Actor-like cP Systems

Alec Henderson and Radu Nicolescu

– Application/Test : Byzantine Agreement –

Department of Computer Science
University of Auckland, Auckland, New Zealand

CMC
Dresden, Germany
4-7 September 2018

1. Greetings
2. Motivation
3. cP Local Evolution Samples
4. cP Communication
5. The Byzantine Agreement
6. Sybil-like Attacks
7. Ruleset
8. Unbounded non-determinism – fairness, beyond Turing?

Kia ora

- Kia ora! G'day!
- Good day!
- Dobryj dyen'!
- Guten Tag!
- Bonjour!
- Buon giorno!
- Buenos días!
- Bună ziua!

Basic features shared by P and cP systems

- Cellular organisation
  - Top cells organised in digraph networks – tissue P systems
  - Top cells contain nested sub-cells – cell-like P systems
- Data given as multisets
- Evolution by multiset rewriting rules – potential parallelism
  - Extended with states, weak priority, promoters, inhibitors, ...
  - ... and communication primitives between top-cells

Basic features shared by P and cP systems

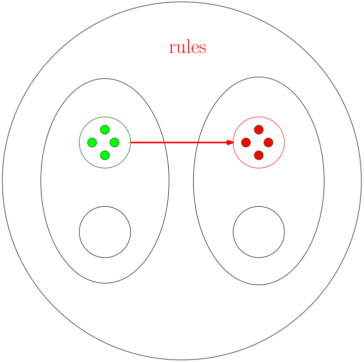
- Cellular organisation
  - Top cells organised in digraph networks – tissue P systems
  - Top cells contain nested sub-cells – cell-like P systems
- Data given as multisets
- Evolution by multiset rewriting rules – potential parallelism
  - Extended with states, weak priority, promoters, inhibitors, ...
  - ... and communication primitives between top-cells

Basic features shared by P and cP systems

- Cellular organisation
  - Top cells organised in digraph networks – tissue P systems
  - Top cells contain nested sub-cells – cell-like P systems
- Data given as multisets
- Evolution by multiset rewriting rules – potential parallelism
  - Extended with states, weak priority, promoters, inhibitors, ...
  - ... and communication primitives between top-cells

Bird's eye view – digraph of top level cells

- Each top cell has
  - passive sub-cellular components (data only – no own rules!)
  - organelles, vesicles, ...
  - high-level rules (that can directly work on subcells' contents)



Inspiration

- Logic programming
  - subcells (aka complex symbols) $\approx$ Prolog-like first-order terms, recursively built from multisets of atoms and variables
- Functional and generic programming
- Actor model

Inspiration

- Logic programming
  - subcells (aka complex symbols) $\approx$ Prolog-like first-order terms, recursively built from multisets of atoms and variables
- Functional and generic programming
- Actor model

Inspiration

- Logic programming
  - subcells (aka complex symbols) $\approx$ Prolog-like first-order terms, recursively built from multisets of atoms and variables
- Functional and generic programming
- Actor model

Previous work – P systems with complex objects, cP

- image processing and computer vision
  - stereo-matching, skeletonisation, segmentation
- graph theory
- high-level P systems programming
- numerical P systems
- NP-complete problems
- distributed algorithms
  - Byzantine agreement – continued here

cP Local Evolution Samples

- Local evolution: one top cell and its subcells
- No communication between top cells
- Model for parallelism with shared memory

Natural numbers

Ad-hoc convention: *1* – unary digit

- $x = 0 \equiv x() \equiv x(\lambda)$
- $x = 1 \equiv x(\mathit{1})$
- $x = 2 \equiv x(\mathit{11})$
- $x = n \equiv x(\mathit{1}^n)$
- $x \leftarrow y + z \equiv$
  - $y(Y)\ z(Z)\ \rightarrow\ x(YZ)$ (destructive add)
  - $\rightarrow\ x(YZ)\ \mid\ y(Y)\ z(Z)$ (preserving add)
- $x \leq y \equiv \mid\ x(X)\ y(XY)$
- $x < y \equiv \mid\ x(X)\ y(XY\mathit{1})$

Natural numbers

Ad-hoc convention: $\mathit{1}$ – unary digit

- $x = 0 \equiv x() \equiv x(\lambda)$
- $x = 1 \equiv x(\mathit{1})$
- $x = 2 \equiv x(\mathit{11})$
- $x = n \equiv x(\mathit{1}^n)$
- $x \leftarrow y + z \equiv$
  - $y(Y)\ z(Z)\ \rightarrow\ x(YZ)$ (destructive add)
  - $\rightarrow\ x(YZ)\ \mid\ y(Y)\ z(Z)$ (preserving add)
- $x \leq y \equiv \mid\ x(X)\ y(XY)$
- $x < y \equiv \mid\ x(X)\ y(XY\mathit{1})$

Natural numbers

Ad-hoc convention: $\mathit{1}$ – unary digit

- $x = 0 \equiv x() \equiv x(\lambda)$
- $x = 1 \equiv x(\mathit{1})$
- $x = 2 \equiv x(\mathit{11})$
- $x = n \equiv x(\mathit{1}^n)$
- $x \leftarrow y + z \equiv$
  - $y(Y)\ z(Z) \ \rightarrow \ x(YZ)$ (destructive add)
  - $\rightarrow \ x(YZ) \ \mid \ y(Y)\ z(Z)$ (preserving add)
- $x \leq y \equiv \mid \ x(X)\ y(XY)$
- $x < y \equiv \mid \ x(X)\ y(XY\mathit{1})$

