

# Perfect Failure Detection with Very Few Bits

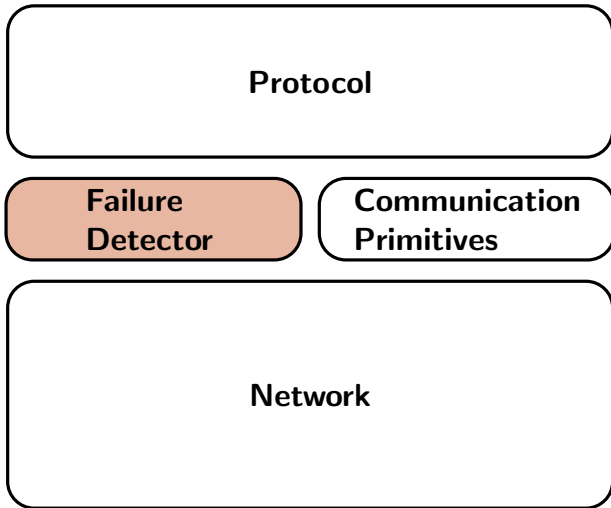
Pierre Fraigniaud <sup>1</sup> Sergio Rajsbaum <sup>2</sup> **C. Travers**<sup>3</sup>  
Petr Kuznetsov <sup>4</sup> Thibault Rieutord <sup>4</sup>

<sup>1</sup>IRIF, Paris    <sup>2</sup>UNAM, Mexico    <sup>3</sup>LaBRI, Bordeaux  
<sup>4</sup>ParisTech, Paris

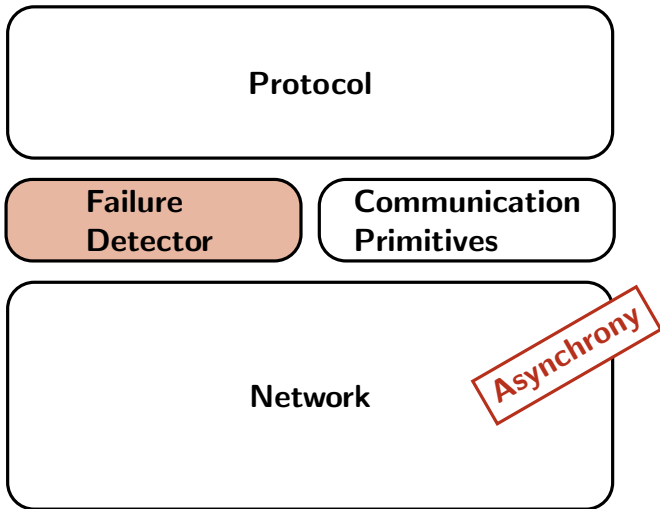
ANR Descartes, Chasseneuil, Octobre 2017

- Distributed device
- Give (unreliable) information on failures

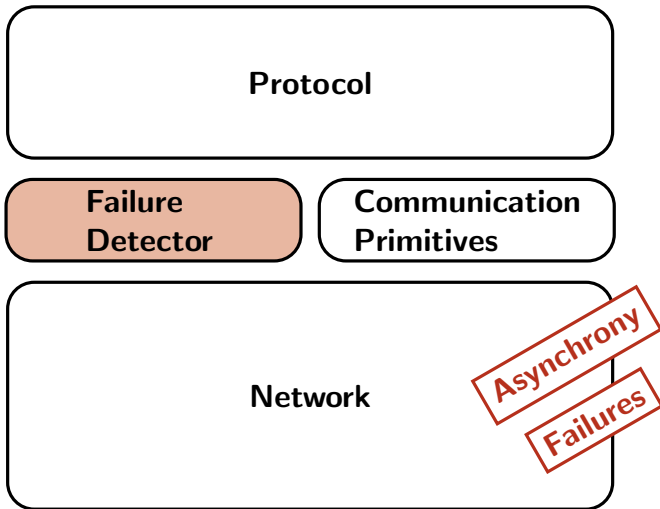
# Modular Distributed Computing



# Modular Distributed Computing



# Modular Distributed Computing



# Relative Hardness of Distributed Task

Failure detector  $D$  is the weakest for task  $T$



- 1 There is a protocol for  $T$  using  $D$
- 2 Any f.d.  $D'$  that can be used to solve  $T$  can emulate  $D$

# Relative Hardness of Distributed Task

Failure detector  $D$  is the weakest for task  $T$



- 1 There is a protocol for  $T$  using  $D$
- 2 Any f.d.  $D'$  that can be used to solve  $T$  can emulate  $D$

Minimum information on failures required to solve  $T$

# Failure Detectors



1



2



3



4



5



6

- Local failure detection module at each proc.



# Failure Detectors



1



2



3



4



5



6

- Local failure detection module at each proc.
- Provide information on other proc. failures

# Perfect Failure Detector



$\{\mathbf{p}_1, \mathbf{p}_4\}$



$p_1$

$p_2$

$p_3$



$p_4$

$p_5$

Perfect failure detector  $\mathbf{P}$

# Perfect Failure Detector



$\{\mathbf{p}_1, \mathbf{p}_4\}$



$p_1$

$p_2$

$p_3$



$p_4$

$p_5$

Perfect failure detector  $\mathbf{P}$

- Provide each proc. with a list of proc ids.

# Perfect Failure Detector



$\{\mathbf{p}_1, \mathbf{p}_4\}$



$p_1$

$p_2$

$p_3$



$p_4$



$p_5$

Perfect failure detector  $\mathbf{P}$

- Provide each proc. with a list of proc ids.
- No false alarm

# Perfect Failure Detector



$\{p_1, p_4, p_5\}$



$p_1$

$p_2$

$p_3$



$p_4$



$p_5$

Perfect failure detector  $\mathbf{P}$

- Provide each proc. with a list of proc ids.
- No false alarm
- Eventually outputs the set of non-faulty processes

# Failure Detector $\phi$



2



$p_1$

$p_2$

$p_3$



$p_4$

$p_5$

Failure detector  $\phi$



2



$p_1$

$p_2$

$p_3$



$p_4$

$p_5$

Failure detector  $\phi$

- Provide each proc. with an integer



2



$p_1$

$p_2$

$p_3$



$p_4$



$p_5$

Failure detector  $\phi$

- Provide each proc. with an integer
- Lower bound on the number of failures





3



$p_1$

$p_2$

$p_3$



$p_4$



$p_5$

Failure detector  $\phi$

- Provide each proc. with an integer
- Lower bound on the number of failures
- Eventually tight

In a  $n$ -process system

**P**

$\phi$

In a  $n$ -process system

**P**

- **List** of proc ids.

$\phi$

- **integer**  $f, 0 \leq f \leq n$

In a  $n$ -process system

**P**

- **List** of proc ids.
- **n** bits per process

$\phi$

- **integer**  $f, 0 \leq f \leq n$
- **log n** bits per process

In a  $n$ -process system

**P**

- **List** of proc ids.
- **n** bits per process

$\phi$

- **integer**  $f, 0 \leq f \leq n$
- **log n** bits per process

And yet:

**Theorem** (Mostefaoui, Raynal, T.)

*P and  $\phi$  are equivalent: any task that can be solved using P (resp.  $\phi$ ) can also be solved using  $\phi$  (resp. P)*

How many bits per proc. are needed to achieve perfect failure detection ?

