Demon

in distributed systems

Colette Johnen
http://www.labri.fr/~johnen

Distributed System [Tel]

A distributed system is an interconnected collection of autonomous computers, processes, or processors.

The computers, processes, or processors are referred to as the nodes of the distributed system.

To be qualified as autonomous, the nodes must at least be equipped with their own private control.

To be qualified as interconnected, the nodes must be able to exchange information.

Model of distributed System

S : Distributed System S = (G + A)

G : Communication Graph – connected and simple
G = (V , E) (| V |= n, | E |= m)

A : distributed Algorithm : each node performs a local code

Each node has its own memory, code, clock

Computation Model : State Model

Computation Model (1)

State of a node :

The values of its variables

A node can only modify the value of its own variables

Local configuration of a node (communication model) :

Its state and the state of its neighbors

Configuration of the system :

The set of node states

Computation Model (2)

Each node **p** has its own code : a set of guarded rules

A rule has the following pattern :

R : Guard → Action

Guard : predicate on local configuration of the node

Action : modification of its state (value of **p**'s variables)

In a global configuration c, a rule **R** is said enabled by **p** if its **R**'s guard is verified by **p**

Computation Model (3)

During a computation step, some nodes simultaneously execute an enabled rule (updating their local variables) : $c_0\{p1p2\}c_1$

Computation is a sequence of consecutive computation steps

What computations are realistic ?

Demon – Undeterminism oracle

at each step of the computation, the Demon/Scheduler chooses some enabled nodes that will perform an rule action

- synchronous demon "chooses" all enabled nodes
- fair demon : along a computation, a node always enabled performs infinitely often a rule
- central demon chooses a single node
- distributed demon chooses a subset of enabled nodes

An execution is a game between an algorithm and a scheduler (non-deterministic adversary)

Demon for deterministic distributed systems

S = (G+A) a deterministic distributed system

A Demon is a set of computations on **S**

$(c1p2c2,c2p4c3,c3p1c4,c4p3c5,c5p5c1)^*$
$(c1p2c1)^*$

The unfair distributed demon UDD contains all computations

The synchronous demon contains a single computation starting on a given configuration **c1**

Fairness of the demon

- Weak-fair demon : a node continuously enabled (never disabled) executes infinitely often a rule
- Strong fair demon : a node infinitely often enabled performs infinitely often a rule
- k-bounded demon : till a node **p** stay enabled, any other node perform at most **k** action.

Deterministic Token Circulation on **Odd** size Unidirectional Rings

DTC: $(v_p = v_l) \rightarrow v_p := v_l + 1 \bmod (2)$

Each node, **p**, has a boolean variable v_p .

l is the left neighbor of **p**.

Computations under any demon of **DTC** are k-fair : between two actions of a node **p** any other node performs at most k=**|V|** actions.

[1990, T. Herman] : Probabilistic Token Circulation on **Odd** size Unidirectional Rings

$Token_p \rightarrow$ **to flip a coin,** ***if*** **« head »** ***then*** $Pass_Token_p$

Each node, **p**, has a boolean variable v_p .
l is the left neighbor of **p**.

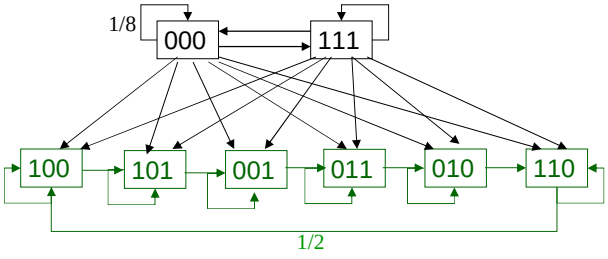
$Token_p \equiv (v_p = v_l)$
$Pass_Token_p \equiv v_p := v_l + 1 \bmod (2)$

Synchronous Scheduler

Herman's Alg under synchronous scheduler

$(v_p = v_l) \rightarrow$ **to flip a coin,** ***if*** **« head »** ***then*** $v_p := v_l + 1 \bmod (2)$

Markov Chain :



Herman Alg.

