Renaming

c. travers
travers@labri.fr

Displexity December 2012

Origin: Mutual Exclusion

```
a := 2;
// enter critical section
{ // critical section
      b:= 2 +a;
      c:= 2*b - 4;
}
// exit critical section
```

[Dijsktra 1965]

Exclusive access

p_1 p_2 p_3 p_4 p_5

Resource can be accessed by at most one process at a time

Exclusive access

p_1 p_2 p_3 p_4 p_5

Resource can be accessed by at most one process at a time

Renaming

n processes

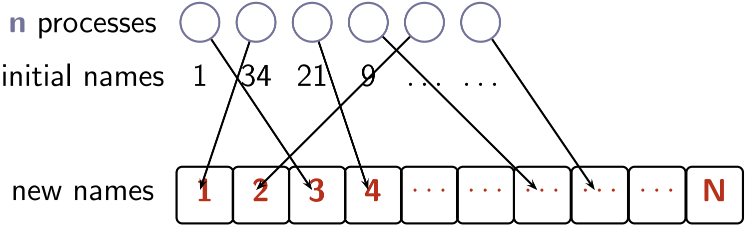
initial names 1 34 21 9 ... ...

new names

1	**2**	**3**	**4**	···	···	···	···	···	**N**

- n processes, provided with an initial name $\in \mathbf{1}..\mathbf{M}$

Renaming



- n processes, provided with an initial name $\in$ **1..M**
- each proc. has to get an **unique** name $\in$ **1..N**, **N** $<<$ **M**

Renaming

$\mathbf{n}$ processes

initial names 1 34 21 9 … …

new names

1	2	3	4	⋯	⋯	⋯	⋯	⋯	N

- n processes, provided with an initial name $\in \mathbf{1..M}$
- each proc. has to get an **unique** name $\in \mathbf{1..N}$, $\mathbf{N} << \mathbf{M}$

Comparison-based algorithms

initial ids van only be compared

Renaming Variants

n processes ◯ ◯ ◯ ◯ ◯ ◯

initial names 1 34 21 9 ... ...

new names | **1** | **2** | **3** | **4** | $\cdots$ | **N** |

Tight/Loose renaming

- Tight : $\mathbf{N} = \mathbf{n}$ / Loose : $\mathbf{N} > \mathbf{n}$

Renaming Variants

$\mathbf{n}$ processes

initial names 1 34 21 9 ... ...

new names

1	**2**	**3**	**4**	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	**N**

Tight/Loose renaming

- Tight : $\mathbf{N} = \mathbf{n}$ / Loose : $\mathbf{N} > \mathbf{n}$

Renaming Variants

n processes ○ ○ ○ ○ ○ ○

initial names 1 34 21 9 ... ...

new names

1	**2**	**3**	**4**	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	**N**

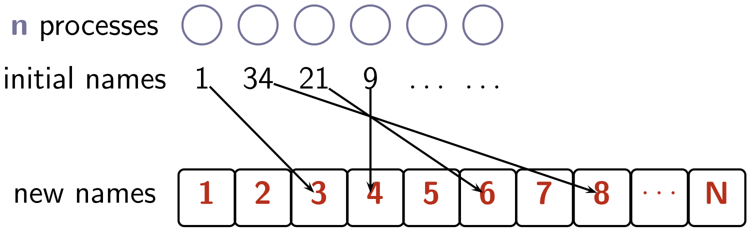
Tight/Loose renaming

- Tight : $\mathbf{N} = \mathbf{n}$ / Loose : $\mathbf{N} > \mathbf{n}$

Adaptive renaming

- Final name space function of # participating procs.

Renaming Variants



Tight/Loose renaming

- Tight : $\mathbf{N} = \mathbf{n}$ / Loose : $\mathbf{N} > \mathbf{n}$

Adaptive renaming

- Final name space function of # participating procs.

Order preserving

- Final names preserve the order of initial names

Questions

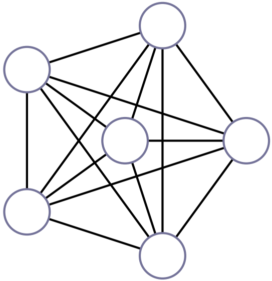
Name space: how many final names?

Complexity: how much work to acquire a new name?

