TOWARDS OPTIMAL DISTRIBUTED CONSENSUS

Extended Abstract

Piotr Berman†‡
Department of Computer Science
The University of Chicago
Chicago, IL 60637

Juan A. Garay†
Department of Computer Science
The Pennsylvania State University
University Park, PA 16802

Kenneth J. Perry
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

ABSTRACT

In a *Distributed Consensus* protocol all processors (of which t may be faulty) are given (binary) initial values; after exchanging messages all correct processors must agree on one of them. We measure the quality of a protocol using the following parameters: total number of processors n, number of rounds of message exchange r and maximal message length m. Their optima are respectively $3t+1$, $t+1$ and 1.

While no known protocol is optimal in all these three aspects simultaneously, the protocols presented in this paper take further steps in this direction. The first protocol, with $n>4t$, $r=t+1$ and polynomial message size, improves on the breakthrough result of Moses and Waarts [12]. The second protocol, with $n>3t$, $r=3t+3$ and $m=2$, improves on [3] and [13]. Notably, it is asymptotically optimal in all three quality parameters while using the optimal number of processors.

Using these protocols as building blocks, we obtain families of protocols with intermediate quality parameters, that offer better tradeoffs than the ones of Bar-Noy *et al.* [2]. All our protocols work in polynomial time and have succint descriptions.

1. Introduction

The need for coordinating decisions and fault tolerance arises in various contexts, e.g. distributed data bases, control of real-time processes, etc. The *Distributed Consensus* problem (and its 'twin', *Byzantine Agreement*) provides perhaps the most abstract setting for the discussion of such issues, and allows for the development of elegant methods, which in turn may influence practical designs and implementations. The problem can be formally stated as follows.

Let P be a set of processors; $T \subset P$ is the set of *faulty* processors, $|T|=t$. Processors from $P-T$ are called *correct.* We assume that the correct processors initially do not know the set T. Every processor i is given an initial value, 0 or 1. After the execution of the protocol, the final values of the correct processors have to satisfy the following two conditions, regardless of the behavior of the faulty processors:

- *Agreement*: they are all equal.
- *Validity*: they are all equal to the initial value, if the latter is unique.

We shall use the simple and standard model of a synchronous network of processors numbered with $\{1, ... ,n\}$. The computation performed by the network evolves as a series of *rounds*, during which the processors send messages, receive them and perform local computations according to the protocol (the computation time in our protocols is always polynomial).

There are three basic aspects contributing to the quality of an algorithm for *Distributed Consensus*: resiliency—the maximum tolerable number t of faulty processors, the number of rounds r, and the maximum message size m (as in [1], we use m to characterize the communication costs; see [6] for another approach). The optima for those quality parameters are respectively $n>3t$ [11], $r=t+1$ [10] and $m=1$.

In the last few years there has been a rapid progress towards higher quality of the protocols (e.g., [1,2,3,6,12]). Protocols are known that are optimal in any two of the aspects, but no protocol is optimal in all three as yet. The first deterministic protocol with optimal number of rounds, linear resiliency ($n>6t$) and polynomial message size was provided by Moses and Waarts [12].

This paper provides two efficient protocols wich further narrow the optimality gap. The first one—*Cloture Vote*—improves on the resiliency of the last result (to $n>4t$). Our method is significantly simpler than their *coordinated traversal* method, and perhaps it provides a better clue for finding a protocol with optimal resiliency. For $n>6t$, *cloture voting* is as efficient as *coordinated traversal*, while the improvement in resiliency is achieved with moderate increase in message size. The method borrows from the well-known parliamentary procedure of cloture votes: weary of long speeches (messages), the senators (processors) may vote for cloture; this way a qualified majority of the senators may swiftly curtail the debate. Similarly as *coordinated traversal, cloture voting* is used in conjuction with the two earlier methods of *fault masking* [5,2] and *early stopping* [9]. Its development was influenced by the *gear shifting* technique of Bar-Noy *et al.* [2].

