Multiplayer Interactive Computation Games Notes

William George and Clément Lesaege
Kleros Cooperative

1 Introduction

A severe limitation to many existing blockchain systems is their requirement that all nodes process all transactions. This has led to limits on how many transactions can be handled by the system while remaining decentralized [3]. Systems such as Truebit [4] have pioneered an approach to address this problem by allowing a computation to be performed off-chain, with participants reporting the result on-chain. Then, to the degree that participants disagree about the result, an interactive game can be triggered under which an on-chain smart contract only needs to perform a limited number of operations to determine a single point of disagreement between two participants which can be resolved by the on-chain contract executing this single step of disagreement on-chain.

Two players that disagree on the final state of a computation can play a relatively straightforward interactive game to determine their point of disagreement:

Game 1. *Suppose a computation requires n steps and players Alice and Bob disagree on the result. Then they can each provide their value at step $\lfloor \frac{n}{2} \rfloor$. They either disagree on the value at step $\lfloor \frac{n}{2} \rfloor$ or they agree; if they agree at step $\lfloor \frac{n}{2} \rfloor$ then there must be a disagreement between this step and step n. In either event, they now have a computation of at most $\frac{n}{2}$ steps on which they disagree on the result. Thus this process can be repeated resulting in a game that requires at most $\log_2(n)$ intermediate steps.*

In this work we will consider the efficiency and security of a particular approach to generalize this idea to multiple players. Particularly, even if some players are assumed to be malicious, one should still have good upper bounds on the number of steps required for this process to halt and the amount of computation that each individual honest participant should be required to perform.

2 Notation

We denote the total number of participants who stake on results by N. Let A denote the set of possible responses that players can submit for the current

state.

Time is divided as follows:

- "Periods", where a smart contract referee waits for information to be submitted to it by one or multiple parties.
- "Rounds", where participants play an interactive game, potentially one player against many, to perform a binary search for a point of disagreement in a calculation. In a given round, we will see that each player need only provide intermediate calculations of a single branch in the Merkle tree of intermediate calculations.
- "Phases" are collections of one or more rounds in which the set of participating players is held constant.

3 Algorithm

Players are awarded or lose points during the course of this process; each participant begins with one point. Then, at the beginning of each phase, active participants are matched into interactive games as follows:

MATCH

- While (there are unmatched parties with different answers):
  - While (true):
    - Take the unmatched party with the highest amount of points, call the party the "asker"
    - Add as "answerer" a party which:
      - Is unmatched.
      - Has a different result from that of the asker.
      - Has the same result as previously matched parties.
      - Has equal or less points than the available points of the asker (available points = asker points - $\sum$(answerer points)).
      - Has the highest amount of points possible subject to the previous conditions.
    - If not possible:
      - Break

During the course of a round, the asker plays the binary search procedure of Game 1 against the group of askerers to find a point in the calculation where they disagree. Note that the askerers can be thought of as acting as a single party for the sake of Game 1 to the degree that they provide unanimous responses. Then, if 2 or more answerers provide contradictory answers, they are removed from the game and put to play a derivative "asker-answerer" game together as follows:

SUB-MATCH

- While (there are answerers with different intermediate answers):
  - While (true):
    - Take the answerer with the highest amount of points as the sub-asker and add as sub-answerer a party which:
      - Was also an answerer in the existing game
      - Has a different intermediary result from that of the sub-asker.
      - Has not already been matched into a sub-game.
      - Has the same intermediary result as previously matched parties.
      - Has equal or less points than the available points of the sub-asker (available points = sub-asker points - $\sum$(sub-answerer points)).
      - Has the highest amount of points possible subject to the previous conditions.
    - If not possible:
      - Break
    - Sub-askers and sub-answerers matched by this process are removed from the game and play a game together. Other answerers continue the original game.

If (sub)-answerers are eliminated, their points are transferred to the corresponding (sub)-asker. If a (sub)-asker is eliminated, her points are redistributed to the (sub)-answerers in amounts corresponding to their existing points, with any leftover points distributed by the labeling order of the players. The set of (sub)-answerers who receive points should include any non-eliminated answerers who were separated off to play sub-games, so this redistribution of points should be performed on any terminated sub-games before being performed on the parent game.

Once, after some number of rounds but within a given phase, a sub-asker-sub-answerer game terminates, participants who are not eliminated return to the original game generated by MATCH as answerers.