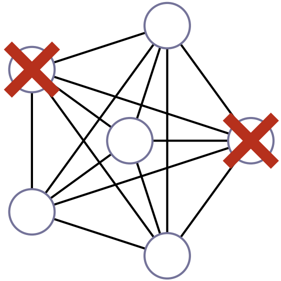
Distributed models

Message passing

- Complete network
- **Failures**: crashes
- Synchronous/Asynchronous



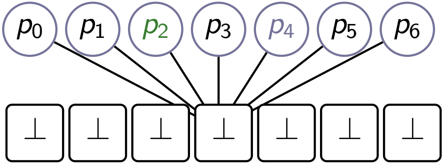
Distributed models



Message passing

- Complete network
- **Failures**: crashes
- Synchronous/Asynchronous

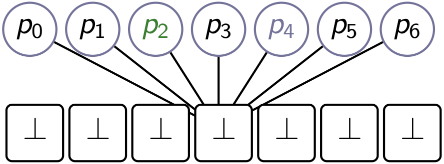
Distributed Models



Shared memory

- n Processes $\{p_1, \ldots, p_n\}$

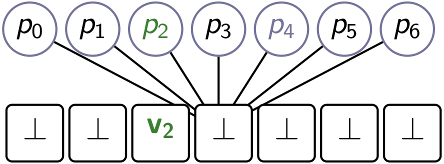
Distributed Models



Shared memory

- n Processes $\{p_1, \ldots, p_n\}$
- Asynchronous communications
    - read() from/write() to any memory cell
    - finite but unbounded delay between steps

Distributed Models



Shared memory

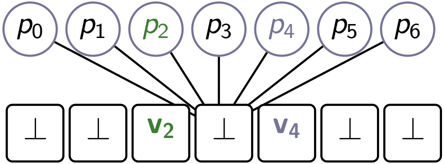
- n Processes $\{p_1, \ldots, p_n\}$
- Asynchronous communications
  - read() from/write() to any memory cell
  - finite but unbounded delay between steps

$\mathbf{p_2}$ $R[2].\text{write}(v_2)$

$\mathbf{p_4}$ $R[4].\text{write}(v_4)$

$\mathbf{p_2}$ $R.\text{read}()$ $[\ldots, \mathbf{v_2}, \perp, \mathbf{v_4}, \ldots]$

Distributed Models

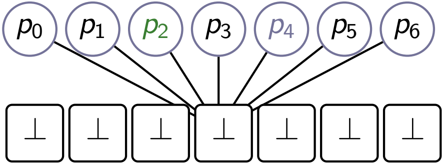


Shared memory

- n Processes $\{p_1, \ldots, p_n\}$
- Asynchronous communications
  - read() from/write() to any memory cell
  - finite but unbounded delay between steps

$\mathbf{p_2}$	$R[2].\text{write}(v_2)$	
$\mathbf{p_4}$	$R[4].\text{write}(v_4)$	
$\mathbf{p_2}$	$R.\text{read}()$	$[\ldots, \mathbf{v_2}, \perp, \mathbf{v_4}, \ldots]$

Distributed Models



Shared memory

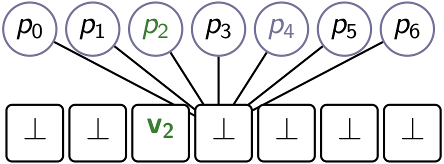
- n Processes $\{p_1, \ldots, p_n\}$
- Asynchronous communications
  - read() from/write() to any memory cell
  - finite but unbounded delay between steps

$\mathbf{p_2}$ $R[2].\mathrm{write}(v_2)$

$\mathbf{p_2}$ $R.\mathrm{read}()$ $[\ldots, \mathbf{v_2}, \perp, \perp, \ldots]$

$\mathbf{p_4}$ $R[4].\mathrm{write}(v_4)$

Distributed Models

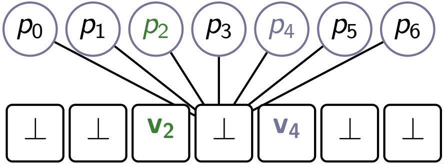


Shared memory

- n Processes $\{p_1, \ldots, p_n\}$
- Asynchronous communications
  - read() from/write() to any memory cell
  - finite but unbounded delay between steps

$\mathbf{p_2}$	$R[2].\mathrm{write}(v_2)$	
$\mathbf{p_2}$	$R.\mathrm{read}()$	$[\ldots, \mathbf{v_2}, \perp, \perp, \ldots]$
$\mathbf{p_4}$	$R[4].\mathrm{write}(v_4)$	

Distributed Models



Shared memory

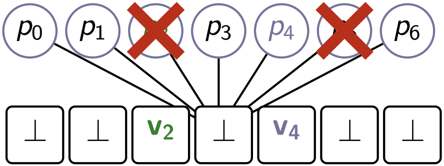
- n Processes $\{p_1, \ldots, p_n\}$
- Asynchronous communications
  - read() from/write() to any memory cell
  - finite but unbounded delay between steps