Natural numbers

Ad-hoc convention: *1* – unary digit

- $x = 0 \equiv x() \equiv x(\lambda)$
- $x = 1 \equiv x(\mathit{1})$
- $x = 2 \equiv x(\mathit{11})$
- $x = n \equiv x(\mathit{1}^n)$
- $x \leftarrow y + z \equiv$
  - $y(Y)\ z(Z)\ \rightarrow\ x(YZ)$ (destructive add)
  - $\rightarrow\ x(YZ)\ \mid\ y(Y)\ z(Z)$ (preserving add)
- $x \leq y \equiv \mid\ x(X)\ y(XY)$
- $x < y \equiv \mid\ x(X)\ y(XY\mathit{1})$

Efficient summary statistics

- Consider a multiset of 'a' numbers, such as:

$$a(1^5)\ \ a(1^3)\ \ a(1^7)\ \ \ldots$$

- Min finding in two steps (regardless of the data cardinality)

1 $S_1 \rightarrow_+ S_1'\ b(X) \mid a(X)$
2 $S_1'\ b(XY1) \rightarrow_+ S_2 \mid a(X)$

- Rule (2): delete all b's having values strictly higher than anyone a

- Result (non-destructive):

$$a(1^5)\ \ a(1^3)\ \ a(1^7)\ \ \ldots$$
$$b(1^3)$$

Efficient summary statistics

- Consider a multiset of 'a' numbers, such as:

$$a(1^5) \;\; a(1^3) \;\; a(1^7) \;\; \ldots$$

- Min finding in two steps (regardless of the data cardinality)

1 $S_1 \;\; \rightarrow_+ \;\; S_1' \;\; b(X) \;\mid\; a(X)$

2 $S_1' \;\; b(XY1) \;\; \rightarrow_+ \;\; S_2 \;\mid\; a(X)$

- Rule (2): delete all b's having values strictly higher than anyone a

- Result (non-destructive):

$$a(1^5) \;\; a(1^3) \;\; a(1^7) \;\; \ldots$$
$$b(1^3)$$

Efficient summary statistics

- Consider a multiset of 'a' numbers, such as:

$$a(1^5) \quad a(1^3) \quad a(1^7) \quad \ldots$$

- Min finding in two steps (regardless of the data cardinality)

1 $S_1 \ \rightarrow_+ \ S_1' \ b(X) \ | \ a(X)$
2 $S_1' \ b(XY1) \ \rightarrow_+ \ S_2 \ | \ a(X)$

- Rule (2): delete all b's having values strictly higher than anyone a

- Result (non-destructive):

$$a(1^5) \quad a(1^3) \quad a(1^7) \quad \ldots$$
$$b(1^3)$$

Efficient summary statistics

- Consider a multiset of 'a' numbers, such as:

$$a(1^5)\ \ a(1^3)\ \ a(1^7)\ \ \ldots$$

- Min finding in two steps (regardless of the data cardinality)

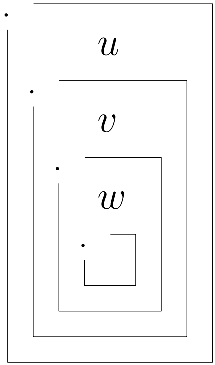
1 $S_1 \ \rightarrow_+ \ S_1' \ \ b(X) \ \mid \ a(X)$
2 $S_1' \ \ b(XY1) \ \rightarrow_+ \ S_2 \ \mid \ a(X)$

- Rule (2): delete all b's having values strictly higher than anyone a

- Result (non-destructive):

$$a(1^5)\ \ a(1^3)\ \ a(1^7)\ \ \ldots$$
$$b(1^3)$$

List x – with . as cons



- $x(.(u\ .(v\ .(w\ .()))))) \equiv$
- $x[u, v, w]$ (sugared notation) $\equiv$
- $x[u \mid [v, w]]$ (sugared notation)

List – basic ops

$\rightarrow_1 \; y[\,]$ *creating empty list y*

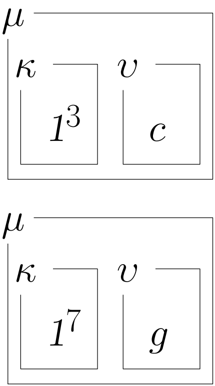
$a \; y[Y] \; \rightarrow_1 \; y[a \mid Y]$ *pushing atom a on list y*

$a(X) \; y[Y] \; \rightarrow_1 \; y[X \mid Y]$ *pushing contents of a on list y*

$y[X \mid Y] \; \rightarrow_1 \; b(X) \; y[Y]$ *popping the top of list y to contents of b*

Associative arrays (mappings, dictionaries)

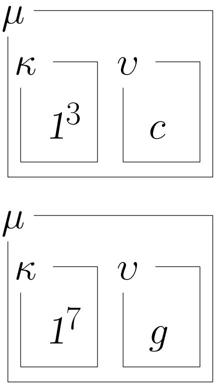
μ – mapping, κ – key, υ – value



- $1^3 \mapsto c \equiv$
  - $\mu(\kappa(1^3)\ \upsilon(c))$
- $\{1^3 \mapsto c, 1^7 \mapsto g\} \equiv$
  - $\mu(\kappa(1^3)\ \upsilon(c)) \quad \mu(\kappa(1^7)\ \upsilon(g))$
- Similarly: finite functions, relations, tables, trees, ...