† This work was partially supported by AFOSR contract 87-0400 and NSF grant CR 8805978.
‡ Current address: CS Department, The Pennsylvania State University.

The *Cloture Vote* protocol, in conjunction with a schema used in [3], yield a family of equally resilient protocols with $t(1+1/d)$ rounds, and $m = O(d^5)$. This tradeoff compares favorably with the exponential (in d) message size in similar protocol family of [5,2].

Even assuming that each communication round has a large cost associated with it, in any practical setting the message size quickly dominates the cost if it is proportional to, say, n^3 or more. Thus it is important to consider protocols with very short messages, even if the number of rounds is not exactly optimal. In [13], Toueg *et al.* achieve optimal resiliency, $r = 2t+1$ and $m = \Omega(\log n)$. [3] contains a protocol with resiliency $n > 4t$, $r = 2(t+1)$ and $m = 1$.

Our second protocol—*Phase King*—improve on both results, having optimal resiliency, $r = 3(t+1)$ and $m = 2$. This protocol also provides a basis for a family of protocols with optimal resiliency, $r = (1+1/d)t$ and $m = C^d$, improving on the similar results of [2]. For $m \sim \log n$ the resulting total number of transferred bits becomes $O(n^2)$, thus matching the known lower bound [8].

The remainder of the paper is organized as follows. Section 2 describes the *Cloture Vote* protocol, Section 3 the *Phase King* protocol and in Section 4 we elaborate on families of protocols with number of rounds versus message size tradeoffs.

2. Cloture voting: t+1-round, polynomial time consensus for n > 4t

2.1. EIG: Exponential information gathering

Below we describe a version of the original Byzantine Agreement algorithm of [11,2]. We reformulate the correctness proof in a way that will allow us to explain three 'tricks' which, applied together, will reduce the message size to a polynomial.

Let $S^{\square}$ denote the set of sequences from S^* without repetitions, and $S_s^{\square}$ the set of sequences from $S^{\square}$ of length at most s. Processors build trees of the following form: $P_{t+1}^{\square}$ is the set of nodes; the root is λ (the empty sequence); the children of σ, $EXT(\sigma)$, are nodes of the form σj. In the proofs we also use the notation $Ext(\sigma) = \{j : \sigma j \in EXT(\sigma)\}$. In each tree, a node σ has a *value* denoted $Val(\sigma)$. A node which has final value v in trees of all correct processors is *v-common*, or just *common*.

In the protocols of this section correct processors communicate using the following two primitives:

- Send(σ): send $Val(\sigma)$ to all other processors.
- Receive(σ,j): create node σj and set $Val(\sigma j)$ to the value received from j about its own node σ (here a processor i also receives a value sent by itself).

The *EIG* protocol is shown in Figure 1 (*CL* is the acronym for *current level*). Below we prove its correctness assuming $n > 3t$.

```
read(initial);
    (* computation of preliminary values *)
create node λ;  Val(λ) := initial;  CL := {λ};
for r := 0 to t begin (* execute t+1 rounds *)
    for σ ∈ CL
        Send(σ);
    for σ∈CL
        for j ∈ Ext(σ)
            Receive(σ,j)
    CL :=   ⋃   EXT(σ)
          σ∈CL
end;
    (* computation of the final values *)
for r := t downto 0 begin
    CL := parents of nodes in CL;
    for σ ∈ CL
        Val(σ) := the majority value of Val(EXT(σ));
end;
write(Val(λ))
```

Fig. 1. Protocol *EIG*: code for processor i.

Lemma 2.1:
a. If all children of a node σ are common, σ is also common.
b. If the majority of children of σ are v-common, so is σ.

The proof follows directly from computing the final values via majority.

Lemma 2.2: Assume $|\sigma| \le t$ and that $i \in Ext(\sigma)$ is a correct processor. Then v is the preliminary value of σ for i if and only if σi is v-common.

