The Space Complexity of Unbounded Timestamps

Faith Ellen, University of Toronto
Panagiota Fatourou, University of Ioannina
Eric Ruppert, York University

Timestamp System

a set of timestamps U

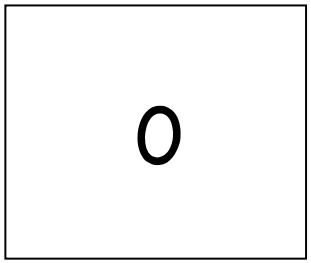
two operations:

- GetTS() returns a timestamp
- Compare(t,t′) returns a Boolean value, indicating which of the timestamps t or t′ was generated first

Timestamps have been used to:

- solve mutual exclusion
- solve randomized consensus
- construct multi-writer registers from single-writer registers
- construct snapshot objects and other data structures

A very simple timestamp system



counter
C

U = N

GetTS():

t ← Fetch&Increment(C)

return (t)

Compare(t,t′):

return (t < t′)

Timestamps can grow without bound during an execution.

This is necessary to describe the ordering among an unbounded number of non-concurrent events.

For some applications, Compare is restricted to recent events, so old timestamps can be reused.

A timestamp system is:

unbounded, if U is infinite, and

bounded, if U is finite.

Bounded Timestamp Systems:

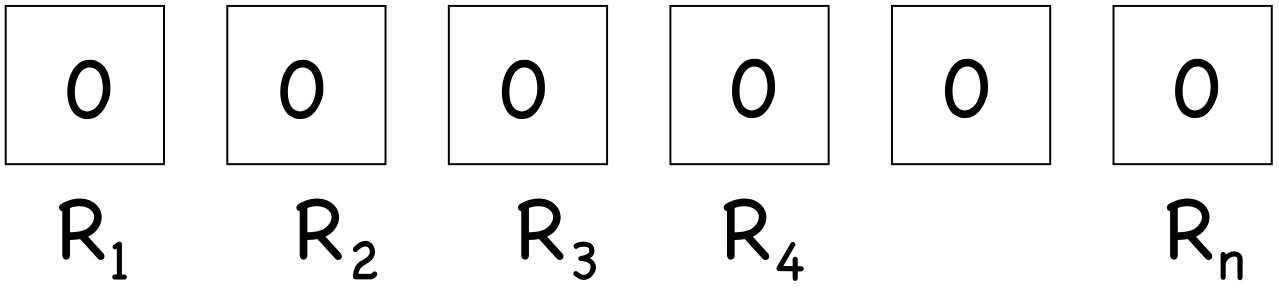
Dolev and Shavit [1997],
Dwork and Waarts [1999],
Dwork, Herlihy, Plotkin, and Waarts [1999],
Haldar and Vitanyi [2002],
Israeli and Li [1993],
Israeli and Pinhasov [1992]

Theorem [Israeli and Li] Any bounded timestamp system shared by n processes must use $\Omega(n)$ bits per timestamp.

Unbounded timestamps can have length logarithmic in the number of GetTS operations performed.

If the number GetTS operations is reasonable, for example less than 2^{64}, then timestamps can fit in a single memory word.

A simple timestamp system using single-writer registers [Lamport, 1974]



U = N

GetTS() by process p_i:

$t \leftarrow 1 + \max\{R_1,\ldots,R_n\}$

$R_i \leftarrow$ write(t)

return (t)

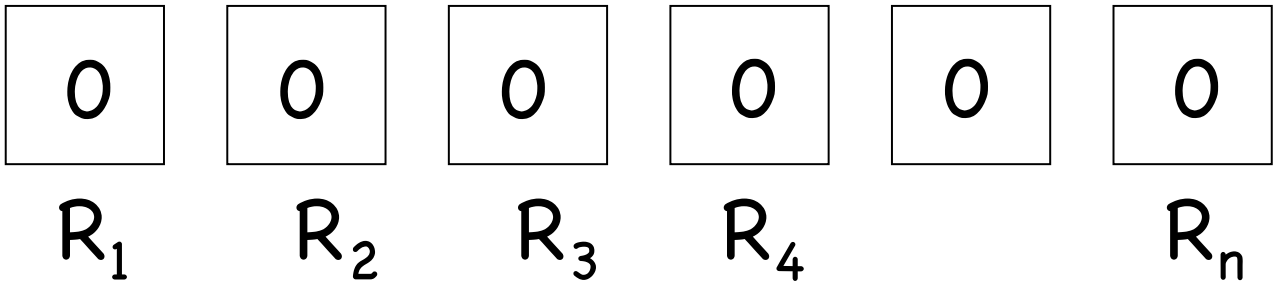
Compare(t,t'):

return (t < t')

If some GetTS operation returns t and another GetTS, which starts after it is complete, returns t′, then Compare(t,t′) = 1.

To ensure that timestamps obtained by different GetTS operations in an execution are different, append the log n bit process ID to the result.

A simple timestamp system using single-writer registers [Dwork and Waarts, 1999]



$U = N^n$

GetTS() by process p_i:

$R_i \leftarrow$ write($1+R_i$)

$t \leftarrow$ read($R_1,\ldots,R_n$)

return (t)

each timestamp is a vector of n integers

Compare(t,t′) can be done

- lexicographically or

  $t < t'$ if there exists k such that $t_i = t_i'$ for $i < k$ and $t_k < t_k'$

- component-wise

  $t < t'$ if $t_i \leq t_i'$ for $i = 1,..,k$ and there exists k such that $t_k < t_k'$

Both these timestamp implementations use n shared registers.

Is it possible to implement an unbounded timestamp system using fewer registers?