- Configuration where there are several tokens are transient state in Markov Chain
- Example :
  - i = [000], $\mathbf{Pr(T_i < \infty) = \sum_{n=1} 2^{-3n} < 1}$
- Expectation time of a token circulation is not the expectation of the hitting time to **j** (a configuration with one token) : $\sum_{n=1} \mathbf{nf_{jj}^n}$

Herman's Alg under synchronous scheduler

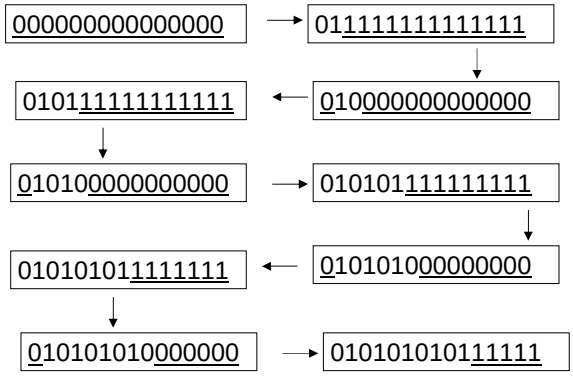
$\mathbf{(v_p = v_l) \rightarrow}$ **to flip a coin,**
if **« head »** ***then*** $\mathbf{v_p := v_l + 1 \bmod 2}$

In any ring of size **n** (**n** being odd)
From any configuration **c**, there is scenario **sc**
- converging with probability $\geq \mathbf{2^{-t.n} \geq 2^{-n.n}}$ to a configuration with a single token where **t** is the number of tokens in **c**
- in at most **n** computation steps

Herman' Alg is a « Las Vegas » Algorithm : the output is correct, the expectation time of convergence is $\leq \mathbf{n.2^{n.n}}$

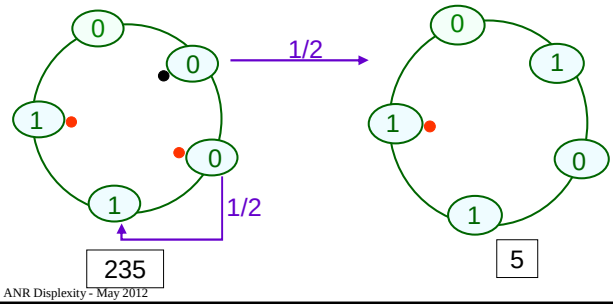
n=15

Convergence of Herman Alg



Herman's Alg under any scheduler

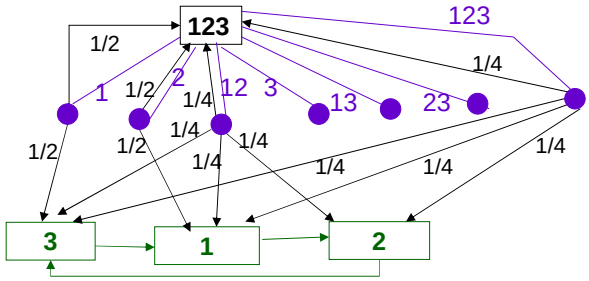
$\mathbf{(v_p \neq v_l + 1 \bmod (M_N)) \rightarrow}$ **to flip a coin,**
if **« head »** ***then*** $\mathbf{v_p := v_l + 1 \bmod (M_N)}$



Herman's Alg under any scheduler

Herman' Algo on ring of size 3

$\mathbf{(v_p \neq v_l + 1 \bmod (M_N)) \rightarrow}$ **to flip a coin,**
if **« head »** ***then*** $\mathbf{v_p := v_l + 1 \bmod (M_N)}$



Markov Decision Process : (**S,Act,A,p**)

- **S** is the set of states
- **Act** is the set of actions
- **A(s)** is the finite set of actions available at s
- **p(s,a,t)∈[0,1]** is the probability of the transition from **s** to **t** when the action a is selected.

