The Control of Burstiness in Fair Queueing Scheduling

Liat Ashkenazi*and Hanoch Levy †
School of Computer Science, Tel-Aviv University
Tel-Aviv, Israel

August 26, 2004

Abstract

Bennett and Zhang [4] demonstrated the existence of large discrepancies between the service provided by the packet-based WFQ system and the fluid GPS system. As demonstrated in [4] these discrepancies can cause cycles of bursting. Such inaccuracy and bursty behavior significantly and adversely affect both best-effort and real-time traffics. The WF^{2}Q algorithm [4] overcomes this difficulty but its computational complexity is high relatively to other scheduling algorithms. Those algorithms, in contrast, are subject to the burstiness problem.

This paper studies the issue of bursty transmission in packet scheduling algorithms. We propose the Burst-Constrain Fair-Queueing (BCFQ) algorithm, a packet scheduler that achieves both high fairness and low burstiness while maintaining low computational complexity. We also propose a new measure (and criterion) which can be used for evaluating the burstiness of arbitrary scheduling algorithms. We use this measure and demonstrate that BCFQ possesses the desired properties of fairness and burstiness.

1 Introduction

Next generation networks are built to support a variety of applications with a wide range of Quality of Service (QoS) requirements. QoS guarantees in a packet network require the use of traffic scheduling algorithms in switches or routers. The choice of a packet service discipline at queueing points in the network is one of the most important issues in designing next generation networks. Fair Queueing algorithms have received much attention, since they provide a separation between different service classes, while letting them to share the available bandwidth.

Processor Sharing (PS), presented in Kleinrock [11], is an idealistic service discipline in which all active sessions are served concurrently and the processor rate is equally shared among them. GPS is a generalized of PS where the service rate granted to a session is proportional to its weight. As a result, it is possible to divide the service among the sessions, at all times, exactly in proportion to the specified service share. In practice packets must be processed one by one and thus GPS is not practical and only serves as an idealistic model.

Number of different packet-by-packet scheduling algorithms that aim at approximating GPS have been proposed. The earliest such algorithm was Weighted Fair Queueing (WFQ), or PGPS, presented by Parekh and Gallager [14],[12],[13]. WFQ emulates GPS's behavior by computing a virtual time function. The virtual time is used to compute a

*E-mail : `liatashk@post.tau.ac.il`

†E-mail : `hanoch@cs.tau.ac.il`

time stamp for each arriving packet, indicating the time at which it would depart the system under GPS. Packets are then transmitted in increasing order of their timestamps. One problem of WFQ is that it leads to high computational complexity and makes the scheme infeasible for high speed applications. Another problem of WFQ, as demonstrated by Bennett and Zhang [4], is that there could be severe discrepancies between the service provided by WFQ and GPS. Specifically it was shown that the amount of service WFQ provides to a session can be much larger than that provided by GPS. This inaccuracy leads to the scheduler producing bursty output. Such burstiness adversely affects (a) delay bounds for real-time traffic when there is hierarchical link sharing; and (b) traffic management algorithms for best-effort traffic.

The WF^{2}Q scheduling proposed in [4] overcomes this problem by choosing the sessions with the minimum finish time just among sessions that already have started their service in GPS. The service provided by WF^{2}Q is almost identical to that of GPS, differing by no more than one maximum size packet. However its computational complexity is still high.

Several scheduling algorithms have been proposed to reduce the computational complexity of WFQ. Self-Clocked Fair Queueing (SCFQ) presented by Golestani [9],[8] and Start-Time Fair Queueing presented by Goyal, Vin, and Cheng [10] compute a self-clock as the index of work progress. Stiliadis and Varma [19],[17] proposed Rate-Proportional Server (RPS). RPS uses a system potential function that maintains the global state of the system by tracking the service offered by the system to all sessions sharing the outgoing link. Greedy algorithms as Greedy Fair Queueing (GrFQ) presented by Shi and Sethu [16] were proposed to minimize the maximum differences between sessions. None of these efficient schedulers provides a solution to the burstiness problems of WFQ.

The objective of this paper is to study the burstiness problem in Fair Queueing scheduling and to device an efficient solution to it. We start (Section 2) by analyzing the existing algorithms. In this context we first provide an explicit discussion of the complexity limitations of emulating GPS; such discussion is necessary, as the literature seems to be somewhat unclear about this issue. We conclude that this complexity is quite high, $O(N)$ operations per packet transmission. We then show that *efficient* scheduling algorithms like SCFQ and GrFQ, that operate via some approximation, are subject to the busrtiness problem, as observed in [4] for WFQ.

Having addressed the efficiency and burstiness problems of existing algorithms, we propose (Section 3) a new scheduler, Burst Constrain Fair Queueing (BCFQ), that is both *efficient* and not subject to the *burstiness* problems. The BCFQ algorithm is based on two principles. Its major scheduling decisions are based on using the *normalized service* of both the system and the individual sessions. Burstiness is prevented by computing an *approximate virtual time* (whose computational complexity is low) and comparing the session's normalized service against it. The complexity of BCFQ is similar to that of SCFQ and GrFQ, namely $O(logN)$ operations per packet; in contrast, the complexity of WFQ and WF^{2}Q is $O(N)$ operations per packet transmission. Having devised BCFQ we then (Section 4) show that its fairness is identical to that of previous algorithms by showing that it admits to the Relative Fairness Bound (RFB).

We then (Section 5) turn to address the burstiness issue. First we recognize that while a measure of algorithm fairness has been proposed and widely used, a similar measure for burstiness does not exist. We thus propose the Relative Burstiness bound (RB) as a criterion of burstiness that can be applied easily to various algorithms to examine whether they are bursty or not. We further show that the RB criterion is equivalent to measuring "proximity to GPS". We also show that a scheduling policy whose Relative Fair Bound is low can have its relative burstiness high and thus the RFB is not appropriate for measuring burstiness.

Having devised a burstiness criterion we then (Section 6) demonstrate the low burstiness of BCFQ. We do that by showing that when the active session set is constant BCFQ behaves exactly as WF^{2}Q. Thus, under this setting BCFQ admits the RB criterion. Further, we discuss one more property of BCFQ, according to which one can achieve very low burstiness for BCFQ if one uses the GPS virtual time to assist the computation of BCFQ. The burstiness achieved then is precisely that of WF^{2}Q (thus admitting the RB criterion) and this is achieved for general conditions (arbitrary arrivals and not-necessarily-constant active set). This, however, is achieved at the expense of higher complexity.

We then (Section 7) examine the burstiness of BCFQ, under general conditions, via simulation. We carry out an array of simulation runs and show that under general conditions (that is, the actives sessions are not necessarily constant) the burstiness of BCFQ is very low. Finally, concluding remarks are given in Section 8. For the reader's convenience a glossary of notation is provided in Appendix A.

2 Existing FQ Schedulers: Properties and Complexity

2.1 Preliminaries

A session consists of an infinite sequence of packets, which are stored in a FIFO queue. The system consists of N sessions and one output link. The server operates at a fixed rate r and is work-conserving[1]. The N sessions share the same output link. Let $S_{i,P}(0,t)$ denote the total amount of service received by session i by time t under scheduler P and w_i the weight of session i.

A session is defined to be *Active* or *Backlogged* at time t if its queue is not empty at that time. Let $A(t)$ denote the active session set at time t. In the event that the active session set does not change during the interval (t,t'), let $A(t,t')$ denote this set.

A *busy period* is a maximal-length interval of time during which at least one of the N sessions is active. Throughout the paper we assume the system is busy, if the system becomes idle all its variables become zero and the time is initiated.

Definition 2.1 System variables: The overall service granted by the system by time t is defined as $S_P(0,t) = \sum_i S_{i,P}(0,t)$. The *system weight* at time t is defined as $W(t) = \sum_{i \in A(t)} w_i$ and the *system weight* during interval (t,t'), when the active session set on this interval is constant, is defined as $W(t,t') = \sum_{i \in A(t,t')} w_i$.

Definition 2.2 The change of a session from active to inactive (or vice versa) is defined as an *event*, where t_k is the time of the k^{th} *event* (simultaneous events are ordered arbitrarily).

Definition 2.3 Two schedulers with different service disciplines are called *corresponding* schedulers of each other if they have the same speed, same set of sessions, same arrival pattern , and if applicable, same service share for each session.

2.2 The Complexity of GPS Emulation

Below we provide a review of the complexity results of GPS emulation, where we conclude that its complexity is $O(N)$ operations per packet transmission. This review is given in some detail, since to the best of our knowledge, these details are not very clear in the literature. An emulation of GPS which is based on a virtual time function $v(t)$ was proposed in [7],[14],[12],[13]. The computational complexity of that emulation is composed

[1] A server is work-conserving if it is busy whenever there are packets to be transmitted. Otherwise, it is non-work-conserving.

of two components: The computation of $v(t)$ and the maintenance of the packet finish times. The virtual time is used to track the progress of GPS. Every packet holds a virtual finish time variable which denotes the virtual time at which the packet transmission ends. As presented in [12] the virtual finish time is computed by using the system virtual time $v(t)$. In order to track the progress of GPS, one needs to compute the real time at which the next packet will depart from the GPS system. This is achieved using the current minimum virtual finish time and the system virtual time ($v(t)$).