Proof: By decreasing induction on $|\sigma|$. For $|\sigma| = t$, in round t processor i sends v via Send(σ), all correct proceesors assign $Val(\sigma i) := v$ in Receive(σ,i); later this value is not modified.

For $|\sigma| < t$, when i sends v via Send(σ), all correct proceesors assign v to $Val(\sigma i)$ in Receive(σ,i); inductively, for every correct $j \in Ext(\sigma i)$ the node σij is v-common. Since the number of such j's is at least $n-2t > t$, Lemma 2.1b applies. □

Lemma 2.3: If i is a correct processor, then all nodes of the form σi are common.

Lemma 2.4: All nodes in $T^{\square}$ (λ included) are common.

Proof: Assume otherwise. Let σ be the longest (lowest) node from $T^{\square}$ which is not common. All children of σ are common: either by Lemma 2.3, or by the assumption. Thus, by Lemma 1a, σ is also common.

Theorem 2.1: Protocol *EIG* satisfies the *agreement* and *validity* properties.

Proof: Agreement is proven in Lemma 2.4. *Validity*: if $initial = v$ for all correct processors, then v is the preliminary value of λ in trees of all correct processors, hence, by Lemma 2.2, for every $i \in P-T$ the node i is v-common, and so is, by Lemma 2.1b, the root λ. □

2.2. The ESFM protocol: EIG with early stopping and fault masking

The entire discussion contained in this subsection is based on the assumption $n > 4t$. The protocol we will describe here (*ESFM*) allows correct processors to reach the same conclusions as *EIG* with lesser amount of communication. This happens because:

1) In many instances the final values of nodes may be predicted before computing all the initial values (thus making the computing of many initial values unnecessary); processors make such predictions through the application of the so-called *Early Stopping* rules.
2) Once a processor discovers that another processor, say j, is faulty (by the *Fault Discovery* rule), it may pretend that all values reported to it by j are 0 (*fault masking*); this increases the number of cases where *early stopping* applies.

To discuss those cases, we borrow notation from epistemic logic. We read $K_i^r\phi$ as 'after round r processor i knows fact ϕ'. We will drop i and r when obvious from the context. Instead of describing data structures, we will provide computationally easy rules allowing to deduce statements of the form $K\phi$. We note that only certain specific ways of inferring knowledge will be allowed. Therefore a statement of the form $\neg K_i^r\phi$ will mean that the rules which we provide do not allow i to deduce ϕ in round r. In particular, in this version of the protocol only initial values of the nodes are explicitely stored; the final values will be implicit from statements of the from K_i^r[σ is v-common].

For every fact ϕ we apply the 'not forgetting' rule

(nf) $K_i^r\phi$ implies $K_i^{r+1}\phi$.

We will also use the 'everybody knows' operator $E^r\phi \equiv \bigwedge_{i\in P-T} K_i^r\phi$. We now provide rules for deducing a fact of the form [σ is v-common]. These rules are valid provided $\sigma \in T^{\square} \cup T^{\square}P$; this is sufficient for our purposes, because we will prove that in *ESFM* processors do not apply this formula to any other nodes.

K_i^r[σ is v-common] if one of the following holds in i's tree at the end of round r:

(a) at least $n-t-|\sigma|$ children of σ have value v;

(b) K_i^r[τ is v-common] for a majority of children τ of σ;

(c) v is the value of σ and node σ is created in round $r=t$.

Below we show important features of K_i^r.

Lemma 2.5: Assume that node $\sigma \in T^{\square}(P-T)$ is created in round $s < r$. Then for some v node σ is v-common and E^r[σ is v-common].