$\mathbf{p_2}$ $R[2].\mathrm{write}(v_2)$

$\mathbf{p_2}$ $R.\mathrm{read}()$ $[\ldots, \mathbf{v_2}, \perp, \perp, \ldots]$

$\mathbf{p_4}$ $R[4].\mathrm{write}(v_4)$

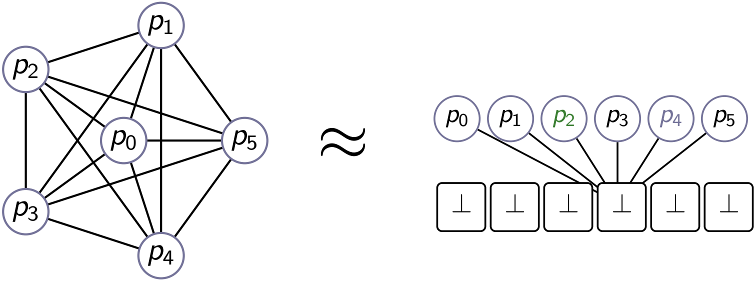
Distributed Models



Shared memory

- n Processes $\{p_1, \ldots, p_n\}$
- Asynchronous communications
  - read() from/write() to any memory cell
  - finite but unbounded delay between steps
- Failures : crash

Equivalence



Asynchronous models

Shared memory can be simulated in message passing
if $\#\text{crashs} < \frac{n}{2}$

Asynchronous Renaming

Wait-free renaming

Model

- **n**-process **asynchronous** shared memory
- Wait-free: all but one process may crash

p_0 p_1 p_2 p_3 p_4

Question

- **Name space** How many final names needed to solve renaming?

Wait-free renaming

Model

- **n**-process **asynchronous** shared memory
- Wait-free: all but one process may crash

Question

- **Name space** How many final names needed to solve renaming?

Wait-free renaming

- $\mathbf{2n-1}$ names are sufficient
  [Attiya et al., Borowsky Gafni, Attiya Fouren, Gafni Rajsbaum]
- $\mathbf{n+1}$ names are necessary [Attiya et al.]
- $\mathbf{2n-1}$ names are necessary
  [Herlihy Shavit, Herlihy Rajsbaum, Attiya Rajsbaum]

Wait-free renaming

- $2n-1$ names are sufficient [Attiya et al., Borowsky Gafni, Attiya Fouren, Gafni Rajsbaum]
- $n+1$ names are necessary [Attiya et al.]
- $2n-1$ names are necessary [Herlihy Shavit, Herlihy Rajsbaum, Attiya Rajsbaum]

$>$ 10 years later

- $2n-1$ names are necessary for **some** values of **n** [Casteñada Rajsbaum, Attiya Paz]

Wait-free renaming

- $2n-1$ names are sufficient
  [Attiya et al., Borowsky Gafni, Attiya Fouren, Gafni Rajsbaum]
- $n+1$ names are necessary [Attiya et al.]
- $2n-1$ names are necessary
  [Herlihy Shavit, Herlihy Rajsbaum, Attiya Rajsbaum]

$>$ 10 years later

- $2n-1$ names are necessary for **some** values of **n**
  [Casteñada Rajsbaum, Attiya Paz]

Theorem

$(2n-2)$-renaming solvable $\iff$ n is not a prime power

Open questions

Asynchronous **message passing** renaming

- [Attiya et al.] (2n-1)-renaming : exponential worst case complexity
- [Alistarh et al.] attempt to transform synchronous algs. into (partially) asynchronous ones

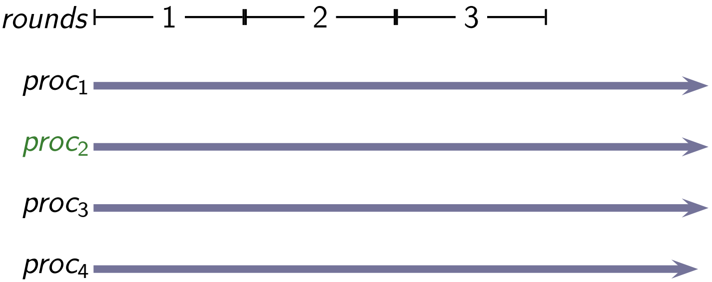
$\Rightarrow$ News ideas are needed

Wait-free **shared memory** renaming

- Explicit algorithm for $(\mathbf{2n-2})$-renaming

Synchronous renaming

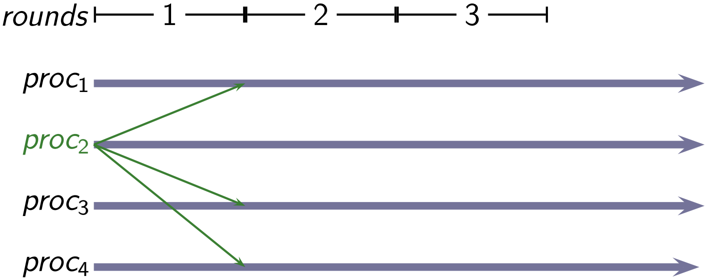
Round-by-round computation



In a round r, each proc.

