Interval Tree Clocks
A Logical Clock for Dynamic Systems

P.S. Almeida, C. Baquero, V. Fonte

Presenter: Yi Zhang

October 29, 2014

Acknowledgement: all figures from the presented paper.

Overview

Problem Setting

- In a distributed system, i.e. a collection of hosts.
- Events happen involving one or more of hosts in some logical order.
- Hosts may be added or removed dynamically.
- Want to keep track of what happened before and after what.

Logical Clocks

- Logical clock: A mechanism keeping track of causalities of events in a distributed system.
- Formally: A partial order on all the events in the system.

Partial Order

A partial order "$\leq$" on the set of event E satisfies, for any $a, b, c \in E$:

- $a \leq a$: reflexive
- if $a \leq b$ and $b \leq a$, then $a = b$: antisymmetric
- if $a \leq b$ and $b \leq c$, then $a \leq c$: transitive

Note: doesn't have to be 'complete'.

Related Work

- Version vectors.
- Vector clocks.
- Matrix clocks.

Stamps

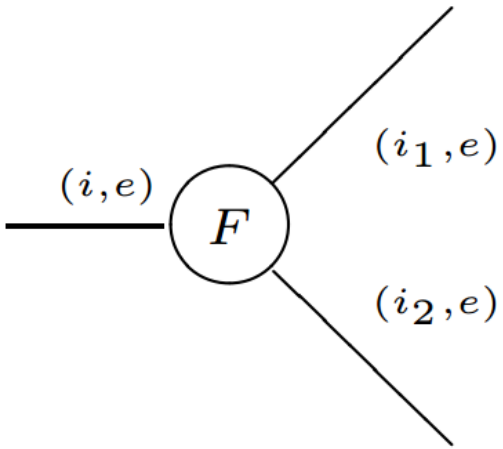
- A stamp is just a record of who did what.
- Formally, a stamp is a pair (i, e), where i defines an id (the 'who') and e defines an event (the 'what').
- i can be bit strings, can be string labels, or functions, etc.

Fork-Event-Join Model

- Use these 3 core operations acting on stamps to model the logic of events.
- They can be composed to describe other events, e.g. send, receive, sync.

Fork

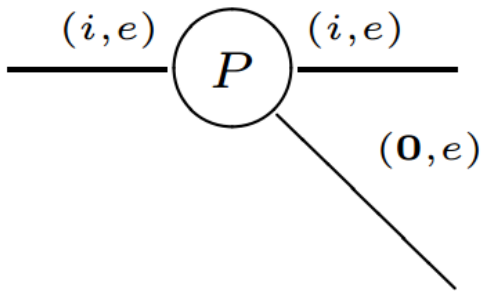
- A fork is a split of a single event to two ids.
- $\texttt{fork}(i, e) = ((i_1, e), (i_2, e))$.



(a) fork

Peek

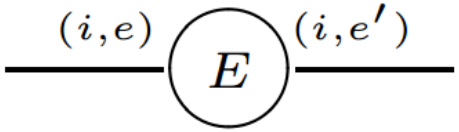
- A peek is a special fork where the event is 'peeked' by an *anonymous* id.
- $\texttt{peek}(i, e) = ((i, e), (\mathbf{0}, e))$.



(b) peek

Event

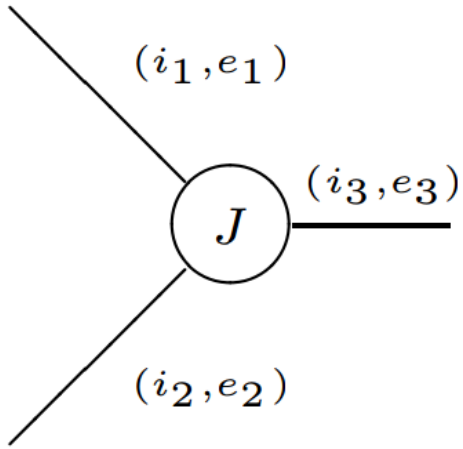
- An event simply updates the event on an id.
- $\texttt{event}(i, e) = (i, e')$.