How many bits per proc. are needed to achieve perfect failure detection ?

### Theorem (upper bound)

*There exists a failure detector  $\mu\mathbf{P}$  as powerful as  $\mathbf{P}$  that outputs  $O(\mathbf{Ack}^{-1}(n))$  bits per proc*

How many bits per proc. are needed to achieve perfect failure detection ?

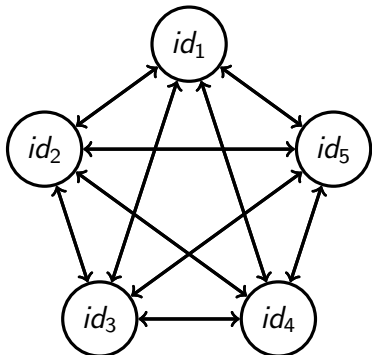
### Theorem (upper bound)

*There exists a failure detector  $\mu\mathbf{P}$  as powerful as  $\mathbf{P}$  that outputs  $O(\mathbf{Ack}^{-1}(n))$  bits per proc*

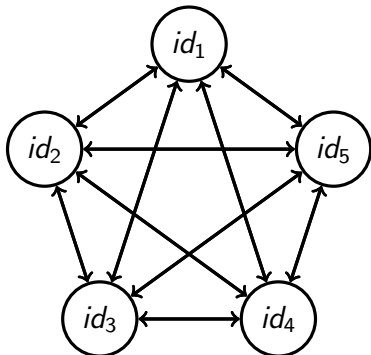
### Theorem (lower bound)

**No failure detector** outputting a **constant** number of bits per proc. can emulate  $\mathbf{P}$

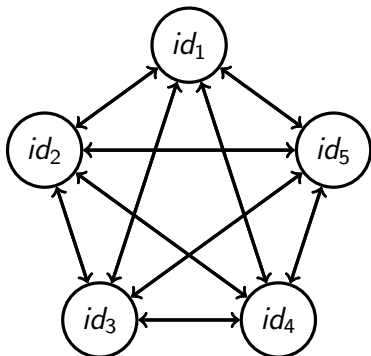




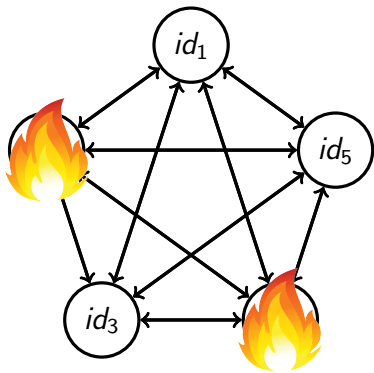
- Message passing



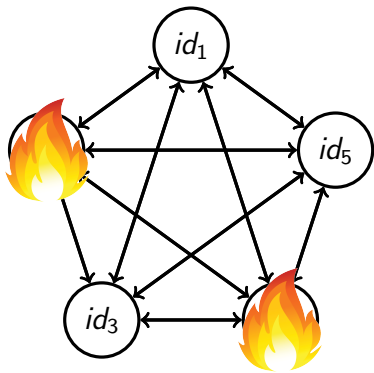
- Message passing
- Asynchronous



- Message passing
- Asynchronous
- $n$  processes



- Message passing
- Asynchronous
- $n$  processes
- Crash failures



- Message passing
- Asynchronous
- $n$  processes
- Crash failures
- Unique ids

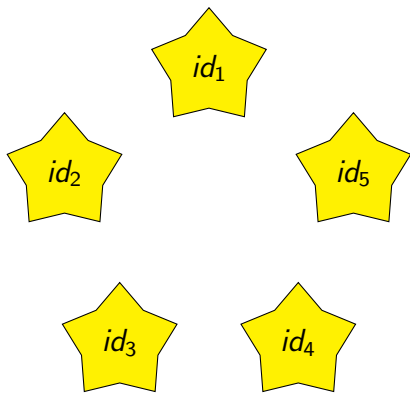
# Distributed Encoding of the Integers

[Fraigniaud, Rajsbaum, T. LATIN'16]

# Counting the Stars

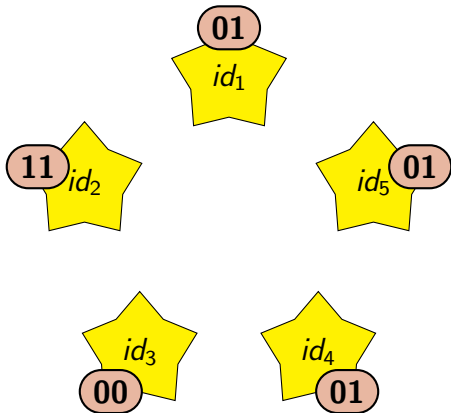


# Counting with Distributed Certificates

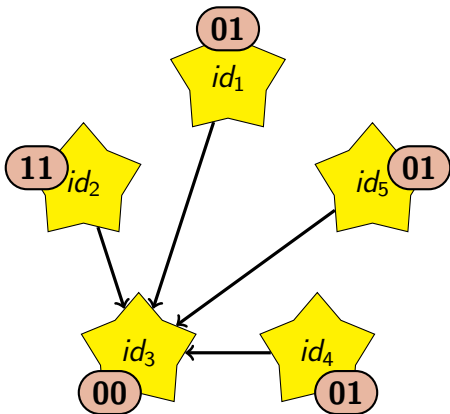




# Counting with Distributed Certificates

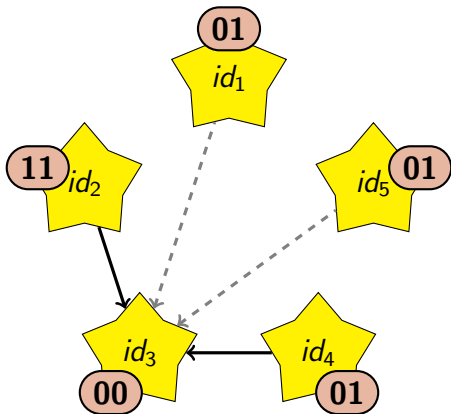


# Counting with Distributed Certificates



- $verify(5, \begin{array}{|c|c|c|c|c|} \hline 01 & 11 & 00 & 01 & 01 \\ \hline \end{array}) ? \rightarrow \text{YES}$

# Counting with Distributed Certificates



- $verify(5, \begin{array}{|c|c|c|c|c|} \hline 01 & 11 & 00 & 01 & 01 \\ \hline \end{array}) ? \rightarrow \text{YES}$
- $verify(3, \begin{array}{|c|c|c|} \hline 11 & 00 & 01 \\ \hline \end{array}) ? \rightarrow \text{NO}$

# Distributed Encoding of the Integers

- $\mathcal{A}$  alphabet

# Distributed Encoding of the Integers

- $\mathcal{A}$  alphabet
- $\mathbf{f} : \mathcal{A}^* \rightarrow \{\mathbf{YES}, \mathbf{NO}\}$

# Distributed Encoding of the Integers

- $\mathcal{A}$  alphabet
- $f : \mathcal{A}^* \rightarrow \{\mathbf{YES}, \mathbf{NO}\}$

such that for each  $n \in \mathbb{N}$   
there exists a code of  $n$   $c_n \in \mathcal{A}^n$  :