- Sends messages to every procs.
- Receives messages sent in round r

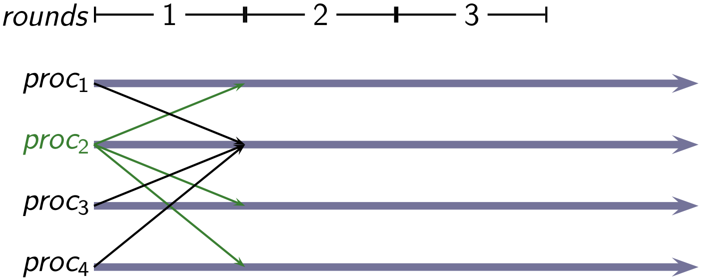
Round-by-round computation



In a round r, each proc.

- Sends messages to every procs.
- Receives messages sent in round r

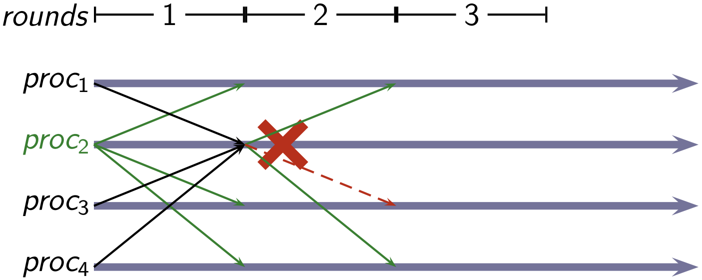
Round-by-round computation



In a round r, each proc.

- Sends messages to every procs.
- Receives messages sent in round r

Round-by-round computation



In a round r, each proc.

- Sends messages to every procs.
- Receives messages sent in round r

Complexity of synchronous renaming

Complexity = # **rounds** before decision

n procs renaming, tolerating up to $\mathbf{n-1}$ failures :

	Complexity	Remarks
Chaudhuri et al [CHT90]	$O(\log n)$	tight
Okun [Okun10]	$O(\log n)$	tight, order preserving

Lower bound :
$\boldsymbol{\Omega}(\log \mathbf{n})$ for tolerating $\mathbf{n-1}$ failures [CHT90]
holds for tight and loose renaming

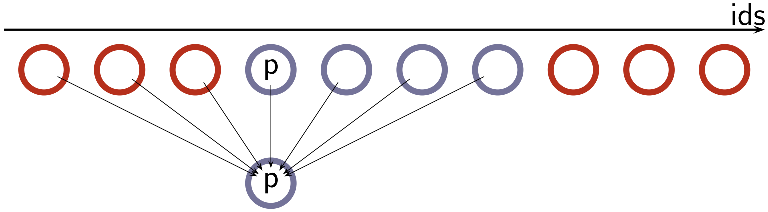
$\Omega(\log \mathbf{n})$ lower bound

[CHT 90]

- $2/3$ processes fail per round

$\mathbf{\Omega}(\log \mathbf{n})$ lower bound

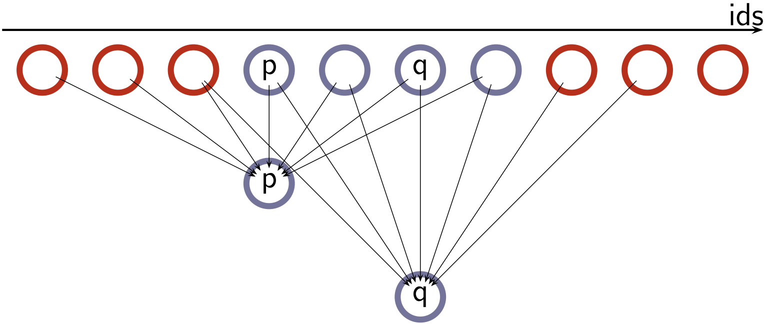
[CHT 90]



- 2/3 processes fail per round

$\Omega(\log \mathbf{n})$ lower bound

[CHT 90]



- 2/3 processes fail per round
- non-faulty proc. are in order equivalent states
  rank(id_p,ids rcved by p) $= 4 =$ rank(id_q,ids rcved by q)