Note that MATCH and SUB-MATCH can be implemented optimistically, so that the smart contract verifier checks the criteria related to whether parties are unmatched, whether their results are different from that of the asker, and whether the answerers have less that or equal the number of points of the asker. Then in order to find sets of parties that have the highest amount of points satisfying these conditions, the smart contract verifier can accept submissions, verify the other properties and compare the number of points to those of a pending submission. This avoids the verifier being required to perform costly sort and search operations on-chain. As this process is not interactive, it can be performed in one period. Moreover, as the number of points in the sub-game is strictly less than the number of points held by the answerer in the original game, each participant will be rematched by SUB-MATCH at most $\log_2 N$ times. Thus, a round requires $O(\log_2 N)$ periods.

Now we define:

Definition 1. *An asker-answerer game is said to be "resolved" when all remaining non-eliminated players in this game provide the same top-level answer. A phase is said to be "resolved" when all all asker-answerer games generated by MATCH are resolved.*

Namely, a phase is resolved when all asker-answerer games have terminated either by the asker begin eliminated or all answerers being eliminated. This includes answerers that play sub-asker-sub-answerer games, so these sub-games must terminate before the main game can be resolved.

Example 1. *Suppose that at the beginning of a phase we have 10 remaining participants with the following results and point totals. We also indicate whether the player is honest, namely whether she will provide correct computations at each stage:*

Player	1	2	3	4	5	6	7	8	9	10
Result	a	b	c	c	c	c	c	c	c	c
Points	4	4	2	1	1	1	1	1	1	1
Honest?	N	N	N	N	N	N	N	N	Y	Y

We provide a possible run of the algorithm of Section 3 for these players in Table 1. We suppose that the number of steps required to perform the computation n is of the form 2^i for simplicity. Here, in any given step the players that are paired together in asker-answerer games are indicated in brackets, with the asker on the left followed by | and then the answerer(s) on the right. For example, player 1 is paired as an asker against answerers 3, 4, and 5. Then due to conflicting answers 3, 4, and 5 are split off to play a sub-game, from which eventually 4 and 5 play another sub-game between each other.

Note that player 10 cannot be paired by the initial application of MATCH at the beginning of Phase 1, and then as the set of players is fixed during this phase she remains inactive during Phase 1, Rounds 2 and 3.

The table shows, for each player the intermediate value of the computation they are asked to provide and their response at each step. For example, during Phase 1, Round 2, player 3 is asked for her value at the $n/4$th step which she provides as c_{11}.

Note that even though there are two non-eliminated players at the end of Phase 2, Round 1, the process terminates as they both share c as their answer for the final state of the system.

4 Analysis

We make a few basic observations about the algorithm of Section 3.

Proposition 1. *In each round, each participant is only required to provide a single branch in the binary search of intermediate computations.*

Table 1: Possible sequence of intermediate calculations provided by different players of Example 1.