Proof: Node σ is of the form τi, where i is correct. Value v from the claim is the preliminary value of τ in the tree of processor i: node σ is created when i executes Send(τ) and other processors execute Receive(τ,i); then v becames their preliminary value for σ. If $s=t$, rule (c) applies. Otherwise, in the next round of $s+1$ ($s+1 \le r$) all correct processors but i execute Send(σ), thus in every correct tree at least $n-t-1 \ge n-t-|\sigma|$ children of σ have preliminary value v. This way σ is v-common and E^r[σ is v-common] is also true—rule (a). □

As we will see, once a processor knows that σ is v-common, it does not try to establish the final values for the descendants of σ, therefore the question of the form 'is σ v-common' never arises for σ not from $T^{\square} \cup T^{\square}P$.

Lemma 2.6: Let $i \in P-T$ and $\sigma \in T^{\square} \cup T^{\square}P$. Then K_i^r[σ is v-common] implies E^{r+1}[σ is v-common].

Proof: For $\sigma \in T^{\square}(P-T)$, K_i^r[σ is v-common] implies E^r[σ is v-common] by Lemma 2.5.

Consider now $\sigma \in T^{\square}$ created in round $r-1$. If $r=t$, then all children of σ are common and the claim holds. If $r<t$, then σ has at least $n-2t$ children σj such that j is correct and σj has preliminary value v (in the tree of i, but since they are correct, also in all other correct trees); by Lemma 2.5 for such children we have E^{r+1}[σj is v-common], and by rule (b) E^{r+1}[σ is v-common].

For σ created before round $r-1$ the claim follows by induction, and application of rule (b). □

Lemma 2.6 shows that once processor i knows that all processors will establish v as the final value of σ, all other processors will know it after reading the messages of the next round. Thus after sending messages in the next round concerning descendants of σ, i may stop reading and sending any such messages. This is exactly the method of *early stopping* [9].

Fault masking [5] exploits the fact that if the children of σj are created in round r and yet $\neg K_i^r$[σj is common] then by Lemma 2.5 processor i knows that j is faulty. This justifies the 'fault discovery' rule [2]:

(fd) If in i's tree at the end of round r the children of σj exist and for $v=0,1$ there exist $t+1-|\sigma|$ τ's from $EXT(\sigma j)$ such that $Val(\tau)=v$, then K_i^r[j is faulty].

Fault masking means that if K_i^r[j is faulty] then in round r processor i replaces messages from j by 0 (i.e. the default value). This does not disturb the correctness of the algorithm, because *EIG* works regardless of the behavior of the faulty processors.

Assume that the children of σ are created in round r. We say that the node σ is *corrupted* if K_i^r[σ is v-common] holds for no correct i and no v. Fault masking assures that when some σj is corrupted, then no node of the form τj created later than σj may be corrupted.

The *ESFM* protocol (see Figure 2) is run by every correct processor with parameter *Start* = 0, the consensus value is stored as *Outcome*[0]. By Lemmas 2.5 and 2.6, the problem of correctness of *ESFM* reduces to the correctness of *EIG*. Later, *Debate* from Figure 2 is used in our polynomial time algorithm. Note that *NTS* and *NNC* used in that procedure form subsets of *CL* (the current level of the tree) of *EIG*. These acronyms stand respectively for *Nodes To be Sent* and *Nodes Needing Children.* Note that if *NNC* becomes empty at the end of round r, processor i stops (early) communicating with other processors in round $r+1$.

2.3. Cloture Voting

The main idea is that each correct processor runs several processes—called *debates*— concurrently, together with an overhead computation—the *MAIN* process. A new debate is started in each of the $t+1$ (global) rounds. Each debate is an execution of procedure *Debate* (given in the previous subsection) and evolves through rounds of message exchange; we shall use $debate_k$ to refer to the debate started in round k. We assume that in every round there is a single time across processes at which messages are sent and, next, a single time during which messages are received.