2.2.1 The complexity of updating $\mathbf{v}(\mathbf{t})$

$v(t)$ is updated using the following recursive equation:

$$v(t_{k-1} + \tau) = v(t_{k-1}) + \frac{\tau}{W(t_{k-1}, t_k)}, \tag{1}$$

where t_k is the time of the k^{th} event (Definition 2.2) and $0 \leq \tau \leq t_k - t_{k-1}$. The total system weight changes every event, therefore the $v(t)$ function must be recomputed at every event. Accordingly the computational complexity of $v(t)$ depends on the number of events that occur. The main problem of the complexity of emulating GPS is that events can occur at an arbitrarily short period. The reason for this phenomenon is that in GPS packets are served simultaneously. Consequently, it is possible that many packets will finish transmission almost in the same time. Thus if many of these packets are the last packets on their queues we observe many events (of queues becoming inactive) occurring in a short period of time. The result is that the number of computations of the $v(t)$ function can, during a single packet transmission, reach the total number of sessions. Assuming that a single computation of $v(t)$ takes $O(1)$ operations, the complexity of computing $v(t)$ is therefore $O(N)$ operations per packet transmission where N is the number of sessions.

2.2.2 The complexity of maintaining packet finish time

In order to track the progress of GPS we need the minimal finish virtual time. Therefore the packet finish time must be stored in a data structure. The data structure must support two operations: Insert a packet finish time, and Extract-Minimum (ExMin) packet finish time; this data structure is called priority queue.

As stated earlier, since in GPS the sessions are served in parallel, there can be many packet departures in an arbitrarily short period. Every packet departure causes ExMin operation. By using a simple and efficient data structure (e.g. Ordered link list), with N departures per packet transmission the complexity can reach $O(N)$, where N is the number of sessions.

To summarize 2.2.1 and 2.2.2, the high complexity in emulating GPS results from the fact that GPS serves the sessions simultaneously and therefore there can be many packet departures during a single packet transmission. When N departures occur during a single transmission unit, both the complexity of computing $v(t)$ is $O(N)$ per packet transmission and the complexity of performing order of N ExMin, using a simple data structure, is $O(N)$ per packet transmission. Under more complicated and efficient data structures that were proposed for maintaining the packet finish time [2],[6],[15], the computation of $v(t)$ is still of high complexity ($O(N)$).

2.3 The Inaccuracy and Burstiness of WFQ and its Variants

Bennett and Zhang presented in [4],[3],[5] the inaccuracy and bursty behavior of WFQ. Below we review that result and show that it applies to many other schedulers such as SCFQ [9] and GrFQ [16].

In [4],[3],[5], the following example is used to illustrate the large discrepancies between the services provided by GPS and WFQ. Assume that there are 11 sessions with packet size of 1 sharing a link with the speed of 1, where $w_1 = 10$, and $w_i = 1$, $i = 2, ..., 11$. Session 1 sends 11 back-to-back packets starting at time 0 while each of all the other 10 sessions sends only one packet at time 0. If the server is GPS, it will take 2 time units to transmit each of the first 10 packets of session 1, one time unit to transmit the 11^{th} packet, and 20 time units to transmit the first packet from each of the other sessions. Denote the k^{th} packet of session j p_j^k, then, under GPS, the packet finish time is $2k$ for p_1^k, $k = 1...10$, 21 for p_1^{11}, and 20 for p_j^1, $j = 2, ..., 11$.

Bennett and Zhang [4] showed that under WFQ [12] the first 10 packets of session 1 (p_1^k,$k = 1...10$) will first be transmitted, followed by one packet from each of sessions 2,...,11 (p_j^1,$j = 2, ..., 11$), and then the 11^{th} packet of session 1 (p_1^{11}). In the example, as explained in [4], between time 0 and 10, WFQ serves 10 packets from session 1 while GPS serves only 5. After such a period, WFQ needs to serve other sessions in order for them to catch up. The difference between the amounts of service provided to each session by WFQ and GPS is the inaccuracy of WFQ in approximating GPS.

Examining SCFQ [9] we observer that packets will be transmitted according to their F_j^k tag. The packet with the lowest F_j^k tag is served first. The F_j^k tag is computed according to the progress of working in the system (the detailed computation can be found in [1], Appendix B there). According to the above example, the F_j^k values are $F_1^k = k/10$, $k = 1...11$ and $F_j^1 = 1$, $j = 2, ..., 11$. Therefore, as in WFQ, the first 10 packets of session 1 (p_1^k,$k = 1...10$) will be transmitted, followed by one packet from each of sessions 2,...,11 (p_j^1,$j = 2, ..., 11$), and then the 11^{th} packet of session 1 (p_1^{11}). Thus SCFQ is subject to the same inaccuracy as WFQ.

Examining GrFQ [16] we observer that packets will be transmitted according to their u_i and $\widehat{u}_i$ session values that are computed according to the progress of working in the system (the detailed computation can be found in [1], Appendix B there). According to the above example, at time 0, $u_i(0) = 0$,$i = 1, ..., 11$, $\widehat{u}_1(0) = 1/10$, and $\widehat{u}_i(0) = 1$, $i = 2, ..., 11$, therefore session 1 is chosen to transmit. At time 1, $u_1(1) = 1/10$, $u_i(1) = 0$,$i = 2, ..., 11$, $\widehat{u}_1(1) = 2/10$, and $\widehat{u}_i(1) = 1$,$i = 2, ..., 11$, therefore session 1 is chosen to transmit, and so on. Thus the first 10 packets of session 1 will be transmitted, followed by one packet from each of sessions 2,...,11, and then the 11^{th} packet of session 1. Thus GrFQ is subject to the same inaccuracy as WFQ and SCFQ. The whole sample path of WFQ, SCFQ, and GrFQ systems is shown in Figure 1.

This cycle of bursting 10 packets and going silent for 10 packet times can continue indefinitely, if more packets arrive to each session. As demonstrated above, there could be large discrepancies between the services provided by WFQ, SCFQ or GrFQ and GPS. As explained in Bennett and Zhang [5],[4],[3], the adverse properties of burstiness are: (a) High delay to real-time traffic in hierarchical link sharing, and (b) Instability of end-to-end control algorithms that caused by the oscillation of service rates introduced by a bursty behavior. Therefore it is important to develop a scheduler that overcomes this bursty behavior with low computational complexity.

WFQ, SCFQ and GrFQ	1 1 1 1 1 1 1 1 1 1 2 3 4 5 6 7 8 9 10 11 1
WF^{2}Q	1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9 1 10 1 11 1

Figure 1: Transmission Order of various schedulers (Packets identified by sessions' index)

2.4 The Properties of WF^{2}Q

The WF^{2}Q scheduler presented by Bennett and Zhang [4] overcomes the inaccuracy problem of WFQ described in Section 2.3. In the WF^{2}Q system, when the next packet is chosen for service at time t, rather than selecting it from among all the packets at the server as in WFQ, the server only considers the set of packets that have started receiving service in the corresponding GPS system at time t.

As Bennett and Zhang showed in [4], for the example discussed in Section 2.3, the WF^{2}Q scheduler will service session 1 alternately as shown in Figure 1. As we realize in Figure 1, the output from the WF^{2}Q scheduler is rather smooth as opposed to the bursty output under the WFQ, SCFQ, and GrFQ schedulers. As proved in [4], the service provided by WF^{2}Q is almost identical to that of GPS. However its computational complexity is still high.

3 BCFQ: Burst Constraint Fair Queueing

Our motivation is to develop a scheduler that overcomes the burstiness problem while having a low computational complexity. The BCFQ scheduler operates by applying a packet selection criterion on a set of eligible packets. The selection criterion is described in Section 3.5 and the eligibility constraint in Section 3.3. Prior to that, Section 3.2 is used to introduce the system normalized service.

3.1 Session Normalized Service

The amount of service a session receives from the system divided by its weight specifies its normalized share of the system. We assign each session the normalized service it got until time t. A similar concept was used in [9],[17],[16],[18]. Let p_i^k denote the k^{th} packet of session i.

Definition 3.1 Assume session i is active in the interval $(0, t)$. The *normalized service* session i receives by time t, $h_i(t)$, is defined as: $h_i(t) = S_{i,P}(0,t)/w_i$.

In the sequel we present the normalized service of a session that was not continuously active. The normalized service of a session increases when the session is served. Thus if L_i^k is the length of packet p_i^k and τ_s and τ_e are its transmission start time and transmission end time, respectively, then:

$$h_i(\tau_e) = h_i(\tau_s) + L_i^k/w_i. \qquad (2)$$

Let t be a packet selection epoch. Let p_i^H denote the head packet of the queue of session i at time t and L_i^H denote the length of p_i^H. The *potential normalized service* of session i at time t, $h_i'(t)$, is defined as:

$$h_i'(t) = h_i(t) + L_i^H/w_i. \qquad (3)$$

3.2 System Normalized Service

Below we present the system normalized service that will serve as a low complexity substitute to the high complexity virtual time (explained in 2.2). The *accumulated system normalized service* for scheduler P during the interval $(0, t_l)$ in which the system is *continuously busy* is,

$$g_P(t_l) = \sum_{k=1}^{l} S_P(t_{k-1}, t_k)/W(t_k^-), \tag{4}$$

where t_k is the time of the k^{th} event under scheduler P, as defined in Definition 2.2, and (t_k^-) denotes the time instant just prior to time t_k. We can compute the accumulated normalized service also by,

$$g_P(t_{k-1} + \tau) = g_P(t_{k-1}) + S_P(t_{k-1}, t_{k-1} + \tau)/W(t_{k-1}, t_k), \tag{5}$$

for $0 \leq \tau \leq t_k - t_{k-1}$. In case that the server rate is 1 we get,

$$g_P(t_{k-1} + \tau) = g_P(t_{k-1}) + \tau/W(t_{k-1}, t_k). \tag{6}$$

As stated above (4), (5), and (6) apply for when the system is continuously busy. When the system becomes idle and then becomes busy again, all the system variables are set to zero and time is reinitiated, that is $t := 0$ and $g := 0$.

