_e} \right] b_{i,t} (1 - \beta) b_{i,t} = \alpha \left[ 1 - \left( \frac{b_{i,t} \tilde{n}}{b_e} \right)^2 \right]
\]

Lemma 2

Independent of the initial bandwidths are from being equal. This is the reciprocal of that used in \( [\ ] \) purpose. It has a number of useful properties.

Proof of Lemma 2:

The main task of the proof is that independent of the current state of the system, \( \mathcal{M}_t \) decreases at a rate of \( \frac{\Delta \mathcal{M}_t}{\Delta t} \leq -\frac{2\alpha}{b_n} (\mathcal{M}_t - 1) \). This change causes the value of \( \mathcal{M}_t - 1 \) to decrease by a factor of \( e \) in at most time \( \frac{b_n}{2\alpha n} \leq \frac{B}{2\alpha n} \). (We are assuming that the total bandwidth \( b_t \) never exceeds the bottleneck’s capacity \( B \).) Because initially \( \mathcal{M}_t \) is at most \( n \), \( \mathcal{M}_t - 1 \) becomes at most \( n - q \) in at most \( \mathcal{O}(\log(\alpha n)) \) such half lives. The result follows.

The change in \( \mathcal{M}_t \) is computed as follows.

\[
\mathcal{M}_t = \frac{n \left( \sum_i b_{i,t}^2 \right)}{\left( \sum_i b_{i,t} \right)^2} = \frac{n \left( \sum_i b_{i,t}^2 \right)}{(b_t)^2}
\]
\[ \frac{\delta M_t}{\delta t} = \frac{n}{b_t} \left[ \left( \sum_i 2b_{i,t} \frac{\delta b_{i,t}}{\delta t} \right) (b_t^2) - \left( \sum_i b_{i,t}^2 \right) \left( 2b_t \frac{\delta b_t}{\delta t} \right) \right] \]

\[ = \frac{2n}{b_t^3} \left[ \sum_i b_{i,t} \left( \alpha \left[ 1 - \left( \frac{b_{i,t}\tilde{n}}{b_t} \right)^2 \right] \right) \right] - \left( \sum_i b_{i,t}^2 \right) \frac{b_t}{b_t^2} \left( \sum_i \alpha \left[ 1 - \left( \frac{b_{i,t}\tilde{n}}{b_t} \right)^2 \right] \right) \]

\[ = \frac{2\alpha n}{b_t^3} \left[ b_t^2 - \tilde{n}^2 \sum_i b_{i,t}^3 - \left( \sum_i b_{i,t}^2 \right) b_t \tilde{n} \right] + \left( \sum_i b_{i,t}^2 \right) \tilde{n}^2 \frac{\delta n}{\delta t} \left[ \sum_i b_{i,t}^2 \right] \]

At this point, it is useful to observe that \( \sum_i b_{i,t}^2 \geq \frac{1}{\delta n} (\sum_i b_{i,t}^2)^2 \) for any values \( b_i \geq 0 \). The intuition is similar to that for the standard fact that \( \sum_i b_i^2 \geq \frac{1}{n} (\sum_i b_i)^2 \). It is more significant to cube the individual large values before summing than only squaring them. It is interesting, however, that equality is achieved both when the values are either completely equal or completely unbalance. The maximum difference occurs when there are two distinct values. The proof has not been included. Using it, our above expression simplifies.

\[ -\frac{\delta M_t}{\delta t} \geq \frac{2\alpha n}{b_t^3} \left[ -b_t^3 + \left( \sum_i b_{i,t}^2 \right) b_t \tilde{n} \right] = \frac{2\alpha n}{b_t} \left[ -1 + n \frac{\left( \sum_i b_{i,t}^2 \right)}{b_t^2} \right] = \frac{2\alpha n}{b_t} (M_t - 1) \]

We will now see that while the individual bandwidths are unbalanced, the total bandwidth \( b_t \) and the approximation \( \tilde{n}(b_t) \) both decreases lower than they should. However, as soon as the bandwidths are close, they converge quickly to the desired levels \( \tilde{b}(n) \) and \( n \). Note that depending on the initial total bandwidth \( b_t \) and the function \( \tilde{n}(\cdot) = \tilde{b}^{-1}(\cdot) \) that was chosen, the initial value of \( \tilde{n}(b_t) \) could even be infinite. Our first step is to ensure that in such a case \( \tilde{n}(b_t) \) decreases quickly to at most \( 2n \).

**Lemma 3** Independent of its initial value, \( \tilde{n}(b_t) \) becomes at most \( 2n \) in time \( O\left( \frac{1}{\alpha \frac{\delta \tilde{b}(n)}{\delta n}} \right) \).

