Causal Total order broadcast algorithm with bounded message size for dynamic systems

Colette Johnen

Luciana Arantes

Pierre Sens

FIFO broadcast when all processes are correct

Reliable broadcast :

Every broadcast message is eventually delivered once by all processes

Any delivered message was broadcast by a process

FIFO order : delivering order of the messages sent by a process *p* is the sending order

If *p* broadcasts *m1* before broadcasting *m2* then every process delivers *m1* before *m2*

Total order broadcast when all processes are correct

FIFO order : delivering order of the messages sent by a process *p* is the sending order

A process may deliver *m'* before or after *m* if these two messages are sent by distinct processes

Total order (atomic broadcast) : all processes deliver all messages in the same order

If a process delivers *m* before *m'* then every process delivers *m* before *m'*

Total order broadcast primitives – interest

Total order broadcast primitives is a fundamental building block, used to implement

- consensus (they are equivalent problem)

[Chandra, Toueg, 1996]

- state machine replication

[Rajsbaum, Raynal 2020]

- Database replication

[Agrawal *et al* 1997][Stanoi *et al* 1998] [Pedone *et al* 2003]

- sequential consistency

[Baldoni *et al*, 2012] [Perrin *et al* 2016]

Survey on Total order broadcast algorithms [Delfago *et al* 2004]

Around **60** total order broadcast algorithms were presented

algorithms are classified according to the mechanism used to order messages :

- communication history
- privilege-based
- sequencer
- destination agreement

Challenge

- Total causal order broadcast in highly dynamic message passing systems
- Dynamics modeled as *Dynamic graph* [Charron-Bost, Moran 2018]

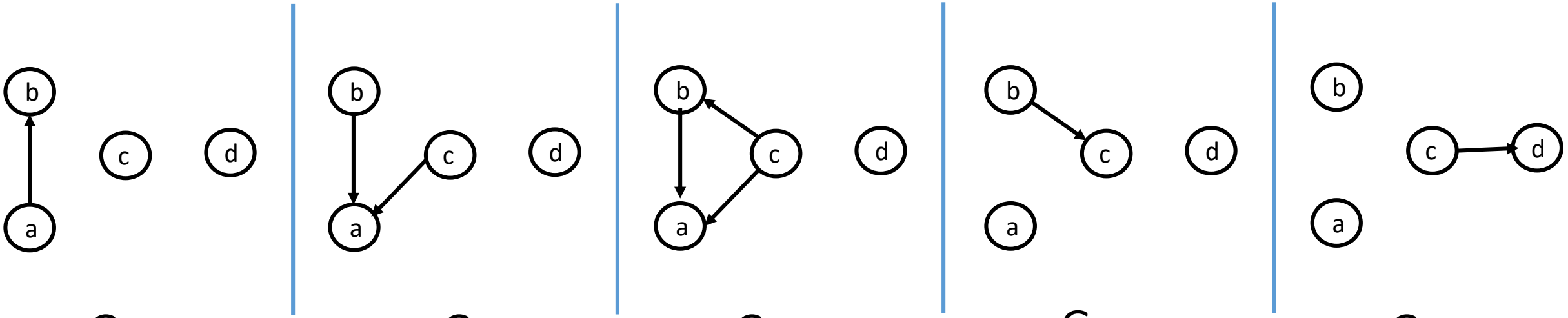
a particular topology is not required at a given time
dynamicity does not follow a probabilistic law

To bound data in transit at any point of time :

1. message size has to be bounded
2. the number of messages in transit has to be bounded

Dynamic Graph [Charron-Bost, Moran 2018]

An infinitely sequence of direct loopless graphs having the same vertex set, V : G_1, G_2, G_3 ….



G_1

$\mathcal{IN}^1(b)=\{a\}$

G_2

$\mathcal{IN}^2(a)=\{b,c\}$

G_3

$\mathcal{IN}^3(a)=\{b,c\}$

$\mathcal{IN}^3(b)=\{c\}$

G_4

$\mathcal{IN}^4(c)=\{b\}$

G_5

$\mathcal{IN}^5(d)=\{c\}$

Computation in Synchronous rounds

During the round i, a process *p* executes 3 steps :

1. *p* sends a message consisting of all or a part of its local state at the beginning of the round i by calling the primitive SEND()

p also sends messages of others processes in transit at i

2. using Primitive RECEIVE(), *p* receives all messages sent by processes in $\mathcal{IN}(p)^i = \{q \in V : (q,p) \in G_i\}$