Remark 3.1 *Note the similarity in shape of $g_P(t)$ to the virtual time function of GPS, $v(t)$. In fact $v(t) = g_{GPS}(t)$. Nonetheless, for an arbitrary scheduler P, $g_P(t) \neq g_{GPS}(t)$, due to the differences between the events of P and GPS.*

The $g_P(t)$ function depends on the events in the system, since the system total weight changes when the active session set changes. Different schedulers lead to different event times, therefore $g_P(t)$ depends on the scheduler P. In a packet-based scheduler P, $g_P(t)$ must also be computed every event. However since in this system packets are served "one at a time", it cannot experience many departures during a single packet transmission. Therefore the complexity of computing $g_P(t)$ is $O(1)$ per packet transmission.

$g_P(t)$ will be used to approximate $g_{GPS}(t)$ (which is the ideal service that a session should be granted). BCFQ will use $g_{BCFQ}(t)$ as a substitute to $g_{GPS}(t)$ when low complexity is desired.

Observation 3.1 *If two non-idling schedulers P and Q have the same output link rate and the same events during $(0, t)$, then their accumulated normalized service is identical, namely $g_P(t) = g_Q(t)$.*

The proof is immediate.

Corollary 3.2 *Given scheduler P and a corresponding GPS scheduler, if the systems weights of P and GPS are constant during the interval $(0, t)$, then $\forall_{0<\tau<t}\ g_P(\tau) = v(\tau)$, where $v(\tau)$ is the GPS virtual function.*

Proof From Equation (6), one can easily verify that when the system weight remains constant in $(0, t)$ and is equal to W then $g_P(\tau) = \tau/W$. Similarly, from Equation (1), under the above conditions $v(\tau) = \tau/W$. Q.E.D.

Corollary 3.3 *If the active session set is constant during the interval* $(0, t)$*, then* $\forall_{0<\tau<t}$ $g_P(\tau) = v(\tau)$*.*

The proof is immediate from Corollary 3.2.

3.3 The Eligibility Constraint

The inaccuracy of WFQ shows up in situations where a session receives too much service in comparison to GPS. The eligibility constraint introduced in this section aims at restricting the amount of service given to a session. We will use the system accumulated normalized service ($g_P(t)$), in comparison to the session normalized service ($h_i(t)$) to carry out this restriction. Specifically, if $g_P(t)$ can be thought of as the amount of service one unit of weight should get by time t, then $w_i \cdot g_P(t)$ represents the amount of service session i should get by t. Therefore to limit the amount of service given to a session we have to maintain $S_{i,P}(0, t) \leq w_i \cdot g_P(t)$, or, alternatively, via Definition 3.1, $h_i(t) \leq g_P(t)$.

Definition 3.2 Session i is said to be *legal* or *eligible* at time t by BCFQ if it meets the eligibility constraint: $h_i(t) \leq g_P(t)$.

Note that in some rare cases under BCFQ all sessions may become ineligible, ($\forall_i$,$h_i(t) > g_{BCFQ}(t)$). A simple remedy , in this case, is to set $g_{BCFQ}(t) := min\{h_i(t)\}$. A detailed example where all the sessions are ineligible is presented in [1] (Appendix C there).

3.4 The Normalized Service of a Newly Active Session

In Section 3.1 we described how to update the session normalized service, $h_i(t)$, when session i is active. An open question is how to update these variable when session i becomes active after being inactive. A natural approach is to grant session i a value that will not discriminate it compared to other sessions in the system. A natural selection that creates little discrimination is to grant session i the value $h_i(t) = g_P(t)$. Further, to avoid a situation where a session being discriminated positively by leaving at t (with a high normalized service $h_i(t) > g_P(t)$) and returning a little while later, at $t + \varepsilon$ (and being granted $h_i(t + \varepsilon) := g_P(t + \varepsilon) < h_i(t)$), we correct it by assigning[2]:

$$h_i(t) = max\{h_i(t^-), g_P(t)\}. \tag{7}$$

3.5 BCFQ: Selecting the Next Session to be Transmitted

Upon packet departure the scheduler must decide on the next packet for transmission among the currently active sessions. BCFQ chooses among the eligible sessions as defined in Section 3.3 the one with the minimum potential normalized service, as defined in Section 3.1. The logic behind this decision is: 1. A session that got more service than it should will not be chosen, and 2. Between the legal sessions we will choose at t the one whose selection will create the least discrimination (when its service completes), namely the one with the minimal potential normalized service.

To demonstrate the effectiveness of BCFQ in reducing burstiness, consider again the bursty example presented in Section 2.3 (Figure 1) and apply BCFQ to it. At time 0, $h_i(0) = 0$, $i = 1, ..., 11$, $h'_1(0) = 1/10$, $h'_i(0) = 1$, $i = 2, ..., 11$, and $g(0) = 0$, thus all the sessions are legal. Among them session 1 has the smallest h'_i and it is selected to transmit. At time 1, $h_1(1) = 1/10$, $h_i(1) = 0$, $i = 2, ..., 11$, $h'_1(1) = 2/10$, $h'_i(1) = 1$,

[2] The notation "t^-" denotes the time instant just before time t.

$i = 2, ..., 11$ and $g(1) = 1/20$, thus, all sessions except for session 1 are legal. Since all the legal sessions have the same h'_i, a tie-breaking rule will select the session with the smallest index, session 2, for transmission. At time 2, session 1 becomes legal again and has the smallest h'_i among all active sessions, therefore it will be chosen to transmit. The reader may easily verify that this pattern of behavior continues and the resulting transmission pattern is a cyclic repetition of 1,2,1,3,...1,10,1,11, exactly identical to the transmission pattern of WF^{2}Q (Figure 1).

Thus, we realize that the output from the BCFQ system is rather smooth, as opposed to the bursty output of a WFQ scheduler.

3.6 An Efficient Implementation of BCFQ

3.6.1 The Priority Queue Data Structure

We maintain an efficient data structure for finding the minimum h'_i under the eligibility constraint. We utilize the data structure, presented by Stoica and Abdel-Wahab [20], that dynamically finds a minimum subject to a constraint. The data structure is based on a binary search tree and supports the following operations: Insertion, deletion, and finding the eligible session with the minimum h'_i. We call this data structure Minimum-under-Constraint Binary Tree (*MCBT*). Detailed explanation of *MCBT* and its operations is provided in [1] (Appendix D there).

As explained in [20] the complexity of each of the operations is $O(tree - heigh)$ per packet. Using one of the balanced-trees (red-black or AVL [20]) one gets an overall time complexity of $O(logN)$ for each elementary operation (where N is the number of active sessions).

3.6.2 The Computation of the $g(t)$ Function

The $g(t)$ function, appearing in Equation (6), must be computed at every event, i.e. when the active session set changes. In the BCFQ implementation we simplify the computation of $g_{BCFQ}(t)$ and compute this function upon completion of each packet transmission. Packets that arrive in the middle of transmission are treated as they arrive at the completion of that transmission. If the j^{th} packet transmission completion occurs at time t_j and the server rate is 1 then the computation of g_{BCFQ} is preformed as follows,

$$g_{BCFQ}(t_{j+1}) = g_{BCFQ}(t_j) + (t_{j+1} - t_j)/W(t_j). \tag{8}$$

3.6.3 The Implementation of BCFQ

We implement the BCFQ scheduler using two main functions: PacketArrived and PacketTransmit. In the PacketArrived function we insert the packet to its queue and if the packet arrives to an empty queue, we update the session parameters, update the system total weight, and insert it to the MCBT (see Section 3.6.1). In the PacketTransmit function we choose the session for transmission and then call FinishTransmit, which deletes the packet from the queue, updates $g_{BCFQ}(t)$ and the session parameters. When the session has no more packets, FinishTransmit deletes it from the MCBT (see Section 3.6.1) and updates the system total weight. The algorithm pseudo code is presented in [1] (Appendix E there).

3.7 The Complexity of BCFQ

Theorem 3.4 *The per packet complexity of BCFQ when using $g_{BCFQ}(t)$, is O(logN) where N is the number of sessions.*

Proof For each packet we perform the PacketArrived procedure and the PacketTransmit procedure (see Section 3.6.3). The computation of $g_{BCFQ}(t)$ takes $O(1)$ per packet transmission as explained in Section 3.6.2. The main complexity results from the *MCBT* operations (see Section 3.6.1): GetLegalMin, Delete and Insert, where each takes $O(logN)$ operations per packet as demonstrated in Section 3.6.1. Q.E.D.

4 Fairness Analysis of BCFQ

The *Relative Fairness Bound (RFB)* has been used widely in the FQ literature [21],[9],[16]. It is based on the maximum difference between the normalized service received by any two active sessions. Since it bounds the gap between sessions, it serves as a fairness measure. In this section we show that BCFQ has a relatively low RFB. The following definition of the relative fairness bound (RFB) is equivalent to the definition in [21],[16].