Associative arrays (mappings, dictionaries)

μ – mapping, κ – key, υ – value



- $1^3 \mapsto c \equiv$
  - $\mu(\kappa(1^3)\ \upsilon(c))$
- $\{1^3 \mapsto c, 1^7 \mapsto g\} \equiv$
  - $\mu(\kappa(1^3)\,\upsilon(c)) \quad \mu(\kappa(1^7)\,\upsilon(g))$
- Similarly: finite functions, relations, tables, trees, ...

Previous cP messaging mechanism

- Sender takes all decisions

$a\ a\ \rightarrow b!_1$	*two a's are deleted and one b is sent over arc 1*

- More emphatically: $\boxed{b!_1 \equiv\ !_1\{b\}}$
- Problem: receiving cell has no control: time, filter, consistency, ...
- In particular, the system is prone to Sybil attacks – i.e. can be subverted by forging identities
  - Name inspired by the book Sybil, a case study of a person diagnosed with dissociative (multiple) identity disorder
- More generally, the network part was subsumed by local evolutions – modelling flaw

Previous cP messaging mechanism

- Sender takes all decisions

$$\boxed{a\ a\ \rightarrow b!_1 \qquad \textit{two a's are deleted and one b is sent over arc 1}}$$

- More emphatically: $\boxed{b!_1 \equiv\ !_1\{b\}}$

- Problem: receiving cell has no control: time, filter, consistency, ...

- In particular, the system is prone to Sybil attacks – i.e. can be subverted by forging identities
  - Name inspired by the book Sybil, a case study of a person diagnosed with dissociative (multiple) identity disorder

- More generally, the network part was subsumed by local evolutions – modelling flaw

Previous cP messaging mechanism

- Sender takes all decisions

$$\boxed{a\ a\ \rightarrow b!_1 \qquad \textit{two } a\textit{'s are deleted and one } b \textit{ is sent over arc 1}}$$

- More emphatically: $\boxed{b!_1 \equiv\ !_1\{b\}}$
- Problem: receiving cell has no control: time, filter, consistency, ...
- In particular, the system is prone to Sybil attacks – i.e. can be subverted by forging identities
  - Name inspired by the book Sybil, a case study of a person diagnosed with dissociative (multiple) identity disorder
- More generally, the network part was subsumed by local evolutions – modelling flaw

Previous cP messaging mechanism

- Sender takes all decisions

$a\ a\ \rightarrow b!_1$	*two a's are deleted and one b is sent over arc 1*

- More emphatically: $\boxed{b!_1 \ \equiv \ !_1\{b\}}$
- Problem: receiving cell has no control: time, filter, consistency, ...
- In particular, the system is prone to Sybil attacks – i.e. can be subverted by forging identities
  - Name inspired by the book Sybil, a case study of a person diagnosed with dissociative (multiple) identity disorder
- More generally, the network part was subsumed by local evolutions – modelling flaw

Previous cP messaging mechanism

- Sender takes all decisions

$a\ a\ \rightarrow b!_1$	*two a's are deleted and one b is sent over arc 1*

- More emphatically: $\boxed{b!_1 \ \equiv \ !_1\{b\}}$
- Problem: receiving cell has no control: time, filter, consistency, ...
- In particular, the system is prone to Sybil attacks – i.e. can be subverted by forging identities
  - Name inspired by the book Sybil, a case study of a person diagnosed with dissociative (multiple) identity disorder
- More generally, the network part was subsumed by local evolutions – modelling flaw

Fallacies of distributed computing – L Peter Deutsch

- Latency is zero
- Transport cost is zero
- Bandwidth is infinite
- The network is reliable
- The network is secure
- Topology doesn't change
- The network is homogeneous
- There is one administrator
- ...

Actor model

- The Actor model is a model of message-based concurrent computation which treats “actors” as universal primitives
- In response to a message that it receives, an actor can
  - make local decisions
  - create more actors
  - send more messages
  - (change state) determine how to respond to the next message received
- There is no assumed sequence to the above actions
- In the (typical) asynchronous case, it could take an unbounded time to receive a sent message

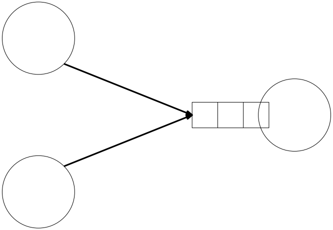
Actor model

- The Actor model is a model of message-based concurrent computation which treats "actors" as universal primitives
- In response to a message that it receives, an actor can
  - make local decisions
  - create more actors
  - send more messages
  - (change state) determine how to respond to the next message received
- There is no assumed sequence to the above actions
- In the (typical) asynchronous case, it could take an unbounded time to receive a sent message

Actor model

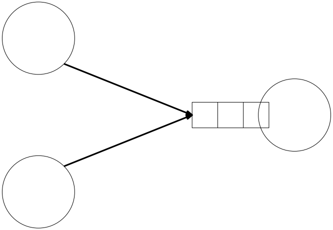
- The Actor model is a model of message-based concurrent computation which treats “actors” as universal primitives
- In response to a message that it receives, an actor can
    - make local decisions
    - create more actors
    - send more messages
    - (change state) determine how to respond to the next message received
- There is no assumed sequence to the above actions
- In the (typical) asynchronous case, it could take an unbounded time to receive a sent message