3. *p* computes its state according its local algorithm – local state of *p* at the beginning of the round i+1

p also defines the received messages it keeps in transit

Journey

Journey from p_1 to p_{k+1} starting at t1

$\mathcal{J} = (e1, t1), (e2, t2), \ldots, (ek, tk)$

such that $\forall\, i \in \{1, \ldots, k\}$

$$ei = (p_i, p_{i+1}) \in G_i$$

$$i < k \Rightarrow ti < ti{+}1$$

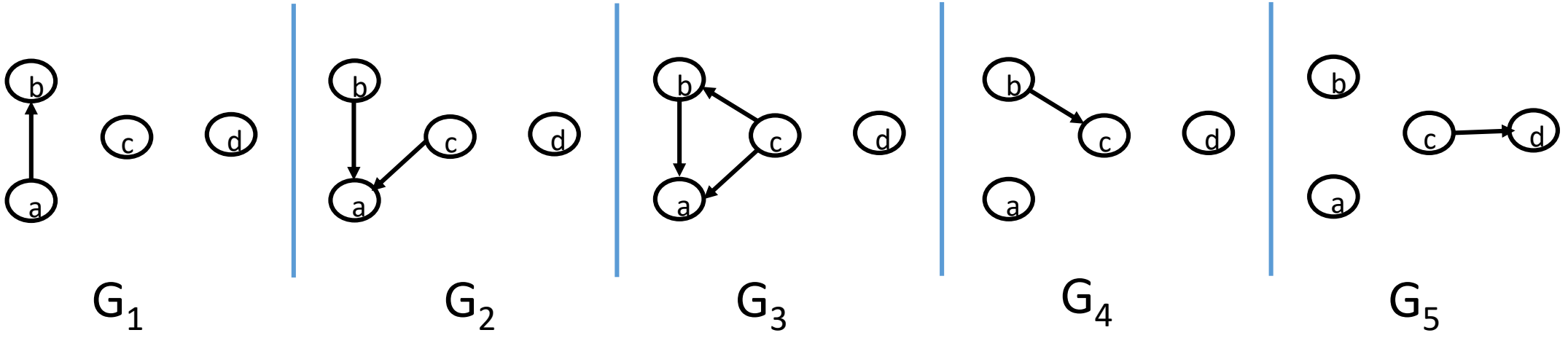
Temporal length: $tk - t1 + 1$

Example:

$((a, b), 1), ((b, c), 4), ((c, d), 5)$

is a journey of length 5

from a to d



Broadcast algorithms in Dynamic Graphs

- At any point in time, the graph is connected

  [O'Dell, Wattenhofer 2005] [Kuhn *et al* 2010] [Ahmadi *et al* 2020]

- If an edge appears once, it appears infinitely often

  [Casteigts *et al* 2010] [Raynal *et a*l 2014] [Casteigts *et al* 2015]

- At any point in time, every node can reach all the others through a journey of temporal length at most Δ

  [Gómez-Calzado *et al* 2015]

- No study of the delivery order
- Stonger requirement on DG than Recurrent Connectivity

Class $\mathcal{TC}^{\mathcal{R}}$ (Recurrent Connectivity): At any point in time, every process can reach all the others through a journey

Causal Total order Broadcast algorithm using FIFO broadcast

Communication between processes is performed via FIFO broadcast primitives :

- FD-broadcast(m) and
- FD-deliver(id(q),m)

Total order Broadcast algorithm

Communication between processes is performed via FIFO broadcast primitives: FD-broadcast(m) and FD-deliver(id(q),m)

When *p* has received a message from each process, via FD-deliver, then

p handles the first message received from each process (i.e. *p* delivers this message if it is not *atomic_⊥*)

the delivery order of the messages is the order of senders identifier

Each process permanently performs a Total order broadcast (i.e. a FD-broadcast)

If *p* has not data to broadcast, it broadcasts the neutral data : *atomic_⊥*

Total order Broadcast algorithm properties

data(s,i) is the i-th data atomically broadcast by process *s*

Lemma : Let *data(p,i)* and *data(q,j)* be two messages distinct of *atomic⊥*.
data(p,i) is delivered before *data(q,j)* by a process if and only if

