Université de Strasbourg

Consensus in Systems with Eventual Quiescence

Brief Announcement

Quentin Bramas

SSS2025, October 11th, 2025

Motivation

Motivation

Paper: *"Reaching Consensus in the Presence of Contention-Related Crash Failures"*
by Anaïs Durand, Michel Raynal & Gadi Taubenfeld, SSS 2022

Motivation

Paper: *"Reaching Consensus in the Presence of Contention-Related Crash Failures"*
by Anaïs Durand, Michel Raynal & Gadi Taubenfeld, SSS 2022

Definition: λ-constrained crash failures are crash faults that can occur only before λ processes have started their execution.

Motivation

Paper: *"Reaching Consensus in the Presence of Contention-Related Crash Failures"*
by Anaïs Durand, Michel Raynal & Gadi Taubenfeld, SSS 2022

Definition: λ-constrained crash failures are crash faults that can occur only before λ processes have started their execution.

Result:
Algorithm solving consensus tolerating up to k λ-constrained crashes, when $\lambda = n - k$, using objects with consensus number k

Consensus number of an object O is the maximal number of processes for which consensus can be solved despite any number of process crashes (occurring at any time) with any number of objects O and read/write registers

Motivation

Paper: *"Reaching Consensus in the Presence of Contention-Related Crash Failures"*
by Anaïs Durand, Michel Raynal & Gadi Taubenfeld, SSS 2022

Definition: λ-constrained crash failures are crash faults that can occur only before λ processes have started their execution.

Result:
Algorithm solving consensus tolerating up to k λ-constrained crashes, when $\lambda = n - k$, ~~using objects with consensus number k~~

Consensus number of an object O is the maximal number of processes for which consensus can be solved despite any number of process crashes (occurring at any time) with any number of objects O and read/write registers

Motivation

Paper: *"Reaching Consensus in the Presence of Contention-Related Crash Failures"*
by Anaïs Durand, Michel Raynal & Gadi Taubenfeld, SSS 2022

Definition: λ-constrained crash failures are crash faults that can occur only before λ processes have started their execution.

Result:

Simple Algorithm solving consensus tolerating up to k λ-constrained crashes, when $\lambda = n - k$, ~~using objects with consensus number k~~

Consensus number of an object O is the maximal number of processes for which consensus can be solved despite any number of process crashes (occurring at any time) with any number of objects O and read/write registers

How ?

Let 0 < k < n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

How ?

Let 0 < k < n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray: 

TrustedArray: 

How ?

Let 0 < k < n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray: 

TrustedArray: 

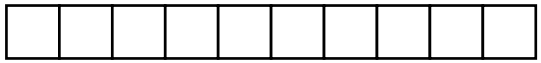
- Say “hello” *(write 1 in my “hello” register)*

How ?

Let 0 < k < n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray: 

TrustedArray: 

- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”

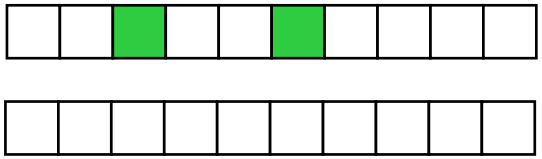
How ?

Let 0 < k < n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray:

TrustedArray:



- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”

How ?

Let 0 < k < n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray: 

TrustedArray: 

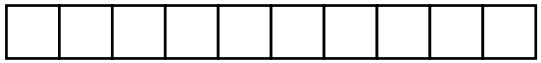
- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”

How ?

Let $0 < k <$ n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray:  $n - 3 \geq n - k$

TrustedArray: 

- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”

How ?

Let $0 < k <$ n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray:  $n - 3 \geq n - k$

TrustedArray: 

- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”
- I know that no more crash can occur

How ?

Let $0 < k <$ n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray:  $n - 3 \geq n - k$

TrustedArray: 

- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”
- I know that no more crash can occur
- Say “I am correct” *(write 1 in my “trusted” register)*

How ?

Let $0 < k <$ n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray:  $n - 3 \geq n - k$

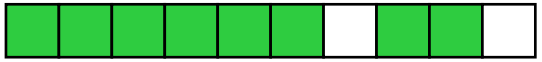
TrustedArray: 

- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”
- I know that no more crash can occur
- Say “I am correct” *(write 1 in my “trusted” register)*

How ?

Let $0 < k <$ n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray:  $n - 3 \geq n - k$

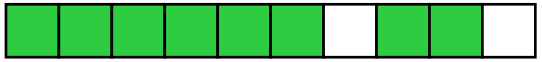
TrustedArray: 

- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”
- I know that no more crash can occur
- Say “I am correct” *(write 1 in my “trusted” register)*

How ?

Let $0 < k <$ n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ *at most* $k = 4$ *crashes*

HelloArray:  $n - 3 \geq n - k$

TrustedArray: 

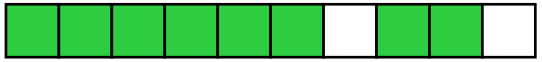
- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”
- I know that no more crash can occur
- Say “I am correct” *(write 1 in my “trusted” register)*
- Now the TrustedArray is an **eventual perfect failure detector**[1]

[1] Eventually: every process that crashes is permanently suspected and correct processes are not suspected

How ?

Let $0 < k <$ n, we want to solve the consensus with at most k $(n-k)$-constrained crashes.

Example with $n = 10$ at most $k = 4$ crashes

HelloArray:  $n - 3 \geq n - k$

TrustedArray: 

- Say “hello” *(write 1 in my “hello” register)*
- Wait to see at least $n - k$ “hello”
- I know that no more crash can occur
- Say “I am correct” *(write 1 in my “trusted” register)*
- Now the TrustedArray is an **eventual perfect failure detector**[1]
- Execute **Lo & Hadzilacos** consensus algorithm with it

[1] Eventually: every process that crashes is permanently suspected and correct processes are not suspected

A general algorithm

A general algorithm

If the system is **eventually quiescent**

(I eventually know when no more crash can occur),

A general algorithm

If the system is **eventually quiescent**

(I eventually know when no more crash can occur),

we can solve consensus:

A general algorithm

If the system is **eventually quiescent**

(I eventually know when no more crash can occur),

we can solve consensus:

- When I know that no more crash can occur
- Write *true* in my “trusted” register
- Now the TrustedArray is an eventual perfect failure detector
- Execute **Lo & Hadzilacos** consensus algorithm with it

A general algorithm

If the system is **eventually quiescent**

(I eventually know when no more crash can occur),

we can solve consensus:

- When I know that no more crash can occur
- Write *true* in my “trusted” register
- Now the TrustedArray is an eventual perfect failure detector
- Execute **Lo & Hadzilacos** consensus algorithm with it

This could be applied to other kind of failures, not only contention-related.

A general algorithm

If the system is **eventually quiescent**

(I eventually know when no more crash can occur),

we can solve consensus:

- When I know that no more crash can occur
- Write *true* in my "trusted" register
- Now the TrustedArray is an eventual perfect failure detector
- Execute **Lo & Hadzilacos** consensus algorithm with it

This could be applied to other kind of failures, not only contention-related.

Thank you!