Player	1	3	4	5	2	6	7	8	9	10
Initial Points	4	2	1	1	4	1	1	1	1	1
Phase 1, Round 1										
Step:Response	$\lceil n : a \rceil$	$n : c$	$n : c$	$n : c\rfloor$	$\lceil n : b \rceil$	$n : c$	$n : c$	$n : c$	$n : c\rfloor$	inactive
	$\lceil n/2 : a_1 \rceil$	$n/2 : c_1$	$n/2 : c_2$	$n/2 : c_2\rfloor$	$\lceil n/2 : b_1 \rceil$	$n/2 : b_1$	$n/2 : b_1$	$n/2 : c_1$	$n/2 : c_2\rfloor$	inactive
	inactive	$\lceil n/4 : c_{11} \rceil$	$n/4 : c_{21}$	$n/4 : c_{22}\rfloor$	$\lceil 3n/4 : b_{11} \rceil$	$3n/4 : c_{11}$	$3n/4 : c_{11}\rfloor$	$\lceil n/4 : c_{12} \rceil$	$n/4 : c_{21}\rfloor$	inactive
	inactive	inactive	$\lceil n/8 : c_{211} \rceil$	$n/8 : c_{221}\rfloor$	$\lceil 5n/8 : b_{111} \rceil$	$5n/8 : c_{111}$	$5n/8 : c_{111}\rfloor$	$\lceil n/8 : c_{121} \rceil$	$n/8 : c_{211}\rfloor$	inactive
	....									
Last step	inactive	inactive	win v. 5	lose to 4	win v. 6,7	lose to 2	lose to 2	lose to 9	win v. 8	inactive
Post-Round Points	4	2	2	0	6	0	0	0	2	1
Phase 1, Round 2										
Step:Response	inactive	$\lceil n/4 : c_{11} \rceil$	$n/4 : c_{21}\rfloor$		$\lceil n/2 : b_1 \rceil$				$n/2 : c_1\rfloor$	inactive
	inactive	$\lceil n/8 : c_{211} \rceil$	$n/8 : c_{211}\rfloor$		$\lceil n/4 : b_{12} \rceil$				$n/4 : b_{12}\rfloor$	inactive
	....									
Last step	inactive	win v. 4	lose to 3		lose to 9				win v. 2	inactive
Post-Round Points	4	4	0	0	0	0	0	0	8	1
Phase 1, Round 3										
Step:Response	$\lceil n/2 : a_1 \rceil$	$n/2 : c_1\rfloor$							inactive	inactive
	$\lceil n/4 : a_{11} \rceil$	$n/4 : c_{11}\rfloor$							inactive	inactive
	....									
Last step	win v. 3	lose to 1							inactive	inactive
Post-Phase Points	8	0	0	0	0	0	0	0	8	1
Phase 2, Round 1										
Step:Response	$\lceil n : a \rceil$								$n : c\rfloor$	inactive
	$\lceil n/2 : a_1 \rceil$								$n/2 : c_2\rfloor$	inactive
	....									
Last step	lose to 9								win v. 1	inactive
Final Points	0	0	0	0	0	0	0	0	16	1

Proof. This is true by construction as each time answerers provide divergent responses to an intermediate computation, they are separated to play a derived game. Hence the asker is only required to perform the binary search along the branch corresponding to at most one value for this intermediate computation. □

Proof. □

We note the following growth property on the number of points of active participants:

Lemma 1. *Let $\{1, \ldots, n\}$ be the set of participants that are matched into an asker-answerer game in a given round, $n > 1$. Denote by k_i the number of points of participant i. Without loss of generality take $k_1 \geq k_2 \geq \ldots \geq k_n$. Then, at the end of the round, each remaining player in $\{1, \ldots, n\}$ has at least $2k_n$ points.*

Proof. Denote by $K = \sum_{i=1}^{n} k_i$ the total number of points belonging to players matched in a phase. We perform induction on $K - k_n$. Note

$$K - k_n = \sum_{i=1}^{n-1} k_i \geq \sum_{i=1}^{n-1} k_n \geq k_n \geq 1,$$

so if we suppose $K - k_n = 1$, this implies $k_n = 1$ and $K = 2$. Hence this is a game between an asker and a single answerer, which is resolved in one round with the winning party holding $K \geq 2k_n$ points.

Now suppose the induction hypothesis holds for all such matchings where $K - k_n \geq t \in \mathbb{N}$. We will show that it also holds when $K - k_n \geq t + 1$. Taking $1 \in \{1, \ldots, n\}$ again to be the asker in the main game, one or more asker-answerer sub-games $\Gamma_1, \Gamma_2, \ldots$ may break off from the main game/have already broken off in a previous round, leaving the (potentially empty) list $i_1 \ldots, i_s$ as the answers that continue the interactive game against 1 in this round. Note that for each Γ_w,

$$\sum_{i \text{ participates in } \Gamma_w} k_i < K.$$

However, for any Γ_w, all sub-answerers are also answerers in the main game, so they have at least k_n points. Hence, for this sub-game the difference between the total points and the fewest points of any player is at most

$$\sum_{i \text{ participates in } \Gamma_w} k_i - k_n \leq t.$$

Then by induction, each non-eliminated participant of Γ_w has at least $2k_n$ at the end of the round. The participants $i_1 \ldots, i_s$ either lose to the asker and are eliminated, or they win against the asker, resolving the game and at least doubling their points after the redistribution of the asker's points.