The reason for having parallel debates is the following: $debate_0$ should reconcile the original vote. When all correct processors are unanimous (*validity* situation) this debate and all subsequent ones 'die' right away. Otherwise, the message size in the original debate may grow. However, if it does grow then clearly the default can be safely chosen. Thus once a processor is required by the algorithm to send a very long message (not unlike a filibuster speech in the American Senate), it will cast a vote for cloture. Thus in every round a new debate starts—shall the deliberation end? (this is exactly what a parliamentary cloture vote is about). Although the cloture vote itself may be debated, once 'filibuster speeches' exceed certain polynomial size, the cloture vote will be unanimous and the deliberation will actually end.

A high-level sketch of *MAIN* is given in Fig. 3; inter-process synchronization and communication details are left implicit. We note that the scope of facts explicitly involving nodes is naturally local, i.e. restricted to the corresponding debates, whereas facts of the form [processor i is faulty] are globally known, and shared by all debates. Parameter S, the 'patience' treshold, is set to $2n^3$; justification for this particular value is given in Lemma 2.8.

```
procedure Debate(Start,InitialValue);
begin
  create node $\lambda$; $Val(\lambda)$ := InitialValue;
  NTS, NNC := {$\lambda$};
  for r := start to t begin (* execute $t+1$ rounds *)
    for $\sigma \in NTS$
      Send($\sigma$);
    NTS := $\bigcup_{\sigma \in NNC} EXT(\sigma)$;
    for all $\sigma j \in NTS$
      Receive($\sigma$,j);
    for all $\tau j \in NTS$ s.t. $K[j$ is faulty]
      $Val(\tau j)$ := 0; (* masking of the faults *)
    for all $\sigma \in NNC$ and v s.t. $K[\sigma$ is v-common] begin
      $\rho$ := $\sigma$;
      repeat
        $\tau$ := $\rho$;
        if $\tau \neq \lambda$ then
          $\rho$ := the parent of $\tau$
      until $\tau = \lambda$ or $\neg K[\rho$ is v-common];
      $Val(\tau)$ := v;
      remove the descendants of $\tau$ from NNC
    end;
    NNC := $\bigcup_{\sigma \in NNC} EXT(\sigma)$;
  end;
  Outcome[Start] := $Val(\lambda)$
end.
```

```
Round 0 of MAIN:
  start Debate(0,initial);

Round k:
    (* cloture *)
  if $K[\lambda$ is 0-common] for some $debate_l$ then begin
    echo only $debate_l$ (the debate with
        smallest NTS, if many);
    output 0;
    exit
  end;
    (* start this round's debate *)
  if for an ongoing debate |NTS| > S then
    start Debate(k,0) (* vote for cloture *)
  else
    start Debate(k,1);

After round t, output $\underset{0 \le k \le t}{MIN}(Outcome[k])$.
```

Fig. 3. The *Cloture Vote* protocol: code for processor i.

Lemma 2.7: (*Validity*) Assume that all processors start the protocol with the same input value v. Then all correct processors output v.

Proof: Under the assumption, in $debate_0$ we have $E^0[\lambda$ is v-common], and so in round 1 for every correct processor $NNC = \emptyset$. Consequently, no further messages are read in this debate and no evidence is gathered which could indicate that a processor is faulty. Clearly, all correct processors invoke $Debate(1,1)$. Inductively, for $k \ge 1$ we have the same situation for $debate_k$: $E^k[\lambda$ is 1-common] will hold. Therefore the output will equal v for every correct processor. (Actually, the protocol may be modified to have the early stopping property) □

The next lemma provides the crucial reason that makes cloture voting work. When the messages become long, the processors know (*Validity* lemma) that any outcome is permissible, and they may opt for the default value. However, any attempt of this kind is essentially equivalent to the original consensus problem. The danger is that the remaining number of rounds is insufficient to assure such a consensus. As proven in [10], every faulty processor which is undetected (i.e. $j \in T$ for which $\neg E^r[j$ is faulty]) may delay the consensus by one round. Thus processors may 'risk' a nonunanimous vote only if there exists a sufficient number of detected faulty processors.

