Weighted Byzantine Agreement

Vijay K. Garg John Bridgman

Parallel and Distributed Systems Lab at The University of Texas at Austin

IPDPS 2011

Byzantine Agreement

- Introduced by Lamport, Shostak and Pease 1980
- Model:
  - n processes
  - f byzantine faults
  - Synchronous system

Byzantine Agreement Requirements

- **Agreement**: Two correct processes cannot decide on different values.

Byzantine Agreement Requirements

- **Agreement**: Two correct processes cannot decide on different values.

	P_0	P_1	P_2	...	P_n
Input	0	1	0	...	1
Good Output	1	1	1	...	1
Bad Output	0	1	0	...	1

Byzantine Agreement Requirements

- **Agreement**: Two correct processes cannot decide on different values.
- **Validity**: The value decided must be proposed by some correct process.

Byzantine Agreement Requirements

- **Agreement**: Two correct processes cannot decide on different values.
- **Validity**: The value decided must be proposed by some correct process.

	P_0	P_1	P_2	...	P_n
Input	1	1	1	...	1
Good Output	1	1	1	...	1
Bad Output	0	0	0	...	0

Byzantine Agreement Requirements

- **Agreement**: Two correct processes cannot decide on different values.
- **Validity**: The value decided must be proposed by some correct process.
- **Termination**: All correct processes decide in finite number of steps.

Byzantine Agreement Lower Bounds

- $n \geq 3f + 1$
- Given by Lamport, Shostak, Pease 1980

Byzantine Agreement Lower Bounds

- $n \geq 3f + 1$
- Given by Lamport, Shostak, Pease 1980
- What if we have 30 processes where 15 of them can fail?

Byzantine Agreement Lower Bounds

- $n \geq 3f + 1$
- Given by Lamport, Shostak, Pease 1980
- What if we have 30 processes where 15 of them can fail?

- $f + 1$ rounds worst case
- Given by Fischer and Lynch 1982

Byzantine Agreement Lower Bounds

- $n \geq 3f + 1$
- Given by Lamport, Shostak, Pease 1980
- What if we have 30 processes where 15 of them can fail?
- $f + 1$ rounds worst case
- Given by Fischer and Lynch 1982
- Can we design a protocol that under certain assumptions can beat these?

Weight Motivation

1. Abstract notion of trust
2. Support multiple classes of processes
3. Beat bounds under certain conditions

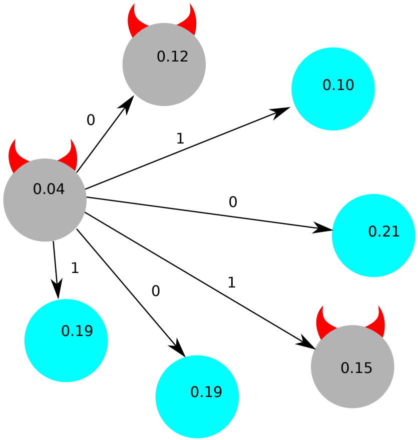
WBA Problem Specification

- Common weight vector, w
- Weight of failed processes no more than ρ
- Must satisfy:
  - Agreement
  - Validity
  - Termination

WBA Lower Bounds

Let α_ρ be the minimum number of processes whose weight exceeds ρ then

- α_ρ rounds
- $\rho < 1/3$



Outline

1 Introduction

2 Algorithms
- Weighted-Queen Algorithm
- Weighted-King Algorithm

3 Initial Weight Assignment

4 Updating Weights

5 Conclusions

Weighted Byzantine Algorithm Examples

- Two algorithms: Weighted Queen and Weighted King
- These have good properties
  - $\leq f + 1$ phases
  - Any failure combination so long as weight $< \rho$

The Weighted-Queen Algorithm

- Based on Phase Queen given by Berman and Garay 1989

	Phase Queen (original)	Weighted Queen (ours)
Fault tolerance	$f < n/4$	$\rho < 1/4$
Rounds	$2(f+1)$	$2\alpha_\rho$

The Weighted-Queen Algorithm

- Based on Phase Queen given by Berman and Garay 1989

	Phase Queen (original)	Weighted Queen (ours)
Fault tolerance	$f < n/4$	$\rho < 1/4$
Rounds	$2(f+1)$	$2\alpha_\rho$

$$\alpha_\rho \leq f + 1$$

The Weighted-Queen Algorithm

For α_ρ phases iterating over the processes starting with highest weight to lowest do:

- First round
  - Exchange own value, v, with everyone
  - Set v to the value with the highest weight
  - Set *supp* to the weight of v
- Second round
  - Queen broadcasts its value
  - If $supp \leq 3/4$, set v to the queen's value

