A SFTS Properties

A.1 Fairness Properties

A.1.1 Share Guarantee.

THEOREM 1. *SFTS provides the share guarantee (a backlogged client should receive an equal share, i.e., key usage, of one of the resources it uses, regardless of the demand of other clients.).*

PROOF. Consider a system with a fixed number of clients, and all clients are backlogged. Without loss of generality, we assume that all clients are active during an interval $[t_1, t_2)$. Let us say that all clients start with the same virtual time at the beginning of the backlogged interval. By design, SFTS equalizes client virtual times, and virtual time is determined by the dominant key usage per transaction (Section 5.1). For a given client i that executes transaction x_i^k, let us say its dominant key usage is $s_{i,j}^k$. Assume that the backlogged interval starts at time 0. At time t, the virtual time of this client will advance by $V(t)$ +/- $s_{i,j}^k$. Let us say that client i's dominant resource at time t is j. Then, client i has used j by at least $V(t)$ +/- $s_{i,j}^k$. Thus, each client's virtual time increases by its dominant key usage, and this is equalized by SFTS. □

A.1.2 Strategy-Proofness.

THEOREM 2. *SFTS ensures strategy-proofness (a client cannot increase its dominant resource usage by lying about its demand, i.e., by adding extraneous operations and/or transactions).*

PROOF. Assume for a contradiction that in a continuous backlogged interval $[t_1, t_2)$, client i gets more service by lying in that interval. We consider two scenarios: (a) client i lies about its demand and (b) client i does not lie. We assume that client i gets more service in case (a) than (b). For this to be true, there must be a point in time during $[t_1, t_2)$ when client i has a transaction scheduled in (a) but not in (b). Otherwise, client i would get the same usage in both cases. Let t_0 be the first time in $[t_1, t_2)$ when this happens. Let us say that client j's transaction is scheduled at t_0 in (b) while client i's transaction is scheduled in (a). The only way this can occur is if client j's dominant share is great than client i's dominant share in (a) but not (b). For client j to have higher dominant share in (a) than (b), client i must artificially inflate client j's dominant share. There are two ways this can happen: client i adds a dummy transaction or it adds dummy operations to an existing transaction.

First, we consider the case when a client i adds an extraneous transaction $\hat{x}_i^k$ to try to increase its dominant resource usage by causing conflicts with another client j's transaction earlier in the schedule. Let us say that $\hat{x}_i^k$ causes an additional conflict with $x_l^{k'}$, which would cause client l's usage to increase. We consider two cases: (i) when client l's dominant resource changes and (ii) when client l's dominant resource does not change. For (i), we have client l's dominant resource change from j to j'. We denote the old usage on key j as $s_{l,j}^k$, the old usage on key j' as $b_{l,j'}^k + \frac{d_{l,j'}^k}{m}$, and the new usage on key j' as $b_{l,j'}^k + \frac{\hat{d}_{l,j'}^k}{m}$. Since the dominant resource changes, this means that $s_{l,j}^k > b_{l,j'}^k + \frac{d_{l,j'}^k}{m}$. We have that client j's virtual time increases by $b_{l,j'}^k + \frac{\hat{d}_{l,j'}^k}{m} - s_{l,j}^k$ as a result of $\hat{x}_i^k$. Since $s_{l,j}^k > b_{l,j'}^k + \frac{d_{l,j'}^k}{m}$, this means that the most client j's virtual time can increase by is $\frac{\hat{d}_{l,j'}^k}{m}$. As SFTS spreads the the schedule-dependent usage of client l is evenly across conflicting requests, client i's virtual time will increase by $\frac{\hat{d}_{l,j'}^k}{m}$. Thus, client i's virtual time cannot increase less than client l's, and it cannot manipulate the scheduler to increase its dominant share, which is a contradiction. For (ii), if the dominant resource of client l does not change, then client l's virtual time would also increase by $\frac{\hat{d}_{l,j'}^k}{m}$ at most. By the same logic as (i), client i's virtual time cannot increase less than client l's so that client i's transactions are scheduled earlier.

Next, we consider the case when a client i adds extraneous operation(s) to transaction x_i^k to produce a new transaction $\hat{x}_i^k$ to try to increase its dominant resource usage by causing conflicts with another client j's transaction earlier in the schedule. By similar logic in (i), client i's virtual time will increase by $\frac{\hat{d}_{l,j'}^k}{m}$, at least as much as client l's virtual time, so it cannot increase its dominant resource usage, which is a contradiction. □

A.2 Fairness for Backlogged Clients

Next, we provide several standard properties, which have been established for fair schedulers in prior work [34], bounding the delay experience by clients.

A.2.1 Fairness for Backlogged Clients. First, we ensure the lack of starvation, regardless of client workloads; that is, the unfairness between backlogged clients will be bounded over a given time interval. We adapt the notion of *dominant-resource monotonic* [34] for the transactional setting. A client is called dominant-resource monotonic if, during any of its backlogged periods, its dominant resource does not change. A client in which all transactions have the same dominant resource is trivially a dominant-resource monotonic client. We use $s_{i,r}^{\uparrow}$ to denote $\max_k s_{i,r}^k$, the constant dominant resource of each client. The bound in this lemma depends on Δ because this parameter determines maximum difference between virtual start times of different clients.