- 1  $f(c_n) = \mathbf{YES}$  and

# Distributed Encoding of the Integers

- $\mathcal{A}$  alphabet
- $\mathbf{f} : \mathcal{A}^* \rightarrow \{\mathbf{YES}, \mathbf{NO}\}$

such that for each  $n \in \mathbb{N}$   
there exists a code of  $n$   $\mathbf{c}_n \in \mathcal{A}^n$  :

- ①  $\mathbf{f}(\mathbf{c}_n) = \mathbf{YES}$  and
- ② For every sub-word  $c'$  of  $\mathbf{c}_n$ ,  $\mathbf{f}(c') = \mathbf{NO}$

# Simple Distributed Encoding

distributed code of  $\mathbf{n}$

$$C(\mathbf{n}) = \underbrace{n, n, \dots, n}_{n \text{ times}}$$



# Simple Distributed Encoding

distributed code of  $\mathbf{n}$

$$C(\mathbf{n}) = \underbrace{n, n, \dots, n}_{n \text{ times}}$$

$$f(x_1, \dots, x_\ell) = \mathbf{YES} \iff x_1 = x_2 = \dots = x_\ell = \ell$$

# Simple Distributed Encoding

distributed code of  $\mathbf{n}$

$$C(\mathbf{n}) = \underbrace{n, n, \dots, n}_{n \text{ times}}$$

$$f(x_1, \dots, x_\ell) = \mathbf{YES} \iff x_1 = x_2 = \dots = x_\ell = \ell$$

Alphabet of  $N$  symbols to encode the first  $N$  integers

# Simple Distributed Encoding

distributed code of  $\mathbf{n}$

$$C(\mathbf{n}) = \underbrace{n, n, \dots, n}_{n \text{ times}}$$

$$f(x_1, \dots, x_\ell) = \mathbf{YES} \iff x_1 = x_2 = \dots = x_\ell = \ell$$

Alphabet of  $N$  symbols to encode the first  $N$  integers

Challenge: Compact encoding

# Diagonal Sequence

0000

code of 4

# Diagonal Sequence

0000

00**11**0

code of 4

code of 5

# Diagonal Sequence

0000

code of 4

00**11**0

code of 5

0**11**0**1**0

code of 6

# Diagonal Sequence

0000		code of 4
00110		code of 5
011010		code of 6
1101010		
10101011		
010101111		
1111110010		
11111001011		
111100101111		
1110010111111		
⋮	⋮	
⋮	⋮	
111111111111 ..... 1		code of $2^{257} - 2$





# Diagonal Sequence

0000                    code of 4  
00110                   code of 5  
011010                   code of 6

1101010

**10101011**

010101111

1111110010            not a sub-word

11111001011

111100101111

**1110010111111**

⋮

⋮

⋮

⋮

111111111111 ..... 1            code of  $2^{257} - 2$



$H(4)-1$

Let  $w, w' \in \{0, 1\}^*$

$w \preceq_* w' \iff w$  is a **sub-word** of  $w'$

$w$	<b>1010</b>
$w'$	00 <b>1</b> 11 <b>0</b> 1111 <b>10</b>

Let  $w, w' \in \{0, 1\}^*$

$w \preceq_* w' \iff w$  is a **sub-word** of  $w'$

$w$	<b>1010</b>
$w'$	00 <b>1</b> 11 <b>0</b> 1111 <b>0</b>

## Bad Sequence

A sequence  $w_1, w_2, \dots, w_\ell$  of words of  $\{0, 1\}^*$  is **bad** iff for every  $i < j$ ,  $w_i \not\preceq_* w_j$

Higman's lemma

$(\{0, 1\}^*, \preceq_*)$  is a well-quasi order

## Higman's lemma

$(\{0, 1\}^*, \preceq_*)$  is a **well-quasi order**

That is, every bad sequence over  $\{0, 1\}^*$  is **finite**

## Higman's lemma

$(\{0, 1\}^*, \preceq_*)$  is a **well-quasi order**

That is, every bad sequence over  $\{0, 1\}^*$  is **finite**

**Length Function Theorem** [ Schmitz et al., ICALP'11]

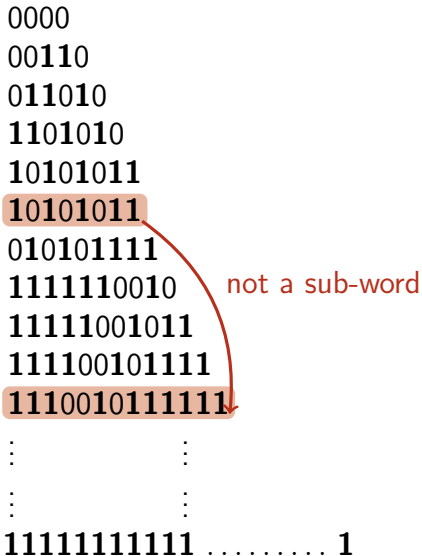
Bad sequences  $w_1, w_2, \dots, w_\ell$  over  $\{0, 1\}^*$  with

- $|w_1| \leq d$
- $|w_i| \leq i$

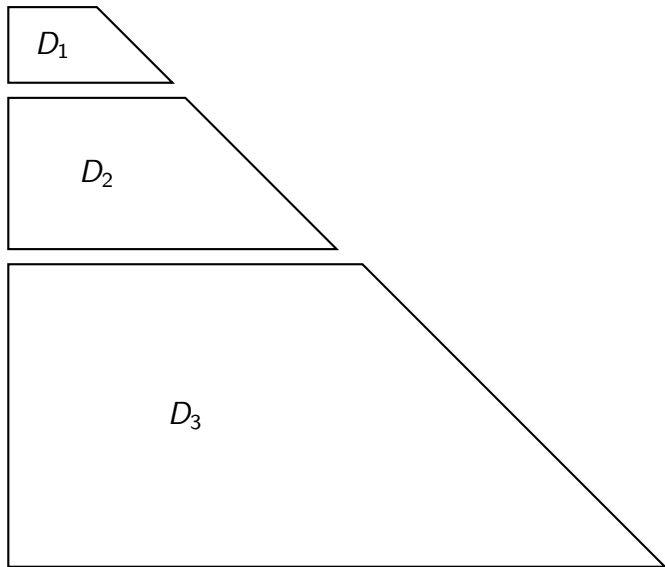
have length bounded by  $L(d)$  where  $L$  is a function of *Ackermannian growth*

# A Bad Sequence

0000  
00110  
011010  
1101010  
10101011  
**10101011**  
010101111  
1111110010 not a sub-word  
11111001011  
111100101111  
**1110010111111**  
:  
:  
:  
:  
111111111111 ..... 1