Output own value

Weighted-Queen Example

- Example: 7 processes with weight assignment $[0.2, 0.2, 0.12, 0.12, 0.4, 0.12, 0.12]$
- Standard algorithm: 1 fault only, Weighted: some 2 faults
- For example, processes 0 and 4 can fail together

Weighted-Queen Example

- Phase 1, Round 1:

0

w[0]: 0.20

1

w[1]: 0.20
v: 1

2

w[2]: 0.12
v: 0

3

w[3]: 0.12
v: 1

4

w[4]: 0.04

5

w[5]: 0.12
v: 1

6

w[6]: 0.12
v: 0

Weighted-Queen Example

- Phase 1, Round 1:

0

1
0: 0.12
1: 0.88

2
0: 0.52
1: 0.48

3
0: 0.52
1: 0.48

4

5
0: 0.12
1: 0.88

6
0: 0.52
1: 0.48

Weighted-Queen Example

- Phase 1, Round 1:

0

1
v: 1
supp: 0.88

2
v: 0
supp: 0.52

3
v: 0
supp: 0.52

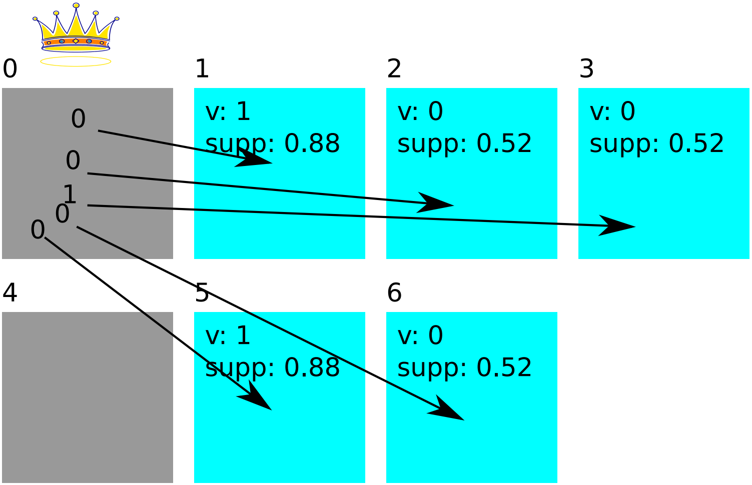
4

5
v: 1
supp: 0.88

6
v: 0
supp: 0.52

Weighted-Queen Example

- Phase 1, Round 2:



Weighted-Queen Example

- Phase 1, Round 2:

0

1
v: 1

2
v: 0

3
v: 1

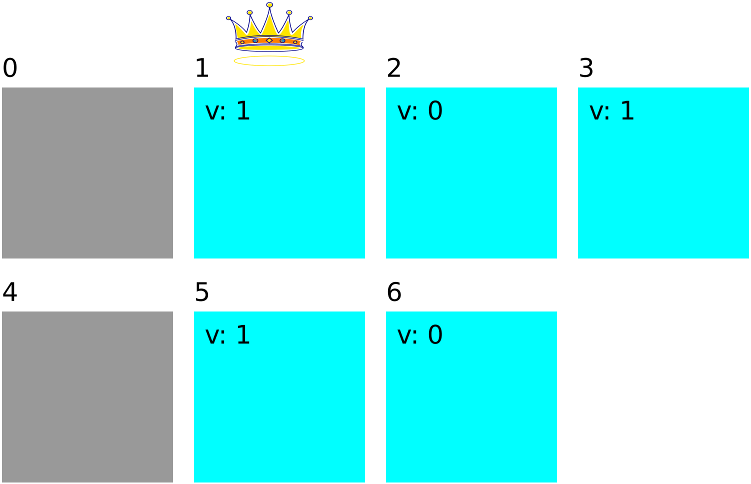
4

5
v: 1

6
v: 0

Weighted-Queen Example

- Phase 2, Round 1:



Weighted-Queen Example

- Phase 2, Round 1:

