Results on a Scalable Byzantine Agreement.

Olumuyiwa Oluwasanmi, University of New Mexico ,
Joint work with
Jared Saia University of New Mexico
Valerie King University of Victoria

Motivation

- *"Unfortunately, Byzantine agreement requires a number of messages quadratic in the number of participants, so it is infeasible for use in synchronizing a large number of replicas" [REGWZK, '03].*
- *"Eventually batching cannot compensate for the quadratic number of messages [of Practical Byzantine Fault Tolerance (PBFT)]" [CMLRS, '05].*
- *"The communication overhead of Byzantine Agreement is inherently large" [CWL '09] .*

Motivation

Why Byzantine Agreement is important:

- Tells us how to build a reliable system from unreliable components.

Byzantine Agreement

- Each processor starts with an initial input bit.
- Each good processor outputs the same bit *b*, this bit must equal one of the input bits.
- A hidden subset of n/3 processors are bad and they may behave arbitrarily.

Leader Election

Another related problem we consider here is Leader Election:

- Some good processor p is elected as the leader and is known by all processors.
- With constant probability, p is good (there is no way to elect a good processor with certainty when there are bad processors).

Universe Reduction

A generalisation of Leader Election:

- Some set C (of size $O(\log^3 n)$) is elected and known by all processors.
- With high probability $(1 - o(1))$, C is good i.e . a majority of the processors in C are good.

Our Results

Theory: Can Solve Byzantine Agreement(Leader Election and Universe Reduction) with each processor sending $\tilde{O}(\sqrt{n})$ bits [KS, '09].

Practice: Significant improvements in bandwidth starting at about 16k processors.[This talk]

Our Approach

1. Almost Everywhere Universe Reduction: There is a set C of size $O(\log^3 n)$ that is good and is known by a $1 - \epsilon$ fraction of good processors.
2. Almost Everywhere to Everywhere:
   - C does B.A.
   - Everybody knows C and C's output.

Universe Reduction

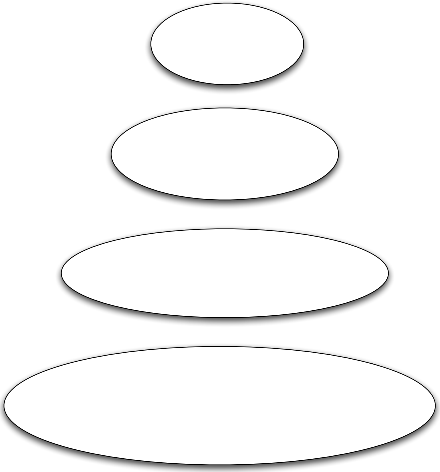
Problem

- The challenge? Our adversary can insert a greater than expected fraction of bad processors in the subset selected. Solution:Use randomness to select the processors.
- How to do this by avoiding sending $O(n)$ messages per processor. Solution:Use election graph to elect this subset.

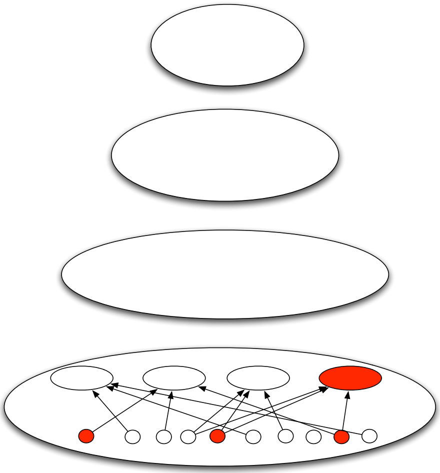
Almost Everywhere Universe Reduction

- We can reduce message complexity by using an election graph [KSSV, 06,07].
- The nodes in this graph are groups of $O(\ln n)$ processors called committees.
- The election proceeds in layers.