□

Corollary 1. *Let $\{1, \ldots, n\}$ be the set of participants that are matched into various asker-answerer games in a given round, $n > 1$. Denote by k_i the number of points of participant i. Without loss of generality take $k_1 \geq k_2 \geq \ldots \geq k_n$. Then, after a finite number of rounds, the phase resolves. In particular at most*

$$\log_2 \left(\frac{k_*}{k_n} \right) \leq \log_2 \left(\frac{k_*}{1} \right) = \log_2 k_* \leq \log_2 N$$

additional rounds are required for the phase to resolve, where k_* is the maximum number of points held by any non-eliminated player in $\{1, \ldots, n\}$ at the end of the last round of the phase.

Proof. As MATCH is only called at the beginning of phases, during the phase the players in asker-answerer games remain the same. Note that k_n is a lower bound on the number of points of participants in each of the asker-answerer games in this round. Then by Lemma 1, if after t subsequent rounds the phase has not resolved, all players must have at least $2^t \cdot k_n$ points. As the number of points of any one player cannot exceed N, we must have $t \leq \log_2 \left(\frac{N}{k_n} \right)$. In fact, $2^t \cdot k_n \leq k_*$, so $t \leq \log_2 \left(\frac{k_*}{k_n} \right)$.

□

We develop some additional notation that will be useful. Denote by A_j the subset $A \subseteq A$ of options $a \in A_j$ for which there remains a non-eliminated participant staked a at the beginning of the phase j. We denote by $L_j(i)$ the number of points held by the player $i \in N$ after the jth phase. Denote by $a(i)$ the option in A that is staked upon by the player i. Then we define, for each $a \in A$,:

$$l_j(a) = \min_{\substack{i \in N, a(i) = a \\ i \text{ not eliminated} \\ \text{at end of } j\text{th phase}}} L_j(i).$$

Note that $L_j(i)$ is monotonically non-decreasing as long as the player i is not eliminated, so $l_j(a)$ also is monotonically non-decreasing in j for all $a \in A$ for which there exists at least one player i at the end of phase j with $a(i) = a$.

Note that, each time MATCH is called at the beginning of a phase, all participants that are non-matched must share the same response, otherwise they would be matched between each other. We define the sequence of elements in A: $a_{*,j}$ by taking this response for each phase j. (Note that in some phases j it is possible that all participants are matched. In this case we write $a_{*,j} = \emptyset$, and any statement of the form $a \neq a_{*,j}$ holds trivially for all $a \in A$.)

Lemma 2. *For all j such that $|A_j| \geq 2$,*

$$l_j(a) \geq 1.5 \min_{a_0 \in A_j, a_0 \neq a_{*,j}} l_{j-1}(a_0)$$

for all $a \neq a_{*,j}$.

Proof. Let i be a participant who is not eliminated at the beginning of the jth phase or equivalently at the end of the $j-1$st phase. Then there are three possibilities of how i is matched by MATCH:

- i is an asker
- i is an answerer, or
- i is not matched, in which case $a(i) = a_{*,j}$.

Note that if i is an answerer, i's points are either (at least) doubled if i wins her game against the asker, or i is eliminated.

If i is an asker, we consider two subcases, either i is matched with a participant that stakes on $a_{*,j}$ or i is matched with a participant that stakes on some other option. If i is matched with a participant that stakes on $a_{*,j}$, then by the definition of $a_{*,j}$ there exists participants that could be matched with i that have $k_1 \geq k_2 \geq \ldots$ points and $\sum_{t=1}^{T} k_t > L_{j-1}(i)$. Without loss of generality take T to be the smallest value for which this holds so that

$$\sum_{t=1}^{T} k_t > L_{j-1}(i) \geq \sum_{t=1}^{T-1} k_t.$$

Then, considering i to be matched with the $T-1$ participants corresponding to $k_1, \ldots, k_{T-1}$, as $k_1 \geq k_2 \geq \ldots \geq k_T$, one has $\sum_{t=1}^{T-1} k_t \geq (T-1)k_T$, which implies

$$1 + \frac{1}{T-1} \geq \frac{\sum_{t=1}^{T-1} k_t + k_T}{\sum_{t=1}^{T-1} k_t} \Rightarrow \frac{T}{T-1} \sum_{t=1}^{T-1} k_t \geq \sum_{t=1}^{T} k_t > L_{j-1}(i)$$

$$\Rightarrow \frac{\sum_{t=1}^{T-1}}{L_j(i)} \geq \frac{T-1}{T}.$$

As i is assumed to be matched, $T-1 \geq 1$, so this bound implies that the collective points of answerers paired against i are at least $1.5L_{j-1}(i)$. On the other hand, suppose i is matched with a participant that stakes on a response other than $a_{*,j}$; moreover note that this answerer must also stake on a response other than $a(i)$. As i must be paired with at least one answerer, so the collective points of such answerers are at least $\min_{a_0 \in A_j, a_0 \neq a(i), a_{*,j}} l_{j-1}(a_0)$.