Model

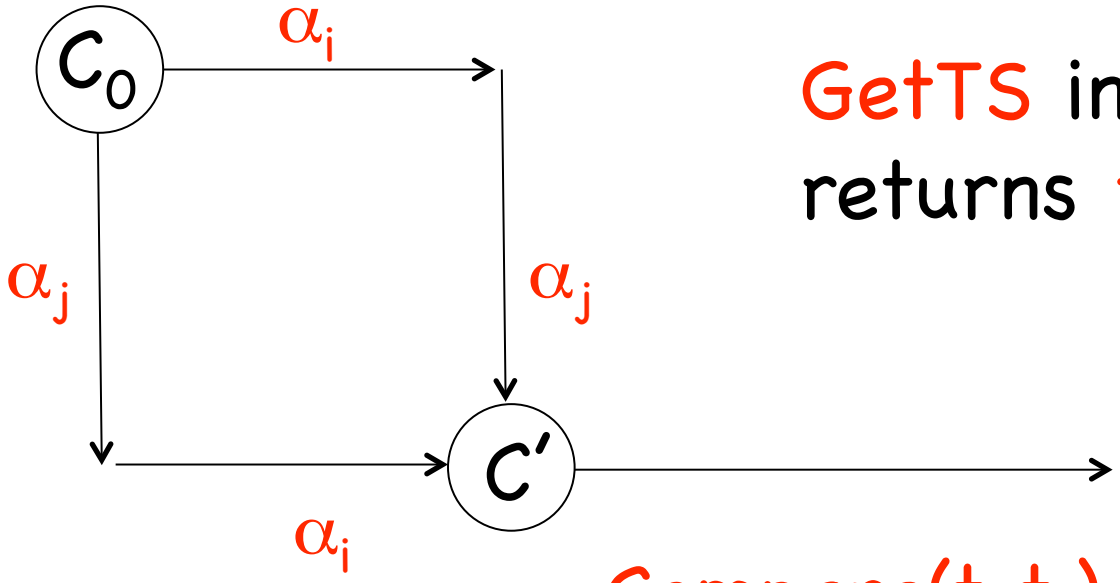
n deterministic asynchronous processes, $P_1, \ldots, P_n$

r shared registers, $R_1, \ldots, R_r$

obstruction free: each GetTS and Compare operation must complete if it is given sufficiently many consecutive steps.

Theorem Every single-writer timestamp implementation for n processes uses at least $n-1$ registers.

Proof Let α_i be the solo execution of GetTS by process p_i starting from C_0. To obtain a contradiction, suppose there exist p_i and p_j that perform no writes during α_i and α_j, respectively.



GetTS in α_i
returns t_i

Compare(t_i, t_j)
by $q \neq p_i, p_j$

Theorem Every timestamp implementation for n processes uses more than $\sqrt{n-1} / 2$ registers.

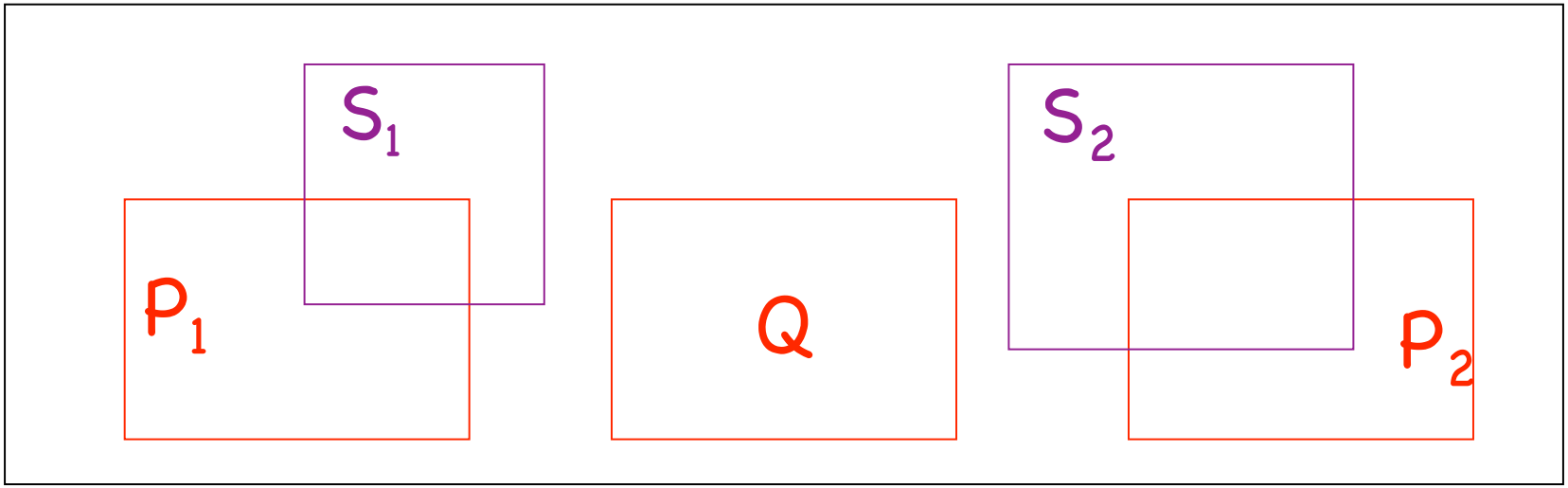
Covering Argument

A process p_i covers a register R_j in a configuration if p_i will write to R_j when it next takes a step.

A set P of k processes covers a set R of k registers if each register in R is covered by some process in P.

If P covers R, a block write by P consists of a consecutive sequence of writes, one by each process in P.

Suppose that, in configuration C, P_1, P_2, Q each cover the set of registers R; P_1, P_2, Q are disjoint; S_1, S_2 are disjoint; S_1, $(P_2 \cup Q)$ are disjoint; and $S_2, (P_1 \cup Q)$ are disjoint.