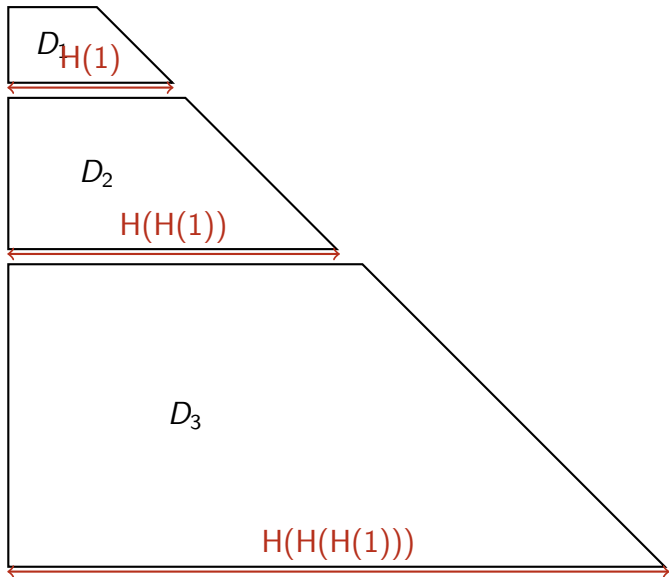


# Multi diagonal sequence

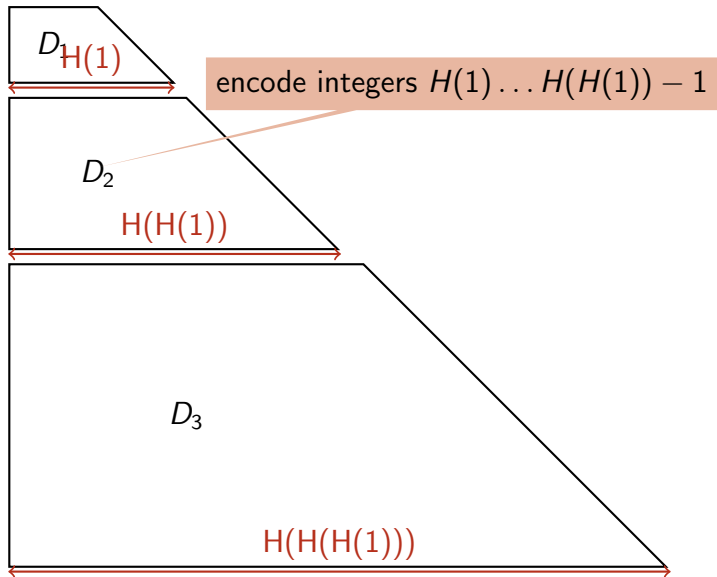




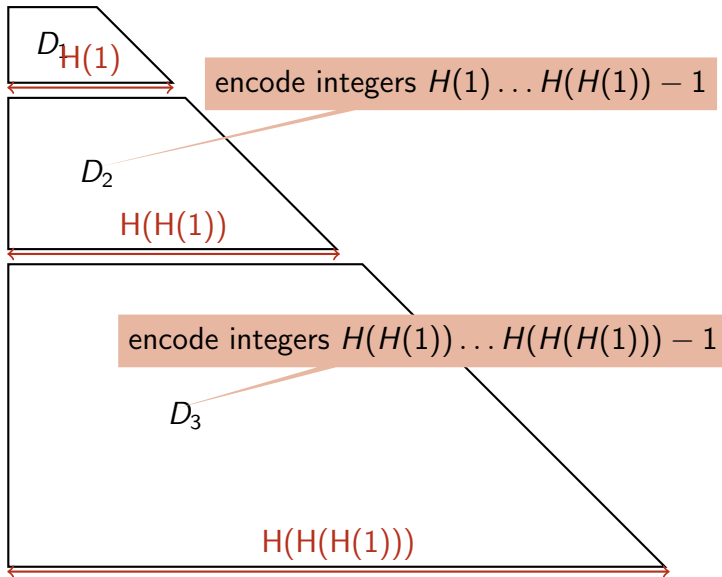
# Multi diagonal sequence



# Multi diagonal sequence



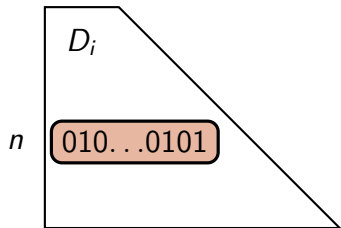
# Multi diagonal sequence



# Encoding from Multi Diagonal Sequence

$$A = \{0, 1\} \times \mathbb{N}$$

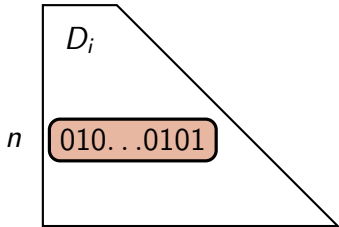
*Code*( $n$ ) :



# Encoding from Multi Diagonal Sequence

$$A = \{0, 1\} \times \mathbb{N}$$

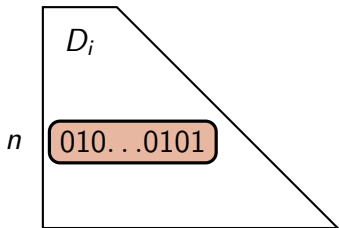
$Code(n) : (0, i), (1, i), (0, i), \dots, (1, i), (0, i), (1, i)$



# Encoding from Multi Diagonal Sequence

$$A = \{0, 1\} \times \mathbb{N}$$

$Code(n) : (0, i), (1, i), (0, i), \dots, (1, i), (0, i), (1, i)$



$f((b_1, d_1), (b_2, d_2), \dots, (b_n, d_n)) = \mathbf{YES} \iff$

①  $d_1 = d_2 = \dots = d_n = i$

②  $b_1, \dots, b_n$  is the sequence of length  $n$  in  $D_i$

*How many bits to distributively encode the first  $n$  integers?*

*Code( $n$ ) : (0,  $i$ ), (1,  $i$ ), (0,  $i$ ),  $\dots$ , (1,  $i$ ), (0,  $i$ ), (1,  $i$ )*

*How many bits to distributively encode the first  $n$  integers?*

$Code(n) : (0, i), (1, i), (0, i), \dots, (1, i), (0, i), (1, i)$

$\Rightarrow 1 + \log(i)$  bits

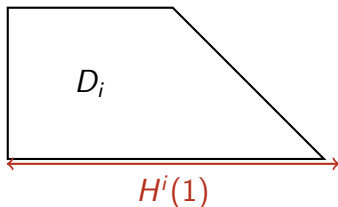


*How many bits to distributively encode the first  $n$  integers?*

$Code(n) : (0, i), (1, i), (0, i), \dots, (1, i), (0, i), (1, i)$

$\implies 1 + \log(i)$  bits

where  $i = \min\{j : n < H^j(1)\} \leq \mathbf{Ack}^{-1}(n)$

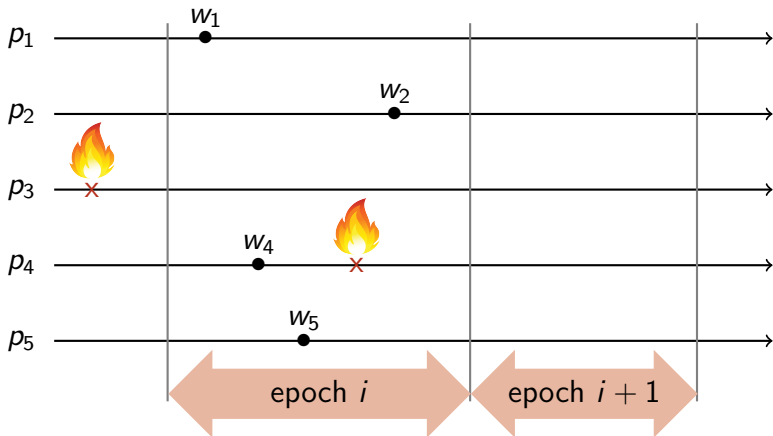


Perfect Failure Detection  
from  
Distributed Encoding

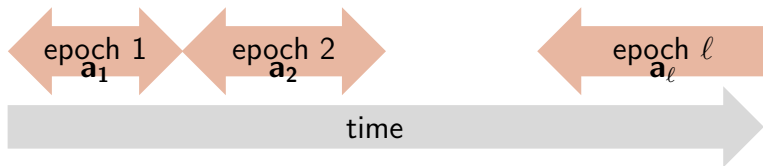
# Perfect failure detection from distributed encoding