**Proof of Lemma 3:** By Lemma 1, the rate of change of the approximation \( b_t \) and hence of \( \tilde{n}(b_t) \) are

\[ \frac{\delta b_t}{\delta t} = \sum_i \frac{\delta b_{i,t}}{\delta t} = \sum_i \alpha \left[ 1 - \left( \frac{b_{i,t}\tilde{n}}{b_t} \right)^2 \right] = \alpha \left[ n - \frac{\left( \sum_i b_{i,t}^2 \right)}{b_t^2} \tilde{n} \right] = -\alpha \left[ M_t \left( \frac{\tilde{n}}{n} \right)^2 - 1 \right] \]

\[ \frac{\delta (\tilde{n} - n)}{\delta t} = \frac{\delta \tilde{n}}{\delta b_t} \frac{\delta b_t}{\delta t} = -\frac{\delta \tilde{n}}{\delta b_t} \alpha n \left[ M_t \left( \frac{\tilde{n}}{n} \right)^2 - 1 \right]. \]

When the individual bandwidths \( b_{i,t} \) are unbalanced, the measure \( M_t \) is large, and hence both \( b_t \) and \( \tilde{n}(b_t) \) decrease lower than they should. However, for our purposes of decreasing \( \tilde{n}(b_t) \) from possibly infinity to \( 2n \), this only helps.

Recall that the function \( \tilde{b}(n) \), which dictates the desired total bandwidth when there are \( n \) jobs, is chosen by the designer. However, we impose on it the requirement that its second derivative is negative. Hence, \( 1/\frac{\delta \tilde{b}(b_t)}{\delta n} = \frac{\delta \tilde{b}(\tilde{n})}{\delta n} \leq \frac{\delta \tilde{b}(n)}{\delta n} \) when \( \tilde{n} \geq n \). This gives:

\[ \frac{\delta (\tilde{n} - n)}{\delta t} \leq -\frac{1}{\frac{\delta \tilde{b}(n)}{\delta n}} \alpha \left[ n^2 - \tilde{n}^2 \right] \leq -\frac{1}{\alpha \frac{\delta \tilde{b}(n)}{\delta n}} \frac{\alpha}{n} \left( \tilde{n} - n \right)^2. \]

Again, the form of this differential equation is \( \frac{\delta \tilde{n}}{\delta t} = -c\Delta^2 \), which as solution \( \Delta_t = \frac{1}{T(\Delta_0 + ct)} \). Hence, \( |\tilde{n} - n| \) becomes \( \Delta_T = n \), independent of its initial value \( \Delta_0 \), in time \( T \leq \frac{1}{c\Delta} = \frac{\alpha}{\frac{\delta \tilde{b}(n)}{\delta n}} \frac{\alpha}{n} = \frac{1}{\alpha} \frac{\delta \tilde{b}(n)}{\delta n} \).
The initial total bandwidth $b_t$ and the function $\tilde{n}(\cdot)$ may be such that the initial value of $\tilde{n}(b_t)$ is as small as zero. As well, having the bandwidths unbalanced may decrease $\tilde{n}(b_t)$ even lower than it already is. Our second step is to ensure that in such a case, as soon as the bandwidths are balanced, $\tilde{n}(b_t)$ increases quickly to at least $\frac{1}{2}n$.

**Lemma 4** When the job’s bandwidths are close to being equal, i.e. $M_t \leq 1 + 2^{-q}$, independent of its initial value, $\tilde{n}(b_t)$ becomes at least $\frac{1}{2}n$ in time $O\left(\frac{\delta b}{\alpha n}\right)$.

**Proof of Lemma 4:** The proof of Lemma 3 gives that the change in the total bandwidth is $\frac{db}{dt} = \alpha n[1 - M_t(\tilde{\frac{n}{2}})^2] \geq \alpha n[1 - (1 + 2^{-q})(\frac{1}{2})^2]$, when $M_t \leq \alpha n$ and $\tilde{n} \leq \frac{1}{2}n$. At this rate of increase, in time $O\left(\frac{\delta b}{\alpha n}\right)$, this total bandwidth $b_t$ would increase by more than the capacity $B$ of the bottleneck, at which time we would be sure that $\tilde{n}(b_t) \geq \frac{1}{2}n$.

The remaining step is to bound the time required for the approximation $\tilde{n}$ to fine tune itself to $n$.

**Lemma 5** When the job’s bandwidths are close to being equal, i.e. $M_t \leq 1 + 2^{-q}$, and the approximation $\tilde{n}(b_t)$ is bounded within $[\frac{1}{2}n, 2n]$, the total bandwidth $b_t$ converges to being within a factor of $(1 - 2^{-q})$ of $b(n)$ within time $O\left(\frac{q \delta b(n)}{\alpha n}\right)$.