Early Decision

Failures occur but are rare in practice

- Decide earlier when there are few failures
- Complexity = function of $\mathbf{f}$ actual #failures

Early deciding agreement:

- $\mathbf{O}(\mathbf{f})$ early deciding for consensus
- $\mathbf{O}(\frac{\mathbf{f}}{\mathbf{k}})$ *k*-set-agreement

Early Deciding Renaming

[Alistarh, Attiya, T. 2012]

Two early deciding renaming algorithms

	name-space	complexity	
Alg. 1	**loose** 1..2n	$\log f + 5$	
Alg. 2	**tight** 1..n	**cst**	for $f \leq \sqrt{n}$
		$5\log(f) + 10$	otherwise

n = # procs.
f = # failures

Early Deciding Renaming

[Alistarh, Attiya, T. 2012]

Two early deciding renaming algorithms

	name-space	complexity	
Alg. 1	**loose** 1..2n	$\log f + 5$	
Alg. 2	**tight** 1..n	**cst**	for $f \leq \sqrt{n}$
		$5\log(f) + 10$	otherwise

n = # procs.
f = # failures

Alg. 1	based on [CHT90]
Alg. 2	based on [Okun10]

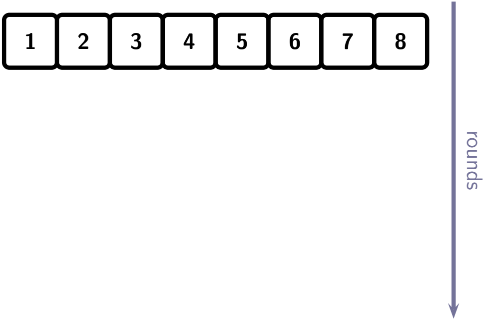
Algorithm 1

CHT 90 Renaming

- Tight name-space
- Complexity $O(\log n)$ rounds

CHT 90 Renaming

- Tight name-space
- Complexity $O(\log n)$ rounds

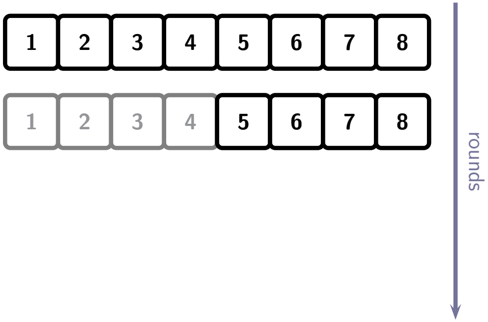


For each proc p

- **interval $\mathbf{I_p}$** of preferred names

CHT 90 Renaming

- Tight name-space
- Complexity $O(\log n)$ rounds

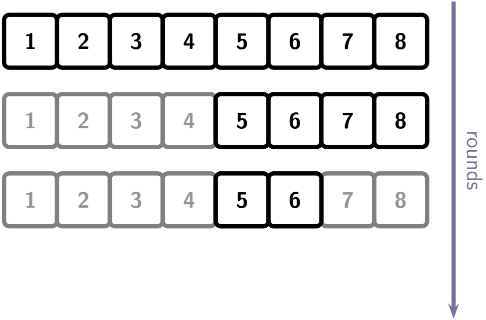


For each proc p

- **interval $\mathbf{I_p}$** of preferred names
- interval periodically **halved**

CHT 90 Renaming

- Tight name-space
- Complexity $O(\log n)$ rounds

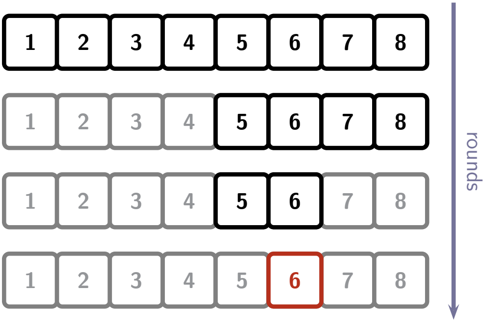


For each proc p

- **interval $\mathbf{I_p}$** of preferred names
- interval periodically **halved**

CHT 90 Renaming

- Tight name-space
- Complexity $O(\log n)$ rounds

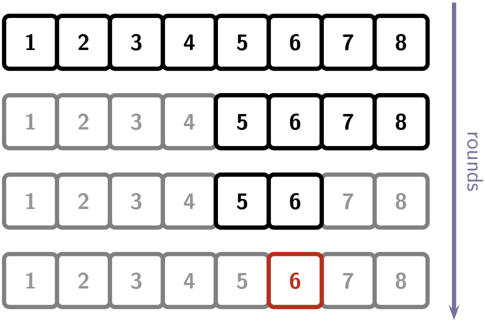


For each proc p

- **interval $\mathbf{I_p}$** of preferred names
- interval periodically **halved**
- **decision** when $|I_p| = 1$

