New Efficient Error-Free Multi-Valued Consensus with Byzantine Failures *

Guanfeng Liang and Nitin Vaidya
Department of Electrical and Computer Engineering, and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
gliang2@illinois.edu, nhv@illinois.edu

June 5, 2011

*This research is supported in part by Army Research Office grant W-911-NF-0710287 and National Science Foundation award 1059540. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

In this report, we investigate the multi-valued Byzantine consensus problem described as follows: There are n processors, namely $P_1, ..., P_n$, of which at most t processors may be *faulty* and deviate from the algorithm in arbitrary fashion. Denote the set of all fault-free processors as P_{good}. Each processor P_i is given an L-bit input value v_i, and they want to agree on a value v' such that the following properties are satisfied:

- *Termination*: every fault-free P_i eventually decides on an output value v'_i,
- *Consistency*: the output values of all fault-free processors are equal, i.e., for every fault-free processor P_i, $v'_i = v'$ for some v',
- *Validity*: if every fault-free P_i holds the same input $v_i = v$ for some v, then $v' = v$.

Algorithms that satisfy the above properties in all executions are said to be **error-free**.

The discussion in this report is not self-contained, and relies heavily on the material in [2] and [1] – please refer to these papers for necessary background.

1 A More Efficient Consensus Algorithm

In our recent paper [2] we introduced an algorithm that solves this problem error-free with communication complexity approximately $\frac{n(n-1)}{n-2t}L$, for large enough L. In this report, we are going to present a more efficient algorithm. The consensus algorithm in this report achieves communication complexity

$$\frac{n(n-1)}{n-t}L \ bits \tag{1}$$

for $t < n/3$ and sufficiently large L.

Our algorithm achieves consensus on a long value of L bits deterministically. Similar to the algorithm in [2], the proposed algorithm progresses in generations. Each processor P_i is given an input value v_i of L bits, which is divided into L/D parts of size D bits each. These parts are denoted as $v_i(1), v_i(2), \cdots, v_i(L/D)$. For the g-th generation ($1 \le g \le L/D$), each processor P_i uses $v_i(g)$ as its input in Algorithm 1. Each generation of the algorithm results in processor P_i deciding on g-th part (namely, $v'_i(g)$) of its final decision value v'_i.

The value $v_i(g)$ is represented by a vector of $n-t$ symbols, each symbol represented with $D/(n-t)$ bits. For convenience of presentation, we assume that $D/(n-t)$ is an integer. We will refer to these $n-t$ symbols as the *data symbols*.

An $(n, n-t)$ distance-$(t+1)$ Reed-Solomon code, denoted as C_{n-t}, is used to encode the $n-t$ data symbols into n *coded symbols*. We assume that $D/(n-t)$ is large enough to allow the above Reed-Solomon code to exist, specifically, $n \le 2^{D/(n-t)} - 1$. This condition is met only if L is large enough (since $L > D$).

In each generation g, a set of at least $n-t$ processors that appear to have identical inputs up to generation $g-1$ is maintained. More formally, our algorithm maintain a set P_{match} of size at least $n-t$ such that for every $P_i, P_j \in P_{match}$, $v_i(h) = v_j(h)$ appears to be true for all $h < g$. P_{match} is updated in every generation. Notice that, in a particular generation, if P_{match} does not exist, i.e., there are at least $t+1$ processors that appear to have input values different from the other

Algorithm 1 Multi-Valued Consensus (generation g)

1. **Matching Stage:**
   In the following steps, for every processor P_i: $R_i[k] \leftarrow S_j[k]$ whenever P_i receives $S_j[k]$ from its trusted processor P_j.
   Each processor $P_i \in P_{match}$ performs steps 1(a) and 1(b) as follows:
   (a) Compute $(S_i[1], \ldots, S_i[n]) = C_{n-t}(v_i(g))$, and *send* $S_i[i]$ to every trusted processor P_j. (including those not in P_{match}, and P_i itself.).
   (b) $\forall P_j$ that trusts P_i:
   If $P_i = \min\{l | P_l \in P_{match}$ and P_j trusts $P_l\}$, then P_i sends $S_i[k]$ to P_j for each k such that P_j does not trust $P_k \in P_{match}$.