**Proof of Lemma 5:** The difference, $|b_t - b(n)|$, between the actual and the desired total bandwidth will be at most $2^{-q}\tilde{b}(n)$, when the difference, $|\tilde{n}(b_t) - n|$, between the bottleneck’s approximation and the actual of the number of active jobs is at most $\left(\frac{1}{\tilde{b}(n)}\right) \cdot 2^{-q}\tilde{b}(n)$. By the statement of the lemma, $|\tilde{n}(b_t) - n|$ is initially at most $n$. Hence, we require $|\tilde{n}(b_t) - n|$ to decrease by a factor of $n \cdot \frac{\delta b(n)}{\alpha n} \cdot 2^{-q}$. In the proof of Theorem 1, it was proved that this is at most $2^q$. Hence, the number of time that $|\tilde{n}(b_t) - n|$ must decrease by a factor of 2 is at most $O(q)$.

The remaining step is to compute how long it takes for the error $|\tilde{n} - n|$ to decrease by a factor of 2. Suppose that $|\tilde{n} - n| = \epsilon n$ for some $\epsilon \geq 2^{-q}$.

$$\frac{\delta |\tilde{n} - n|}{\delta t} = -\frac{\delta n}{\delta b_t} \alpha n |M_t n^2 - n^2| \leq -\frac{\delta n}{\delta b_t} \alpha n (1 + 2^{-q}) ((1 + \epsilon)n)^2 - n^2| \leq -\Theta \left(\frac{1}{\alpha n} \cdot 2^{-q}\tilde{b}(n)\right) \cdot \epsilon n.$$  

Note that because $\tilde{n}(b_t) = \Theta(n)$, $1/\frac{\delta \tilde{n}(b_t)}{\delta n} = \Theta\left(\frac{\delta \tilde{b}(n)}{\delta n}\right)$.

For $|\tilde{n} - n| = \epsilon n$ to decrease by a factor of 2, it must decrease by $\frac{\epsilon n}{2}$. At the above rate of change, this will take time $\epsilon n \cdot \frac{\delta b(n)}{\alpha n} \cdot \frac{1}{\alpha n} = O\left(\frac{\epsilon n}{\alpha n} \cdot \frac{\delta b(n)}{\delta n}\right)$.  

One measure worth considering is the length of an adjustment period. Having this long has the advantage of decreasing the frequency in which the bottleneck and the senders must deal with adjustments. Having it short has the advantage of decreasing the time $D(J)$ that jobs must wait until its gets its fair allocation of bandwidth.

**Lemma 6** The length of an adjustment period is $|\tau_{j+1} - \tau_j| = \frac{(1 - \beta) B}{\alpha n^T} + (1 - \beta) \delta$, where $n^T$ denotes the (average) number jobs alive under TCP during the period.

**Proof of Lemma 6:** At the point in time when the bottleneck reaches capacity, the total bandwidth allocated to jobs is clearly the bottleneck’s capacity $B$. If there is a delay of $\delta$ in time before the senders detect packet loss, then during this time each sender continues to increase its transmission rate at the additive rate of $\alpha$. The total transmission rate after this delay will be $B + \alpha n^T \delta$.  

\[\text{\begin{tabular}{c} \end{tabular}}\]
This paper considers two strategy that TCP might take at this point. The first strategy is for
the sender to decrease its transmission rate to the fraction $\beta$ of its current rate of sending data.
Doing this would decrease the total transmission rate to $\beta(B + \alpha n^T_t \delta)$. (It is problematic if this
delay $\delta$ is so big that this adjusted rate is still be bigger than the capacity $B$ of the bottleneck.)
The second strategy is to decrease its transmission rate to a fraction $\beta$ of the current rate that
data passes through the bottleneck without getting dropped. Doing this would decrease the total
transmission rate to only $\beta B$. (Here there is no limit on how large the delay $\delta$ can be.)

With either strategy, the total bandwidth allocated continues to increase at a rate of $\alpha n^T_t$. The
time required for the total to increase again to $B$ is $\frac{B-\beta B}{\alpha n^T_t}$ in the first strategy and only
$\frac{B-\beta B}{\alpha n^T_t}$ in the second. The total length of the adjustment period is this plus the $\delta$ delay time, which
is either $|\tau_{j+1} - \tau_j| = \frac{(1-\beta)B}{\alpha n^T_t} + (1 - \beta)\delta$ or $|\tau_{j+1} - \tau_j| = \frac{(1-\beta)B}{\alpha n^T_t} + \delta$. ■

\begin{theorems}
Let $q \geq 1$ be an integer, $s$ be any value, and $\mathcal{J}$ be any set of jobs. For each job $J_i$
and for all times $t = \tau_j^q + q^j, j \geq 0, b^T_{i,t} \geq (1 - \beta^q) \frac{B}{n^T_t}$, where $b^T_{i,t}$ denotes the bandwidth allocated
by TCP$_s(\mathcal{J})$ to job $J_i$ at time $t$ and $n^T_t$ denotes the number jobs alive at this time. (On the other
hand, at all times $t \geq \tau_j^q + \log(n)/\log(1/\beta) + q$, $b^T_{i,t} \leq (1 + \beta^q) \frac{B}{n^T_t}$.
\end{theorems}