Definition 4.1 Let (t_1, t_2) be an interval of time during which all sessions under consideration are all active. Given a scheduler P, for a pair of sessions i and j, that are continuously active during interval (t_1, t_2), the $RF_{(i,j)}(t_1, t_2)$ measure is defined as,

$$RF_{(i,j)}(t_1, t_2) = \left| \frac{S_{i,P}(t_1, t_2)}{w_i} - \frac{S_{j,P}(t_1, t_2)}{w_j} \right|. \tag{9}$$

The *relative fairness bound*, *RFB*, can now be defined as,

$$RFB = max_{\forall(i,j)\forall(t_1,t_2)} RF_{(i,j)}(t_1, t_2). \tag{10}$$

Lemma 4.1 *Under BCFQ any pair of sessions i and j that are continuously active in the interval $(0, t)$ obey the following,*

$$RF_{(i,j)}(0, t) = \left| \frac{S_i(0, t)}{w_i} - \frac{S_j(0, t)}{w_j} \right| \leq \max\{\frac{L_{i,max}}{w_i}, \frac{L_{j,max}}{w_j}\}. \tag{11}$$

The proof is given in Appendix B.

Corollary 4.2 *Under BCFQ, any pair of sessions i and j that are continuously active in the interval $(0, t_2)$, obey the following,*

$$RF_{(i,j)}(t_1, t_2) = \left| \frac{S_i(t_1, t_2)}{w_i} - \frac{S_j(t_1, t_2)}{w_j} \right| \leq 2\max\{\frac{L_{i,max}}{w_i}, \frac{L_{j,max}}{w_j}\}. \tag{12}$$

Proof Substitute $t = t_2$ and $t = t_1$ to bound $RF_{(i,j)}(0, t_2)$ and $RF_{(i,j)}(0, t_1)$ respectively. Use these two bounds combined with $S_i(t_1, t_2) = S_i(0, t_2) - S_i(0, t_1)$ to get (12). Q.E.D.

Theorem 4.3 *The Relative Fairness Bound of BCFQ is bounded as follows, $RFB \leq 2\frac{L_{max}}{w_{min}}$.*

Proof Follows directly from the definition of RFB in Section 4.1 and Corollary 4.2. Q.E.D.

We thus conclude that the RFB of BCFQ is identical to that of some other schedulers such as SCFQ [9] and GrFQ [16].

5 A Burstiness Criterion

In Section 2.3, we demonstrated that schedulers which are proved to have a low RFB are still subject to the burstiness problem. Thus a simple criterion for examining scheduler burstiness is needed. In this section we propose a new criterion called Relative Burstiness (RB), that does not depend on the GPS system. We further prove that the RB criterion is equivalent to the set of criteria provided for WF^{2}Q [4] (Equations (10),(11), and (12) there) and that a low RFB is not sufficient criterion for measuring burstiness. Let i^c denote the complement of i, namely the set of all active sessions except i, and let $S_{i^c,P}(0,t)$ denote the total amount of service received by the set of sessions i^c by time t under scheduler P, and let $W_{i^c} = W - w_i$.

Definition 5.1 Let P be a scheduling policy, then the *relative burstiness* with respect to session i over the interval (t_1, t_2), denoted by $RB_i(t_1, t_2)$, is defined:

$$RB_i(t_1, t_2) = \left| \frac{S_{i,P}(t_1, t_2)}{w_i} - \frac{S_{i^c,P}(t_1, t_2)}{W_{i^c}} \right|. \tag{13}$$

Definition 5.2 Let $(0, t)$ be an interval during which the active session set is constant. A scheduler P is said to be *non-bursty* if $\forall_{t \geq 0}$ the following holds:

$$\forall i RB_i(0, t) \leq \frac{L_{i,max}}{w_i}. \tag{14}$$

Corollary 5.1 *Let $(0, t)$ be an interval during which the active session set is constant and (t_1, t_2) be an interval such that $0 \leq t_1 < t_2 \leq t$. If P is a* non-bursty *scheduler then the following holds,*

$$\forall i RB_i(t_1, t_2) \leq 2\frac{L_{i,max}}{w_i}. \tag{15}$$

The proof is similar to that of Corollary 4.2.

Remark 5.1 *Note that the use of constant active session set is common in the literature. See, e.g., the RFB criterion defined in [21],[16].*

5.1 The Burstiness Criterion and the Proximity of Schedulers to GPS

In this section we will show an equivalence between a scheduler being non-bursty (according to Definition 5.2) and its proximity to GPS (Theorem 5.5).

According to Bennett and Zhang [4], given a WF^{2}Q system and the corresponding *GPS* system, the following properties hold for any i and t:

$$S_{i,GPS}(0, t) - S_{i,WF^2Q}(0, t) \leq L_{max}, \tag{16}$$

$$S_{i,WF^2Q}(0, t) - S_{i,GPS}(0, t) \leq (1 - \frac{w_i}{W})L_{i,max}. \tag{17}$$

The first property (16) holds for both WFQ [12] and WF^{2}Q [4], while the second (17) holds only for WF^{2}Q. As presented in [4], since the service provided by WF^{2}Q can be neither too far behind (Equation (16)), nor too far ahead (Equation (17)), when compared to the corresponding GPS system, it must be that WF^{2}Q provides service which is almost identical to that of GPS. This is formally defined next.

Definition 5.3 Given a scheduler P and a corresponding GPS scheduler, P is said to be *proximate* to GPS if the following holds for all i and t:

$$S_{i,GPS}(0,t) - S_{i,P}(0,t) \leq L_{max}, \tag{18}$$

$$S_{i,P}(0,t) - S_{i,GPS}(0,t) \leq (1 - \frac{w_i}{W})L_{i,max}. \tag{19}$$

Scheduler P is *tightly proximate* to GPS if we replace Equation (18) with,

$$S_{i,GPS}(0,t) - S_{i,P}(0,t) \leq (1 - \frac{w_i}{W})L_{i,max}. \tag{20}$$

Let $d^k_{i,GPS}$ and $d^k_{i,WFQ}$ denote the times that packet p^k_i departs under GPS and WFQ respectively. In the following claim we provide a tight relation between $d^k_{i,WFQ}$ and $d^k_{i,GPS}$ when the active session set is constant. Lemma 5.3 is assisted by this claim.

Claim 5.2 *If the active session set is constant, then for all k and i,*

$$d^k_{i,GPS} \geq d^k_{i,WFQ}. \tag{21}$$

Proof Suppose that the server finishes transmission at time τ under WFQ and must select the next packet to be transmitted. Consider a session i that belongs to the constant active session set. According to [12] (in the analysis prior the Theorem 1 there), the only packets that are delayed more in WFQ than in GPS are those that arrive too late to be transmitted in their GPS order. Since we assume that the active session set is constant, at every packet selection epoch τ there must exist at least one packet in the queue for session i. Thus the earliest-to-finish packet of session i (under GPS) must be in the queue. Therefore there exist no packet that arrive too late to be transmitted under the GPS order and the proof follows. Q.E.D.

The following lemma establishes a bound for WFQ/WF^{2}Q which is tighter than (16) when the active session set is constant. Theorem 5.4 is assisted by this lemma.

Lemma 5.3 *If the active session set is constant during the interval* $(0,t)$*, then under WFQ (and* WF^{2}Q*):*

$$S_{i,GPS}(0,t) - S_{i,WFQ}(0,t) \leq (1 - \frac{w_i}{W})L_{i,max}. \tag{22}$$

Proof We first prove the lemma for WFQ and then for WF^{2}Q. We follow the proof of Equation (16) for WFQ as presented in [12], but add the fact that the active session set is constant. Let $b^k_{i,WFQ}$ be the time that p^k_i begins transmission under WFQ, and let L^k_i be the length of p^k_i. Then p^k_i completes transmission at $b^k_{i,WFQ} + L^k_i/r$. Since session i packets are served in the same order under both GPS and WFQ,

$$S_{i,GPS}(0, d^k_{i,GPS}) = S_{i,WFQ}(0, b^k_{i,WFQ} + L^k_i/r). \tag{23}$$

From Claim 5.2, $d^k_{i,GPS} \geq b^k_{i,WFQ} + L^k_i/r$. Therefore, $S_{i,GPS}(0, d^k_{i,GPS}) \geq S_{i,GPS}(0, b^k_{i,WFQ} + L^k_i/r)$. From this equation and (23) we get,

$$S_{i,WFQ}(0, b^k_{i,WFQ} + L^k_i/r) \geq S_{i,GPS}(0, b^k_{i,WFQ} + L^k_i/r). \tag{24}$$

The processing rate given to session i in GPS is $r\frac{w_i}{W}$, therefore,

$$S_{i,GPS}(0, b^k_{i,WFQ} + L^k_i/r) = S_{i,GPS}(0, b^k_{i,WFQ}) + L^k_i \frac{w_i}{W}. \tag{25}$$

The processing rate of WFQ is r, therefore, $S_{i,WFQ}(0,b^k_{i,WFQ}+L^k_i/r) = S_{i,WFQ}(0,b^k_{i,WFQ})+L^k_i$. Substituting this equation and (25) in (24) we get,

$$S_{i,GPS}(0,b^k_{i,WFQ}) - S_{i,WFQ}(0,b^k_{i,WFQ}) \leq (1-\frac{w_i}{W})L_{i,max}. \tag{26}$$

The proof for arbitrary t under WFQ follows from the fact that the value of $S_{i,GPS}(0,t) - S_{i,WFQ}(0,t)$ reaches its maximal value when session i packets begin transmission under WFQ. To complete the proof for WF^{2}Q one can use the same proof as in [4] (the proof to Theorem 1 there) and Equation (26). Q.E.D.