$i < j$ or $(i = j$ and $id(p) < id(q))$

Every process delivers the messages in the same order that is a FIFO order

Total order Broadcast algorithm properties

data(s,i) is the ith data atomically broadcast by process *s*

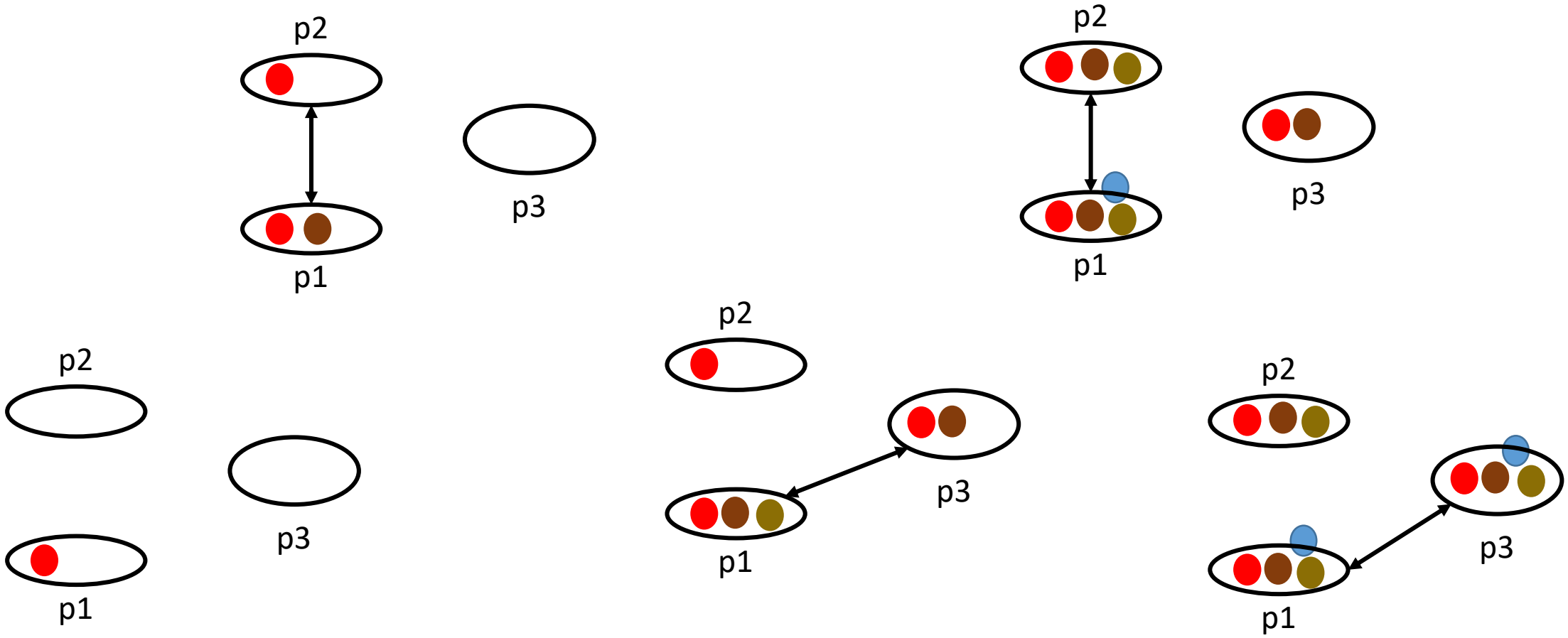
Local order property *:* if a process *s* atomically delivers *data(q,j)* before atomically broadcasts *data(s,i)* then every process delivers *data(q,j)* before *data(s,i)*