Failure detector  $\mu\mathbf{P}$ :

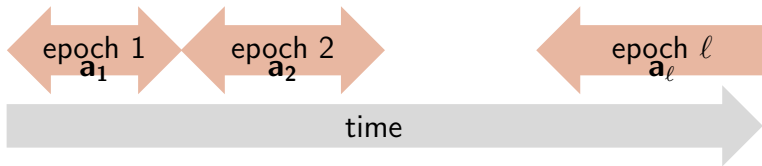
- Encode an upper bound on the number of alive processes
- Eventually converge to the (code of the) number of non-faulty processes



- Constant fd output at each proc in each epoch
- $w_1 w_2 w_4 w_5 \preceq_* \text{code}(a_i)$ , where  $\# \text{ alive}(\text{epoch } i) \leq a_i$

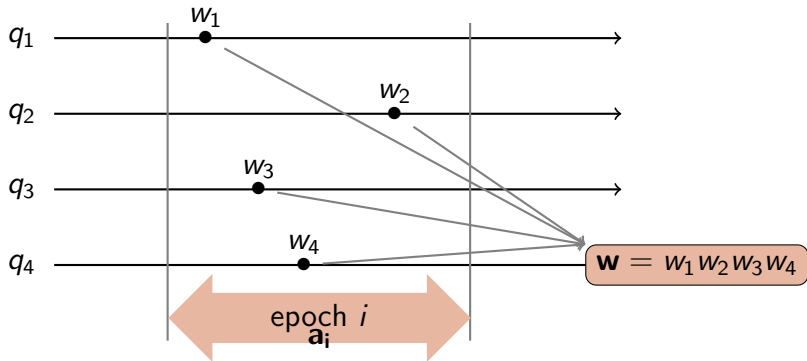


- At most  $n$  epochs
- $a_1 \geq a_2 \geq \dots \geq a_l$



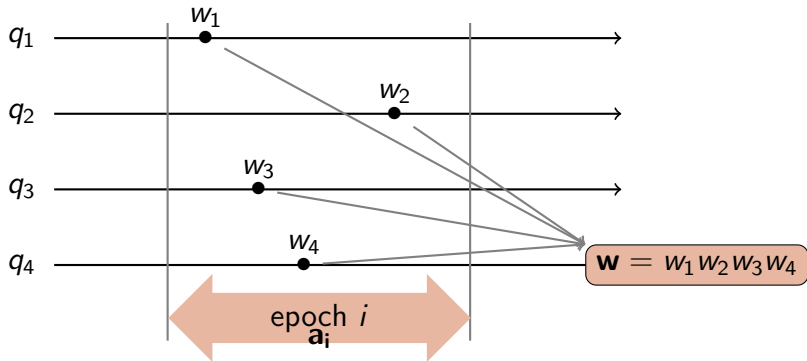
- At most  $n$  epochs
- $a_1 \geq a_2 \geq \dots \geq a_l$
- $a_l = \# \text{ alive}(\text{last epoch}) = \# \text{ correct procs}$

Let  $Q = \{q_1, \dots, q_4\} \subseteq \{p_1, \dots, p_n\}$



- Recall:  $\mathbf{w} \preceq_* \text{code}(a_i)$  and  $|\text{Alive}(\text{epoch } i)| \leq \mathbf{a}_i$

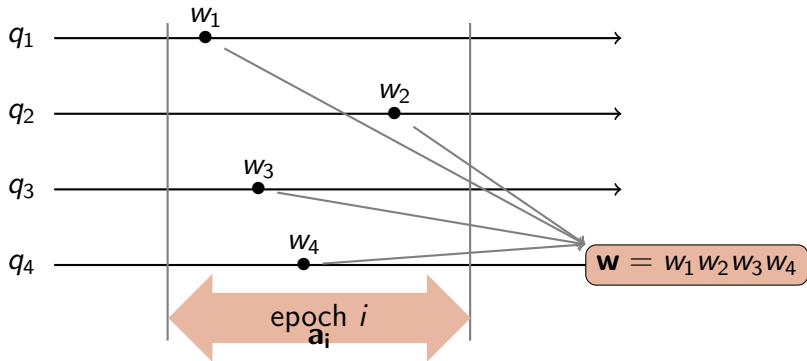
Let  $Q = \{q_1, \dots, q_4\} \subseteq \{p_1, \dots, p_n\}$



- Recall:  $\mathbf{w} \preceq_* \text{code}(a_i)$  and  $|\text{Alive}(\text{epoch } i)| \leq \mathbf{a}_i$
- Code def.:  $w \preceq_* \text{code}(\mathbf{a}_i) \implies \mathbf{f}(w) = \text{false}$   
and  $\mathbf{f}(\text{code}(a_i)) = \text{true}$

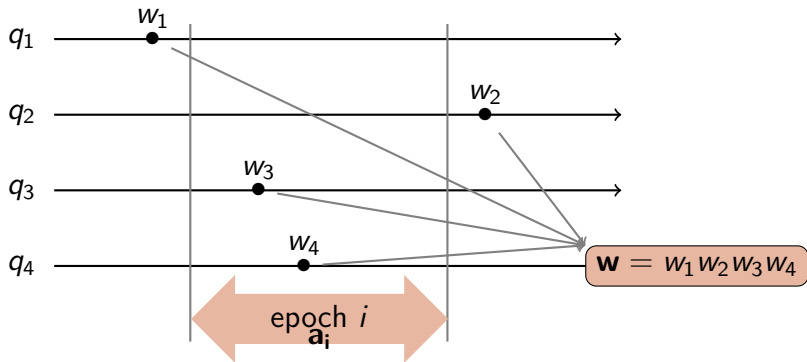


Let  $Q = \{q_1, \dots, q_4\} \subseteq \{p_1, \dots, p_n\}$



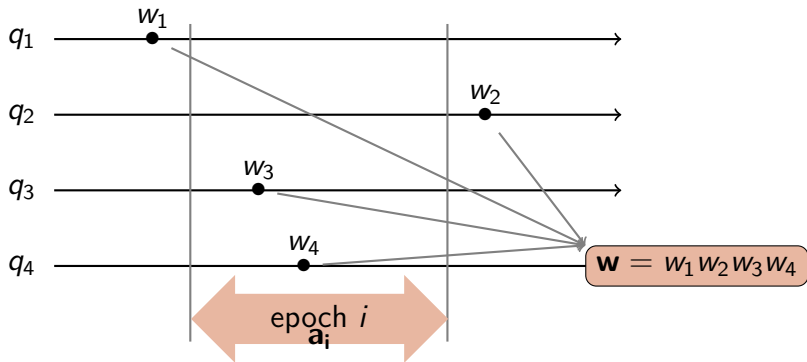
- Recall:  $\mathbf{w} \preceq_* \text{code}(a_i)$  and  $|\text{Alive}(\text{epoch } i)| \leq a_i$
- Code def.:  $w \preceq_* \text{code}(a_i) \implies \mathbf{f}(w) = \text{false}$   
and  $\mathbf{f}(\text{code}(a_i)) = \text{true}$
- Hence, if  $\mathbf{f}(w) = \text{true}$  then  $\{p_1, \dots, p_n\} \setminus Q \subseteq \text{Faulty}$