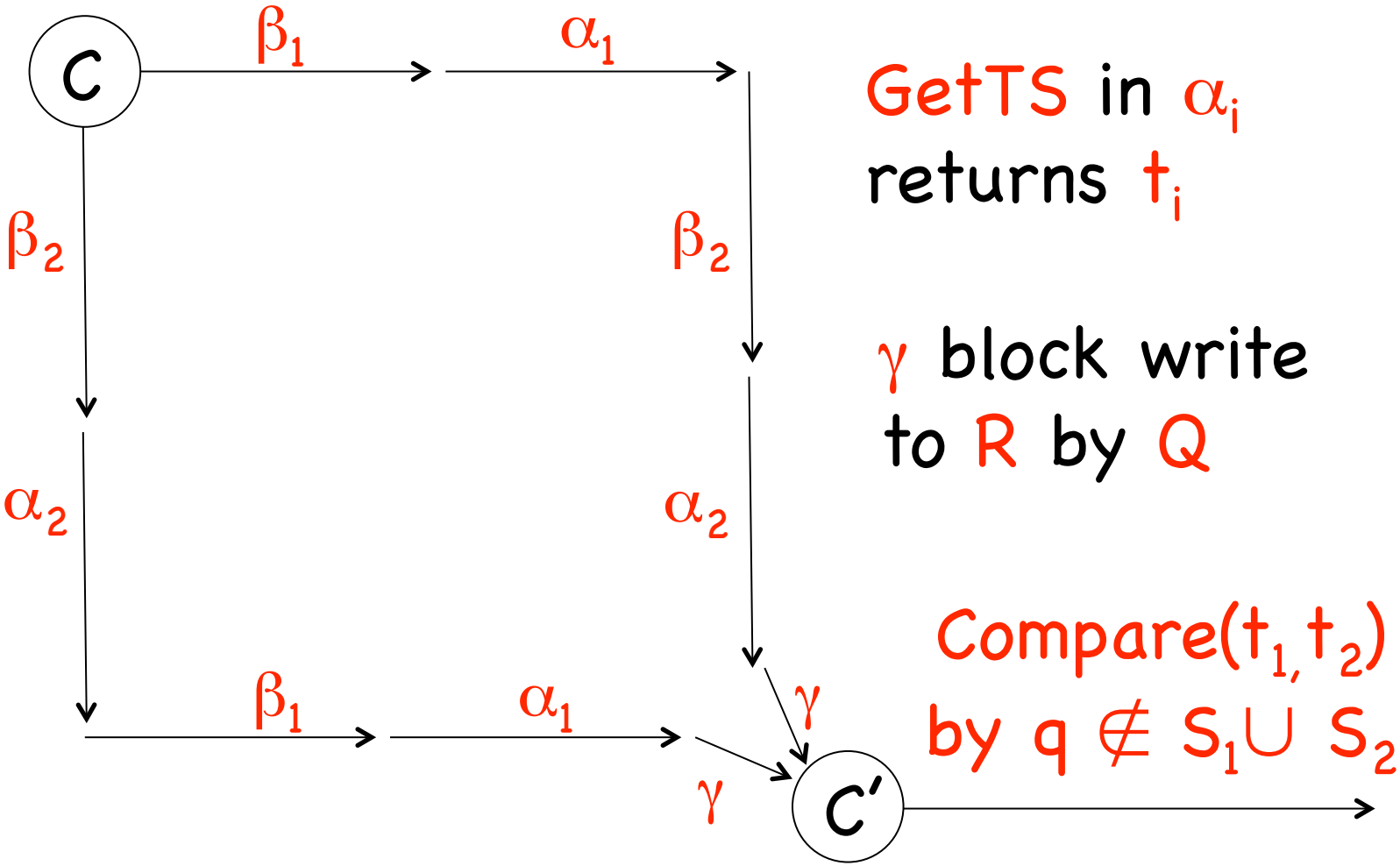


Lemma Suppose that, in configuration C, P_1, P_2, Q each cover the set of registers R; P_1, P_2, Q are disjoint; S_1, S_2 are disjoint; S_1, $(P_2 \cup Q)$ are disjoint; and $S_2, (P_1 \cup Q)$ are disjoint.

Let β_1, β_2 be block writes by P_1, P_2.

Then, for some $i \in \{1,2\}$, all S_i-only executions α_i starting from $C\beta_i$ containing a complete GetTS contain a write to a register not in R.

Proof Suppose not. Then α_i, β_i only write to R.

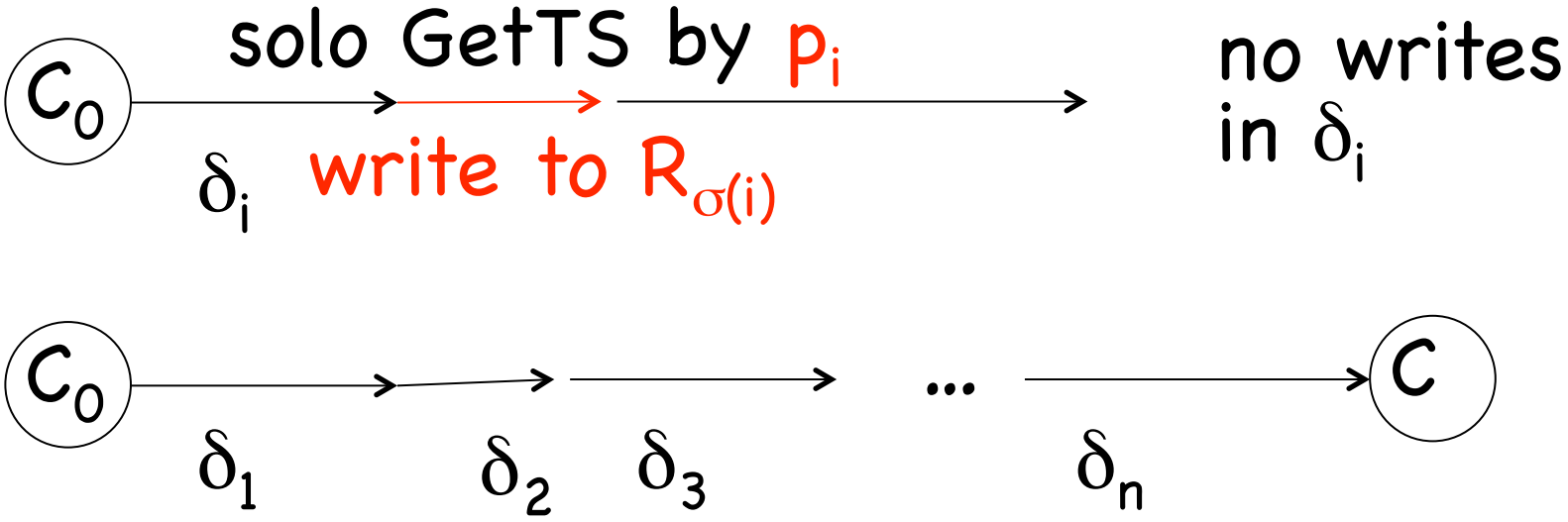


Theorem Every timestamp implementation for n processes uses more than $\sqrt{n-1} / 2$ registers.