CHT 90 Renaming

- Tight name-space
- Complexity $O(\log n)$ rounds



For each proc p

- **interval $\mathbf{I_p}$** of preferred names
- interval periodically **halved**
- **decision** when $|I_p| = 1$

Round complexity depends on initial size of $\mathbf{I_p}$

CHT analysis

Invariant 1

Preferences interval are well formed

$\forall \mathbf{I}, \mathbf{I}' : \mathbf{I} \cap \mathbf{I}' = \emptyset$ or $\mathbf{I} \subseteq \mathbf{I}'$ or $\mathbf{I}' \subseteq \mathbf{I}$

CHT analysis

Invariant 1

Preferences interval are well formed
$\forall \mathbf{I}, \mathbf{I}' : \mathbf{I} \cap \mathbf{I}' = \emptyset$ or $\mathbf{I} \subseteq \mathbf{I}'$ or $\mathbf{I}' \subseteq \mathbf{I}$

Invariant 2

For each preferences interval $\mathbf{I}$,
at most $|\mathbf{I}|$ procs with preferences $\subseteq \mathbf{I}$

CHT analysis

Invariant 1

Preferences interval are well formed
$\forall \mathbf{I}, \mathbf{I}' : \mathbf{I} \cap \mathbf{I}' = \emptyset$ or $\mathbf{I} \subseteq \mathbf{I}'$ or $\mathbf{I}' \subseteq \mathbf{I}$

Invariant 2

For each preferences interval $\mathbf{I}$,
at most $|\mathbf{I}|$ procs with preferences $\subseteq \mathbf{I}$

Complexity

- In each round, largest preferences intervals are **halved**
- Initial interval of the form $\mathbf{1..2^b}$ with $\mathbf{b} \leq \lceil \log \mathbf{n} \rceil$

Early-deciding CHT

Idea: carefully select initial preferences interval
based on an estimation of #failures in round 1

round 1 send $\mathbf{id_p}$ to all; rank $\mathbf{id_p}$ among ids received

$$\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow \textit{rank} = \mathbf{4}$$

Early-deciding CHT

Idea: carefully select initial preferences interval
based on an estimation of #failures in round 1

round 1 send $\mathbf{id_p}$ to all; rank $\mathbf{id_p}$ among ids received
$\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow \mathit{rank} = \mathbf{4}$
round 2 send $\langle \mathbf{id_p}, \mathbf{rank_p} \rangle$, pick initial preferences interval

Early-deciding CHT

Idea: carefully select initial preferences interval
based on an estimation of #failures in round 1

round 1 send $\mathbf{id_p}$ to all; rank $\mathbf{id_p}$ among ids received
$\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow \mathit{rank} = \mathbf{4}$
round 2 send $\langle \mathbf{id_p}, \mathbf{rank_p} \rangle$, pick initial preferences interval
round r ≥ 3 CHT algorithm

Initial preferences selection

round 1 *p* ranks its **id** among ids received

$\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow rank = \mathbf{4}$
$\{2, 6, \cancel{12}, \mathbf{34}, \mathbf{35}, 41\} \rightarrow rank = \mathbf{4}$
$\{2, \cancel{6}, \cancel{12}, \mathbf{34}, \mathbf{35}, \mathbf{41}\} \rightarrow rank = \mathbf{4}$

Initial preferences selection

round 1 p ranks its **id** among ids received

$$\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow rank = \mathbf{4}$$
$$\{2, 6, \cancel{12}, \mathbf{34}, \mathbf{35}, 41\} \rightarrow rank = \mathbf{4}$$
$$\{2, \cancel{6}, \cancel{12}, \mathbf{34}, \mathbf{35}, \mathbf{41}\} \rightarrow rank = \mathbf{4}$$

p has the *ith* id $\Rightarrow i - f_1 \leq rank_p \leq i \qquad f_1 = \#failures$ in rd 1