Let  $Q = \{q_1, \dots, q_4\} \subseteq \{p_1, \dots, p_n\}$

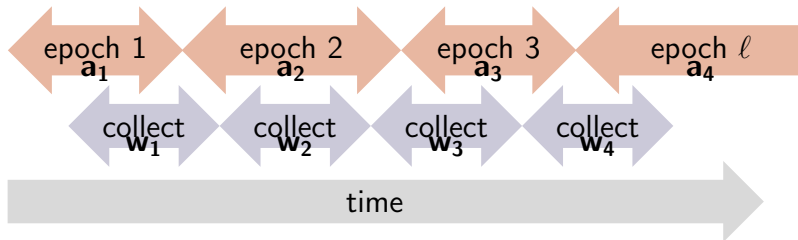


- $w_1, w_2, w_3, w_4$  sampled in  $\neq$  epochs

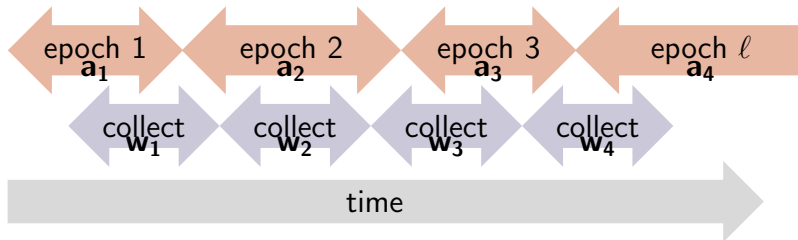
Let  $Q = \{q_1, \dots, q_4\} \subseteq \{p_1, \dots, p_n\}$



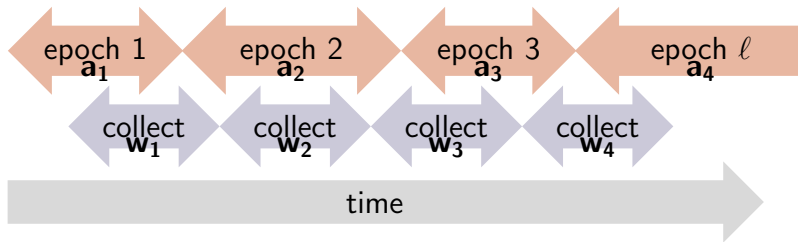
- $w_1, w_2, w_3, w_4$  sampled in  $\neq$  epochs
- $f(w) = true ?? f(w) = false ??$



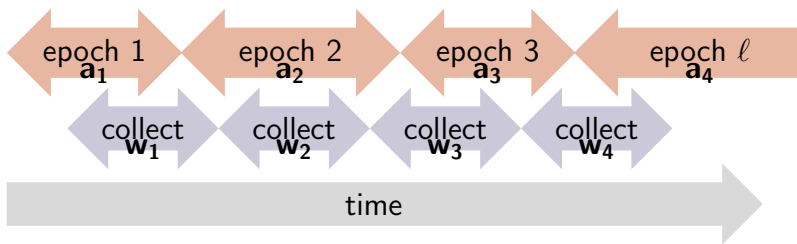
- At most  $n$  epochs



- At most  $n$  epochs
- ⇒ In a sequence of  $n$  collects, at least one is **clean**



- Collect  $i$  is *successful* if (1) terminates and (2)  $\mathbf{f}(w_i) = true$



- Collect  $i$  is *successful* if (1) terminates and (2)  $f(w_i) = true$
- If for some set  $Q$ , there are  $n$  *successful* collects  
 $P$  output =  $\{p_1, \dots, p_n\} \setminus Q$

Failure detector  $\mu P$

- Outputs  $O(\log \text{Ack}^{-1}(n))$  bits per processes
- Can **emulate** the perfect failure detector **P**



Failure detector  $\mu P$ 

- Outputs  $O(\log \text{Ack}^{-1}(n))$  bits per processes
- Can **emulate** the perfect failure detector  $P$
- ( $P$  can also emulate  $\mu P$  – see the paper)

Failure detector  $\mu\mathbf{P}$

- Outputs  $\mathbf{O}(\log \mathbf{Ack}^{-1}(\mathbf{n}))$  bits per processes

Failure detector  $\mu P$

- Outputs  $O(\log \text{Ack}^{-1}(n))$  bits per processes

Is there a f.d.  $D$  that

- ① can emulate  $P$
- ② outputs less than  $\log \text{Ack}^{-1}(n)$  bits per process ?

Failure detector  $\mu\mathbf{P}$

- Outputs  $\mathbf{O}(\log \mathbf{Ack}^{-1}(n))$  bits per processes

Is there a f.d.  $D$  that

- ① can emulate  $P$
- ② outputs less than  $\log \mathbf{Ack}^{-1}(n)$  bits per process ?

Theorem

**No failure detector with constant-size output can emulate  $\mathbf{P}$**

Assume for contradiction  $D$  f.d. such that

- Constant range  $R$ , (independent of  $n$ )

Assume for contradiction  $D$  f.d. such that

- Constant range  $R$ , (independent of  $n$ )
- $T_{D \rightarrow P}$  (can emulate  $P$ )

Assume for contradiction  $D$  f.d. such that

- Constant range  $R$ , (independent of  $n$ )
- $T_{D \rightarrow P}$  (can emulate  $P$ )

Ingredients

- Ramsey's theorem
- Well quasi-order theory

Construct two executions  $e$  and  $e'$  :

- *indistinguishable* for some non-faulty processes
- with  $Correct(e) \subsetneq Correct(e')$



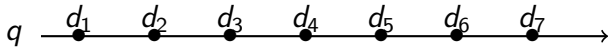
Construct two executions  $e$  and  $e'$  :