[Hadzilacos and Toueg 1994] fifo order + local order $\Rightarrow$ causal order

Every process delivers the messages in the same order : a causal order

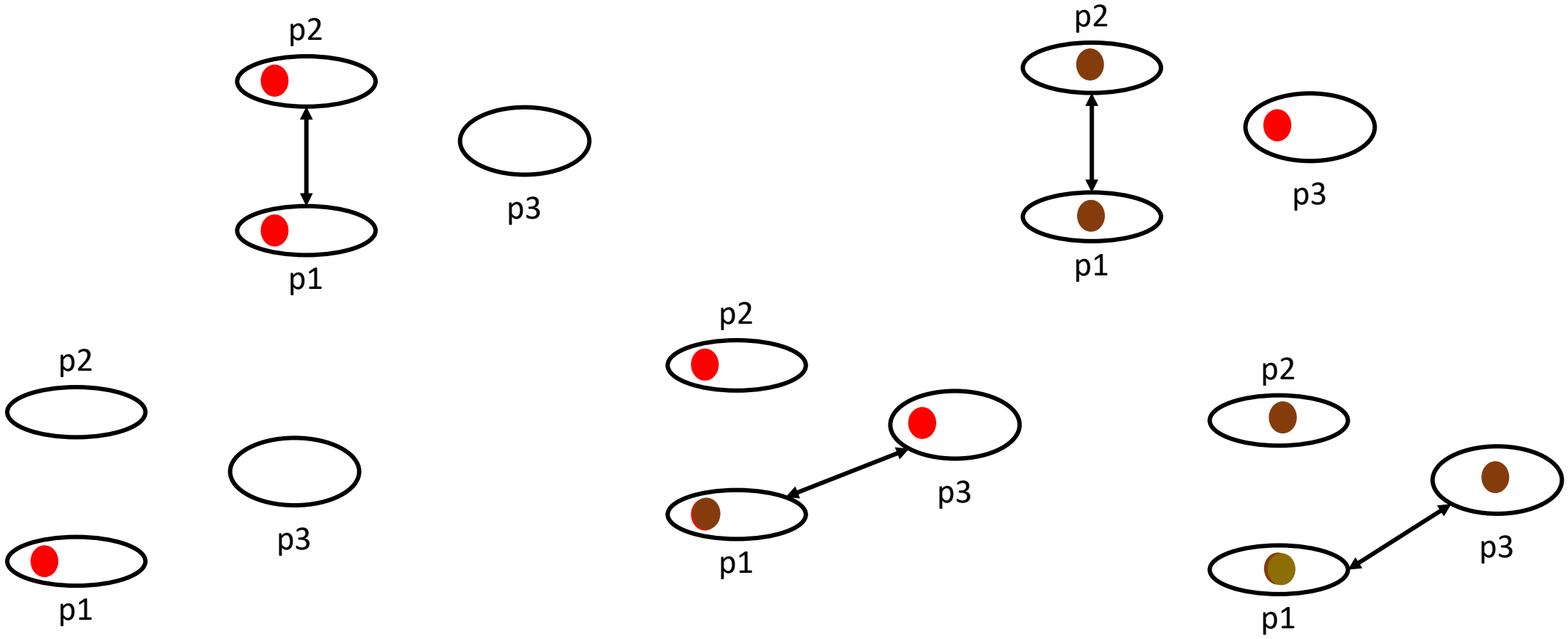
FIFO broadcast

FIFO broadcast



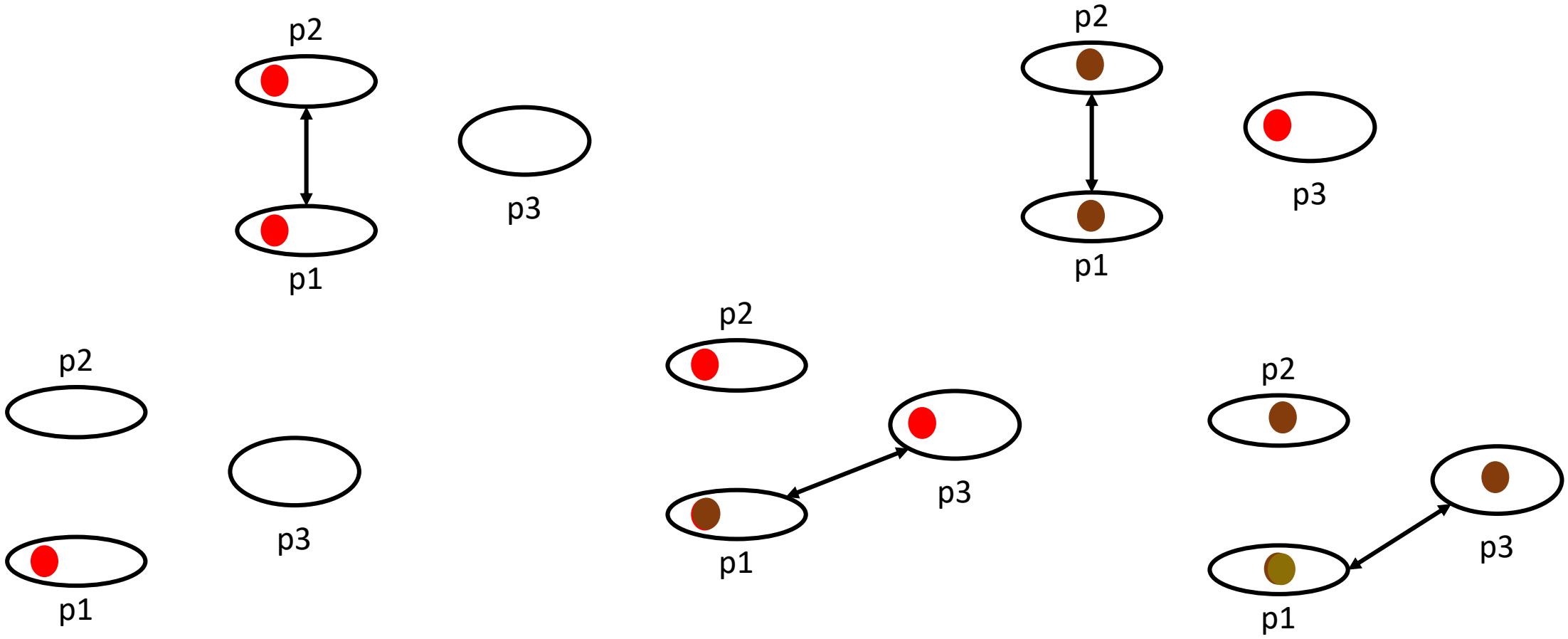
Each broadcast data by *p1* has a sequential number (unbounded value)