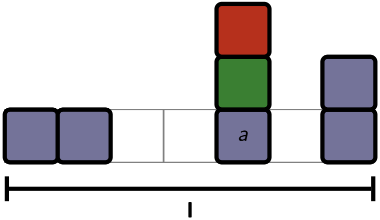
Initial preferences selection

round 1 p ranks its **id** among ids received

$$\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow rank = \mathbf{4}$$
$$\{2, 6, \cancel{12}, \mathbf{34}, \mathbf{35}, 41\} \rightarrow rank = \mathbf{4}$$
$$\{2, \cancel{6}, \cancel{12}, \mathbf{34}, \mathbf{35}, \mathbf{41}\} \rightarrow rank = \mathbf{4}$$

p has the ith id $\Rightarrow i - f_1 \leq rank_p \leq i$ $\quad f_1 = \#failures$ in rd 1

round 2 p receives a set of $\langle id, rank \rangle$



$\#ranks \in \mathbf{I} \leq |I| + f_1$

Initial preferences selection

round 1 p ranks its **id** among ids received

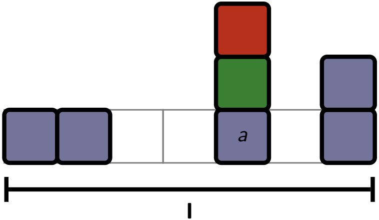
$\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow rank = \mathbf{4}$

$\{2, 6, \cancel{12}, \mathbf{34}, \mathbf{35}, 41\} \rightarrow rank = \mathbf{4}$

$\{2, \cancel{6}, \cancel{12}, \mathbf{34}, \mathbf{35}, \mathbf{41}\} \rightarrow rank = \mathbf{4}$

p has the *ith* id $\Rightarrow i - f_1 \leq rank_p \leq i \quad f_1 = \#failures$ in rd 1

round 2 p receives a set of $\langle id, rank\rangle$



$\#ranks \in \mathbf{I} \leq |I| + f_1$

$$\boxed{est_f = \max_I\{(\#ranks \in \mathbf{I}) - |I|\}}$$

Initial preferences selection cont'

Invariants

1. Well formed: any two intervals do not intersect or one is included in the other
2. At most $|\mathbf{I}|$ procs with preferences $\mathbf{I}' \subseteq \mathbf{I}$

For proc p, choose

- $j = \lceil \log est_f \rceil$
- $d : d2^j \leq rk_p \leq (d+1)2^j$

Preferences interval $J = \mathbf{d2^j ... (d+1)2^j}$?

well-formed

but # procs with rank in $J \leq |J| + \mathbf{f_1}$

$\boxed{J = \mathbf{d2^{j+1} ... (d+1)2^{j+1}}}$

Early-deciding CHT

Name-space	$1..2n$
Complexity	$5 + \log f_1$

where $f_1 = \#\textit{failures}$ in the first round

Algorithm 2

Okun Renaming

- Tight name space
- (Order preserving)
- Complexity $O(\log n)$
- Based on *approximate agreement*

Okun Renaming

- Ranking initial ids
  $\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow rank_{34} = \mathbf{4}$
  $\{2, 6, \cancel{12}, \mathbf{34}, \mathbf{35}, 41\} \rightarrow rank_{34} = \mathbf{3}$
  $\{2, \cancel{6}, \cancel{12}, \mathbf{34}, \mathbf{35}, \mathbf{41}\} \rightarrow rank_{34} = \mathbf{2}$

Okun Renaming

- Ranking initial ids
  $\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow rank_{34} = \mathbf{4}$
  $\{2, 6, \cancel{12}, \mathbf{34}, \mathbf{35}, 41\} \rightarrow rank_{34} = \mathbf{3}$
  $\{2, \cancel{6}, \cancel{12}, \mathbf{34}, \mathbf{35}, \mathbf{41}\} \rightarrow rank_{34} = \mathbf{2}$
- Associate an **agreement protocol** (AP) with each id

Okun Renaming

- Ranking initial ids
  $\{2, 6, 12, \mathbf{34}, 35, 41\} \rightarrow rank_{34} = \mathbf{4}$
  $\{2, 6, \cancel{12}, \mathbf{34}, \mathbf{35}, 41\} \rightarrow rank_{34} = \mathbf{3}$
  $\{2, \cancel{6}, \cancel{12}, \mathbf{34}, \mathbf{35}, \mathbf{41}\} \rightarrow rank_{34} = \mathbf{2}$
- Associate an **agreement protocol** (AP) with each id

id_1	id_2	id_3	id_4
$\boxed{\text{AP}}$	$\boxed{\text{AP}}$	$\boxed{\text{AP}}$	$\boxed{\text{AP}}$

Agree on the rank of each id $\rightarrow$ **new names**

Okun renaming

id_1 $\quad$ id_2 $\quad$ id_3 $\quad$ id_4

$\boxed{\text{AP}}$ $\quad$ $\boxed{\text{AP}}$ $\quad$ $\boxed{\text{AP}}$ $\quad$ $\boxed{\text{AP}}$