Typical Actor implementations use message “queues”



- The actor encapsulates an “inbox” message “queue” that supports multiple-writers and a single reader (the actor itself)
- Writers can send one-way messages to the actor by using the Post method and its variations
- Actors can receive messages using the Receive method and its variations (with optional timeouts)
- Actors can also scan through all their available messages using the Scan method and its variations

Typical Actor implementations use message "queues"



- The actor encapsulates an "inbox" message "queue" that supports multiple-writers and a single reader (the actor itself)
- Writers can send one-way messages to the actor by using the Post method and its variations
- Actors can receive messages using the Receive method and its variations (with optional timeouts)
- Actors can also scan through all their available messages using the Scan method and its variations

Typical Actor extensions

- Multiple inboxes
- Supervision hierarchy
  - Supervisors delegate tasks to subordinates...
  - ... then receive and treat subordinates' failures
- Monitoring relationships
  - Each actor may watch any other actor for termination

Typical Actor extensions

- Multiple inboxes
- Supervision hierarchy
    - Supervisors delegate tasks to subordinates...
    - ... then receive and treat subordinates' failures
- Monitoring relationships
    - Each actor may watch any other actor for termination

Typical Actor extensions

- Multiple inboxes
- Supervision hierarchy
  - Supervisors delegate tasks to subordinates...
  - ... then receive and treat subordinates' failures
- Monitoring relationships
  - Each actor may watch any other actor for termination

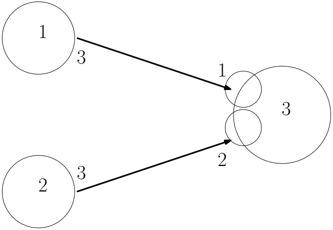
Actor systems – hard practical problems

- Exactly once message delivery
  - At most once
  - At least once
- FIFO messaging
  - Distributed algorithms should not rely on this assumption

Actor systems – hard practical problems

- Exactly once message delivery
  - At most once
  - At least once
- FIFO messaging
  - Distributed algorithms should not rely on this assumption

New cP messaging mechanism – Actor inspired

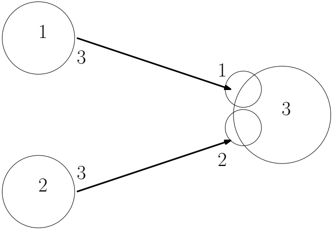


- Receiver has an active role
- Receiving cell has one system provided message multiset for each incoming arc

$\boxed{b?_1\ b\ \rightarrow c \quad \textit{can fire when one 'b' is in the message multiset 1}}$

- More emphatically: $\boxed{b?_1 \;\equiv\; ?_1\{b\}}$

New cP messaging mechanism – Actor inspired

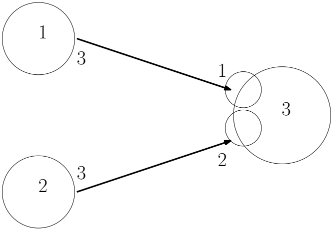


- Receiver has an active role
- Receiving cell has one system provided message multiset for each incoming arc

$$\boxed{b?_1 \ b \ \rightarrow c \quad \textit{can fire when one 'b' is in the message multiset 1}}$$

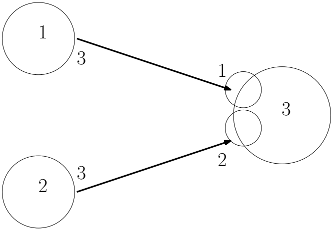
- More emphatically: $\boxed{b?_1 \ \equiv \ ?_1\{b\}}$

New cP messaging mechanism – Actor inspired



- Receiving cell has full control: time, filter, consistency, ...
- In particular, if the communication arcs are secure and reliable, then the system is resilient to Sybil attacks – i.e. cannot be subverted by forging identities

New cP messaging mechanism – Actor inspired



- Receiving cell has full control: time, filter, consistency, ...
- In particular, if the communication arcs are secure and reliable, then the system is resilient to Sybil attacks – i.e. cannot be subverted by forging identities

New cP messaging mechanism – CML inspired

- Message multisets can be implemented in a straightforward way, by automatically encapsulating incoming messages and tagging these with the id of the in-arc, e.g. $\boxed{?_1(b)}$
- The same syntax may have a CML (Concurrent Meta Language) inspired semantics!

$b?_1\ b \ \rightarrow c$	*can fire when a b arrives over in-arc 1*

- The sender could be blocked until the receiver "picks up" the message
- Work in progress – note some similarities with symport/antiport systems

New cP messaging mechanism – CML inspired

- Message multisets can be implemented in a straightforward way, by automatically encapsulating incoming messages and tagging these with the id of the in-arc, e.g. $\boxed{?_1(b)}$
- The same syntax may have a CML (Concurrent Meta Language) inspired semantics!