FIFO broadcast with termination detection



Each broadcast data by *p1* has a sequential number (unbounded value),
each process keeps in transit a single message from *p1* - the most recent message one

FIFO broadcast with termination detection



Each broadcast data by *p1* has a sequential number (**bounded value**),
each process keeps in transit a single message from *p1* - the most recent message one

Overview of FIFO Broadcast with termination detection

m(p,i) is the ith data broadcast by process *p* – its size is bounded by msgSize

A message of *p* containing *m(p,i)* contains also its label : *i mod 3*

A message from *p* contains the acknowledgment of a last received message from *q* (its label)

A message from *p* contains :

- *id(p)* : sender identifier – O(log(N)) bits
- *m(p, -)* : data – msgSize bits
- *TS[p]* : data label - 2 bits
- *TS[q]* with $q \neq p$: message acknowledgments - N-1.2 bits
- *updtCnt* : counter of *TS* updates – log(N)+1 bits

Overview of FIFO Broadcast with termination detection

m(s,i) is the ith data broadcast by process *s* – its size is bounded by msgSize

A message of *s* containing *m(s,i)* contains also its label : *i mod 3*

Detection of termination of ith broadcast by *s* :

Since the beginning of its ith broadcast, *s* has received acknowledgments from *N* distinct processes of *m(s,i)* (i.e. *TS[s]=i mod 3*)

Difficulties as *m(s,i)*, *m(s,i+3)*, and *m(s,i+6)* have the same label

- Acknowledgement of *m(s,i)* is identical to the one of *m(s,i+3k)*
- A process r cannot distinct a message containing *m(s,i)* of a message containing m(s,*i+3k)*

Overview of FIFO Broadcast with termination detection

m(s,i) is the ith data broadcast by process *s* – its size is bounded by msgSize

A message of *s* containing *m(s,i)* contains also its label : *i mod 3*

Detection of termination of ith broadcast by *s* :

Since the beginning of its ith broadcast, *s* has received acknowledgments from *N* distinct processes of a data labeled *i mod 3* (ie *TS[s]=I mod3*)

Requirements to ensure correct broadcasts with termination detection :

at the beginning of the ith broadcast from *s*

1. every process is ready to deliver a data of *s* labeled *i mod 3*
2. every message from *s* in transit has the label *i-1 mod 3*
3. every message in transit verifies $TS[s] \neq i \bmod 3$

Overview of FIFO Broadcast with termination detection

m(s,i) is the ith data broadcast by process *s* – its size is bounded by msgSize
A message of *s* containing *m(s,i)* contains also its label : *i mod 3*

Requirements to ensure correct broadcasts with termination detection :
at the beginning of the ith broadcast from *s*

1. every process is ready to deliver a data labeled *i mod 3* from *s*
2. every message from *s* in transit has the label *i-1mod3*
3. every message in transit verifies $TS[s] \neq imod3$

Each process *q* keeps in transit a single message of *p*, the most recent one sent
$\Rightarrow$ the messages of *p* sent during its ith broadcast has to be dated

The field *updtCnt* contains the number of times *p* updates its message during its current broadcast

Overview of FIFO Broadcast with termination detection

m(s,i) is the ith data broadcast by process *s* – its size is bounded by msgSize
A message of *s* containing *m(s,i)* contains also its label : *i mod 3*