THEOREM 4. *Consider dominant-resource monotonic clients i and j, both backlogged during the interval $[t_1, t_2)$. Let $W_i(t_1, t_2)$ and $W_j(t_1, t_2)$ be the total processing times consumed by clients i and j, respectively, on their dominant resource during interval $[t_1, t_2)$. Then we have*

$$\left| \frac{W_i(t_1, t_2)}{w_i} - \frac{W_j(t_1, t_2)}{w_j} \right| < \frac{s_{i,d_i}^{\uparrow}}{w_i} + \frac{s_{i,d_j}^{\uparrow}}{w_j} + \Delta$$

where $s_{q,d_q}^{\uparrow}$ represents the maximum processing time of a transaction of client q on its dominant resource d_q.

PROOF. Let $v_1 = V(t_1)$ and $v_2 = V(t_2)$. Consider a client i, with dominant resource d_i, that is backlogged in that entire interval of time. We prove the theorem using two cases: when $v_2 - v_1 > \frac{s_{i,d_i}^{\uparrow}}{w_i} + \Delta$, such that client i can have at least have one transaction with start time inside the interval and when the opposite is true. For both, we establish lower and upper bounds on the amount of processing time client i receives in the interval. The maximum discrepancy between these two is then used to establish the theorem.

Consider the following lower bound on the amount of processing time client i receives in the interval $[t_1, t_2]$ on its dominant resource d_i. We observe that the amount of processing time that client i receives on its dominant resource is at least $v_2 - S(t_i^k, d_i)$, where transaction t_i^k is client i's first transaction served in the interval $[t_1, t_2]$. This is because, by definition of V, virtual time can only progress to a greater value v_g when all transactions with maximum starting time less than v_g have been served. Since client i is backlogged, the start time of each of client i's transactions, after the k^{th}, is equal to the finish time of the previous transaction. Thus, it received processing time on its dominant resource equal to $\sum_{j=k}^{m} s_{i,d_i}^{j}$, where t_i^m is the last transaction in the window (i.e., $S(t_i^m, d_i) \in [v_1, v_2]$, $F(t_i^m, d_i) \geq v_2$). Thus, $S(t_i^k, d_i)$ determines the total processing time received on d_i in the interval, assuming that client i appears only once in the schedule since we seek the lower bound. By the same assumption, client i's previous request t_i^{k-1} has a start time less than v_1. As client i is backlogged, $S(t_i^k, d_i) \leq v_1 + \frac{s_{i,d_i}^{\uparrow}}{w_i}$. Thus, i will receive at least $(v_2 - v_1) - \frac{s_{i,d_i}^{\uparrow}}{w_i}$ processing time on d_i in the interval $[t_1, t_2]$.

Consider an upper bound on the amount of processing i receives on d_i in $[t_1, t_2]$. The processing time is maximized when the first transaction k starts at $S(t_i^k, d_i) = v_1$, and the last transaction t_i^m starts at the end of the interval (i.e., $S(t_i^m, d_i) = v_2$). Since t_i^m's processing time on d_i is at most $\frac{s_{i,d_i}^{\uparrow}}{w_i} + \Delta$, it can receive processing time in the virtual time interval $[v_1, v_2 + \frac{S(t_i^m, d_i)}{w_i} + \Delta]$. Thus, i receives at most $(v_2 - v_1) + \frac{s_{i,d_i}^{\uparrow}}{w_i} + \Delta$ processing time on d_i in $[t_1, t_2]$.

The maximum discrepancy of processing times received by the the backlogged clients i and j on their respective dominant resource in the interval $[t_1, t_2]$ is therefore $\frac{s_{i,d_i}^{\uparrow}}{w_i} + \frac{s_{j,d_j}^{\uparrow}}{w_j} + \Delta$, which proves the theorem for the first case.

Now, consider the case when $v_2 - v_1 \leq \frac{s_{q,dq}^{\uparrow}}{w_q} + \Delta$, for any client $q \in \{i, j\}$. Then, q might not get any service in $[t_1, t_2]$. The upper bound is that q has one transaction that happens to have start time in the interval. Thus, the maximum discrepancy is $\max_{q \in \{i,j\}} \left\{ \frac{s_{q,dq}^{\uparrow}}{w_q} \right\} + \Delta$, which proves the theorem for this case. □

A.2.2 Bounded Delay. We also bound on the maximum delay observed by a client, ensuring that no client will be starved indefinitely. This result does not assume that clients are dominant-resource monotonic.

Theorem 5. *Assume transaction t_i^k of client i arrives at time t, and assume client i is idle at time t. Let r encompass all the resources that client i will access in its next transaction. Let n be the total number of clients in the system. Then the maximum delay to start serving transaction t_i^k, denoted as $D(t_i^k)$, is bounded above by*

$$D(t_i^k) \leq \max_r \left(\sum_{j=1, j \neq i}^{n} \left(s_{j,r}^{\uparrow} + \Delta \right) \right)$$

where $s_{j,d_j}^{\uparrow}$ represents the maximum processing time of a transaction of client j on its dominant resource d_j.

Proof. If no transaction is being processed at time t, t_i^k will immediately be processed, and the theorem trivially holds. Thus, assume a transaction is currently being processed with maximum start time $v_t = \max_r v(t, r)$. Since client i is assumed to not be backlogged, we have $S(t_i^k) = v_t$. Since processing happens with increasing start tags, all backlogged transactions p must have $\max_r S(p, r) \geq v_t$. Since each client has strictly non-zero processing time on at least one resource (its dominant resource), at most one transaction from each client can have starting time v_t. In the worst case, each of these clients requires $s_{j,d_j}^{\uparrow} + \Delta$ processing time, and there are $n - 1$ other clients. Consequently, the total delay is no greater than the resource that finishes last when processing all the $n - 1$ client transactions that have virtual start times within $[v_t, v_t + \Delta]$. □

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009