What is important for *cloture voting* is that as long as the number of detected processors is insufficient to assure that a new nonunanimous vote will lead to consensus, the message size remains bounded by a polynomial. Note that the message size within a debate is proportional to the size of *NTS*, and it is its 'large' size which triggers 'vote for cloture' (see Fig. 3).

Let $F^r = \{j: E^r[j \text{ is faulty}]\}$, i.e. the set of processors universally known to be faulty at round r.

Lemma 2.8: (*Sufficient fault detection*) If a correct processor i starts *Debate*$(r,0)$, then $|F^{r-3}| \geq r$.

Proof: Since i starts *Debate*$(r,0)$, we know that for some debate, $|NTS|$ is at least $2n^3$ at the beginning of round r. Assume first that this is *debate*$_0$. Consider a node from *NTS*, say $j_1...j_r$. We can show that for $1 \leq k \leq r-3$, we have $j_k \in F^k - F^{k-1}$.

Would j_k be in F^{k-1}, then at round $k-1$ every correct processor would set $Val(j_1...j_k) = 0$, in round k $j_1...j_k$ would be removed from *NNC*, and so $j_1...j_r$ would not be a member of *NTS*.

Would $j_k \notin F^k$, $K_l^k[j_1...j_k$ is v-common] would hold for some correct processor l, which implies (Lemma 2.6) $E_l^{k+1}[j_1...j_k$ is v-common]. Consequently, in round $k+1$ all correct processors (i in particular) remove descendants of $j_1...j_k$ from *NNC*, and after round $k+2$ *NTS* would not contain descendants of $j_1...j_k$ either. Therefore, at the beginning of round $r \geq k+3$ $j_1...j_k$ cannot be a member of *NTS*, a contradiction.

Let $a_k = |F^k - F^{k-1}|$. As we see, for $k \leq r-3$ we have $a_k \geq 1$. The number of possible $j_1...j_{r-3}$ is at most $\prod_{k=1}^{r-3} a_k$. Would it happen that $|F^{r-3}| = \sum_1^{r-3} a_k \leq r$, then the number of possible $j_1...j_{r-3}$ would be at most 8 (basic arithmetic), thus the number of possible $j_1...j_{r-2}$ would be at most $8t < 2n$ and the number of possible $j_1...j_r$ would be less than $2n^3$. This finishes the case $|NTS| \geq 2n^3$ for *debate*$_0$.

Now assume that $|NTS| \geq 2n^3$ for *debate*$_l$, $l > 0$. Of course, a correct processor had to start this debate as *Debate*$(l,0)$, otherwise it would 'die' in two rounds. Inductively, it means that $|F^{l-1}| \geq l$. By reasoning identical to the one given for *debate*$_0$, $|F^{r-3} - F^l| \geq r-l$, hence the claim. □

Lemma 2.9: (*Agreement*) All correct processors output the same value after executing the protocol.

Proof:

1. Assume first that in a round $r < t$ a correct processor i reaches the conclusion $K_i^r[\lambda$ is 0-common] for some debate. Let r be the first round that this happens. Then for any other correct processor h we have $K_h^{r+1}[\lambda$ is 0-common] for the same debate (or some other, if two such situations occured simultaneously). Thus all correct processors output 0.
2. In the remaining case we may assume that no correct processor interrupts its computation early due to the signal from *MAIN*. If for some k all correct processors start *Debate*$(k,1)$, then they reach unanimously *Outcome*$[k] = 1$. On the other hand, if at least one correct processor starts *Debate*$(k,0)$, then $E^{k-1}[j$ is faulty] is true for at least k processors (*Sufficient fault detection* and (nf) rule). Therefore *debate*$_k$ runs through $t+1-k$ rounds and only $t-k$ faulty processors have their messages read; this is an instance of *ESFM*. This allows us to prove that *Outcome*$[k]$ is unanimous. □

Lemma 2.10: (*Message size*) The maximal message size of the *Cloture Vote* protocol is $O(n^5)$.