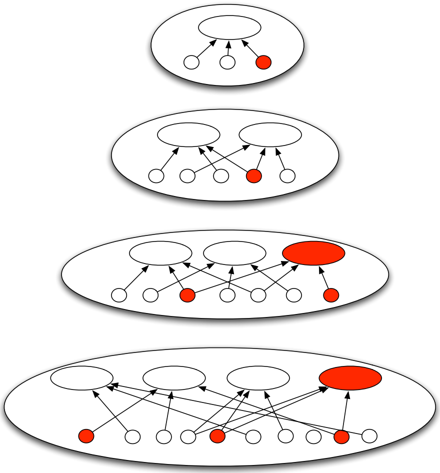
Election Graph



Election Graph



Election Graph



The Election graph.

- Initially at layer 0 we have the leaves of the election graph.
- Each node in the election graph elects a committee of $O(\log n)$ size selected from the nodes below.

Gains of using cryptography

New Ideas: We use [AS, 2006] to elect random processors within committees of size $\Theta(\ln n)$.

- Can reduce committee size of $\Theta(\ln n)$ from [KSSV,05].
- This reduces message complexity at each layer.

Election Scheme.

- Select the processors to advance by running the [AS, 2006] algorithm within each committee.
- A committee is good if >2/3 of the processors are good.
- Use samplers to spread out bad processors so that with high probability (probability $1 - 1/n^c$ where c is a constant and $c > 0$) most of the committees in the next layer of elections are good.

Result

At the end of the A.E. protocol:

- There is a set C of size $O(\log^3 n)$ with a 2/3 fraction of good processors.
- Each processor p, has a guess C_p for C.
- For a majority of good processors p, $C_p = C$

Next step: Ensure everyone knows C.

Almost Everywhere to Everywhere.

Goal: Ensure everyone knows C.

- Idea: Each processor polls $O(\log n)$ processors.
- Problem: Spam! Bad processors send spurious requests.
- Idea: Polling requests sent through C, shich enforces few requests per processor
- Problem: Not everyone knows C!

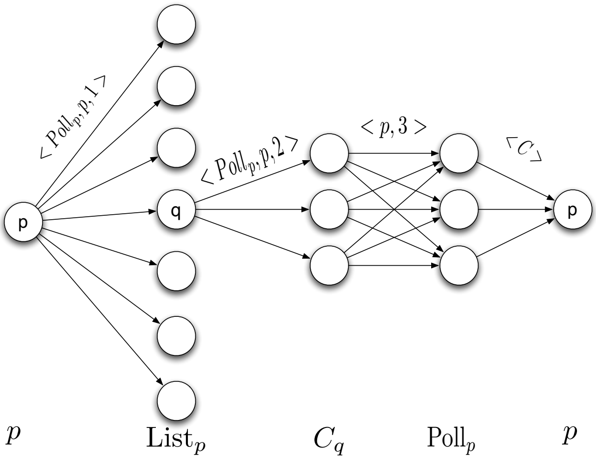
AE2E contd.

- Idea: p sends it's poll (of $O(\log n)$) to $O(\sqrt{n})$ randomly selected processors. Hopefully, someone in this set will forward the poll to C.
- Each processor only forwards messages received from a set of $O(\sqrt{n}\log^2 n)$ random processors.
- Birthday paradox ensures some processor will forward p's poll.

AE2E contd.

- C forwards p's poll to the appropriate processors.
- A processor answers a requests that it receives from a majority of C's members.

Sketch of communication flow in AE2E.



AE2E contd

- Problem: If a confused processor thinks it is in C, it will send many messages.
- Solution: Protocol starts with a check to see if a processor is in C.

Theorem

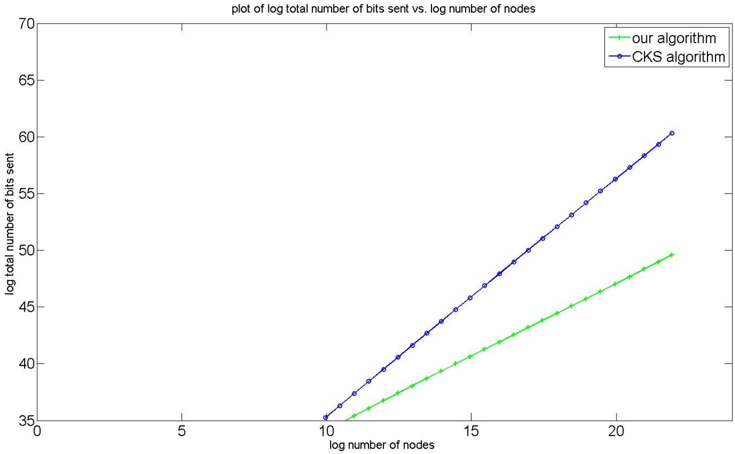
Our algorithm has the following properties:

- With high probability all of the good processors learn the value of the bit.
- Each processor sends $O(\sqrt{n}\log^2 n)$ messages.