0

1
v: 1
supp: 0.88

2
v: 0
supp: 0.52

3
v: 0
supp: 0.52

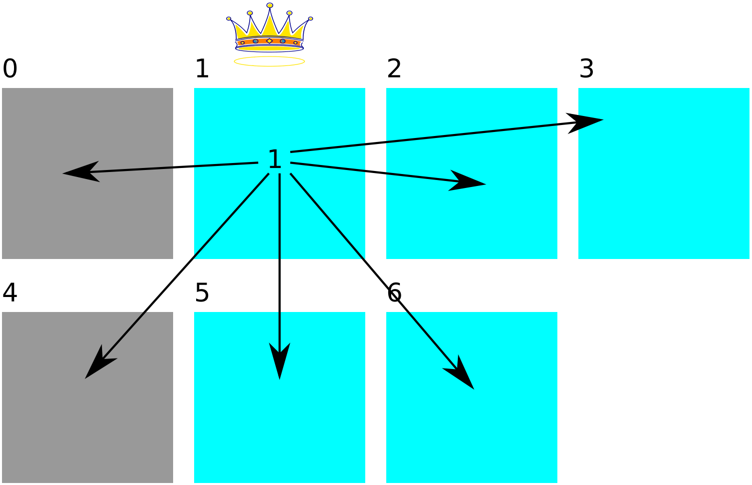
4

5
v: 1
supp: 0.88

6
v: 0
supp: 0.52

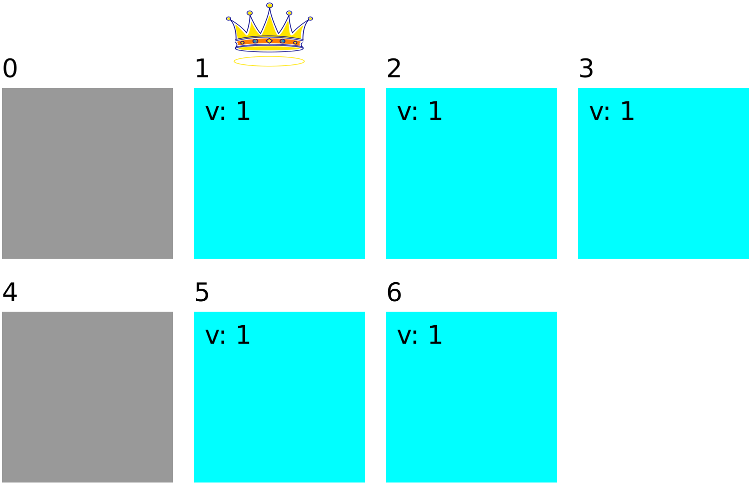
Weighted-Queen Example

- Phase 2, Round 2:



Weighted-Queen Example

- Phase 2, Round 2:



Persistence of Agreement

Lemma (Persistence of Agreement)

Assuming $\rho < 1/4$, if all correct processes prefer a value v at the beginning of a round; then, they continue to do so at the end of the round.

At Least One Correct Queen

Lemma

There is at least one round among the first α_ρ rounds in which the queen is correct.

Weighted-Queen Satisfies the WBA Problem

Theorem

The Weighted-Queen Algorithm solves the agreement problem for all $\rho < 1/4$.

Weighted-King Algorithm

- Three round algorithm based on algorithm given by Berman, Garay and Perry 1989

	Phase King (orig.)	Weighted King (ours)
Fault tolerance	$f < n/3$	$\rho < 1/3$
Rounds	$3(f+1)$	$3\alpha_\rho$

Initial Weight Assignment

- Weight assignment dramatically changes the nature of these algorithms.
- Simple examples:
  - $[1/n, 1/n, \ldots, 1/n]$
  - $[1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 0, \ldots, 0]$
  - $[1, 0, 0, \ldots, 0]$

Initial Weight Assignment

- Weight assignment dramatically changes the nature of these algorithms.
- Simple examples:
  - $[1/n, 1/n, \ldots, 1/n]$
  - $[1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 0, \ldots, 0]$
  - $[1, 0, 0, \ldots, 0]$
- A more involved example with two sets of processes:
  - Set A is a collection of six highly reliable processes with probability of failure $f_a = 0.1$.
  - Set B is a collection of unreliable processes with probability of failure $f_b = 0.3$.

Initial Weight Assignment Policies

- Uniform (Same as regular Byzantine Agreement)
- All weight to set A
- $w[i] \propto 1 - Pr\{P_i \text{ fails}\}$
- $w[i] \propto \frac{1}{Pr\{P_i \text{ fails}\}}$

Updating Weights

Can we update weights?
Some issues with updating weights:

- Weight vector at each process must be the same
- Each process may see different views of what other have sent

Weight Update Algorithm

- A simple solution of agreeing on weights
- Process can detect a faulty process j if:
  - j sends a no message or corrupted message
  - j is queen, queen value is different from v and $supp > 3/4$
- After detect can reduce the weight of the process
- Have to be careful, faulty process can claim good process faulty

Weight Update Algorithm

- Round one
  - Broadcast *faultySet*
  - For each process j that is suspected by some process if the weight of all processes that suspect is greater than ρ then add j to *faultySet*
- Round two
  - Use WBA to agree upon *faultySet*
  - Add to *consensusFaulty* each one agreed to be faulty
- Round three
  - Set the weight of processes in *consensusFaulty* to 0 and renormalise

Weighted Versus Unweighted

- Pros:
  - Simple
  - Can tolerate more than $n/3$ faults in certain circumstances
  - Always $\leq f + 1$ rounds
- Cons:
  - Even with fewer than $n/3$ faulty processes the algorithm may not work in some cases

Future Work

- Better update methods
- Approximately identical weight vectors