$b?_1\ b\ \rightarrow c$	*can fire when a b arrives over in-arc 1*

- The sender could be blocked until the receiver "picks up" the message
- Work in progress – note some similarities with symport/antiport systems

Consensus problems

- Consensus in the presence of faults
  - Node faults
    - Stopping failures
    - Byzantine failures
  - Communication faults
- Models
  - Synchronous
  - Asynchronous

Consensus problems

- Consensus in the presence of faults
    - Node faults
        - Stopping failures
        - Byzantine failures
    - Communication faults
- Models
    - Synchronous
    - Asynchronous

Consensus problems

- Consensus in the presence of faults
  - Node faults
    - Stopping failures
    - Byzantine failures
  - Communication faults
- Models
  - Synchronous
  - Asynchronous

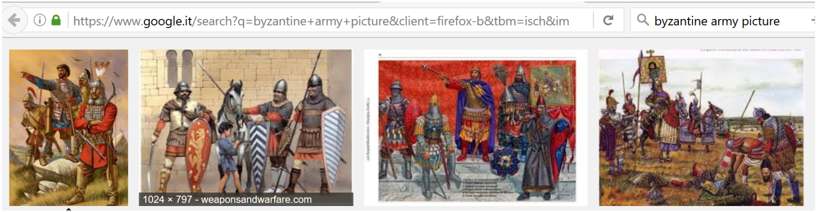
Consensus problems

- Consensus in the presence of faults
  - Node faults
    - Stopping failures
    - Byzantine failures
  - Communication faults
- Models
  - Synchronous
  - Asynchronous

Consensus problems

- Consensus in the presence of faults
  - Node faults
    - Stopping failures
    - Byzantine failures
  - Communication faults
- Models
  - Synchronous
  - Asynchronous

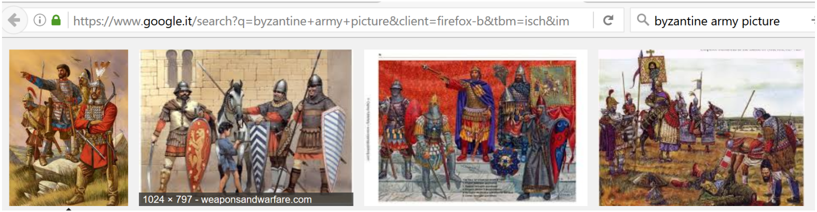
The Byzantine agreement



- $N = 4$ Byzantine armies, physically separated
- Generals start with their own initial decisions, 0 or 1
- They can communicate via $N(N-1)/2 = 6$ reliable channels
- They must reach a common decision
- Problem: among them there may be F Byzantine traitors
- Deterministic agreement between loyal generals possible iff $N \geq 3F + 1$ and communications are reliable and synchronous

Pease, Shostak, Lamport 1980; Lamport, Shostak, Pease 1982; Fischer, Lynch, Paterson 1985

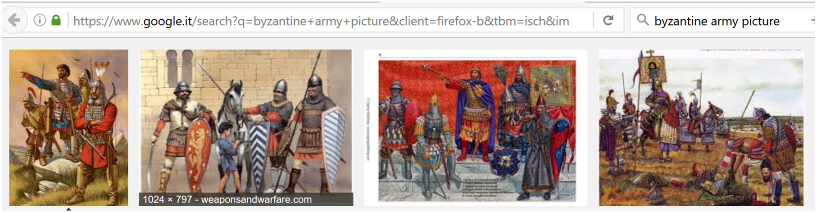
The Byzantine agreement



- $N = 4$ Byzantine armies, physically separated
- Generals start with their own initial decisions, 0 or 1
- They can communicate via $N(N-1)/2 = 6$ reliable channels
- They must reach a common decision
- Problem: among them there may be F Byzantine traitors
- Deterministic agreement between loyal generals possible iff $N \geq 3F+1$ and communications are reliable and synchronous

Pease, Shostak, Lamport 1980; Lamport, Shostak, Pease 1982; Fischer, Lynch, Paterson 1985

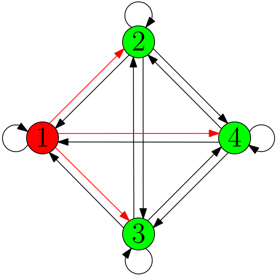
The Byzantine agreement



- $N = 4$ Byzantine armies, physically separated
- Generals start with their own initial decisions, 0 or 1
- They can communicate via $N(N-1)/2 = 6$ reliable channels
- They must reach a common decision
- Problem: among them there may be F Byzantine traitors
- Deterministic agreement between loyal generals possible iff $N \geq 3F + 1$ and communications are reliable and synchronous

Pease, Shostak, Lamport 1980; Lamport, Shostak, Pease 1982; Fischer, Lynch, Paterson 1985

The Byzantine agreement



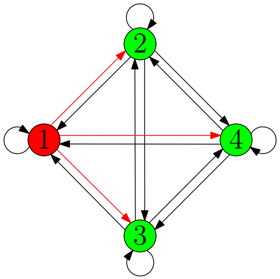
Process	ι_1	ι_2	ι_3	ι_4
Initial choice	0	0	1	1
Faulty	Yes	No	No	No
Round 1 messages	$(1, \mathbf{x})$	$(2, 0)$	$(3, 1)$	$(4, 1)$
Round 2 messages	$(2.1, 0)$ $(3.1, y)$ $(4.1, 1)$	$(1.2, 0)$ $(3.2, 1)$ $(4.2, 1)$	$(1.3, 0)$ $(2.3, 0)$ $(4.3, 1)$	$(1.4, 1)$ $(2.4, 0)$ $(3.4, 1)$
... Final decision	?	0	0	0