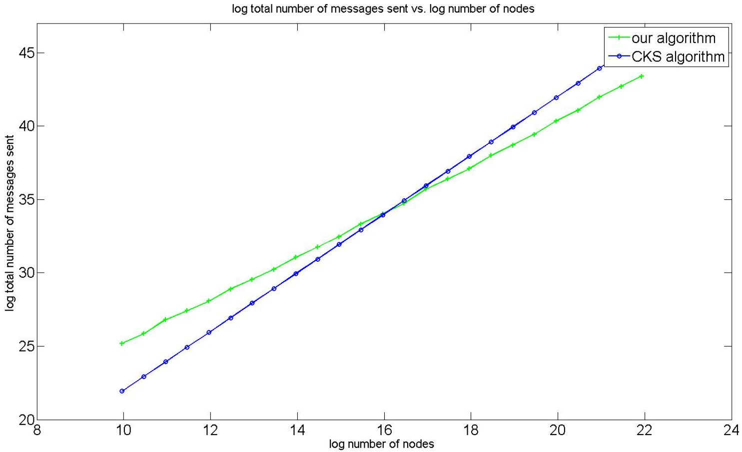
Experiments

- We performed a simulation of our algorithm for n from 1000 to about 4,000,000 processors.
- Compared with CKS algorithm which uses cryptography.
- Measured bandwidth and latency.

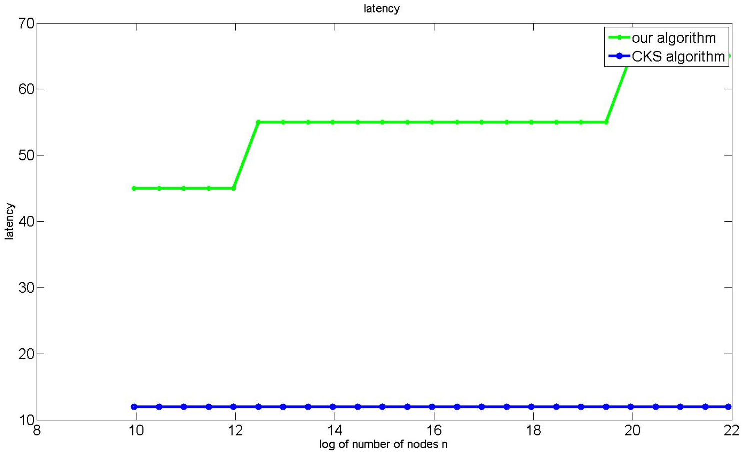
Log log plot total bits sent



Log log plot total messages sent



Latency


latency
our algorithm
CKS algorithm
latency
70
60
50
40
30
20
10
8
10
12
14
16
18
20
22
log of number of nodes n

Conclusion

1. Can Solve BA(and Universe Reduction) with $\tilde{O}\sqrt{n}$ bits communication per processor.
2. Practical improvement on networks of size about 16k nodes.

Future directions

- Further reduce message complexity?
- Use a sparse communication network?
- More realistic simulations?
- Handle the asynchronous case?

Future work

Less is more:

- Further reduce message complexity to $O(\log^2 n)$ per processor.
- Ideas : Better algorithm for choosing a random peer, running elections recursively.

Sparse communication network

Want:

- A sparse communication network is more practical.
- Need communication network with lots of vertex disjoint nodes, so routing messages can be fault tolerant.

Detailed simulation

- Some p2p networks could be as large as ten million nodes.
- Simulate on a cluster, as this more closely simulates real world conditions.

Asynchronous case.

- The asynchronous communication is a more realistic model of network communication.
- Can we make the algorithm asynchronous and keep the bandwidth bounds on the algorithm the same?

Questions

- Questions ?