Proof: Assume that at the beginning of round r, in some *debate*$_k$ a correct processor has $|NTS| > 2n^4$. For every $\sigma j \in NTS$ this processor was executing Receive(σ,j) in round $r-1$. The correctness argument of *ESFM* is that all messages which are needed are actually sent. Thus for every correct processor σ was in *NTS* at the beginning of round $r-1$, consequently for all correct processors $|NTS| > 2n^3$ at the beginning of round $r-1$, and all of them started *Debate*$(r-1,0)$ and the cloture is applied in round r.

Since no *NTS* which is actually sent has size exceeding $2n^4$, the total message size is $O(n^5)$ (actually, $O(n^3t^2)$).□

Lemmas 2.7, 2.9 and 2.10 yield the following:

Theorem 2.2: The *Cloture Vote* protocol solves the Distributed Consensus problem for $n > 4t$, in $t+1$ rounds using messages of size $O(n^5)$ (indeed, at most n^5).

Remark 2.1: The protocol can be modified to satisfy the early stopping property, allowing all correct processors to find the final outcome in $MIN\{t+1, f+2\}$ rounds, where f is the actual number of processors that fail in the run.

Remark 2.2: $n > 6t$. In this case it is sufficient to set the 'patience' threshold to n^2, which allows for messages n times shorter (somewhat shorter than in the more complicated *Coordinated Traversal* technique of Moses and Waarts [12]).

3. Constant message size with optimal resiliency

```
$V := v_i$; (* $i$'s initial value *)
for $m := 1$ to $t+1$ begin
        (* Exchange 1 *)
    send($V$);
    $V := 2$;
    for $k := 0$ to 1 do begin
        $C(k)$ := the number of received $k$'s;
        if $C(k) \geq n-t$ then $V := k$
    end;
        (* Exchange 2 *)
    send($V$);
    for $k := 2$ downto 0 do begin
        $D(k)$ := the number of received $k$'s;
        if $D(k) > t$ then $V := k$
    end;
        (* Exchange 3 *)
    if $m = i$ then
        send($V$);
    if $V = 2$ or $D(V) < n-t$ then
        $V := MIN$(1,received message);
end;
```

Fig. 4. The *Phase King* protocol: code for processor i.

In this section we assume $n > 3t$. The *Phase King* protocol is shown in Figure 4. As in [3], this is an asymptotically optimal protocol for Distributed Consensus in all the cost measures, but improves (to optimal) the resiliency of the one given there. It uses messages of constant size (2 bits), and runs in $t+1$ phases, each consisting of three exchange rounds. Each processor has a local variable V, and integer arrays C and D. Value 2 represents 'undecided', whereas 1 is used as the default value.

Theorem 3.1: The *Phase King* protocol solves the Distributed Consensus problem in $3(t+1)$ rounds using messages from $\{0,1,2\}$ and optimal number of processors.

Proof: First note that the following statements are obvious after inspecting Fig. 4:

- At the end of Exchange 1 there exists $v \in \{0,1\}$ such that, for all correct processors, the value of V is v or 2.
- At the end of Exchange 2, for all correct processors the value of V is v or 2 (same v as above).
- If at the beginning of a given phase the value of V is equal to v for all correct processors, the same is true at the end of this phase. (*Persistency of agreement*)

Now let g be the smallest number of a correct processor. At the end of Exchange 3 of phase g agreement is reached, because either

a) all correct processorss accept the phase king's message (and thus reach consensus), or else

b) some correct processors ignore this message because they find $D(V) < n-t$ false. But then $D(V) > t$ for all correct processors, and so their values of V are equal after Exchange 3 whether they accept the king's message or not.

Since $g \le t+1$, due to *Persistency* agreement exists at the end of the last phase. *Validity* also follows from *Persistency*. □

Remark 3.1: In Exchange 3 the king can just transmit the default value instead of "2". This leads to an $m = 1$, $r = 4t+4$ protocol.