Thus, if i is not eliminated, at the end of phase j,

$$L_j(i) \geq \min \left\{ \begin{array}{l} 1.5L_{j-1}(i) \\ L_{j-1}(i) + \min_{a_0 \in A_j, a_0 \neq a(i), a_{*,j}} l_{j-1}(a_0) \end{array} \right\}.$$

As $L_{j-1}(i) \geq l_{j-1}(a(i))$, and this bound holds for all i not eliminated at the end of the jth phase such that $a(i) \neq a_{*,j}$,

$$l_j(a) \geq \min \left\{ \begin{array}{l} 1.5l_{j-1}(a) \\ l_{j-1}(a) + \min_{a_0 \in A_j, a_0 \neq a, a_{*,j}} l_{j-1}(a_0) \end{array} \right\} \geq 1.5 \min_{a_0 \in A_j, a_0 \neq a_{*,j}} l_{j-1}(a_0),$$

for $a \neq a_{*,j}$ giving the desired result.

□

We will find that it is easier to work with lower bounds on the $l_j(a)$ given by taking the minimum in Lemma 2 over a fixed set. Hence we define: $k_j(a)$ for all $a \in A$, $j \in \mathbb{N}_{\geq 0}$ recursively by by $k_0(a) = 1$ for all $a \in A$, and

$$k_j(a) = \left\{ \begin{array}{lcl} \max\left\{k_j(a), 1.5 \min_{a_0 \neq a_{*,j}} k_{j-1}(a_0)\right\} & : & a \neq a_{*,j} \\ k_{j-1}(a_{*,j}) & : & a = a_{*,j} \end{array} \right\}$$

for $j \geq 1$.

Lemma 3. *$l_j(a) \geq k_j(a)$ for all $a \in A_{j+1}$.*

Proof. We proceed by induction. Suppose $j = 1$. Note by definition $k_1(a) \leq 1.5$ for all $a \in A$ and $k_1(a_{*,1}) = 1$. However, $l_1(a) \geq 1$ for all $a \in A_2$ and if $a \neq a_{*,1}$, any participant staked on a that is not eliminated in the first phase will win at least one point from an opposing asker or answerer, so $l_1(a) \geq 2$. Hence, the induction hypothesis holds in this case.

Then, assuming the statement holds through the jth phase. Suppose $a \in A_{j+2}$ such that $a \neq a_{*,j+1}$. We see that

$$\begin{aligned} l_{j+1}(a) &\geq 1.5 \min_{a_0 \in A_{j+1}, a_0 \neq a_{*,j+1}} l_j(a_0) \geq 1.5 \min_{a_0 \in A_{j+1}, a_0 \neq a_{*,j+1}} k_j(a_0) \\ &\geq 1.5 \min_{a_0 \neq a_{*,j+1}} k_j(a_0) = k_{j+1}(a), \end{aligned}$$

where the first inequality uses Lemma 2, the second inequality uses the induction hypothesis, and the third inequality uses the fact that we are taking the minimum over a larger set $A \supseteq A_{j+1}$. On the other hand, if $a = a_{*,j+1}$,

$$l_{j+1}(a) \geq l_j(a) \geq k_j(a) = k_{j+1}(a),$$

where the first inequality uses the monotonicity of $l_j(a)$ and the second inequality uses the induction hypothesis.

□

Finally, we prove:

Theorem 1. *The algorithm of Section 3 terminates; namely after a finite number of rounds all non-eliminated players will have provided the same top-level answer for the state of the system. Moreover, let k_* be the maximum number of points held by any non-eliminated player at the end of the last phase. Then the algorithm requires at most*

$$\left(2\frac{\log_2 k_*}{\log_2 1.5} + 1\right) \cdot \log_2 k_* \leq \left(2\frac{\log_2 N}{\log_2 1.5} + 1\right) \cdot \log_2 N$$

rounds. Consequently, the algorithm requires $O((\log_2 N)^3)$ periods.