- *indistinguishable* for some non-faulty processes
- with  $Correct(e) \subsetneq Correct(e')$



in  $e'$   $T_{D \rightarrow P}$  erroneously outputs a non-faulty process

Let  $e$  an (infinite) execution



- As  $R_D$  is finite,  $\exists d \in D$  output infinitely many times at  $q$

# From Executions to Words

Let  $e$  an (infinite) execution



- As  $R_D$  is finite,  $\exists d \in D$  output infinitely many times at  $q$

# From Executions to Words

Let  $e$  an (infinite) execution



- As  $R_D$  is finite,  $\exists d \in D$  output infinitely many times at  $q$

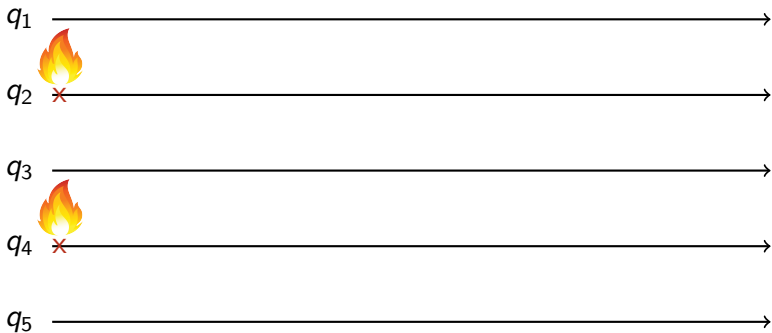
Execution  $\tilde{e}$



- **Constant failure detector output**

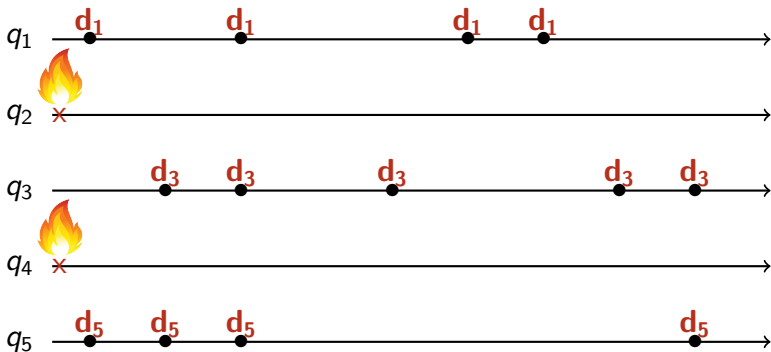
# From Executions to Words

Let  $e$  an (infinite) execution in which crashes are initial



# From Executions to Words

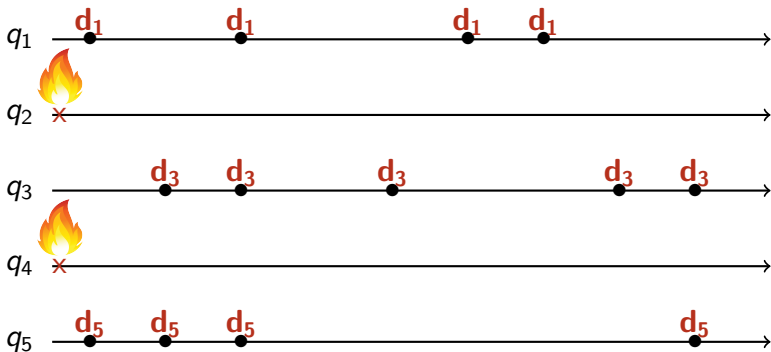
Let  $e$  an (infinite) execution in which crashes are initial



Constant f.d. output ( $d_i$ ) at each non-faulty process  $q_i$

# From Executions to Words

Let  $e$  an (infinite) execution in which crashes are initial



Constant f.d. output ( $d_i$ ) at each non-faulty process  $q_i$

$$e \longrightarrow w_e = d_1 d_3 d_5 \in R_D^*$$

# Towards Indistinguishable Executions

<i>execution</i>	<i>associated word</i> $\in R_D^*$
$e_1$	$w_1$
$e_2$	$w_2$
$e_3$	$w_3$
$\vdots$	$\vdots \quad \vdots \quad \vdots \quad \ddots$
$e_L$	$w_L$

$|w_i| = i$



# Towards Indistinguishable Executions

<i>execution</i>	<i>associated word</i> $\in R_D^*$
$e_1$	$w_1$
$e_2$	$w_2$
$e_3$	$w_3$
$\vdots$	$\vdots \quad \vdots \quad \vdots \quad \ddots$
$e_L$	$w_L$

$|w_i| = i$

- (Higman's Lemma)  $(R_D^* \preceq_*)$  is a *wqo*

# Towards Indistinguishable Executions

<i>execution</i>	<i>associated word</i> $\in R_D^*$
$e_1$	$w_1$
$e_2$	$w_2$
$e_3$	$w_3$
$\vdots$	$\vdots \quad \vdots \quad \vdots \quad \ddots$
$e_L$	$w_L$

$|w_i| = i$

- (Higman's Lemma)  $(R_D^* \preceq_*)$  is a *wqo*
- $\implies$  For large enough  $L$ ,
- $\exists i, j : 1 \leq i < j \leq L$  and  $w_i$  **subword** of  $w_j$

# Towards Indistinguishable Executions

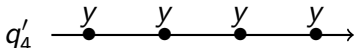
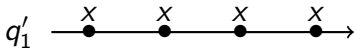
execution  $e$

$w = \mathbf{abc}$



execution  $e'$

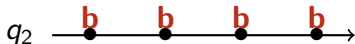
$w' = x\mathbf{a}b\mathbf{y}c$



# Towards Indistinguishable Executions

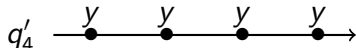
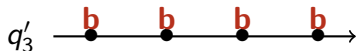
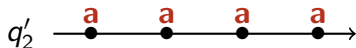
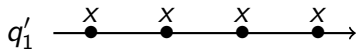
execution  $e$

$w = \mathbf{abc}$



execution  $e'$

$w' = x\mathbf{abc}$

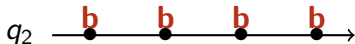


**a** (or **b**, **c**) may be output at processes with **distinct ids** in  $e$  and  $e'$ .

# Towards Indistinguishable Executions

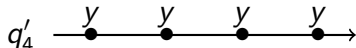
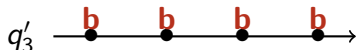
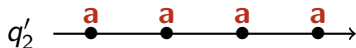
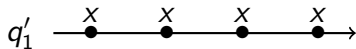
execution  $e$

$w = \mathbf{abc}$



execution  $e'$

$w' = x\mathbf{a}b\mathbf{y}c$



$\mathbf{a}$  (or  $\mathbf{b}$ ,  $\mathbf{c}$ ) may be output at processes with **distinct ids** in  $e$  and  $e'$ . Rely on **Ramsey's Theorem** to get rid of ids

## Summary:

- Perfect failure detection with  $O(Ack^{-1}(n))$  bits per process
- Perfect failure detection with constant output is impossible

## Summary:

- Perfect failure detection with  $O(Ack^{-1}(n))$  bits per process
- Perfect failure detection with constant output is impossible
- Applications of wqo theory to distributed computing

## Summary:

- Perfect failure detection with  $O(\text{Ack}^{-1}(n))$  bits per process
- Perfect failure detection with constant output is impossible
- Applications of wqo theory to distributed computing

## Future work:



## Summary:

- Perfect failure detection with  $O(\text{Ack}^{-1}(n))$  bits per process
- Perfect failure detection with constant output is impossible
- Applications of wqo theory to distributed computing

## Future work:

- Close the gap between lower and upper bounds

## Summary:

- Perfect failure detection with  $O(\text{Ack}^{-1}(n))$  bits per process
- Perfect failure detection with constant output is impossible
- Applications of wqo theory to distributed computing

## Future work:

- Close the gap between lower and upper bounds
- Failure detector as (distributed) encoder: Relation between output size and failure detector power

## Summary:

- Perfect failure detection with  $O(\text{Ack}^{-1}(n))$  bits per process
- Perfect failure detection with constant output is impossible
- Applications of wqo theory to distributed computing

## Future work:

- Close the gap between lower and upper bounds
- Failure detector as (distributed) encoder: Relation between output size and failure detector power
- Other application of the distributed encoding of the integers

Thanks!