$\mathbf{\forall s \in S,\ \forall t \in S\ p(s,a,t)=0 \text{ if } a \notin A(s)}$
$\mathbf{\forall s \in S, \forall a \in A(s),\ \sum_{t \in S} p(s,a,t)=1}$

Markov Decision Process : (S,Act,A,p) in distributed system

- **S** is the set of configurations

  S ={[000],[111],[011],[100],[101], [010], [110], [001]}

- **Act** contains all subsets of nodes set

  Act ={{p1},{p2},{p3},{p1p2},{p2p3}, {p3p1},{p2p3}, {p1p2p3}}

Markov Decision Process : (S,Act,A,p) in distributed system

- **A(c)** is the set of un-empty subsets of enabled nodes at the configuration **c if c** is not a terminal configuration otherwise **A(c)** is empty

  A([000])={{p1},{p2},{p3},{p1p2},{p2p3}, {p3p1},{p2p3},{p1p2p3}}

- **p(c1,set,c2)∈[0,1]** is the probability of the transition from **c1** to **c2** when the nodes of **set** perform an enabled rule.

  p([000],{p1p2},[010])=0,25

  p([000],{p2p1},[100])=0,25

Deterministic policy on a MDP

A policy μ is a function “Action choice” .
μ selects the action executed according to the state history

Notation:

Q_μ**(a|s0s1…sn)=1** iff **a** is the chosen action otherwise Q_μ**(a|s0s1…sn)=0**

Q_μ : $\Sigma_{a \in A(sn)}\ Q_\mu(a|s0s1…sn)=1$

$\exists\ a \in A(sn)\ |\ Q_\mu(a|s0s1…sn)=1$

Deterministic policy on a MDP

A deterministic policy μ is a memory k-bounded policy if for all possible sequences of n states **s0,…,sn** we have:

$Q_\mu(a|s0s1…snc1c2…ck) = Q_\mu(a|c1c2…ck)$

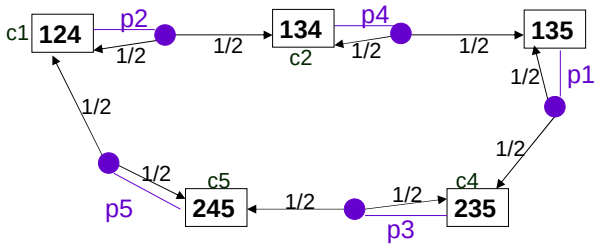
A deterministic policy is called memoryless if it is a memory 1-bounded policy :

$Q_\mu(a|s0s1…sn) = Q_\mu(a|sn)$

Policy

A policy μ on a MDP determines a single Markov Chain having a finite or unfinite state space

Herman’s Alg on ring of size 5



Probabilistic space

- S = (G+A) : the distributed system
- a policy : a schedule
- c0 : the initial configuration

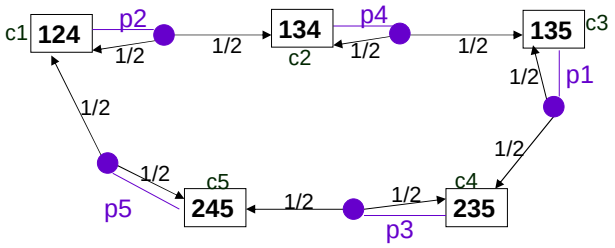
define a single probabilistic space

The events are the computations

Let **h=s0a0s1a1…cn** be a sequence of computation steps. The probability of **h** from **c0** in a MDP equipped of μ is

$p_{c0,\mu}(h) = \Pi_{k \in [0,n-1]}\ p(ck,ak,ck+1)Q_\mu(ak|c0c1…ck)$

μ Policy - exemple



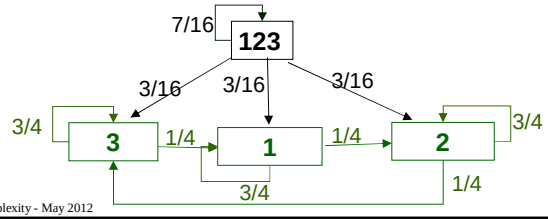
Let **h1=c1p2c2**
Let **h1=c1p1c1**
Let **h1=c1p2c2p4c3p1c4**

$p_{c1,\mu}(h1) = ½$
$p_{c1,\mu}(h1) = 0$
$p_{c1,\mu}(h3) = 1/8$

The uniform probabilisitic memoryless : μ_0

The uniform probabilisitic memoryless policy is
$Q_{\mu 0}(a|s0s1…sn)=Q_{\mu 0}(a|sn)=2^{-|enabled\text{-}nodes\text{-}at\text{-}sn|}$
A enabled node is chosen by the scheduler with the probability ½.