Theorem 5.4 *If the active session set is constant during interval* $(0,t)$*, then* WF^{2}Q *is tightly proximate to GPS (according to Definition 5.3).*

Proof Equations (16) and (17), and Lemma 5.3 yield the proof. Q.E.D.

According to [4], WF^{2}Q possesses properties (16) and (17) and therefore it is non-bursty and accurate. Next (Theorem (5.5)) we will prove the relation between non-burstiness of a scheduler (according to Definition 5.2) and tight proximity to GPS (according to Definition 5.3).

Theorem 5.5 *Under constant active session set, scheduler P is non-bursty (Definition 5.2), iff it is tightly proximate to GPS (Definition 5.3).*

For conciseness of presentation the proof is given in Appendix C.

5.2 The Relation between Relative Fairness and Relative Burstiness

The following theorem establishes a bound on the burstiness of a scheduler as a function of its fairness. Unfortunately, however, as shown in Remark 5.2 tight fairness does not necessarily lead to tight burstiness. Let $X(i,j)$ be an arbitrary variable possibly depending on i and j (session indices).

Theorem 5.6 *If scheduler P obeys the following relative fairness (RF) criterion,*

$$\left|\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{j,P}(0,t)}{w_j}\right| \leq X(i,j), \forall_{i,j}, \tag{27}$$

then it obeys the following relative burstiness (RB) criterion,

$$\left|\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}}\right| \leq max_{\{k,j\}\in A(t)}\{X(k,j)\}, \forall_i. \tag{28}$$

Proof Summing Equation (27) for all $j \neq i$ we get,

$\sum_{j\neq i}\left|\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{j,P}(0,t)}{w_j}\right| \leq \sum_{j\neq i} X(i,j),$

or

$\sum_{j\neq i}|w_j S_{i,P}(0,t) - w_i S_{j,P}(0,t)| \leq \sum_{j\neq i} w_i w_j X(i,j),$

or

$|S_{i,P}(0,t)\sum_{j\neq i} w_j - w_i \sum_{j\neq i} S_{j,P}(0,t)| \leq w_i \sum_{j\neq i} w_j X(i,j),$

or

$|S_{i,P}(0,t)W_{i^c} - w_i S_{i^c,P}(0,t)| \leq w_i \sum_{j\neq i} w_j X(i,j),$

or

$\left|\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}}\right| \leq \frac{1}{W_{i^c}} \sum_{j\neq i} w_j X(i,j).$

Now since $X(i,j) \leq max_{\{k,j\} \in A(t)}\{X(k,j)\}$,
$\frac{1}{W_{i^c}} \sum_{j \neq i} w_j X(i,j) \leq max_{\{k,j\} \in A(t)}\{X(k,j)\} \frac{1}{W_{i^c}} \sum_{j \neq i} w_j = max_{\{k,j\} \in A(t)}\{X(k,j)\}$, and then we get,
$\left| \frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}} \right| \leq max_{\{k,j\} \in A(t)}\{X(k,j)\}$.

Q.E.D.

We thus may conclude that any scheduler possessing the relative fairness criterion, possesses an upper bound on the relative burstiness criterion.

Corollary 5.7 *If scheduler P obeys the following relative fairness criterion,*

$$\left| \frac{S_{i,P}(0,t)}{w_i} - \frac{S_{j,P}(0,t)}{w_j} \right| \leq \max\{\frac{L_{i,max}}{w_i}, \frac{L_{j,max}}{w_j}\}, \forall_{i,j}, \tag{29}$$

then it obeys the following relative burstiness criterion,

$$\left| \frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}} \right| \leq max_{l \in \{A(t)\}}\{\frac{L_{l,max}}{w_l}\}, \forall_i. \tag{30}$$

Proof Using Theorem 5.6 we get,

$$\left| \frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}} \right| \leq max_{\{k,j\} \in A(t)}\{\max\{\frac{L_{k,max}}{w_k}, \frac{L_{j,max}}{w_j}\}\}, \forall_i \tag{31}$$

which leads to the proof. Q.E.D.

Note that one session with large packets and low weight can cause the bound to be high and therefore non-tight.

Remark 5.2 *Note that a scheduler with a low relative fairness (RF) may still have a high relative burstinees (RB), as (30) may be significantly higher than (29). Such a scheduler may have bursty behavior since its relative burstiness bound is higher than it should be as defined in Definition 5.2. For example the GrFQ scheduler has a relative fairness bound as in (29), as proved in [16], while its relative burstiness bound is much higher (30), as can be easily seen from the example presented in Section 2.3. The same can be shown also for the SCFQ scheduler [9].*

6 Non-Burstiness of BCFQ and Its Equivalence to WF^{2}Q

WF^{2}Q is an accurate and non-bursty scheduler whose service approximates GPS very closely. In this section we analyze BCFQ and determine conditions under which its schedule is identical to that of WF^{2}Q. This will provide an evidence (see Section 6.2) to the non-burstiness of BCFQ (in the form of admitting the RB criterion). Specifically, we will show the equivalence (in the sense of yielding exactly identical schedules) of BCFQ to WF^{2}Q in the following settings: First (Subsection 6.1) we show that if the active session set is constant during the interval $(0,t)$ then BCFQ is equivalent to WF^{2}Q during that interval. Second (Subsection 6.3) we discuss additional property of BCFQ according to which if BCFQ uses the GPS virtual time function as its eligibility constraint (and one more minor change is introduced) then the behavior of BCFQ becomes exactly identical to that of WF^{2}Q, that is, it achieve very low burstiness if very small. This property holds for arbitrary arrival patterns (that is, applying to non-constant active session sets well).

Let $b^k_{i,BCFQ}$ ($b^k_{i,GPS}$) and $d^k_{i,BCFQ}$ ($d^k_{i,GPS}$) denote the times at which p^k_i starts transmission and ends transmission, respectively, in BCFQ (GPS). Let a^k_i denote the arrival time of p^k_i.

Definition 6.1 Two packet-based schedulers P and Q are *equivalent* if $\forall_{i,k}\ b^k_{i,P} = b^k_{i,Q}$, namely all the packets' transmission start times are identical.

6.1 Equivalence of BCFQ and WF^{2}Q: Constant Active Session Set

Below, in Theorem 6.3 we will show that if the active session set is constant in the interval $(0,t)$ then BCFQ is equivalent to WF^{2}Q and thus is as close to GPS as WF^{2}Q. Recall from Corollary 3.3, that under these condition $g_{BCFQ}(\tau) = v(\tau)$ for every $0 < \tau < t$. Since the active session set is constant in $(0,t)$, let W denote the sum of weights of the active sessions.

Lemma 6.1 *If the active session set is constant in* $(0,t)$*, then the eligibility constraint of BCFQ,* $h_i(\tau) \leq g_{BCFQ}(\tau)$*, holds iff* $S_{i,BCFQ}(0,\tau) \leq S_{i,GPS}(0,\tau)$*.*

Proof When the active session set is constant in $(0,t)$ session i remains active in $(0,t)$ and thus from Definition 3.1 we get for $0 < \tau < t$:

$$h_i(\tau) = \frac{S_{i,BCFQ}(0,\tau)}{w_i}. \tag{32}$$

Since the active session set is constant in $(0,t)$ we have:

$$S_{i,GPS}(0,\tau) = w_i \cdot \frac{S_{GPS}(0,\tau)}{W}. \tag{33}$$

The service rate of BCFQ and GPS is equal, thus, the service for all sessions obeys: $S_{BCFQ}(0,\tau) = S_{GPS}(0,\tau)$. Further, under these condition Equation (4) can be rewritten as, $g_{BCFQ}(\tau) = \frac{S_{BCFQ}(0,\tau)}{W}$. Plugging the last two equalities and (33) in each other yields $g_{BCFQ}(\tau) = \frac{S_{i,GPS}(0,\tau)}{w_i}$, which together with (32) yields the proof. Q.E.D.

Lemma 6.1 will be used in the sequel (Theorem 6.3) to show that the sets of eligible sessions of WF^{2}Q and BCFQ are identical to each other.

Lemma 6.2 *If the active session set is constant in* $(0,t)$*, then under BCFQ the potential normalized service of session* i *at epoch* $d^k_{i,BCFQ}$*,* $h'_i(d^k_{i,BCFQ})$*, is identical to the GPS finish virtual time of* p^{k+1}_i*,* $F^{k+1}_{i,GPS}$*.*

Proof According to Equation (1), when the active session set is constant, $v(\tau) = \frac{S_{GPS}(0,\tau)}{W}$. Plugging (33) to this equality we get:

$$v(t) = \frac{S_{i,GPS}(0,t)}{w_i}. \tag{34}$$

Since session i's packets are served in the same order under both schemes, $S_{i,GPS}(0, d^k_{i,GPS}) = S_{i,BCFQ}(0, d^k_{i,BCFQ})$. Therefore, from this equality and (34) we get: $v(d^k_{i,GPS}) = S_{i,GPS}(0, d^k_{i,GPS})/w_i = S_{i,BCFQ}(0, d^k_{i,BCFQ})/w_i = h_i(d^k_{i,BCFQ})$. Namely, the virtual time at $d^k_{i,GPS}$ is equal to the normalized service of session i at $d^k_{i,BCFQ}$. Adding L^{k+1}_i/w_i to both sides and using $v(d^k_{i,GPS}) + L^{k+1}_i/w_i = F^{k+1}_{i,GPS}$ (implied from (34) and from the definition of GPS) we finally get: $h'_i(d^k_{i,BCFQ}) = F^{k+1}_{i,GPS}$. Q.E.D.