(c) event

Join

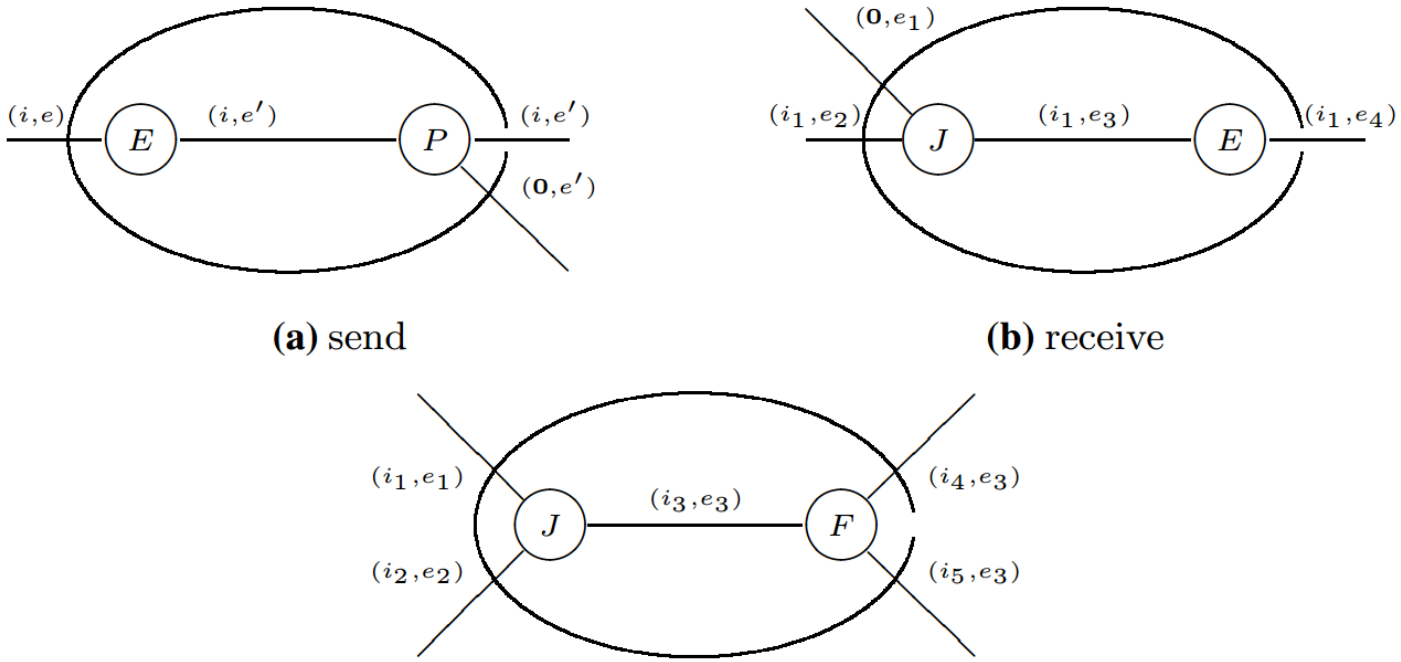
- A join simply joins two stamps together and produces an updated stamp.
- $\texttt{join}((i_1, e_1), (i_2, e_2)) = (i_3, e_3)$.



(d) join

Other Classic Examples

- send = event + peek.
- receive = join + event.
- sync = join + fork.



(a) send

(b) receive

(c) sync

Function Based Mechanism

- We use functions on the interval $[0, 1)$ to represent both the id and the event in the stamp (i, e).
- Functions for i are from $[0, 1)$ to $\{0, 1\}$.
- Functions for e are from $[0, 1)$ to $\mathbb{Z}_{\geq 0}$.

Function Based Mechanism: Example

id:

$$(1,(0,1)) \sim$$

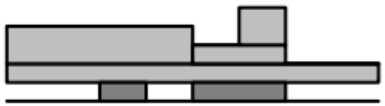
$$((0,(1,0)),(1,0)) \sim$$

event:

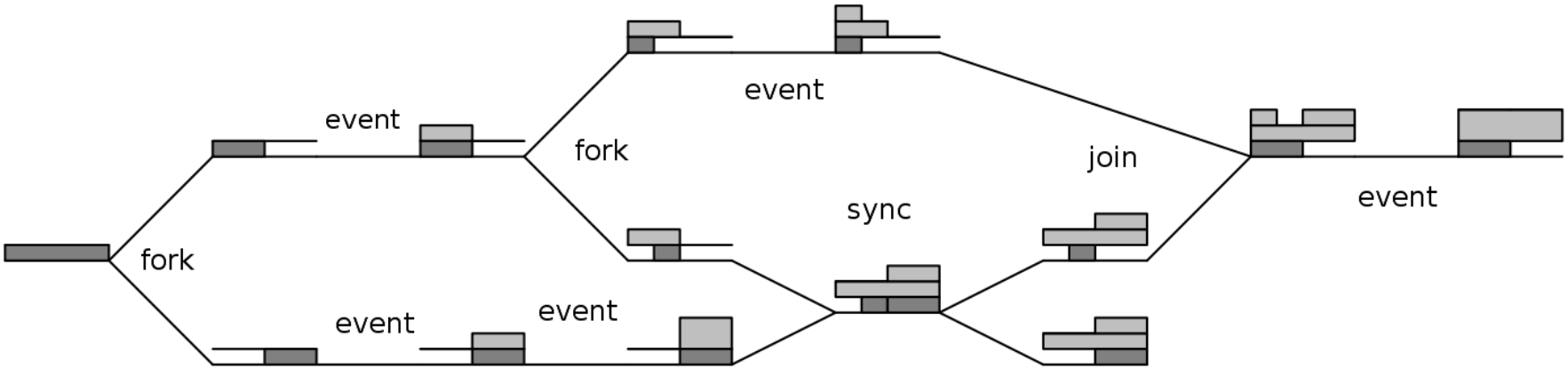
$$(1,2,(0,(1,0,2),0)) \sim$$

together:

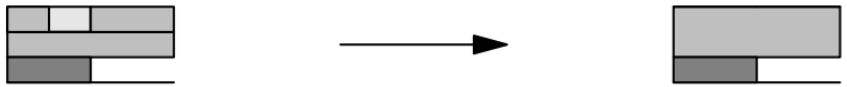
$$(((0,(1,0)),(1,0)),(1,2,(0,(1,0,2),0))) \sim$$



The Model: Example



Normalization and Operation

- Normalization: 
- Comparison: $(i_1, e_1) \leq (i_2, e_2)$ if and only if

$$e_1(x) \leq e_2(x) \quad \forall x \in [0, 1)$$

- Fork, Join, Event, etc.

Experimental Results

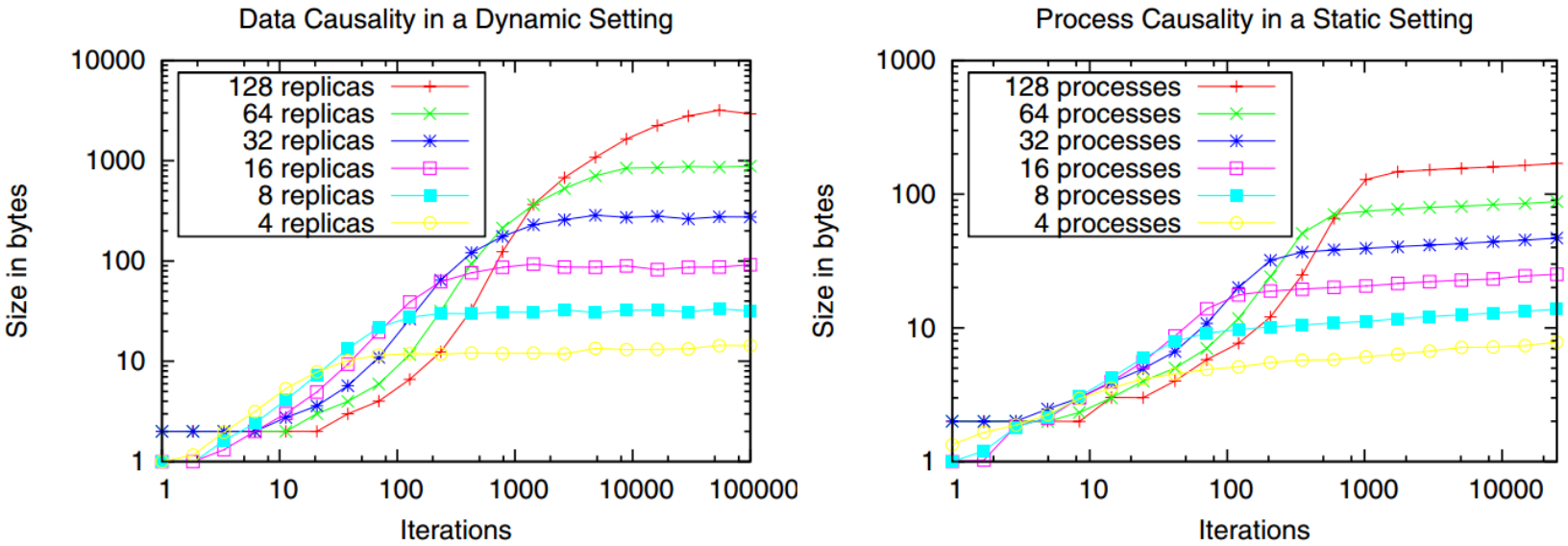


Fig. 3. Average space consumption of an ITC stamp, in dynamic and static settings

- Storage size stabilizes!

Conclusion

- Interval Tree Clock is a novel and convenient logical clock.
- Completely decentralized.
- Works really well in dynamic systems where hosts come and go.
- Practically feasible, verified by experimental results.

Interval Tree Clock

Thank you.