   Each processor $P_j \notin P_{match}$ performs step 1(c) as follows:
   (c) Using the first $n-t$ symbols it has received in steps 1(a) and 1(b), P_j computes $S_j[j]$ according to C_{n-t}, then sends $S_j[j]$ to all trusted processors (Including P_j itself.).
2. **Checking Stage:**
   Each processor P_i (in P_{match} or not) performs Checking Stage as follows:
   (a) If $R_i \in C_{n-t}$ then $Detected_i \leftarrow$ **false** ; else $Detected_i \leftarrow$ **true** .
   (b) If $P_i \in P_{match}$ and $R_i \neq S_i$ then $Detected_i \leftarrow$ **true** .
   (c) Broadcast $Detected_i$ using *Broadcast_Single_Bit* .
   (d) Receive $Detected_j$ from each processor P_j (broadcast in step 2(c)).
   If $Detected_j =$ **false** for all P_j, decide on $v'_i(g) = C^{-1}_{n-t}(R_i)$; else enter Diagnosis Stage.
3. **Diagnosis Stage:**
   Each processor P_i (in P_{match} or not) performs Diagnosis Stage as follows:
   (a) Broadcast S_i and R_i using *Broadcast_Single_Bit* .
   (b) $S^{\#}_j \leftarrow S_j$ and $R^{\#}_j \leftarrow R_j$ received from P_j as a result of broadcast in step 3(a).

   Using the broadcast information, all processors perform the following steps identically:
   (c) For each edge (i, j) in *Diag_Graph* : Remove edge (i, j) if $\exists k$, such that P_j receives $S_i[k]$ from P_i in Matching stage and $R^{\#}_j[k] \neq S^{\#}_i[k]$
   (d) For each $P_i \in P_{match}$: If $S^{\#}_i \notin C_{n-t}$, then P_i must be faulty. So remove i and the adjacent edges from *Diag_Graph* .
   (e) For each $P_j \notin P_{match}$: If $S^{\#}_j[j]$ is not consistent with the subset of $n-t$ symbols of $R^{\#}_j$, from which $S^{\#}_j[j]$ is computed, P_j must be faulty. So remove j and the adjacent edges from *Diag_Graph* .
   (f) If at least $t+1$ edges at any vertex i have been removed, then P_i must be faulty. So remove i and the adjacent edges.
   (g) Find the maximum set of processors $P_{new} \subseteq P_{match}$ such that $S^{\#}_i = S^{\#}_j$ for every pair of $P_i, P_j \in P_{new}$. In case of a tie, pick any one.
   (h) If $|P_{new}| < n-t$, terminate the algorithm and decide on the default output.
   Else, decide on $v'_i(g) = C^{-1}_{n-t}(S^{\#}_j)$ for any $P_j \in P_{new}$, and update $P_{match} = P_{new}$.

processors, it can be guarantee that the fault-free nodes do not have identical inputs. Then our algorithm will terminate and all fault-free nodes will decide on a default output.

Initially (generation 1), P_{match} is the set of all n processors. The operations in each generation g are presented in Algorithm 1

1.1 Proof of Correctness

In this section, we prove the correctness of Algorithm 1. In the proofs of the following lemmas, we assume that the fault-free processors always trust each other [2].

Lemma 1 *If* $Detected_j$ =**false** *for all* P_j *in Line 2(d), all fault-free processors* $P_i \in P_{good}$ *decide on the identical output value* $v'(g)$ *such that* $v'(g) = v_j(g)$ *for all* $P_j \in P_{good} \cap P_{match}$.

Proof: According to the algorithm, every fault-free processor $P_i \in P_{good}$ has sent $S_i[i]$ (computed from $v_i(g)$ directly if $P_i \in P_{match}$, or computed using symbols received in Lines 1(a) and 1(b) if $P_i \notin P_{match}$) to all the other fault-free processors. As a result, $R_i|P_{good} = R_j|P_{good}$ is true for every pair of fault-free processors $P_i, P_j \in P_{good}$. Since $|P_{good}| \geq n - t$ and C_{n-t} is a distance-$(t+1)$ code, it follows that either all fault-free processors P_{good} decide on the same output, or at least one fault-free processor $P_i \in P_{good}$ sets $Detected_i \leftarrow$**true** in Line 2(a). In the case all $Detected_j$=**false** , all fault-free processors decide on an identical $v'(g)$. Moreover, according to Line 2(b), every fault-free processor $P_j \in P_{good} \cap P_{match}$ finds $R_j = S_j$. It then follows that $v'(g) = C_{n-t}^{-t}(R_j) = C_{n-t}^{-t}(S_j) = v_j(g)$.