Lemma 6.1 (eligibility constraint) and Lemma 6.2 (selection criterion) are next used to show the equivalence of WF^{2}Q to BCFQ.

Theorem 6.3 *If the active session set is constant in* $(0, t)$*, then BCFQ and* WF^{2}Q *are equivalent.*

Proof Let $t_{j,P}$ be the time of the j^{th} departure under scheduler P. We will prove by induction on the departure times under BCFQ that the behavior of the two schedulers is identical. Assuming that the two schemes are equal during interval $(0, t_{j,BCFQ})$, we will prove that they are equal also during interval $(0, t_{j+1,BCFQ})$. According to the assumption, $t_{j,BCFQ} = t_{j,WF^2Q}$, therefore at $t_{j,BCFQ}$ both BCFQ and WF^{2}Q have to choose the next packet for transmission.

BCFQ selects the session with the minimum potential normalized service $(h'_i(t_{j,BCFQ}))$ among the set of sessions obeying $h_i(t_{j,BCFQ}) \leq g_{BCFQ}(t_{j,BCFQ})$. WF^{2}Q will select at t_{j,WF^2Q} the session with the minimum finish virtual time (F_i^k) among the sessions whose packets already started service under GPS (that is, the service they got by t_{j,WF^2Q} under WF^{2}Q is smaller than or equal to what they got by GPS). Lemma 6.1 implies that the set of eligible sessions under BCFQ and the set of sessions that already start service under GPS are the same. And from Lemma 6.2 we can conclude that selecting the session with the minimum potential normalized service and selecting the session with the minimum finish virtual time is equivalent. Therefore, BCFQ and WF^{2}Q will select at $t_{j,BCFQ}$ the same packet for transmission. Thus, the two schemes are also equal during interval $(0, t_{j+1,BCFQ})$. Since the above holds also for $t = 0$, the equivalence holds for any time t. Q.E.D.

Lastly, the next theorem establishes the accuracy of BCFQ.

Theorem 6.4 *If the active session set is constant in* $(0, t)$*, then the following relations hold for any* i*,* k*,* $0 < \tau < t$*:*

$$S_{i,GPS}(0, \tau) - S_{i,BCFQ}(0, \tau) \leq (1 - \frac{r_i}{r})L_{i,max}, \tag{35}$$

$$S_{i,BCFQ}(0, \tau) - S_{i,GPS}(0, \tau) \leq (1 - \frac{r_i}{r})L_{i,max}, \tag{36}$$

where $r_i := r\frac{w_i}{W}$ *and* r *is the server rate.*

Proof The equations, where WF^{2}Q replaces BCFQ were proved in [4] (Equations (11), and (12) there). This, combined with Lemma 5.3 and then Theorem 6.3, leads to the proof. Q.E.D.

We thus conclude, that if the active session set is constant in $(0, t)$, then the service provided by BCFQ during this interval is equivalent to that of WF^{2}Q and similar to that of GPS, differing by no more than one maximal size packet.

6.2 The Non-Burstiness of BCFQ: Constant Active Session Set

The next theorem establishes that BCFQ is non-bursty when the active session set is constant.

Theorem 6.5 *BCFQ is a non-bursty scheduler (Definition 5.2).*

Proof The theorem is implied directly from Theorem 6.4 and Theorem 5.5. Q.E.D.

6.3 Using the GPS virtual time in BCFQ: Equivalence to WF^{2}Q

To further demonstrate the accuracy of BCFQ, we show in [1] (Section 6.3 there) that if BCFQ uses the GPS virtual time for its eligibility constraint instead of g_{BCFQ} (and if one more minor modification is incorporated), then its operation is exactly identical to that of WF^{2}Q. It should be noted that this result holds for general and arbitrary set of sessions and that the active session set needs not be constant. Thus, with these modifications it becomes as accurate as WF^{2}Q. Note also that the incorporation of these changes in BCFQ increases its *complexity* to be identical to that of WF^{2}Q. Thus, this modified version in its current form may be considered to be neither inferior nor superior to WF^{2}Q. The importance of the result is therefore in two aspects:

1. Demonstrating the theoretical accuracy of the BCFQ approach.

2. The use of the BCFQ principles, that is using the normalized service times for prioritizing the flows while the virtual time serves as a constraint, might lead to the development of better algorithms based on the normalized service.

7 Simulation Results

In Section 6 we showed that when the active session set is constant BCFQ is not bursty. In this section, we use simulation experiments to examine the burstiness of BCFQ in a general environment (that is, when the active session set is not necessarily constant) and demonstrate that the burstiness of BCFQ is very low[3].

We conduct a simulation of BCFQ and compare it to that of WFQ under exactly the same arrival patterns. We evaluate the burstiness of BCFQ by measuring its positive deviation from GPS ($S_{i,BCFQ}(0,t) - S_{i,GPS}(0,t)$) and computing the full statistics of this deviation. Similarly we compute the deviation of WFQ ($S_{i,WFQ}(0,t) - S_{i,GPS}(0,t)$).

We examine five cases representing a wide range of parameters, differing from each other in their sessions' relative rate and weight and in the number of bursty sessions. Cases A, B, and C consist of several heavy traffic and bursty sessions and many light traffic sessions. They differ from each other in the burstiness and weights given to the high traffic sessions. Cases D and E examine situations where the average rate of all the sessions is identical; in Case D five of the sessions are highly prioritized (weight 5) while in Case E all sessions are of the same weight. Note that we do not examine cases with sessions that have high rate and low weight since these cases are not stable.

The details of the experiments are as follows: The output link rate is 100 packets/second and the total number of packets (throughout the simulation) is approximately 500,000. Case A consists of 110 sessions, 10 of which are high rate sessions, each transmitting at average rate of 4.16 packets/sec, and the other 100 are low rate sessions, each transmitting at average rate of 0.434 packets/sec. The traffic of each session is bursty, and follows an on-off model where the on-period and the off-period are uniformly distributed between 48 to 96 seconds (for the high rate sessions) or between 460 to 920 seconds (for the low rate sessions). The overall utilization of the output link is approximately 0.85. The high rate sessions receive a high weight, $w_i = 10$ while the low rate sessions receive low weight, $w_i = 1$. Cases B, C, D, and E, are similar in their description but significantly differ from Case A in their parameters. The parameters of all five cases are depicted in Table 1.

[3] The reader may recall that when BCFQ is modified via the modification analyzed in Section 6.3 it is equivalent to WF^{2}Q and therefore it is not bursty.

Case (utilization)	Number of sessions		Average session rate (pkt/sec)		Duration of on (off) period uniform at range (sec):		Weights	
	High	Low	High	Low	High	Low	High	Low
A (0.85)	10	100	4.16	0.434	48-96	460-920	10	1
B (0.85)	5	100	8.33	0.434	24-48	460-920	20	1
C (0.88)	1	100	41.666	0.462	4.8-9.6	432-864	30	1
D (0.8)	5	100	0.769	0.769	260-520	260-520	5	1
E (0.86)	-	100	-	0.862	-	232-264	-	1

Table 1: Detailed parameters of simulation cases

The simulation measures the deviation $S_{i,BCFQ}(0,t) - S_{i,GPS}(0,t)$ and $S_{i,WFQ}(0,t) - S_{i,GPS}(0,t)$ for all epochs t which are packet termination epochs under GPS. The deviation is measured in units of packets. We combine this data into a histogram reflecting the percentage of epochs (packets) which are subject to certain deviation. Figures 2(a) and 2(b) depict the results for Case A and Case B, respectively [4]. The figures demonstrate that under BCFQ only a very small fraction of the packets are subject to large deviation (1%-2% of the packets experience deviation of more than 1 time unit). In contrast, WFQ experiences much larger deviations (20% of the packets or more experience deviation of more than 1 time unit, some experiencing deviation of up to 13 time units). Note that Case B is subject to significantly more bursty inputs than Case A, namely the heavy sessions are with higher rate and higher weight. For this reason its output, for WFQ, is also more bursty.