Proof Suppose there is an implementation using $r \leq \sqrt{n-1} / 2$ registers. Then, for $k=1, \ldots, r$, there is a configuration in which k registers are each covered by $r-k+3$ processes (by induction, using the lemma). When $k=r$, apply the lemma again. This is a CONTRADICTION since no more registers exist.

Base Case: k = 1

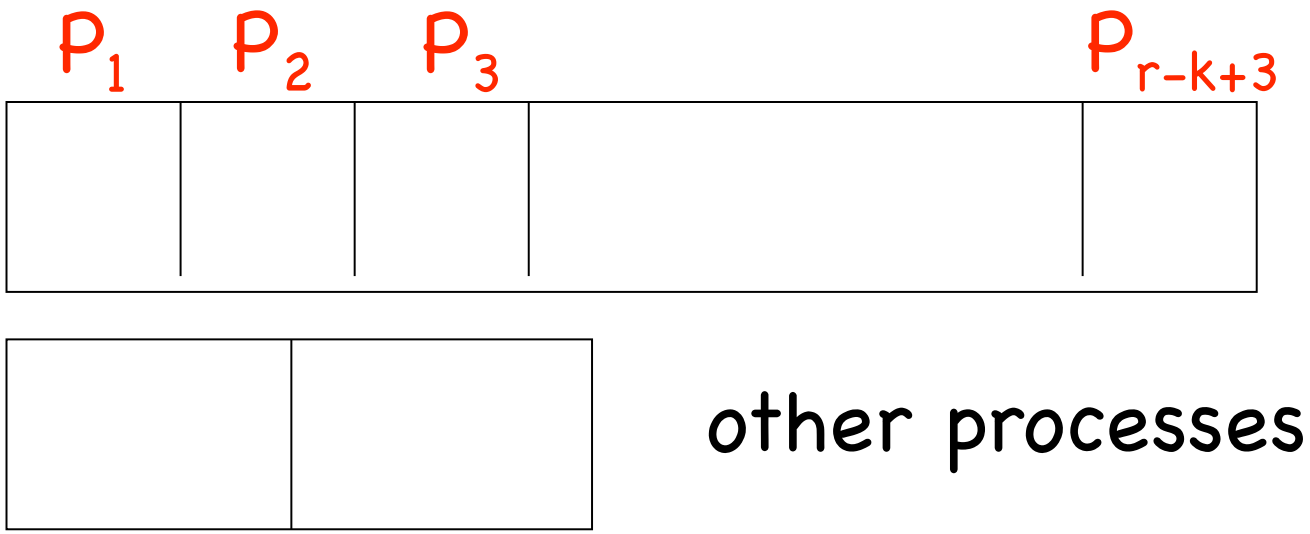
By the lemma with $R = \phi$, for every process p_i, except possibly one,



At least $n-1 > r(r+1)$ processes covering r registers $\Rightarrow$ some register is covered by at least $r+2 = r - k + 3$ processes.

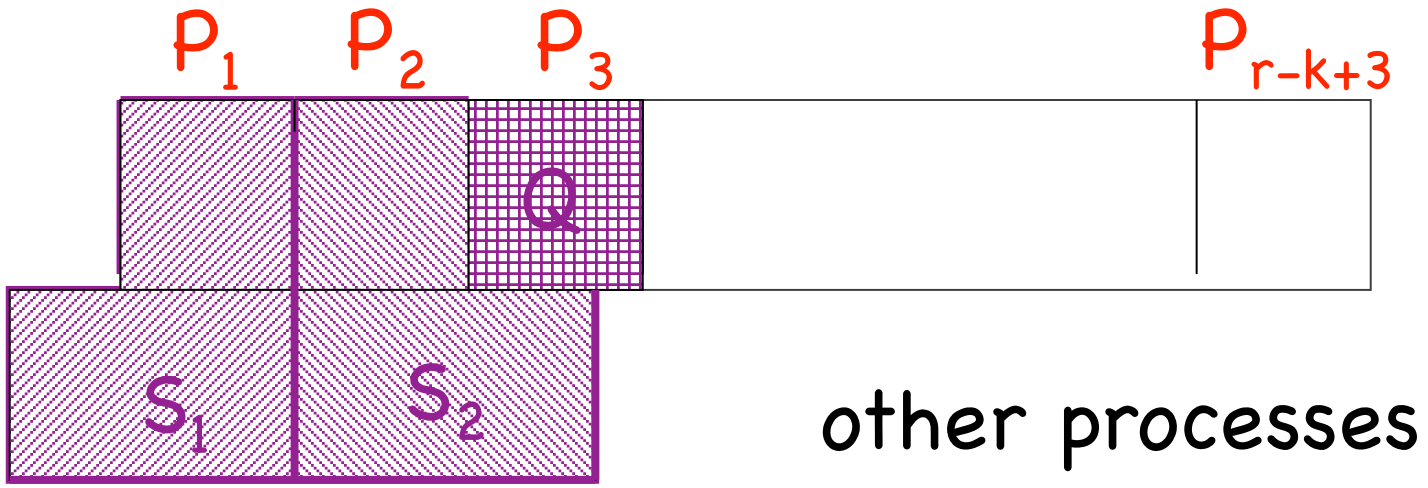
Induction Step:

Consider a configuration C with P_1, ...,P_{r-k+3} disjoint sets of processes each covers R, a set of k registers

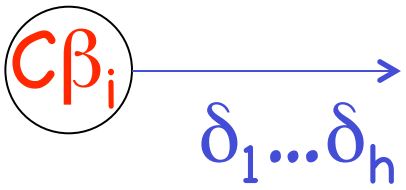


Induction Step:

Consider a configuration C with P_1, ...,P_{r-k+3} disjoint sets of processes each covers R, a set of k registers



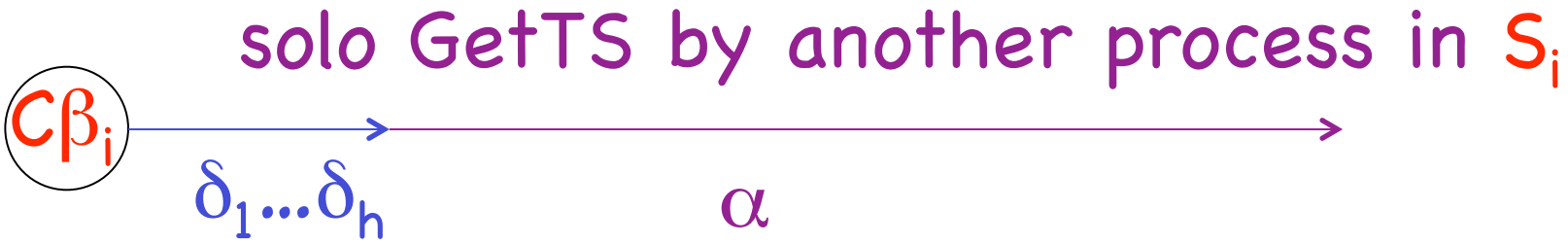
By the lemma, for some $i \in \{1,2\}$, every S_i-only execution starting from $C\beta_i$ containing a complete GetTS contains a write to a register not in R.



$\delta_1, \ldots, \delta_h$ solo executions by h different processes in S_i that only write to R.

At $C\beta_i\delta_1 \ldots \delta_h$ these processes cover registers not in R.

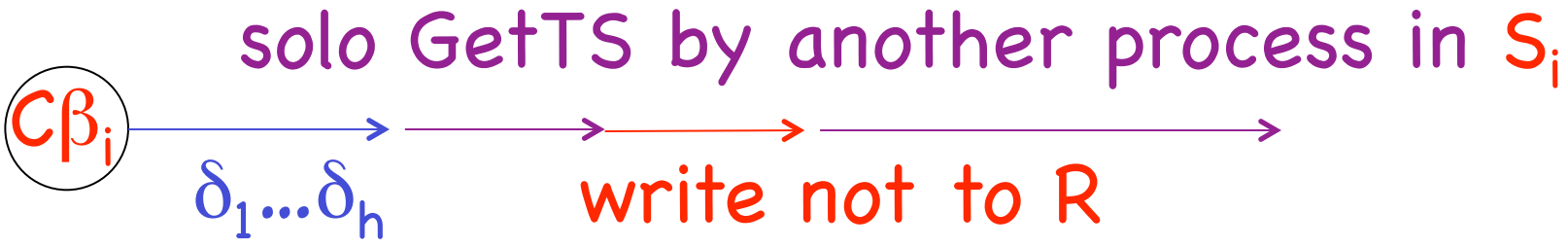
By the lemma, for some $i \in \{1,2\}$, every S_i-only execution starting from $C\beta_i$ containing a complete GetTS contains a write to a register not in R.



$\delta_1, \ldots, \delta_h$ solo executions by h different processes in S_i that only write to R.

At $C\beta_i\delta_1 \ldots \delta_h$ these processes cover registers not in R.

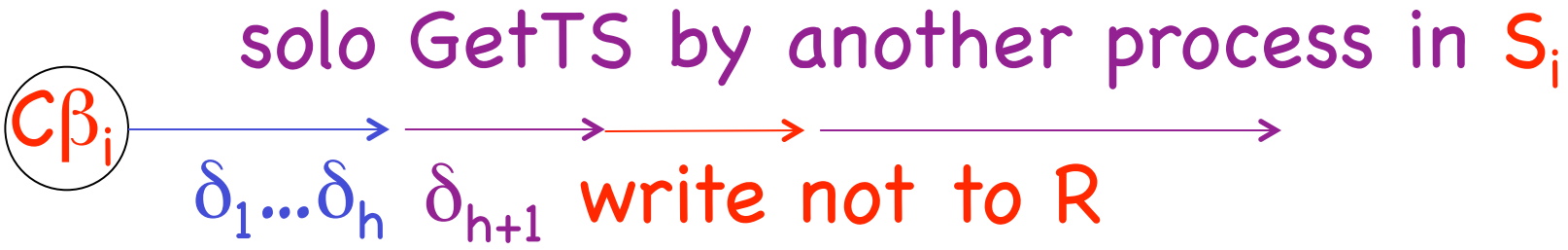
By the lemma, for some $i \in \{1,2\}$, every S_i-only execution starting from $C\beta_i$ containing a complete GetTS contains a write to a register not in R.



$\delta_1, \ldots, \delta_h$ solo executions by h different processes in S_i that only write to R.

At $C\beta_i\delta_1 \ldots \delta_h$ these processes cover registers not in R.

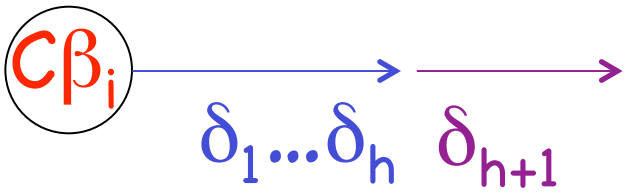
By the lemma, for some $i \in \{1,2\}$, every S_i-only execution starting from $C\beta_i$ containing a complete GetTS contains a write to a register not in R.