4. Number of rounds versus message size tradeoffs

Both in the *Phase King* protocol presented above and in the similar *Single Bit* protocol of [3], the number of phases can be reduced by replacing the phase kings by disjoint committees of processors. Internally, the committees run an efficient round-optimal consensus protocol. Details, to be provided in the full version, can be found in [3].

If in the *Single Bit* protocol ($n > 4t$), the committees run the *Cloture Vote* protocol presented in this paper, for committees of size $4d-3$ we obtain consensus in t/d phases, each with $d+1$ rounds, and messages of size $O(d^5)$.

In [14], Waarts describes early stopping for the case of optimal resiliency. This approach is round-optimal, and in conjunction with fault masking yields a $O(n^3 2.1^t)$ message size. To reduce the number of rounds of the *Phase King* protocol to $t(1+1/d)$, we replace the phase kings by committees of size $6d-2$. The resulting message size is $O(d^3 4.33^d)$.

Remark 4.1: (*Optimal total bit transfer*) By having merely 2 phases (and committtees), and the committees running the same protocol recursively, we obtain $n = 3t+1$, $r = t+o(t)$ and total number of transferred bits $O(t^2)$. Details can be found in [4]. Independently, the optimal bit transfer has also been obtained by Coan and Welch [7].

References

[1] A. Bar-Noy and D. Dolev, "Families of Consensus Algorithms," *Proc. 3rd. Aegean Workshop on Computing*, June/July 1988, pp. 380-390.

[2] A. Bar-Noy, D. Dolev, C. Dwork and H.R. Strong, "Shifting gears: changing algorithms on the fly to expedite Byzantine Agreement," *Proc. 6th PODC*, pp. 42-51, August 1987.

[3] P. Berman and J.A. Garay, "Asymptotically Optimal Distributed Consensus," *Proc. ICALP 89*, Lecture Notes in Computer Science, Vol. 372, pp. 80-94.

[4] P. Berman, J.A. Garay and K.J. Perry, "Recursive Phase King Protocols for Distributed Consensus," The Pennsylvania State University, Computer Science Dept. Tech. Report *CS-89-24*, August 1989.

[5] B. Coan, "A communication-efficient canonical form for fault-tolerant distributed protocols," *Proc. 5th PODC*, pp. 63-72, August 1986.

[6] B. Coan and J. Welch, "Modular Construction of Nearly Optimal Byzantine Agreement Protocols," *Proc. 9th PODC*, August 1989.

[7] B. Coan and J. Welch, personal communication.

[8] D. Dolev and R. Reischuk, "Bounds of Information Exchange for Byzantine Agreement," *JACM*, Vol. 32, No. 1, (1985), pp. 191-204.

[9] D. Dolev, R. Reischuk and H.R. Strong, "Early Stopping in Byzantine Agreement," *IBM Research Report RJ5406-55357*, 1986.

[10] D. Dolev and H.R.Strong, "Polynomial Algorithms for Multiple Processor Agreement," *Proc. 14th STOC*, pp. 401-407, May 1982.

[11] L. Lamport, R.E. Shostak and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, Vol. 4, No. 3, pp. 382-401, July 1982.

[12] Y. Moses and O. Waarts, "Coordinated Traversal: $(t+1)$-Round Byzantine Agreement in Polynomial Time," *Proc. 29th FOCS*, pp. 246-255, October 1988.

[13] S. Toueg, K.J. Perry and T.K. Srikanth, "Fast Distributed Agreement," *SIAM Journal on Computing*, Vol. 16, No. 3, pp. 445-458, June 1987.

[14] O. Waarts, "Coordinated Traversal: Byzantine Agreement in polynomial time," M.Sc. Thesis, Weizmann Institute of Science, Rehovot, Israel, August 1988.