□

Lemma 2 *If a* P_{new} *such that* $|P_{new}| \geq n - t$ *is found in Line 3(g), all fault-free processors* $P_i \in P_{good}$ *decide on the identical output value* $v'(g)$ *such that* $v'(g) = v_j(g)$ *for all* $P_j \in P_{good} \cap P_{new}$.

Proof: Since $|P_{new}| \geq n-t$ and since at most t processors are faulty, there must be at least $n-2t$ fault-free processors in $P_{good} \cap P_{new}$, which have broadcast the same $S^{\#}$'s in Line 3(b). So at Line 3(h), all fault-free processors decide on the identical output $v'(g) = v_j(g)$ for all $P_j \in P_{good} \cap P_{new}$.
□

Lemma 3 *If a* P_{new} *such that* $|P_{new}| \geq n - t$ *can not be found in Line 3(g), then there must be two fault-free processors* $P_i, P_j \in P_{good}$ *such that* $v_i \neq v_j$.

Proof: It is easy to see that if all fault-free processors in P_{good} are given the same input, then a P_{new} such that $|P_{new}| \geq n - t$ can always be found in Line 3(g). Then the lemma follows. □

For the correctness of the way *Diag_Graph* is updated, please see [1] and [2]. Now we can conclude the correctness of Algorithm 1 as the following theorem:

Theorem 1 *Given* n *processors with at most* $t < n/3$ *are faulty, each given an input value of* L *bits, Algorithm 1 achieves consensus correctly in* L/D *generations, with the diagnosis stage performed for at most* $t + t(t+1)$ *times.*

Proof: According to Lemmas 1 and 2, the decided output $v'(g)$ always equals to v_j for some $P_j \in P_{good} \cap P_{match}$, unless $|P_{new}| < n - t$ in Line 3(h). So consistency and validity properties are satisfied until $|P_{new}|$ becomes $< n - t$. In the case $|P_{new}| < n - t$, according to Lemma 3, there must be two fault-free processors that are given different inputs. Then it is safe to decide on a default output and terminate. So the L-bit output satisfies the consistency and validity properties.

Every time the diagnosis stage is performed, either at least one edge associated with a faulty processor is removed, or at least one processor is removed from P_{match}. So it takes at most $t(t+1)$ instances of the diagnosis stage before all faulty processors are identified. In addition, it will take at most t instances to remove fault-free processors from P_{match} until two fault-free processors are identified as having different inputs, and the algorithm terminates with a default output. □

1.2 Complexity

According to Theorem 1, we can compute the communication complexity of Algorithm 1 in a similar way as in [1] and [2]. With a appropriate choice of D, the complexity of Algorithm 1 can be made equal to

$$\frac{n(n-1)}{n-t}L + O(n^4 L^{0.5}). \tag{2}$$

So for sufficiently large L ($\Omega(n^6)$), the complexity is $O(nL)$.

2 More Efficient q-validity Consensus

In [2], we also introduced an algorithm that solves consensus while satisfying the "q-validity" property, as stated below, for all $t+1 \le q \le n-t$ with communication complexity $\frac{n(n-1)}{q-t}L$.

- q-Validity: If at least q fault-free processors hold an identical input v, then the output v' agreed by the fault-free processors equals input v_j for some fault-free processor P_j. Furthermore, if $q \ge \lceil \frac{n+1}{2} \rceil$, then $v' = v$.

When $q = t+1$, its complexity becomes $n(n-1)L$, which is not linear in n any more. In fact, this algorithm achieves communication complexity $O(nL)$ only when $q - t = \Omega(n)$.

On the other hand, Algorithm 1 can achieve q-validity for $q \ge \lceil \frac{n+1}{2} \rceil$ with communication complexity $\frac{n(n-1)}{q}L$, if we substitute every "$n-t$" with "q" in the algorithm. This formulation of complexity is independent of t, and remains to be $O(n)$ as long as $q = \Omega(n)$. However, Algorithm 1 with the mentioned modification cannot achieve q-validity for any $q < \lceil \frac{n+1}{2} \rceil$.

In this section, we present an algorithm that achieves q-validity for all $t+1 \le q \le n-t$ while keeping the complexity $O(nL)$, as long as $q = \Omega(n)$. This algorithm uses the "clique formation" technique from our previous algorithm in [2] to achieve q-validity when q is small, and uses the technique from Algorithm 1 presented in the previous section to improve communication complexity.