Algorithm sketch

phase 1 send $\mathbf{id}_p$ to all,
    V set of ids received
    $rank(id, V)$: rank of id in V

phase 2 Participate simultaneously in each AP_{id}
    with initial value $rank(id, V)$

decide Output of AP_{myid}

Okun renaming

id_1	id_2	id_3	id_4
$\boxed{\text{AP}}$	$\boxed{\text{AP}}$	$\boxed{\text{AP}}$	$\boxed{\text{AP}}$
2	**4**	**5**	**6**
1	**2**	**4**	**5**

Okun renaming

id_1	id_2	id_3	id_4
$\boxed{AP}$	$\boxed{AP}$	$\boxed{AP}$	$\boxed{AP}$
2	**4**	**5**	**6**
1	**2**	**4**	**5**

Parallel composition of *AP*

If for every proc. p, $prop_j - prop_i \geq \delta$ then $dec_j - dec_i \geq \delta$

Okun renaming

	id_1	id_2	id_3	id_4
	$\boxed{\text{AP}}$	$\boxed{\text{AP}}$	$\boxed{\text{AP}}$	$\boxed{\text{AP}}$
	2	**4**	**5**	**6**
	1	**2**	**4**	**5**
decisions	**2**	**4**	**5**	**6**

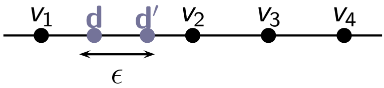
Parallel composition of *AP*

If for every proc. p, $prop_j - prop_i \geq \delta$ then $dec_j - dec_i \geq \delta$

Agreement protocol

- Consensus
  - decide one of the proposed value (validity)
  - no two processes decide differently (agreement)
  - no non-faulty proc never decide (termination)

  Complexity **$O(f)$**
- Approximate agreement with parameter ϵ



Okun renaming

$$\begin{array}{cccc} id_1 & id_2 & id_3 & id_4 \\ \boxed{\epsilon\text{-AA}} & \boxed{\epsilon\text{-AA}} & \boxed{\epsilon\text{-AA}} & \boxed{\epsilon\text{-AA}} \end{array}$$

- final rank of id_i = output of AA rounded to nearest integer
- ϵ small enough such that no two distinct ids receive same rank after rounding

Complexity = complexity of the AA protocol

Okun renaming

id_1	id_2	id_3	id_4
$\boxed{\epsilon\text{-AA}}$	$\boxed{\epsilon\text{-AA}}$	$\boxed{\epsilon\text{-AA}}$	$\boxed{\epsilon\text{-AA}}$

- final rank of id_i = output of AA rounded to nearest integer
- ϵ small enough such that no two distinct ids receive same rank after rounding

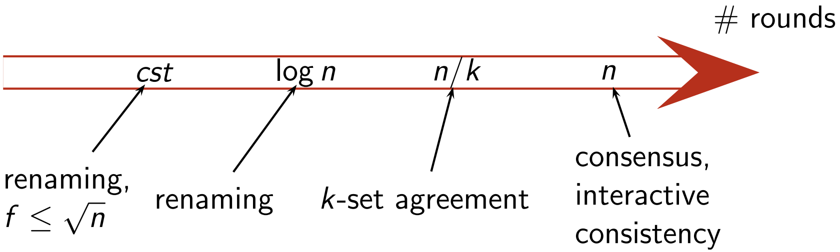
Complexity = complexity of the AA protocol

[Alistarh, Attiya T.]

Finer analysis of the complexity of AA, function of actual # failures

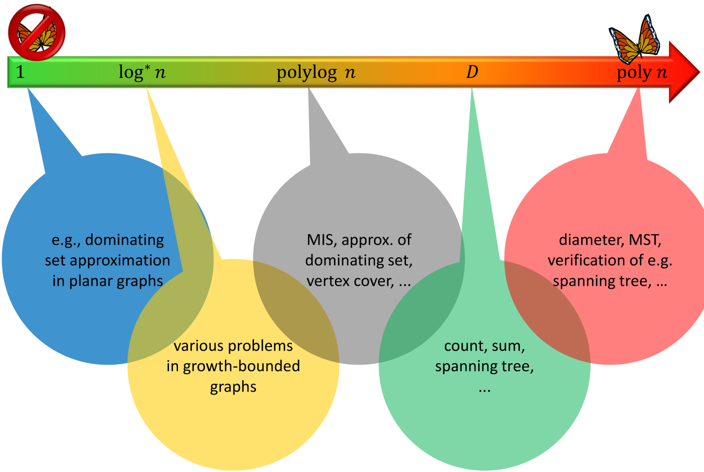
- **cst** (≤ 10) when $f \leq \sqrt{n}$
- $O(\log f)$ otherwise

Synchronous Complexity



Thanks!

Distributed Complexity Classification



[Wattenhofer, Sirocco 2012 Prize Lecture]