# Coloring Subsets of Processes



$c$  assigns a color to each subset of processes

$$c(Q = \{q_1, \dots, q_k\}) \in R_D^k$$

$q_1$  —————→

$q_2$  —————→

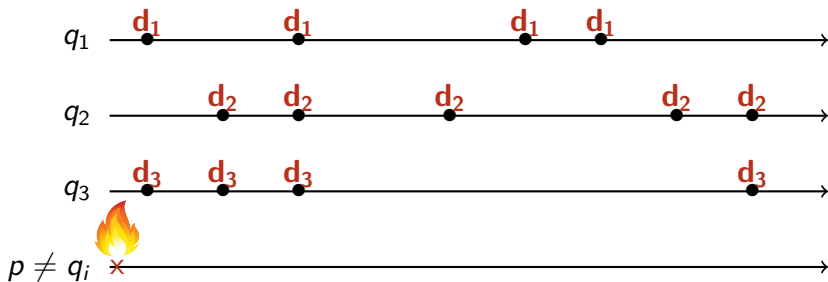
$q_3$  —————→

$p \neq q_i$    —————→

# Coloring Subsets of Processes

$c$  assigns a color to each subset of processes

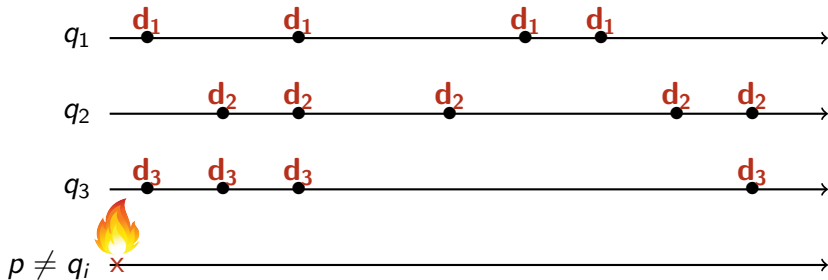
$$c(Q = \{q_1, \dots, q_k\}) \in R_D^k$$



# Coloring Subsets of Processes

$c$  assigns a color to each subset of processes

$$c(Q = \{q_1, \dots, q_k\}) \in R_D^k$$



$$c(\{q_1, \dots, q_3\}) = (d_1, d_2, d_3)$$

$$\mathbf{c} : Q = \{q_1, \dots, q_k\} \rightarrow R_D^k$$



$$c : Q = \{q_1, \dots, q_k\} \rightarrow R_D^k$$

## Ramsey's Theorem

- For any  $m, k$

$$c : Q = \{q_1, \dots, q_k\} \rightarrow R_D^k$$

## Ramsey's Theorem

- For any  $m, k$
- There exists  $n = g(m, k)$  such that

$$\mathbf{c} : Q = \{q_1, \dots, q_k\} \rightarrow R_D^k$$

## Ramsey's Theorem

- For any  $m, k$
- There exists  $n = g(m, k)$  such that
- There exists a  $m$ -subset  $\mathbf{S}$  of the  $n$  procs such that

$$\mathbf{c} : Q = \{q_1, \dots, q_k\} \rightarrow R_D^k$$

## Ramsey's Theorem

- For any  $m, k$
- There exists  $n = g(m, k)$  such that
- There exists a  $m$ -subset  $\mathbf{S}$  of the  $n$  procs such that
- Every  $k$ -subset of  $\mathbf{S}$  has the same color

$$c : Q = \{q_1, \dots, q_k\} \rightarrow R_D^k$$

## Ramsey's Theorem

- For any  $m, k$
- There exists  $n = g(m, k)$  such that
- There exists a  $m$ -subset  $\mathbf{S}$  of the  $n$  procs such that
- Every  $k$ -subset of  $\mathbf{S}$  has the same color

## Intuitively

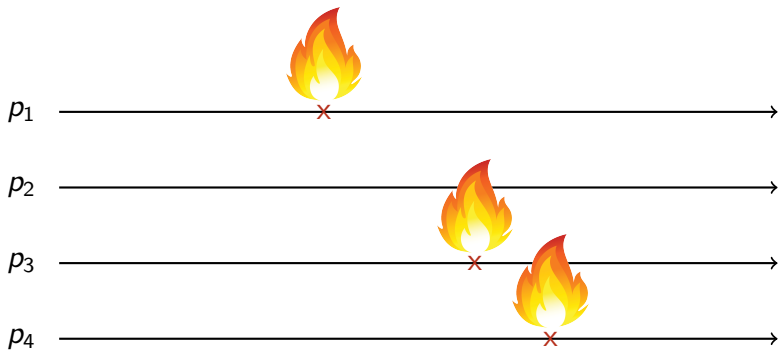
For any  $q \in \mathbf{S}$ ,

- For failure pattern with  $k$  correct procs and initial crashes
- F.d output at  $q$  depends only on the rank of its id

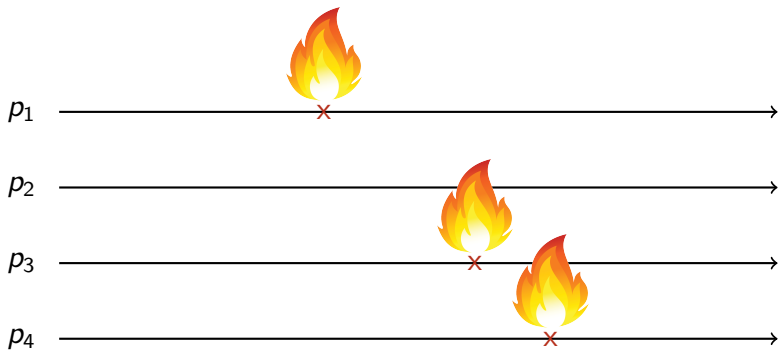
A failure detector  $D$

- Outputs symbols in some *range*  $R_D$
- Is defined with respect to *failure patterns*

# Failure Pattern



# Failure Pattern



$$\mathcal{F} : \mathbb{N} \rightarrow 2^{\{p_1, \dots, p_n\}}$$



# Failure Detector Specification

For each failure pattern,  $D$  defines which outputs are valid

- History :  $H(p, t)$  is the output at process  $p$  at time  $t$
- $D(\mathcal{F}) =$  valid histories for failure pattern  $\mathcal{F}$

# Failure Detector Equivalence

Failure detectors  $D$  and  $D'$  are **equivalent**



There exist two asynchronous, crash-resilient protocols

- $T_{D \rightarrow D'}$  that emulates  $D'$  using  $D$
- $T_{D' \rightarrow D}$  that emulates  $D$  using  $D'$