Proof. Denote, for $j \geq 1$,

$$\mu_j = \#\left\{j_0 = 1, \ldots, j : \emptyset \neq a_{*,j_0} \in \arg\min_{a_0 \in A} k_{j_0 - 1}(a_0)\right\}$$

and

$$\nu_j = \#\left\{j_0 = 1, \ldots, j : a_{*,j_0} = \emptyset \text{ and } \arg\min_{a_0 \in A} k_{j_0-1}(a_0) = A\right\}.$$

We will first prove by induction that:

- $k_j(a) = 1.5^{\mu_j+\nu_j}$ for all $a \notin \arg\min_{a_0 \in A} k_j(a_0)$
- $k_j(a) = 1.5^{j-\mu_j}$ for all $a \in \arg\min_{a_0 \in A} k_j(a_0)$
- $\arg\min_{a_0 \in A} k_j(a_0) = A$ if and only if $2\mu_j + \nu_j = j$; otherwise $|\arg\min_{a_0 \in A} k_j(a_0)| = 1$.

We first consider the $j = 1$ case. Note that $k_0(a) = 1$ for all $a \in A$, so $\arg\min_{a_0 \in A} k_0(a_0) = A$, hence either $a_{*,1} \neq \emptyset$ and $\mu_1 = 1$ OR $a_{*,1} = \emptyset$ and $\nu_1 = 1$. If $a_{*,1} \neq \emptyset$, we compute $k_1(a) = 1.5 = 1.5^{\mu_1+\nu_1}$ for all $a \neq a_{*,1}$, and $k_1(a_{*,1}) = 1^{1-\mu_1} = 1$. Particularly, note that $|\arg\min_{a_0 \in A} k_1(a_0)| = |\{a_{*,1}\}| = 1$. If $a_{*,1} = \emptyset$, then $k_1(a) = 1.5 = 1.5^{1-\mu_j}$ for all $a \in A$, and indeed $2 \cdot 0 + 1 = 1 = j$ and $A = \arg\min_{a_0 \in A} k_1(a_0)$.

Now suppose the induction hypothesis holds for all values less than or equal to j. We see that

$$2\mu_{j_0} + \nu_{j_0} = j_0 \Leftrightarrow 1.5^{\mu_{j_0}+\nu_{j_0}} = 1.5^{j_0-\mu_{j_0}}, \tag{1}$$

so if we prove the formula for the values of $k_{j+1}(a)$ the condition that $\arg\min_{a_0 \in A} k_{j_0+1}(a_0) = A$ if and only if $2\mu_{j_0} + \nu_{j_0} = j_0$ follows.

We consider several cases. First suppose that $a_{*,j+1} \in \arg\min_{a_0 \in A} k_j(a_0)$, $a_{*,j+1} \neq \emptyset$. In this case $\mu_{j+1} = \mu_j + 1$ and $\nu_{j+1} = \nu_j$, and $k_j(a) \geq k_j(a_{*,j+1})$ for all $a \in A$. Then

$$k_{j+1}(a) = \max\left\{k_j(a), 1.5 \min_{a_0 \neq a_{*,j+1}} k_j(a_0)\right\} \geq 1.5 k_j(a_*, j+1)$$

for all $a \neq a_{*,j+1}$. On the other hand, $k_{j+1}(a_{*,j+1}) = k_j(a_{*,j})$. Hence, $\arg\min_{a_0 \in A} k_{j+1}(a_0) = \{a_{*,j+1}\}$.

We compute

$$k_{j+1}(a_{*,j+1}) = k_j(a_*, j+1) = 1.5^{j-\mu_j} = 1.5^{j+1-\mu_{j+1}},$$

where the second equality uses the induction hypothesis. On the other hand, we divide the case where $a \neq a_{*,j+1}$ into several subcases. If $a \in \arg\min_{a_0 \in A} k_j(a_0)$, then by the induction hypothesis $\arg\min_{a_0 \in A} k_j(a_0) = A$ which implies $2\mu_j + \nu_j = j$. Then $k_j(a) = 1.5^{j-\mu_j} = 1.5^{\mu_j+\nu_j}$. If $a \notin \arg\min_{a_0 \in A} k_j(a_0)$, then by the induction hypothesis $k_j(a) = 1.5^{\mu_j+\nu_j}$. So, in either case,

$$k_{j+1}(a) = \max\{1.5^{\mu_j+\nu_j}, 1.5 \min_{a_0 \neq a_{*,j+1}} k_j(a_0)\} = 1.5 \cdot 1.5^{\mu_j+\nu_j} = 1.5^{\mu_{j+1}+\nu_{j+1}}.$$