Faulty process ι_1 sends out conflicting messages:

- $x = 0, y = 1$ to process ι_2
- $x = 0, y = 0$ to process ι_3
- $x = 1, y = 1$ to process ι_4

Still, non-faulty processes do reach a common decision, 0 ($v_0 = 0$)

The Byzantine agreement



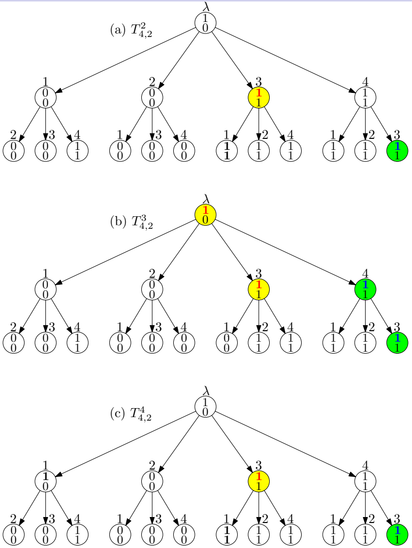
Process	ι_1	ι_2	ι_3	ι_4
Initial choice	0	0	1	1
Faulty	Yes	No	No	No
Round 1 messages	$(1, \mathbf{x})$	$(2, 0)$	$(3, 1)$	$(4, 1)$
Round 2 messages	$(2.1, 0)$ $(3.1, y)$ $(4.1, 1)$	$(1.2, 0)$ $(3.2, 1)$ $(4.2, 1)$	$(1.3, 0)$ $(2.3, 0)$ $(4.3, 1)$	$(1.4, 1)$ $(2.4, 0)$ $(3.4, 1)$
... Final decision	?	0	0	0

Faulty process ι_1 sends out conflicting messages:

- $x = 0, y = 1$ to process ι_2
- $x = 0, y = 0$ to process ι_3
- $x = 1, y = 1$ to process ι_4

Still, non-faulty processes do reach a common decision, 0 ($v_0 = 0$)

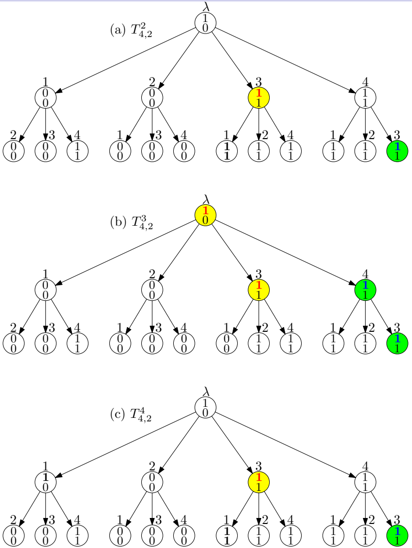
EIG trees for non-faulty processes



Process	ι_1	ι_2	ι_3	ι_4
Initial choice	0	0	1	1
Faulty	Yes	No	No	No
Round 1 messages	$(1, x)$	$(2, 0)$	$(3, 1)$	$(4, 1)$
Round 2 messages	$(2.1, 0)$ $(3.1, y)$ $(4.1, 1)$	$(1.2, 0)$ $(3.2, 1)$ $(4.2, 1)$	$(1.3, 0)$ $(2.3, 0)$ $(4.3, 1)$	$(1.4, 1)$ $(2.4, 0)$ $(3.4, 1)$
... Final decision	?	0	0	0

- α by top-down messaging
- L_1: (initial) $\iota_3 \xrightarrow{(3,1)} \iota_2, \iota_3, \iota_4$
- L_2: (relay) $\iota_3 \xrightarrow{(4.3,1)} \iota_2, \iota_3, \iota_4$
- β by bottom-up local voting
- common final decision, 0

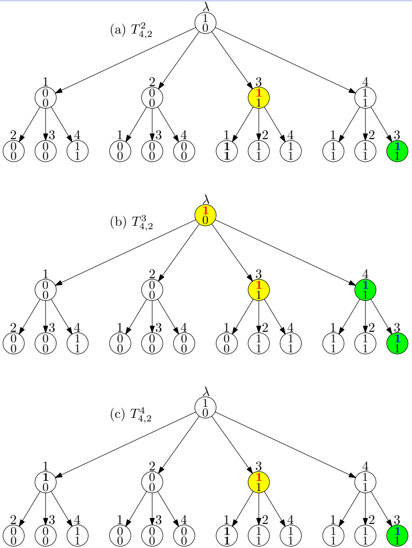
EIG trees for non-faulty processes



Process	ι_1	ι_2	ι_3	ι_4
Initial choice	0	0	1	1
Faulty	Yes	No	No	No
Round 1 messages	$(1, x)$	$(2, 0)$	$(3, 1)$	$(4, 1)$
Round 2 messages	$(2.1, 0)$ $(3.1, y)$ $(4.1, 1)$	$(1.2, 0)$ $(3.2, 1)$ $(4.2, 1)$	$(1.3, 0)$ $(2.3, 0)$ $(4.3, 1)$	$(1.4, 1)$ $(2.4, 0)$ $(3.4, 1)$
... Final decision	?	0	0	0