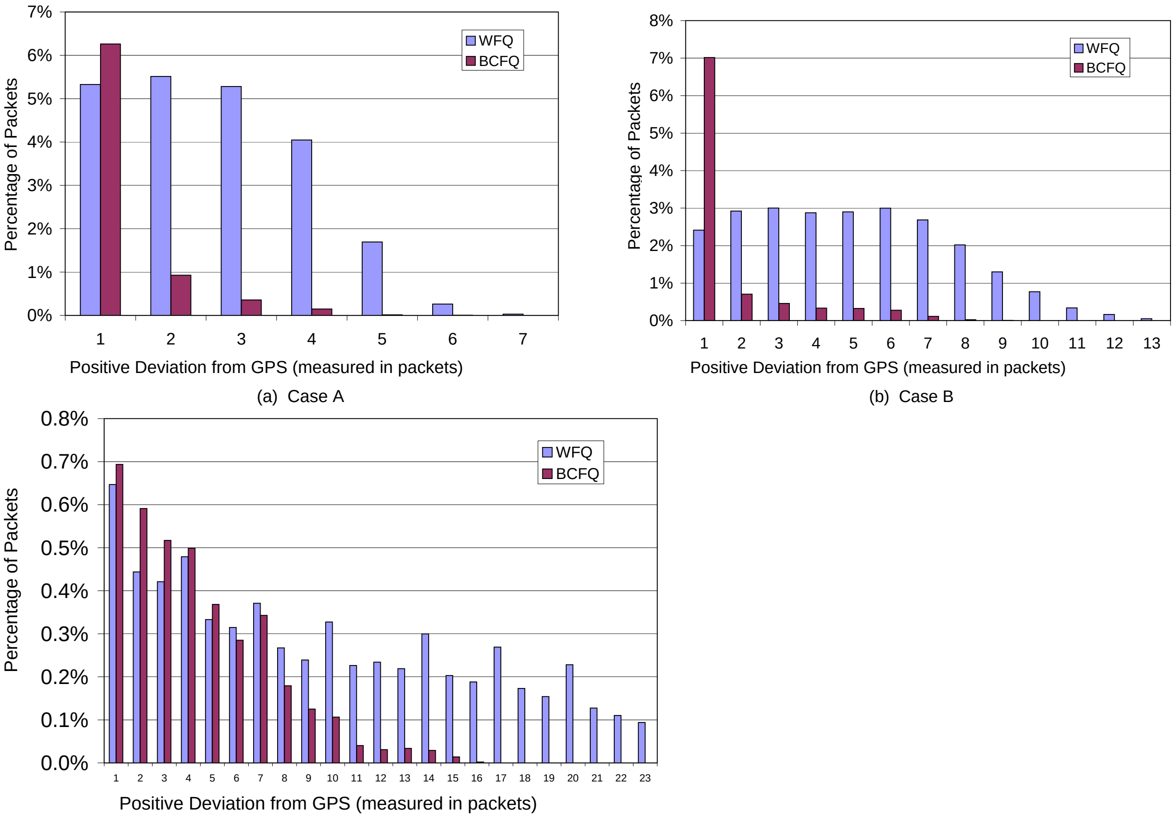
Figure 2(c) depicts the results for Case C. Due to the very high burstiness of the heavy session, the output of both WFQ and BCFQ becomes more bursty (compared to A and B). The figure demonstrates that the performance of BCFQ is significantly better than that of WFQ: Under BCFQ only a very small fraction of the packets are subject to large deviation (approximately 0.1% of the packets experience deviation of more than 10 time unit). In contrast, WFQ experiences much larger deviations (approximately 3% of the packets experience deviation of more than 10 time unit, some experiencing deviation of up to 22 time units).

For Case D WFQ and BCFQ experience (figure is not provided) similar results: For both of them, no packet experiences deviation of more than 1 packet. This stems from the fact that the average rates of all sessions are equal for each other, and the sessions differ only in their weights. Thus, the input rate of the high priority sessions is not very bursty, and the behavior of both scheduling policies is good. For Case E (figure is not provided) the results are similar to that of Case D. This stems from the fact that in Case E all the sessions are identical in their rate and their weight and therefore there are no bursty sessions.

8 Concluding Remarks

We studied the issue of bursty transmission in packet scheduling algorithms. We proposed the Burst-Constrain Fair-Queueing (BCFQ) algorithm, a packet scheduler that achieves both high fairness and low burstiness while maintaining low computational complexity. We also proposed a new measure (and criterion) which can be used for evaluating the

[4] Note that the packets that are subject to 0 deviation are not provided in the figure and thus the numbers do not add up to 100%



(a) Case A

(b) Case B

(c) Case C

Figure 2: Histograms of positive deviation from GPS

burstiness of arbitrary scheduling algorithms. We used this measure and demonstrated that BCFQ possesses the desired properties of fairness and burstiness while maintaining low computational complexity.

References

[1] Ashkenazi, L. Controlling burstiness in fair queueing scheduling, March 2004. Master thesis. To appear. URL: http://www.cs.tau.ac.il/∼liatashk.

[2] Bennett, J. C. R., Stephens, D. C., and Zhang, H. High speed, scalable, and accurate implementation of fair queueing algorithms in atm networks. In *Proceedings of ICNP '97* (October 1997), pp. 7–14.

[3] Bennett, J. C. R., and Zhang, H. Hierarchical packet fair queueing algorithms. In *Proceedings of ACM SIGCOMM'96,Palo Alto, CA* (August 1996), pp. 143–156.

[4] Bennett, J. C. R., and Zhang, H. WF^{2}Q: worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM* (March 1996), pp. 120–128.

[5] Bennett, J. C. R., and Zhang, H. Why wfq is not good enough for integrated services networks. In *Proceedings of NOSSDAV '96, Zushi, Japan* (April 1996), pp. 524–532.

[6] Chiussi, F. M., and Francini, A. Implementing fair queueing in atm switches part 1: A practical methodology for the analysis of delay bounds. In *Proceedings of IEEE Globecom'97* (November 1997).

[7] Demers, A., Keshav, S., and Shenker, S. Analysis and simulation of a fair queueing algorithm. *Internetworking Research and Experience* (October 1990), 3–26. Also in Proceedings of ACM SIGCOMM'89, pp. 3-12.

[8] Golestani, S. J. Network delay analysis of a class of fair queueing algorithms. *IEEE Selected Areas in Communications 13*, 6 (August 1995), 1057–1070.

[9] Golestani, S. J. A self-clocked fair queueing scheme for broadband application. In *Proceedings of IEEE INFOCOM, Toronto, Canada* (June 1996), pp. 636–646.

[10] Goyal, P., Vin, H. M., and Cheng, H. Start-time fair queueing: a scheduling algorithm for integrated service packet switching networks. *IEEE Trans. Networking 5*, 5 (October 1997), 690–704.

[11] Kleinrock, L. *Queueing Systems, Volume 2: Computer Applications.* Wiley, 1976.

[12] Parekh, A. K., and Gallager, R. G. A generalized processor sharing approach to flow control in integrated services networks:the single-node case. *IEEE/ACM Trans. Networking 1*, 1 (June 1993), 344–357.

[13] Parekh, A. K., and Gallager, R. G. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Trans. Networking 2* (April 1994), 137–150.

[14] Parekh, A. K. J. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks.* PhD thesis, Massachusetts Institute of Technology, February 1992.

[15] Rexford, J., Greenberg, A., and Bonomi, F. Hardware-efficient fair queueing architectures for high-speed networks. In *Proceedings of IEEE INFOCOM* (March 1996).

[16] Shi, H., and Sethu, H. Greedy fair queueing: A goal-oriented strategy for fair real-time packet scheduling. In *Proceedings of the Real-Time Systems Symposium (RTSS)* (December 2003).

[17] Stiliadis, D., and Varma, A. Efficient fair queueing algorithms for packet-switched networks. *IEEE/ACM Tansactions on Networking 6*, 2 (April 1998), 175–185.

[18] Stiliadis, D., and Varma, A. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE Tansactions on Networking 6*, 5 (October 1998), 611–624.

[19] Stiliadis, D., and Varma, A. Rate-proportional servers: A design methodology for fair queueing algorithms. *IEEE/ACM Tansactions on Networking 6*, 2 (April 1998), 164–174.

[20] Stoica, I., and Abdel-Wahab, H. Earliest eligible virtual deadline first: A flexible and accurate mechanism for proportional share resource allocation. Tech. Rep. 9522, Old Dominion University, November 1995.

[21] Zhou, Y., and Sethu, H. On the relationship between absolute and relative fairness bounds. *IEEE Comm. Letters 6*, 1 (January 2002), 37–39.

A Glossary of Notation

p_i^k The k^{th} packet of session i

L_i^k The length of packet p_i^k

$S_{i,P}(0,t)$ The total amount of service received by session i by time t under scheduler P

w_i The weight of session i

$W(t)$ The sum of weights of the active session set at time t

$v(t)$ The GPS virtual time at time t

$h_i(t)$ The normalized service of session i at time t

$h'_i(t)$ The potential normalized service of session i at time t

$g_P(t)$ The accumulated system normalized service for scheduler P at time t

$A(t)$ The active session set at time t

B The Proof of Lemma 4.1

$RF_{(i,j)}(0,t)$ must get its maximum at an epoch τ_e when either i or j completes transmission. Let τ_s be the transmission start epoch prior to τ_e. The proof will be conducted via induction on τ_s and τ_e. To this end let τ'_e be the latest epoch prior to τ_s at which either i or j completes transmission. Obviously $\tau'_e \leq \tau_s$. Obviously neither i nor j are served in (τ'_e, τ_s) and thus if Equation (11) holds for $t = \tau'_e$ it must hold for $t = \tau_s$. What remains to show is that if Equation (11) holds for $t = \tau_s$ it must hold for $t = \tau_e$, which we show next. Without loss of generality assume that session i gets service during interval (τ_s, τ_e). At τ_s session i is chosen to transmit, therefore there are two exclusive possibilities: A. Session i is legal, session j is either legal or illegal, and $h'_i(\tau_s) \leq h'_j(\tau_s)$. B. Session i is legal, session j is illegal, and $h'_j(\tau_s) \leq h'_i(\tau_s)$. Note that either A or B must hold, since if all the active sessions are illegal at τ_s, we fix the $g(\tau_s)$ value that at least one active session becomes legal, as explained in Section 3.3. Since i transmits at (τ_s, τ_e) and j does not, we have $h_j(\tau_s) = h_j(\tau_e)$ and $h'_i(\tau_s) = h_i(\tau_e)$. We will use also the fact that the potential normalized service is always larger than the current normalized service, $\forall_l\ h'_l(t) > h_l(t)$.