Note, if $2\mu_{j+1}+\nu_{j+1} = j+1$, we could use Equation 1 to see that $\arg\min_{a_0 \in A} k_{j+1}(a_0) = A$ which contradicts our results above, so we must, in fact, have that $2\mu_{j+1} + \nu_{j+1} \neq j+1$.

Now we consider the case where either

- $a_{*,j+1} \notin \arg\min_{a_0 \in A} k_j(a_0)$, or
- $a_{*,j+1} = \emptyset$ and $\arg\min_{a_0 \in A} k_j(a_0) \neq A$.

In both cases $\mu_{j+1} = \mu_j$ and $\nu_{j+1} = \nu_j$. Also, as in both cases we have that $\arg\min_{a_0 \in A} k_j(a_0) \neq A$, by the induction hypothesis we must have that $|\arg\min_{a_0 \in A} k_j(a_0)| = 1$. Take a_1 to be the unique element in $\arg\min_{a_0 \in A} k_j(a_0)$. Note that $a_1 \neq a_{*,j+1}$. Moreover, $k_j(a_1) < k_j(a)$ for all $a \neq a_1$. Note

$$k_{j+1}(a) = \max\{k_j(a), 1.5 \min_{a_0 \neq a_{*,j+1}} k_j(a_0)\} = \max\{k_j(a), 1.5k_j(a_1)\}$$

for all $a \neq a_{*,j+1}$. As the values of $k_j(a)$ are powers of 1.5 to integer powers, $k_j(a) > k_j(a_1)$ implies $k_j(a) \geq 1.5k_j(a_1)$. Hence, for a such that $a \neq a_1, a_{*,j+1}$, $k_{j+1}(a) = k_j(a)$. Similarly, if $a_{*,j+1} \neq \emptyset$, then by definition $k_{j+1}(a_{*,j+1}) = k_j(a_{*,j+1})$. In either case,

$$k_{j+1}(a) = k_j(a) = 1.5^{\mu_j + \nu_j} = 1.5^{\mu_{j+1}\nu_{j+1}}$$

using the induction hypothesis. Finally, we compute

$$k_{j+1}(a_1) = \max\{k_j(a_1), 1.5k_j(a_1)\} = 1.5k_j(a_1) = 1.5 \cdot 1.5^{j-\mu_j} = 1.5^{j+1-\mu_{j+1}}$$

again using the induction hypothesis.

We use Equation 1 to see that

$$\arg\min_{a_0 \in A} k_{j+1}(a_0) = A \Leftrightarrow 1.5^{\mu_{j+1}+\nu_{j+1}} = 1.5^{j+1-\mu_{j+1}} \Leftrightarrow 2\mu_{j+1} + \nu_{j+1} = j + 1.$$

Otherwise, $\arg\min_{a_0 \in A} k_{j+1}(a_0) = \{a_1\}$.

Finally, we consider the case where $a_{*,j+1} = \emptyset$ and $\arg\min_{a_0 \in A} k_{j_0-1} = A$. Here $\mu_{j+1} = \mu_j$ and $\nu_{j+1} = \nu_j + 1$. As $\arg\min_{a_0 \in A} k_{j_0-1} = A$, $k_j(a)$ must have a fixed value for all $a \in A$, which by using the induction hypothesis is $1.5^{j-\mu_j}$. Then, $k_{j+1}(a) = 1.5^{j+1-\mu_j} = 1.5^{j+1-\mu_{j+1}}$ for all $a \in A$. Thus, $\arg\min_{a_0 \in A} k_{j_0} = A$ as well, and indeed $2\mu_j + \nu_j = j$ implies $2\mu_{j+1} + \nu_{j+1} = j+1$ completing the induction.