- α by top-down messaging
- L_1: (initial) $\iota_3 \overset{(3,1)}{\rightarrow} \iota_2, \iota_3, \iota_4$
- L_2: (relay) $\iota_3 \overset{(4.3,1)}{\rightarrow} \iota_2, \iota_3, \iota_4$
- β by bottom-up local voting
- common final decision, 0

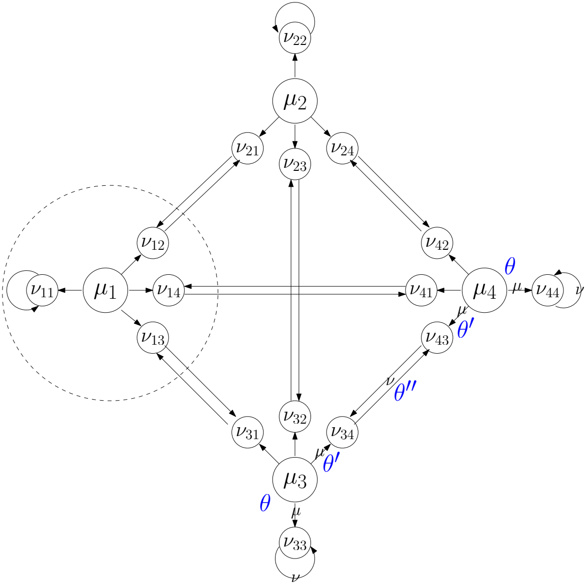
EIG trees for non-faulty processes



Process	ι_1	ι_2	ι_3	ι_4
Initial choice	0	0	1	1
Faulty	Yes	No	No	No
Round 1 messages	$(1, \mathbf{x})$	$(2, 0)$	$(3, 1)$	$(4, 1)$
Round 2 messages	$(2.1, 0)$ $(3.1, y)$ $(4.1, 1)$	$(1.2, 0)$ $(3.2, 1)$ $(4.2, 1)$	$(1.3, 0)$ $(2.3, 0)$ $(4.3, 1)$	$(1.4, 1)$ $(2.4, 0)$ $(3.4, 1)$
... Final decision	?	0	0	0

- α by top-down messaging
- L_1: (initial) $\iota_3 \stackrel{(3,1)}{\longrightarrow} \iota_2, \iota_3, \iota_4$
- L_2: (relay) $\iota_3 \stackrel{(4.3,1)}{\longrightarrow} \iota_2, \iota_3, \iota_4$
- β by bottom-up local voting
- common final decision, 0

Previous cP solution – without Actor features (2016)



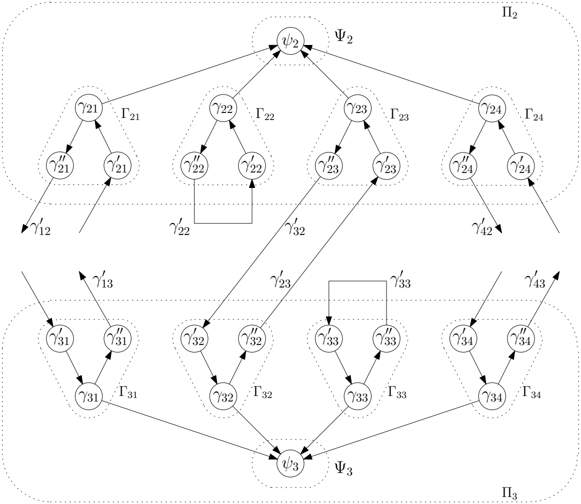
Firewall cells to protect:

- Against very badly formed messages
- Against wrongly timed messages
- Against Sybil-like attacks

Note: firewalls slow down the evolution, 5 or 4 times

An earlier more traditional P solution (2010)

- Just two nodes – even more firewall cells



Summary of complexity measures (where $L = \lfloor (N+2)/3 \rfloor$)

Measure	tP (2010)	cP (2016)	**This mod**
Cells per process	$3N+1$ $(2N+1)$	$N+1$	**1**
Atomic symbols	$\mathcal{O}(N!)$	18	1
States	$\mathcal{O}(L)$	14	5
Rules	$\mathcal{O}(N!)$	23	1
Ruleset size – Raw	2338	2218	14
Ruleset size – Compressed	624	591	52
Raw/Compressed ratio	3.75	3.75	2.
Steps per top-down level	5	4	2
Steps per bottom-up level	1	3 (1)	1

Note: cP systems have fixed-size alphabets and rulesets (no uniform families...)

Ruleset for sending messages (5 rules)

S_0	$\rightarrow_1$	$S_1\ \ell(0)\ \theta(\ell(0)\ \pi[]\ \rho()\ \alpha(V))$ $\|\|\ \bar{\alpha}(V)$
S_1	$\rightarrow_1$	$S_3\ \|\|\ \bar{\ell}(L)\ \|\|\ \ell(L)$
S_1	$\rightarrow_+$	$S_2\ !_\forall\{\theta'(\ell(L1)\ \pi[X\|P]\ \alpha(V))\}$ $\|\|\ \bar{\mu}(X)\ \|\|\ \ell(L)$ $\|\|\ \theta(\ell(L)\ \pi[P]\ \alpha(V)\ \rho(Z))$ $\neg\ (Z = XQ')$
S_1	$\rightarrow_+$	$S_2\ \theta(\ell(L1)\ \pi[X\|P]\ \alpha(V))$ $\|\|\ \ell(L)\ \|\|\ \bar{\pi}[X]\ \|\|\ \bar{v}_0(V)$ $\|\|\ \theta(\ell(L)\ \pi[P]\ \alpha(\_)\ \rho(Z))$ $\neg\ (Z = XQ')$
$S_1\ \ell(L)$	$\rightarrow_1$	$S_2\ \ell(L1)$