Case A: $h'_i(\tau_s) \leq h'_j(\tau_s)$ and $h_i(\tau_s) \leq g(\tau_s)$(session i is legal). Divide this case to three sub cases: A1. $h_j(\tau_s) \geq h'_i(\tau_s)$. A2. $h_i(\tau_s) \leq h_j(\tau_s) < h'_i(\tau_s)$. A3. $h_j(\tau_s) < h_i(\tau_s)$.

Case B: $h'_j(\tau_s) \leq h'_i(\tau_s)$ and $h_i(\tau_s) \leq g(\tau_s) < h_j(\tau_s)$(session i is legal and j not).

Case A1: In case A1 $h_j(\tau_s) \geq h'_i(\tau_s)$, therefore $h_j(\tau_e) - h_i(\tau_e) = h_j(\tau_s) - h'_i(\tau_s) \geq 0$ thus $\left|\frac{S_i(0,\tau_e)}{w_i} - \frac{S_j(0,\tau_e)}{w_j}\right| = h_j(\tau_e) - h_i(\tau_e)$. This obeys $h_j(\tau_e) - h_i(\tau_e) = h_j(\tau_s) - h'_i(\tau_s) < h_j(\tau_s) - h_i(\tau_s)$, where the equality results from i being served and j not and the from equality $h'_i(\tau_s) > h_i(\tau_s)$. Using the inductive assumption, the lemma holds for τ_s, that is $h_j(\tau_s) - h_i(\tau_s) \leq \max\{\frac{L_{i,max}}{w_i}, \frac{L_{j,max}}{w_j}\}$, thus implying $h_j(\tau_e) - h_i(\tau_e) \leq \max\{\frac{L_{i,max}}{w_i}, \frac{L_{j,max}}{w_j}\}$.

Case A2: $h_j(\tau_s) < h'_i(\tau_s)$, therefore $h_i(\tau_e) - h_j(\tau_e) = h'_i(\tau_s) - h_j(\tau_s) > 0$ thus $\left|\frac{S_i(0,\tau_e)}{w_i} - \frac{S_j(0,\tau_e)}{w_j}\right| = h_i(\tau_e) - h_j(\tau_e)$. Case A2 additionally assumes that $h_i(\tau_s) \leq h_j(\tau_s)$, therefore $h_i(\tau_e) - h_j(\tau_e) = h'_i(\tau_s) - h_j(\tau_s) \leq h'_i(\tau_s) - h_i(\tau_s) = L_i/w_i \leq L_{i,max}/w_i \leq \max\{\frac{L_{i,max}}{w_i}, \frac{L_{j,max}}{w_j}\}$.

Case A3: $h_j(\tau_s) < h_i(\tau_s)$, therefore $h_i(\tau_e) - h_j(\tau_e) = h_i'(\tau_s) - h_j(\tau_s) > 0$ thus $\left|\frac{S_i(0,\tau_e)}{w_i} - \frac{S_j(0,\tau_e)}{w_j}\right| = h_i(\tau_e) - h_j(\tau_e)$. In case A, $h_i'(\tau_s) \leq h_j'(\tau_s)$, therefore $h_i(\tau_e) - h_j(\tau_e) = h_i'(\tau_s) - h_j(\tau_s) \leq h_j'(\tau_s) - h_j(\tau_s) = L_j/w_j \leq L_{j,max}/w_j \leq \max\{\frac{L_{i,max}}{w_i}, \frac{L_{j,max}}{w_j}\}$.

Case B: $h_i'(\tau_s) \geq h_j'(\tau_s)$, therefore $h_i(\tau_e) - h_j(\tau_e) = h_i'(\tau_s) - h_j(\tau_s) \geq h_i'(\tau_s) - h_j'(\tau_s) \geq 0$ thus $\left|\frac{S_i(0,\tau_e)}{w_i} - \frac{S_j(0,\tau_e)}{w_j}\right| = h_i(\tau_e) - h_j(\tau_e)$. Case B additionally assumes that $h_j(\tau_s) > g(\tau_s) \geq h_i(\tau_s)$, therefore $h_i(\tau_e) - h_j(\tau_e) = h_i'(\tau_s) - h_j(\tau_s) < h_i'(\tau_s) - h_i(\tau_s) = L_i/w_i \leq L_{i,max}/w_i \leq \max\{\frac{L_{i,max}}{w_i}, \frac{L_{j,max}}{w_j}\}$.

C The Proof of Theorem 5.5

We divide the proof to two parts: (a) Equation (19) holds iff $\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}} \leq \frac{L_{i,max}}{w_i}$, and (b) Equation (20) holds iff $\frac{S_{i^c,P}(0,t)}{W_{i^c}} - \frac{S_{i,P}(0,t)}{w_i} \leq \frac{L_{i,max}}{w_i}$.

(a) Equation (19) holds iff $\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}} \leq \frac{L_{i,max}}{w_i}$ holds.

1) Assuming $\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}} \leq \frac{L_{i,max}}{w_i}$, we prove (19). From the assumption we have $S_{i,P}(0,t)W_{i^c} - (S_P(0,t) - S_{i,P}(0,t))w_i \leq L_{i,max} \cdot W_{i^c}$, or

$$S_{i,P}(0,t) - S_P(0,t)\frac{w_i}{W_{i^c} + w_i} \leq L_{i,max}\frac{W_{i^c}}{W_{i^c} + w_i}. \tag{37}$$

If the active session set is constant during interval $(0,t)$ then, $S_{i,GPS}(0,t) = S_{GPS}(0,t)\frac{w_i}{W}$. The service rate of GPS and of scheduler P is equal thus, $S_{GPS}(0,t) = S_P(0,t)$. From these two we get,

$$S_{i,GPS}(0,t) = S_P(0,t)\frac{w_i}{W}. \tag{38}$$

Substituting (38) in (37) we get, $S_{i,P}(0,t) - S_{i,GPS}(0,t) \leq L_{i,max}(1 - \frac{w_i}{W})$.

2) Assuming (19), we prove $\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}} \leq \frac{L_{i,max}}{w_i}$. Substituting (38) in (19) we get, $S_{i,P}(0,t) - S_P(0,t)\frac{w_i}{W} \leq (1 - \frac{w_i}{W})L_{i,max}$, or $S_{i,P}(0,t)(W_{i^c} + w_i) - S_P(0,t)w_i \leq (W - w_i)L_{i,max}$, or $S_{i,P}(0,t)W_{i^c} - (S_P(0,t) - S_{i,P}(0,t))w_i \leq W_{i^c} \cdot L_{i,max}$. Then we get, $\frac{S_{i,P}(0,t)}{w_i} - \frac{S_{i^c,P}(0,t)}{W_{i^c}} \leq \frac{L_{i,max}}{w_i}$.

(b) Equation (20) holds iff $\frac{S_{i^c,P}(0,t)}{W_{i^c}} - \frac{S_{i,P}(0,t)}{w_i} \leq \frac{L_{i,max}}{w_i}$ holds.

1) Assuming $\frac{S_{i^c,P}(0,t)}{W_{i^c}} - \frac{S_{i,P}(0,t)}{w_i} \leq \frac{L_{i,max}}{w_i}$, we prove (20). From the assumption we have $(S_P(0,t) - S_{i,P}(0,t))w_i - S_{i,P}(0,t)W_{i^c} \leq L_{i,max} \cdot W_{i^c}$, or

$$S_P(0,t)\frac{w_i}{W_{i^c} + w_i} - S_{i,P}(0,t) \leq L_{i,max}\frac{W_{i^c}}{W_{i^c} + w_i}. \tag{39}$$

Substituting (38) in (39) we get, $S_{i,GPS}(0,t) - S_{i,P}(0,t) \leq L_{i,max}(1 - \frac{w_i}{W})$.

2) Assuming (20), we prove $\frac{S_{i^c,P}(0,t)}{W_{i^c}} - \frac{S_{i,P}(0,t)}{w_i} \leq \frac{L_{i,max}}{w_i}$. Substituting (38) in (20) we get, $S_P(0,t)\frac{w_i}{W} - S_{i,P}(0,t) \leq (1 - \frac{w_i}{W})L_{i,max}$, or $S_P(0,t)w_i - S_{i,P}(0,t)(W_{i^c} + w_i) \leq (W - w_i)L_{i,max}$, or $(S_P(0,t) - S_{i,P}(0,t))w_i - S_{i,P}(0,t)W_{i^c} \leq W_{i^c} \cdot L_{i,max}$. Then we get, $\frac{S_{i^c,P}(0,t)}{W_{i^c}} - \frac{S_{i,P}(0,t)}{w_i} \leq \frac{L_{i,max}}{w_i}$.