The value $v_i(g)$ is represented by a vector of q data symbols, each symbol represented with D/q bits. An (n, q) distance-$(n-q+1)$ Reed-Solomon code, denoted as C_q, is used to encode the q data symbols into n coded symbols. The operations in each generation g are presented in Algorithm 2

Algorithm 2 q-Validity Consensus, Matching and Checking stages (generation g)

1. **Matching Stage:**
   In the following steps, for every processor P_i: $R_i[k] \leftarrow S_j[k]$ whenever P_i receives $S_j[k]$ from its trusted processor P_j.
   Every processor P_i performs steps 1(a) to 1(e) as follows:
   (a) Compute $(S_i[1], \ldots, S_i[n]) = C_q(v_i(g))$, and *send* $S_i[i]$ to every trusted processor P_j.
   (b) If $S_i[j] = R_i[j]$ then $M_i[j] \leftarrow$ **true** ; else $M_i[j] \leftarrow$ **false**
   (c) P_i broadcasts the vector M_i using *Broadcast_Single_Bit*

   Using the received M vectors:

   (d) Find a set of processors P_{match} of size q such that
   $M_j[k] = M_k[j] =$ **true** for every pair of $P_j, P_k \in P_{match}$. If multiple possibility exist for P_{match}, then any one of the possible sets is chosen arbitrarily as P_{match} (all fault-free nodes choose a deterministic algorithm to select identical P_{match}).
   (e) If P_{match} does not exist, then decide on a default value and continue to the next generation;
   else continue to the following steps.

   Note: At this point, if P_{match} does not exist, it is, in fact, safe to terminate the algorithm with a default output since it can be asserted that no q fault-free nodes have identical inputs. However, by continuing to the next generation instead of terminating, q-validity is satisfied for the inputs of each individual generation.

   When P_{match} of size q is found, each processor $P_i \in P_{match}$ performs step 1(g) as follows:

   (f) $\forall P_j$ that trusts P_i:
   If $i = \min\{l | P_l \in P_{match}$ and P_j trusts $P_l\}$, then P_i sends $S_i[k]$ to P_j for each k such that P_j does not trust P_k.

   Each processor $P_j \notin P_{match}$ performs step 1(g) as follows:

   (g) Using the first q symbols it has received from the processors in P_{match} in steps 1(a) and 1(f), P_j computes $S_j[j]$ according to C_q, then sends $S_j[j]$ to all trusted processors.

   Note: For every processor P_i trusted by P_j, it has set $R_i[j]$ to the $S_j[j]$ received from P_j in step 1(a). It will be replaced with the new $S_j[j]$ received in step 1(g).

2. **Checking Stage:**
   Each processor P_i (in P_{match} or not) performs Checking Stage as follows:
   (a) If $R_i \in C_q$ then $Detected_i \leftarrow$ **false** ; else $Detected_i \leftarrow$ **true** .
   (b) If $P_i \in P_{match}$ and $R_i \neq S_i$ then $Detected_i \leftarrow$ **true** .
   (c) Broadcast $Detected_i$ using *Broadcast_Single_Bit* .
   (d) Receive $Detected_j$ from each processor P_j (broadcast in step 2(c)).
   If $Detected_j =$ **false** for all P_j, then decide on $v_i'(g) = C_q^{-1}(R_i)$; else enter Diagnosis Stage

Algorithm 2 q-Validity Consensus, Diagnosis stage (generation g)

3. **Diagnosis Stage:**

   Each processor P_i (in P_{match} or not) performs Diagnosis Stage as follows:

   (a) Broadcast S_i and R_i using *Broadcast_Single_Bit* .

   (b) $S_j^{\#} \leftarrow S_j$ and $R_j^{\#} \leftarrow R_j$ received from P_j as a result of broadcast in step 3(a).

   Using the broadcast information, all processors perform the following steps identically:

   (c) For each edge (i, j) in *Diag_Graph* : Remove edge (i, j) if $\exists k$, such that P_j receives $S_i[k]$ from P_i in Matching stage and $R_j^{\#}[k] \neq S_i^{\#}[k]$.

   (d) For each $P_i \in P_{match}$: If $S_i^{\#} \notin C_q$, then P_i must be faulty. So remove i and the adjacent edges from *Diag_Graph* .

   (e) For each $P_j \notin P_{match}$: If $S_j^{\#}[j]$ is not consistent with the subset of q symbols of $R_j^{\#}|P_{match}$, from which $S_j^{\#}[j]$ is computed, P_j must be faulty. So remove j and the adjacent edges from *Diag_Graph* .