$\delta_1, \ldots, \delta_h$ solo executions by h different processes in S_i that only write to R.

At $C\beta_i\delta_1 \ldots \delta_h$ these processes cover registers not in R.

By the lemma, for some $i \in \{1,2\}$, every S_i-only execution starting from $C\beta_i$ containing a complete GetTS contains a write to a register not in R.



$\delta_1, \ldots, \delta_h, \delta_{h+1}$ solo executions by h+1 different processes in S_i that only write to R.

At $C\beta_i\delta_1 \ldots \delta_h\delta_{h+1}$ these processes cover registers not in R.

δ solo executions by all processes in S_i that only write to R.

At $C\beta_i\delta$ these processes cover registers not in R.

By a simple counting argument, at $C\beta_i\delta$ some register $R_j \notin R$ is covered by at least $r-k+2$ processes.

All registers in R are covered by $r-k+2 = r-(k+1) + 3$ processes.

For $k=1,\ldots,r$, there is a configuration in which k registers are each covered by $r-k+3$ processes.

Theorem Every timestamp implementation for n processes uses more than $\sqrt{n-1} \,/\, 2$ registers.

U partially ordered universe under $<$

Compare(t_1,t_2) = 1 if and only if $t_1 < t_2$

U is nowhere dense if for all $x,y \in U$, there are only a finite number of elements $z \in U$ such that $x < z < y$.

Examples

integers under $<$

set of all finite sets of integers under $\subset$

Another Example

set of length k vectors of integers ordered component-wise:

$(u_1,\ldots,u_k) \leq (v_1,\ldots,v_k)$ if and only if $u_i \leq v_i$ for $i = 1,\ldots,k$.

Between $(1,2,1,3)$ and $(1,4,2,3)$ there are:
$(1,2,2,3)$, $(1,3,1,3)$, $(1,3,2,3)$, $(1,4,1,3)$

Not an Example

-Rational numbers

-set of length k vectors of integers ordered lexicographically.

Between (1,0) and (2,0) there are an infinite number of elements:
(1,1), (1,2), (1,3), (1,4),...

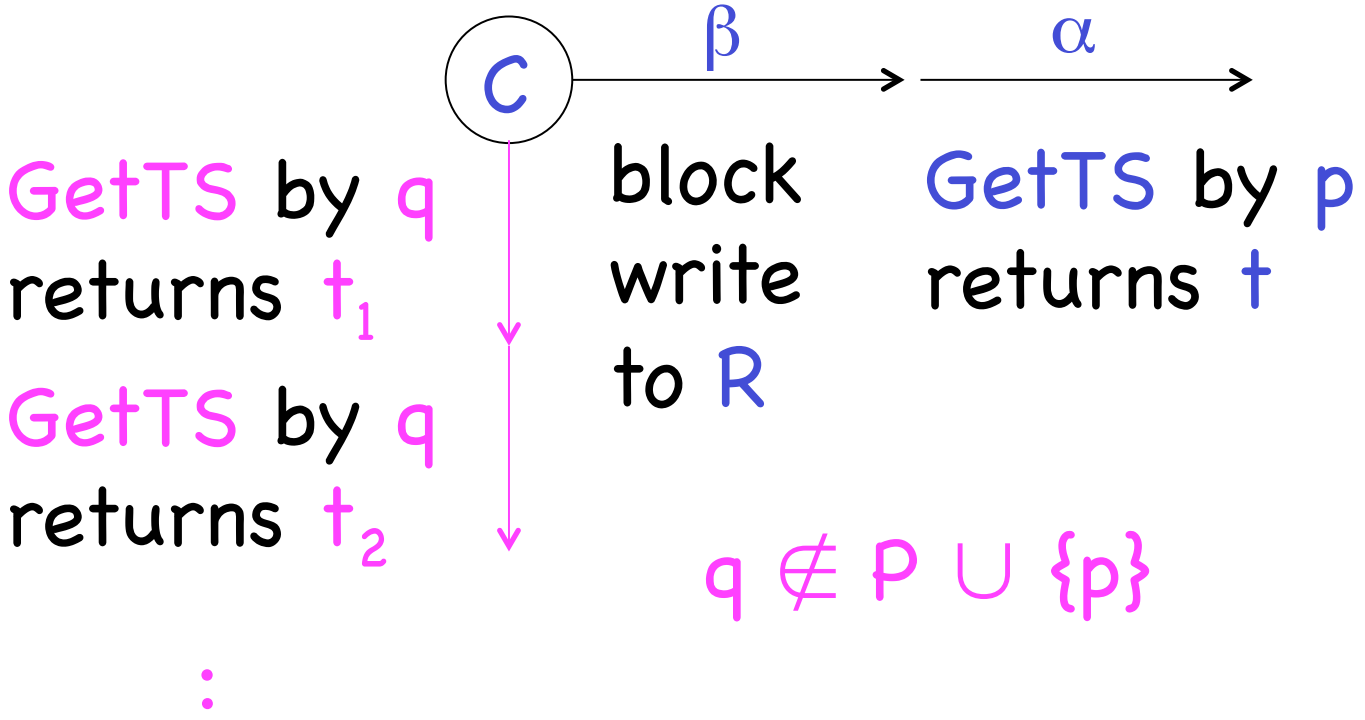
Theorem Every timestamp implementation for n processes that uses a nowhere dense universe requires at least n registers.