$(v_p \neq v_l +1 \bmod(M_N)) \rightarrow$ to flip a coin, *if* « head » *then* $v_p := v_l +1 \bmod(M_N)$



(MDA, policy) in distributed system

The number of deterministic policies is unbounded and also un-enumerable

A demon ***D*** is a set of policies

An algorithms designer has to analysis all probabilistic spaces defined by the policies of **D**

Ω : set of computations (finite and infinite)

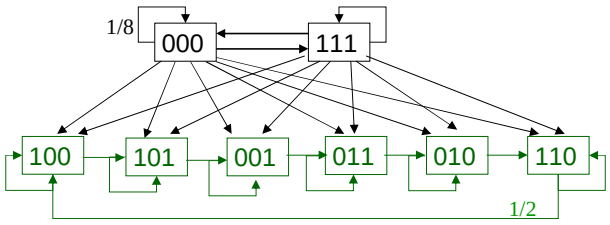
σ-field : the set of histories (finite computations)

$p_{c0,\mu}(h)$

The uniform probabilistic memoryless : μ_0

The uniform probabilistic memoryless policy is
$Q_{\mu 0}(a|s0s1…sn)=Q_{\mu 0}(a|sn)= 2^{-|enabled\text{-}nodes\text{-}at\text{-}sn|}$
A enabled node is chosen by the scheduler (μ_0) with the probability ½.

R: $(v_p \neq v_l +1 \bmod(M_N)) \rightarrow v_p := v_l +1 \bmod(M_N)$



Probabilistic policy on a MDP

A policy μ is a function “Action choice”.

μ associates a probability to the action executions according to the state history

Notation:

$Q_\mu(a|s0s1…sn) \in [0,1[$

Q_μ : $\sum_{a \in A(sn)} Q_\mu(a|s0s1…sn)=1$

Equivalence

- a deterministic algorithm under the probabilistic demon (uniform probabilistic memoryless policy)

  R: $(v_p \neq v_l +1 \bmod(M_N)) \rightarrow v_p := v_l +1 \bmod(M_N)$

- the « probabilized » version of the algorithm under the synchronous demon

Rp: $(v_p \neq v_l +1 \bmod(M_N)) \rightarrow$ to flip a coin, *if* « head » *then* $v_p := v_l +1 \bmod(M_N)$

Are equivalent : the same Markov Chain

About the power of the schedule

Uniform probabilistic memoryless policy - system homogeneous, no failure

Deterministic policies – system heterogeneous, permanent or transient failures

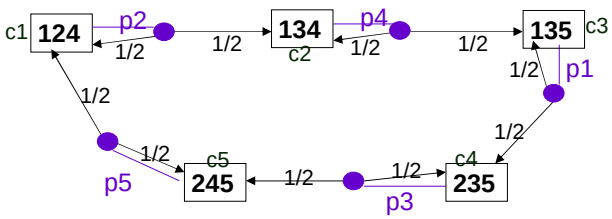
About the power of the schedule

Uniform probabilistic memoryless policy - system homogeneous, no failure

Deterministic fair memory k-bounded policies – system pseudo-heterogeneous, Transient Failures

Deterministic policies – system heterogeneous, permanent or Transient Failures

About the fairness of deterministic policy



Along a fair computation, a node always enabled performs infinitely often a action.

This strategy has an enumerable set of unfair computations (the measure of this set is 0).

(c1p2c1)* , c1p2c2(c2p4c2)*

Thank you

Fair deterministic policy : μ

cp is an unfair infinite computation from **c0** if an always enabled node **p** is never selected by the policy (with or without changing its state)

Fairness of μ

Let **cp** be an unfair infinite computation from **c0.**

There is a prefix of **cp, h,** such that $p_{c0,\mu}(h)=0$