However, $\mu_j \leq j$ for all j, so if $j > 2\frac{\log_2 N}{\log_2 1.5}$, then at least one of $1.5^{\mu_j+\nu_j} > N$ or $1.5^{j-\mu_j} > N$ will be true. As these are lower bounds on the $l_j(a)$ this implies that all points have been accumulated by participants staked on a single answer. As each phase takes at most $\log_2 N$ rounds by Corollary 1, this process takes at most $\left(2\frac{\log_2 N}{\log_2 1.5} + 1\right) \cdot \log_2 N$ total rounds. In fact, if $j > 2\frac{\log_2 k_*}{\log_2 1.5}$, then similarly we see

$$k_* < \max\left\{1.5^{\mu_j+\nu_j}, 1.5^{j-\mu_j}\right\} \leq k_*,$$

which is a contradiction. So, this process takes at most $\left(2\frac{\log_2 k_*}{\log_2 1.5} + 1\right) \cdot \log_2 k_*$ total rounds.

□

We see that the fact that players that are not paired at the beginning of a phase, such as player 10 in Example 1, remain inactive until the beginning of the next phase is (somewhat counter-intuitively) important. Indeed,

Example 2. *Suppose that at the beginning of a phase we had the following k participants only one of whom is honest:*

Player	1	2	3	4	...	$k-1$	k
Result	a	b	b	b	...	b	b
Points	r	$r-2$	1	1	...	1	1
Honest?	N	N	N	N	...	N	Y

Then, under MATCH, player 1 is paired as the asker against answerers 2, 3, and 4. However, player 3 can provide an intermediate state that disagrees with that of player 2 resulting in a sub-game between players 2 and 3. If player 1 ultimately eliminates player 4 and player 2 ultimately eliminates player 3, we are left with with player 1 having $r+1$ points, player 2 having r points. Under the algorithm described in Section 3, players 1 and 2 are required to resume their asker-answerer game before the phase can be resolved and new players can be paired. If this were not the case and we could apply MATCH to

Player	1	2	5	6	...	$k-1$	k
Result	a	b	b	b	...	b	b
Points	$r+1$	$r-1$	1	1	...	1	1
Honest?	N	N	N	N	...	N	Y

we would a game that has the same form as that of the initial table, hence we could recursively continue the above process, requiring a total of $k-1$ rounds.

Remark 1. *If we think of placing a stake on a given state as requiring a fixed monetary deposit D and players receiving a reward in terms of their number of points at the end of the game, then we can reformulate the bound of Theorem 1 into a bound on the griefing factor [2] [1] available to an attacker that maliciously participates in order to cause delay. Some percentage of amounts that are redistributed between players should be burned so that malicious players cannot preserve their funds by winning points against themselves. For example, each player i that finishes with $L_{end}(i)$ points can receive their deposit back as well as a reward of $\frac{L_{end}(i)-1}{2} \cdot D$, namely she receives half of the lost deposits corresponding to the $L_{end}(i)-1$ players from whose points she eventually gained.*

We suppose that we have a single attacking player Eve and one or more honest players. Then if k_* is the maximum number of points held by any non-eliminated player at the end of the last phase, Eve must have lost at least k_*-1 points as honest players always correctly respond at each step of the interactive computation game and never lose points. Hence, the cost of this attack is at least $\frac{k_*-1}{2} \cdot D$ (even if Eve wins back her own points), so using the bound on the amount of delay harm caused by the attack of Theorem 1, this attack has a griefing factor of at most:

$$\frac{\left(2\frac{\log_2 k_*}{\log_2 1.5}+1\right)\cdot \log_2 k_*}{\frac{k_*-1}{2}\cdot D} \leq O\left(\frac{(\log_2 k_*)^2}{k_*}\right).$$

Notably, large scale griefs where an attacker is willing to increase k_* to cause a longer delay have asymptotically decreasing griefing factors.

References

[1] Vitalik Buterin. The triangle of harm. https://vitalik.ca/general/2017/07/16/triangle_of_harm.html, 2017.

[2] Vitalik Buterin. A griefing factor analysis model. Ethresear.ch Forum, https://ethresear.ch/t/a-griefing-factor-analysis-model/2338, 2018.

[3] Trent McConaghy. The DCS triangle. The BigchainDB Blog, https://blog.bigchaindb.com/the-dcs-triangle-5ce0e9e0f1dc, 2016.

[4] Jason Teutsch and Christian Reitwießner. A scalable verification solution for blockchains. http://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf, 2017.