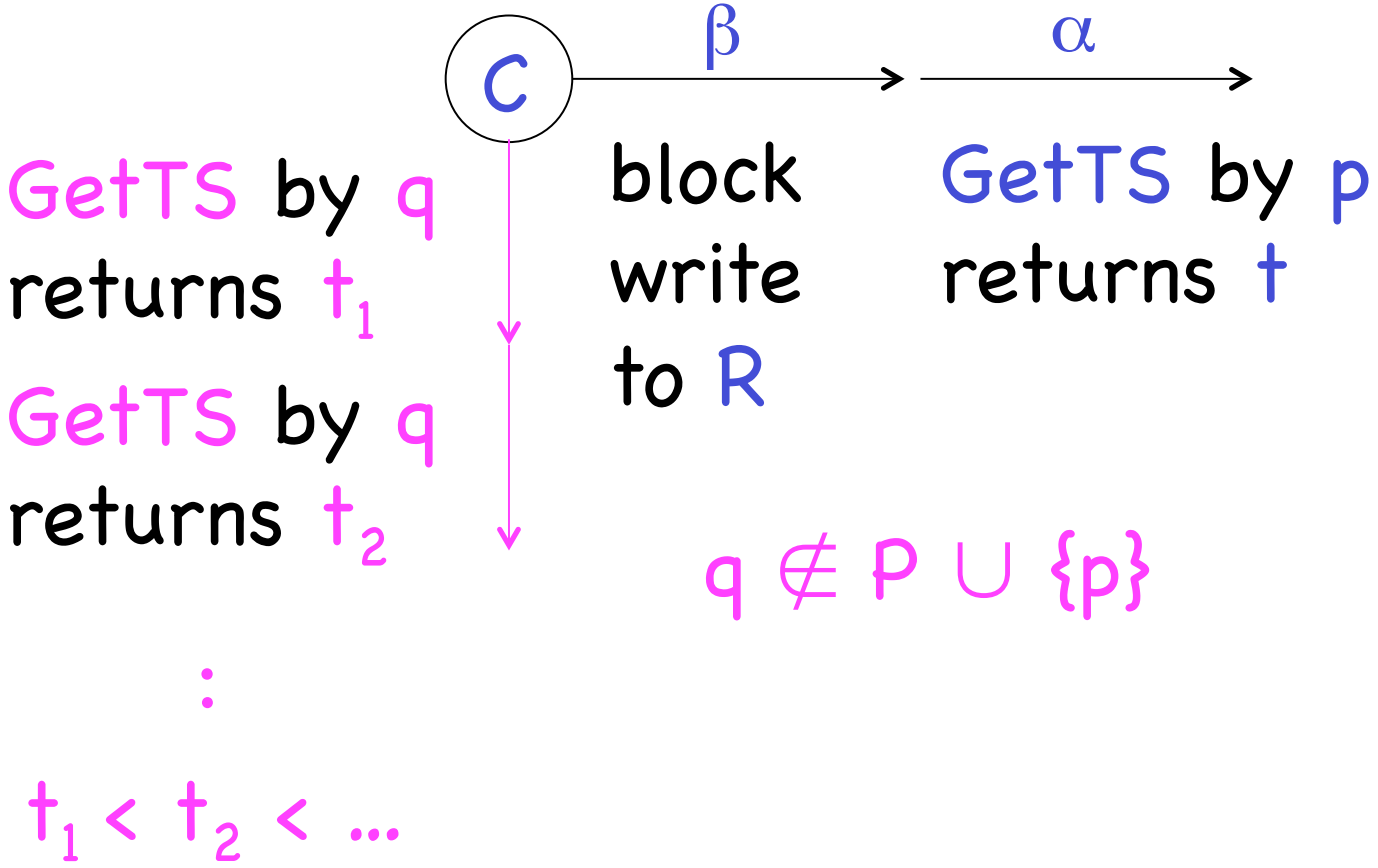
Proof: For $k=0,\ldots,n$, there is a configuration in which k registers are covered.

Base Case: The initial configuration.

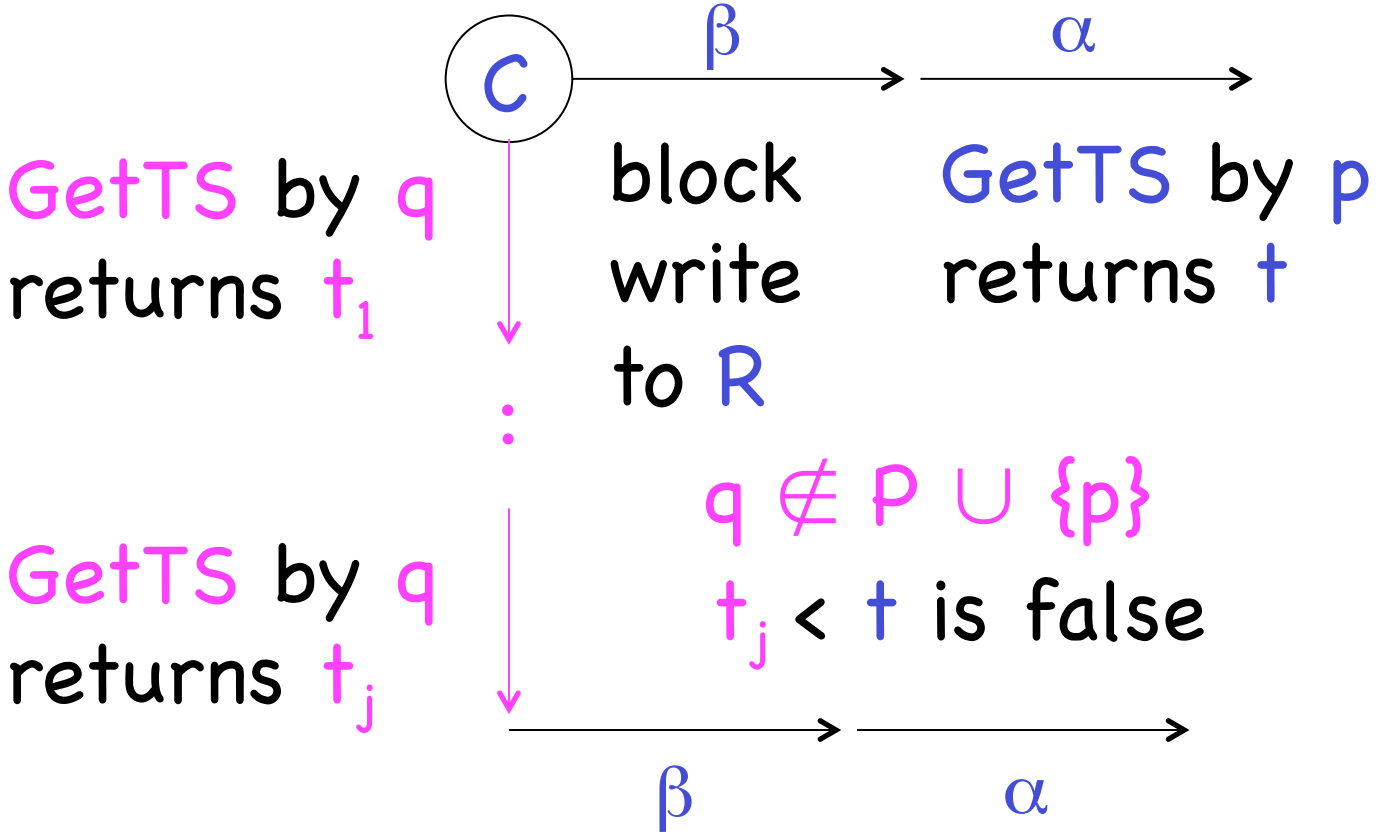
Induction Step:

Consider a configuration C where a set R of $k < n$ registers is covered by a set of k processes P.

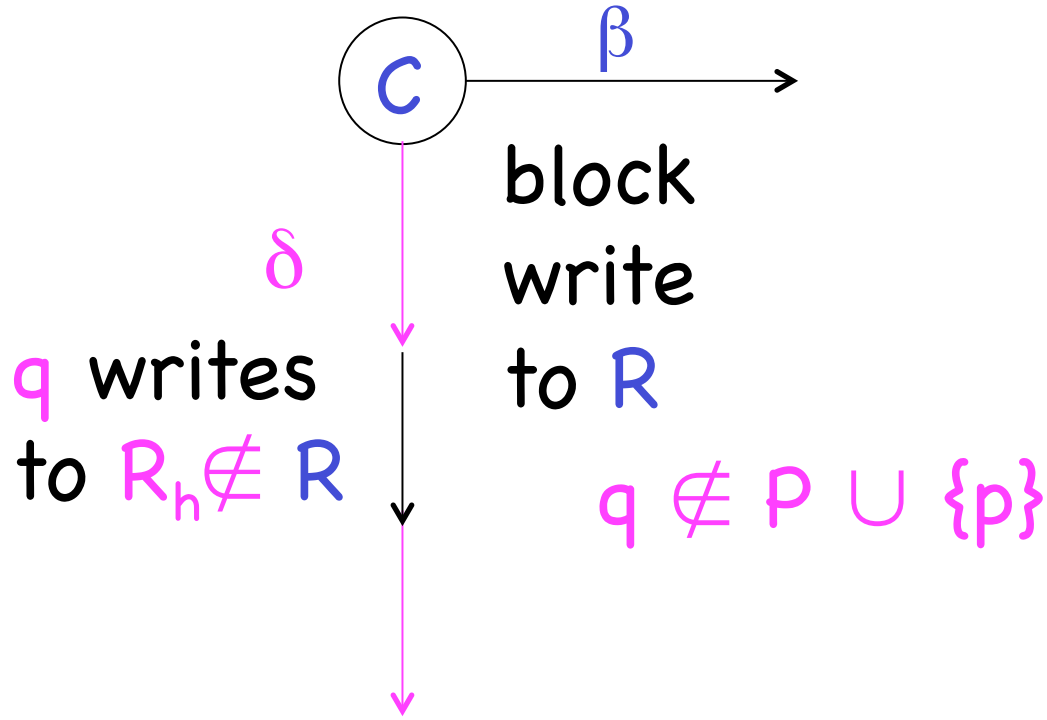




There exists t_j such that $t_j < t$ is false; otherwise there are infinitely many elements of U between t_1 and t.



If q writes only to registers in R



In configuration $C\delta$, a set $R \cup \{R_h\}$ of $k+1$ registers is covered by a set of $k+1$ processes $P \cup \{q\}$.

By induction, there is a configuration in which all n registers are covered.

Theorem Every timestamp implementation for n processes that uses a nowhere dense universe requires at least n registers.

Theorem There is a wait-free timestamp implementation that uses $n-1$ single-writer registers.

$U = N \times N$, ordered lexicographically

GetTS() by process p_i, $i < n$:

$t \leftarrow 1 + \max\{R_1,\ldots,R_{n-1}\}$

$R_i \leftarrow$ write(t)

return (t,0)

GetTS() by process p_i, $i < n$:

$t \leftarrow 1 + \max\{R_1,\ldots,R_{n-1}\}$

$R_i \leftarrow$ write(t)

return (t,0)

GetTS() by process p_n:

$t \leftarrow \max\{R_1,\ldots,R_{n-1}\}$

if $t >$ oldt then $c \leftarrow 0$

$c \leftarrow c + 1$

oldt $\leftarrow t$

return (t,c)

GetTS() by process p_i, $i < n$:

$t \leftarrow 1 + \max\{R_1,\ldots,R_{n-1}\}$

$R_i \leftarrow$ write(t)

return (t,0)

GetTS() by process p_n:

$t \leftarrow \max\{R_1,\ldots,R_{n-1}\}$

if $t >$ oldt then $c \leftarrow 0$

$c \leftarrow c + 1$

oldt $\leftarrow t$

return (t,c)

if GetTS returns (t,c) and then another GetTS begins and returns (t′,c′), then $(t,c) < (t',c')$

Theorem Every timestamp implementation for n processes uses more than $\sqrt{n-1} / 2$ registers.

Theorem Every timestamp implementation for n processes that uses a nowhere dense universe requires at least n registers.

Theorem There is a timestamp implementation for n processes that uses $n-1$ single-writer registers and no such implementation uses fewer registers.

Open Problem

What is the minimum number of registers needed to implement a timestamp system?

If there is a system that uses fewer than n-1 registers, it must use some multi-writer registers and a universe that is NOT nowhere dense.