The field *updtCnt* contains the number of message updates done during the current broadcast

updtCnt value must be bounded

As soon as a process *p* detects the termination of a broadcast, it starts a new one

If *p* has not data to broadcast, it broadcasts the empty data : *brdcst*$\perp$

Correctness of FIFO Broadcast with termination detection

Message size : during a broadcast of *s* , *TS[q]* is updated at most 2 times – in Class $\mathcal{TC}^{\mathcal{R}}$, ***updtCnt*** **≤ 2N**

Safety : during the ith broadcast of *s*, every process delivers once *m(s,i)* if *m(s,i)* ≠ *brdcst*$\perp$

Liveness : in Class $\mathcal{TC}^{\mathcal{R}}$, every broadcast terminates
in Class $\mathcal{TC}^{\mathcal{R}}$: at any point in time, every process can reach all the others through a journey

FIFO Broadcast with termination detection

Process have identifier, $\forall\ p \in V$, *id(p)* is the unique identifier of *p*

[Casteigts *et al* 2015] : FIFO broadcast with termination detection cannot be solved in $\mathcal{TC}^{\mathcal{R}}$ if the number of processes is unknown

$\Rightarrow$ Each process knows the number of processes : *N*

Properties of our FIFO-BTD algo : The size of a message is *2N + O(log(N)) + msgSize* bits where *msgSize* is the size of messages to broadcast

A process has at most N messages in transit

Conclusion

Causal Total order broadcast primitives in dynamic systems

Future direction ??? : Fault-tolerant bradcasts (process crashes, byzantines processes or transient faults, …) in dynamic graph

Pour plus de détail :

- Dépôt hal : https://hal.inria.fr/hal-03332423
- Présentation à SRDS 2021

Atomic Broadcast using FIFO broadcast with termination Detection

Communication between processes is performed via fifo broadcast with termination detection: FD-broadcast(m) and FD-deliver(id(q),m).

When p has received a message from each process, via FD-deliver, then

p handles the first message received from each process (i.e. p atomically delivers this message if it is not *atomic⊥*)

the delivery order of the messages is the order of senders identifier

Each process permanently performs an atomic broadcast (i.e. a FD-broadcast)

If *p* has not data to atomically broadcast, it broadcasts the neutral data : *atomic⊥*

Atomic Broadcast properties

data(s,i) is the ith data atomically broadcast by process *s*

Lemma : Let *data(p,i)* and *data(q,j)* be two messages distinct of *atomic⊥*.
data(p,i) is delivered before *data(q,j)* by a process if and only if

$i < j$ or $(i = j$ and $id(p) < id(q))$

Every process delivers the messages in the same order that is a FIFO order

Atomic Broadcast properties

data(s,i) is the ith data atomically broadcast by process *s*

Local order property *:* if a process *s* atomically delivers *data(q,j)* before atomically broadcasts *data(s,i)* then every process delivers *data(q,j)* before *data(s,i)*

[Hadzilacos and Toueg 1994] fifo order + local order $\Rightarrow$ causal order

Every process delivers the messages in the same order : a causal order

Conclusion

- Causal Total order broadcast primitives in dynamic systems

- Dynamics modeled as *Dynamic graph* [Charron-Bost, Moran 2018]

  Class $\mathcal{TC}^{\mathcal{R}}$ - recurrent connectivity

  a particular topology is not required at a given time

  dynamicity does not follow a probabilistic law

- The size of a message is *2N + O(log(N)) + msgSize* bits where *msgSize* is the size of messages to broadcast

Future direction : Fault-tolerant algorithms (process crashes, byzantines fault or transient fault, …)

Survey on Total order broadcast algorithms [Delfago *et al* 2004]

Around **60** total order broadcast algorithms were presented

algorithms are classified according to the mechanism used to order messages :

- communication history
- privilege-based
- sequencer
- destination agreement

Total order broadcast primitives block

Total order broadcast primitives is a fundamental building block, used to implement

- consensus (they are equivalent problem)

[Chandra, Toueg, 1996]

- state machine replication

[Rajsbaum, Raynal 2020]

- Database replication

[Agrawal *et a*l 1997, Stanoi *et al* 1998, Pedone *et al* 2003]

- sequential consistency

[Baldoni *et al*, 2012, Perrin *et al* 2016]

Considered Class of dynamic graphs

Class $\mathcal{TC}^{\mathcal{R}}$ (Recurrent Temporal Connectivity): At any point in time, every node can reach all the others through a journey