   (f) If at least $t + 1$ edges at any vertex i have been removed, then P_i must be faulty. So remove i and the adjacent edges.

   (g) Find a set of processors $P_{decide} \subseteq P$ such that $S_i^{\#} = S_j^{\#}$ for every pair of $P_i, P_j \in P_{decide}$. In case of a tie, pick any one.

   (h) If $|P_{decide}| < q$, decide on the default output.
   Else, decide on $v_i'(g) = C_q^{-1}(S_j^{\#})$ for any $P_j \in P_{decide}$.

2.1 Proof of Correctness

Lemma 4 *If there are a set of at least q fault-free processors $Q \subseteq P_{good}$ such that for each $P_i \in Q$, $v_i(g) = v(g)$ for some $v(g)$, then a set P_{match} of size q necessarily exists.*

Proof: Since all the fault-free processors in Q have identical input $v(g)$, $S_i = C_q(v(g))$ for all $P_i \in Q$. Since these processors are fault-free and always trust each other, they send each other correct messages in the matching stage. Thus, $R_i[j] = S_j[j] = S_i[j]$ for all $P_i, P_j \in Q$. This fact implies that $M_i[j] = M_j[i] =$**true** for all $P_i, P_j \in Q$. Since there are $|Q| \geq q$ fault-free processors in Q, it follows that a set P_{match} of size q must exist. □

Lemma 5 *If $Detected_j =$***false** *for all P_j in Line 2(d), all fault-free processors $P_i \in P_{good}$ decide on the identical output value $v'(g)$ such that $v'(g) = v_j(g)$ for all $P_j \in P_{match} \cap P_{good}$.*

Proof: Observe that size of set $P_{match} \cap P_{good}$ is at least $q - t \geq 1$, so there must be at least one fault-free processor in P_{match}.

According to the algorithm, every fault-free processor $P_i \in P_{good}$ has sent $S_i[i]$ (computed from $v_i(g)$ directly if $P_i \in P_{match}$, or computed using the q symbols received from P_{match} in Lines 1(a) and 1(f) if $P_i \notin P_{match}$) to all the other fault-free processors. As a result, $R_i|P_{good} = R_j|P_{good}$ is true for every pair of fault-free processors $P_i, P_j \in P_{good}$. Since $|P_{good}| \geq n-t \geq q$ and C_q has dimension q, it follows that either all fault-free processors P_{good} decide on the same output, or at least one fault-free processor $P_i \in P_{good}$ sets $Detected_i \leftarrow$**true** in Line 2(a). In the case $Detected_j$=**false** for all P_j, all fault-free processors decide on an identical $v'(g)$. Moreover, according to Line 2(b), every fault-free processor $P_j \in P_{good} \cap P_{match}$ finds $R_j = S_j$. It then follows that $v'(g) = C_q^{-t}(R_j) = C_q^{-t}(S_j) = v_j(g)$ where $P_j \in P_{good} \cap P_{match}$.

□

Then we can have the following theorem about the correctness of Algorithm 2.

Theorem 2 *Given n processors with at most $t < n/3$ are faulty, each given an input value of L bits, Algorithm 2 achieves q-validity for each one of the L/D generations, with the diagnosis stage performed for at most $t(t+1)$ times.*

Proof: Similar to Theorem 1. □

2.2 Complexity

In Lines 1(a) and 1(f), every processor receives at most $n-1$ symbols, so at most $n(n-1)$ symbols are communicated in these two steps. In Line 1(g), every processor $P_j \notin P_{match}$ sends at most $n-1$ symbols, and there are at most $n-q$ processors not in P_{match}, so at most $(n-q)(n-1)$ symbols are communicated in this step. So in total, no more than $(2n-q)(n-1)$ symbols are communicated in the Matching stage. Then with an appropriate choice of D, the complexity of Algorithm 2 can be made to

$$\leq \frac{(2n-q)(n-1)}{q} L + O(n^4 L^{0.5}). \tag{3}$$

So for any $q = \Omega(n)$ and $t+1 \leq q \leq n-t$, with a sufficiently large L ($\Omega(n^6)$), the complexity is $O(nL)$.

References

[1] Guanfeng Liang and Nitin Vaidya. Complexity of multi-valued byzantine agreement. *Technical Report, CSL, UIUC (http://arxiv.org/abs/1006.2422)*, June 2010.

[2] Guanfeng Liang and Nitin Vaidya. Error-free multi-valued consensus with byzantine failures. In *ACM PODC*, 2011.