Ruleset for receiving messages (2 rules)

$$
\begin{array}{lll}
S_2 \; ?_Y\{\theta'(\ell(L1)\;\pi[Y|P]\;\alpha(V))\} & \rightarrow_+ & S_1 \; \theta(\ell(L1)\;\pi[Y|P]\;\rho(YQ)\;\alpha(V)) \\
\theta(\ell(L1)\;\pi[Y|P]\;\alpha(\_)) & & \| \; \theta(\ell(L)\;\pi[P]\;\rho(Q)\;\alpha(\_)) \\
 & & \| \; (Q \neq YQ') \\
 & & \| \; \bar{\delta}(V) \\
 & & \\
S_2 \; \theta(\ell(L)\;\pi[X|P]\;\alpha(\_)) & \rightarrow_+ & S_1 \; \theta(\ell(L)\;\pi[X|P]\;\alpha(V)) \\
 & & \| \; \ell(L1) \\
 & & \| \; \bar{v_0}(V)
\end{array}
$$

Ruleset for evaluating the EIG tree (5 rules)

$S_3\ \ell()\ \theta(\ell()\ \pi[]\ \alpha(V))$	$\rightarrow_1$	$S_4\ \omega(V)$
$S_3\ \theta(\ell(L1)\ \pi[\_\|P]\ \alpha(1))$ $\theta(\ell(L1)\ \pi[\_\|P]\ \alpha(0))$	$\rightarrow_+$	S_3 $\|\|\ \ell(L1)$
$S_3\ \theta(\ell(L1)\ \pi[\_\|P]\ \alpha(X))$ $\theta(\ell(L)\ \pi[P]\ \alpha(\_))$	$\rightarrow_+$	$S_3\ \theta(\ell(L)\ \pi[P]\ \alpha(X))$ $\|\|\ \ell(L1)$
$S_3\ \theta(\ell(L1)\_)$	$\rightarrow_+$	S_3 $\|\|\ \ell(L1)$
$S_3\ \ell(L1)$	$\rightarrow_1$	$S_3\ \ell(L)$

Thanks

- Thank you for your attention!
- Questions and feedback welcome!

Unbounded non-determinism – fairness beyond Turing?

- A terminating asynchronous non-deterministic system that can generate any number!
- The counter actor cell

$$
\begin{array}{llll}
S_0 & \rightarrow_{1} & S_0 \;\; !_0\{1\} \;\; \iota() \;\; \neg \;\; \iota(X) & (0) \\
S_0 \;\; ?_0\{1\} \;\; \iota(X) & \rightarrow_{1} & S_0 \;\; !_0\{1\} \;\; \iota(X1) & (1) \\
S_0 \;\; ?_1\{1\} \;\; \iota(X) & \rightarrow_{1} & S_1 \;\; !_1\{X\} & (2)
\end{array}
$$

- The main actor cell

$$
\begin{array}{llll}
S_0 & \rightarrow_{1} & S_1 \;\; !_1\{1\} & (0) \\
S_1 \;\; ?_1\{X\} & \rightarrow_{1} & S_2 \;\; \ldots & (1)
\end{array}
$$

Unbounded non-determinism – fairness beyond Turing?

- A terminating asynchronous non-deterministic system that can generate any number!
- The counter actor cell

$$
\begin{array}{llllll}
S_0 & & \rightarrow_1 & S_0 \;\; !_0\{1\} \;\; \iota() \;\; \neg \;\; \iota(X) & & (0) \\
S_0 & ?_0\{1\} \;\; \iota(X) & \rightarrow_1 & S_0 \;\; !_0\{1\} \;\; \iota(X1) & & (1) \\
S_0 & ?_1\{1\} \;\; \iota(X) & \rightarrow_1 & S_1 \;\; !_1\{X\} & & (2)
\end{array}
$$

- The main actor cell

$$
\begin{array}{llll}
S_0 & \rightarrow_1 & S_1 \;\; !_1\{1\} & (0) \\
S_1 \;\; ?_1\{X\} & \rightarrow_1 & S_2 \;\; \ldots & (1)
\end{array}
$$

Unbounded non-determinism – fairness beyond Turing?

- A terminating asynchronous non-deterministic system that can generate any number!
- The counter actor cell

$$\begin{array}{llll}
S_0 & \rightarrow_1 & S_0 \; !_0\{1\} \; \iota() \; \neg \; \iota(X) & (0) \\
S_0 \; ?_0\{1\} \; \iota(X) & \rightarrow_1 & S_0 \; !_0\{1\} \; \iota(X1) & (1) \\
S_0 \; ?_1\{1\} \; \iota(X) & \rightarrow_1 & S_1 \; !_1\{X\} & (2)
\end{array}$$

- The main actor cell

$$\begin{array}{llll}
S_0 & \rightarrow_1 & S_1 \; !_1\{1\} & (0) \\
S_1 \; ?_1\{X\} & \rightarrow_1 & S_2 \; \ldots & (1)
\end{array}$$