EpTO: An Epidemic Total Order Algorithm for Large-Scale Distributed Systems

2015 ACM/IFIP/USENIX Middleware conference

Miguel Matos, **Hugues Mercier**,
Pascal Felber, Rui Oliveira, José Pereira

Institutes of Computer Science and Mathematics
Université de Neuchâtel, Switzerland

HASLab – INESC TEC & University of Minho, Portugal

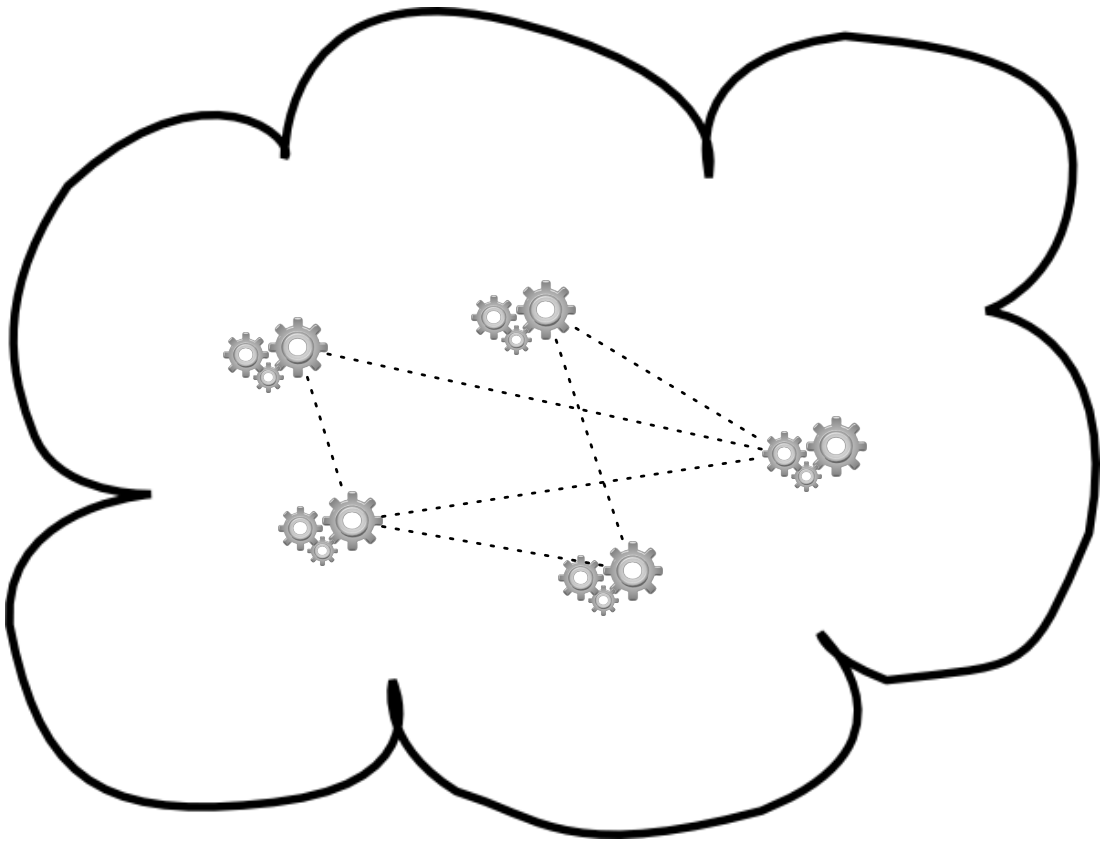
hugues.mercier@unine.ch

Vancouver, Canada, 9 December 2015

Disclaimer:
Everyone in this room is more knowledgeable
about middleware systems than I am.

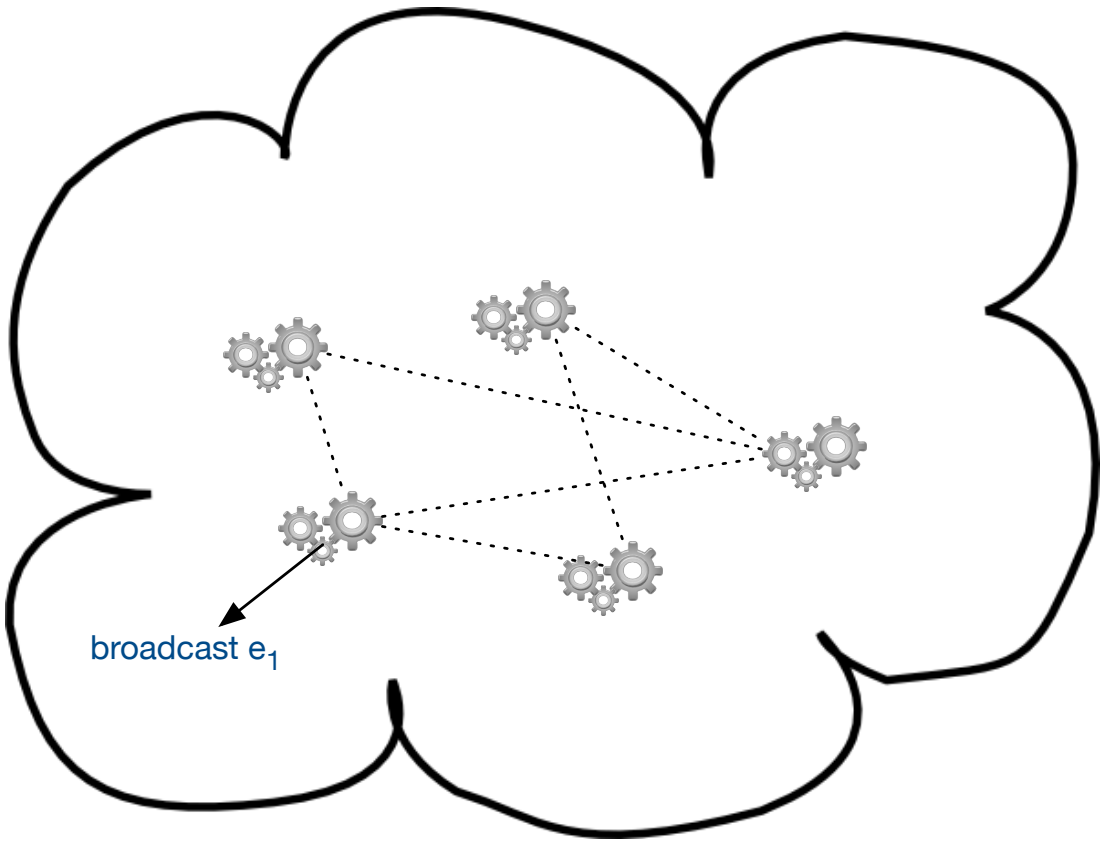
A very simple problem ...

- A set of processes in a network broadcast events, and each process must deliver every event to the application in total order.



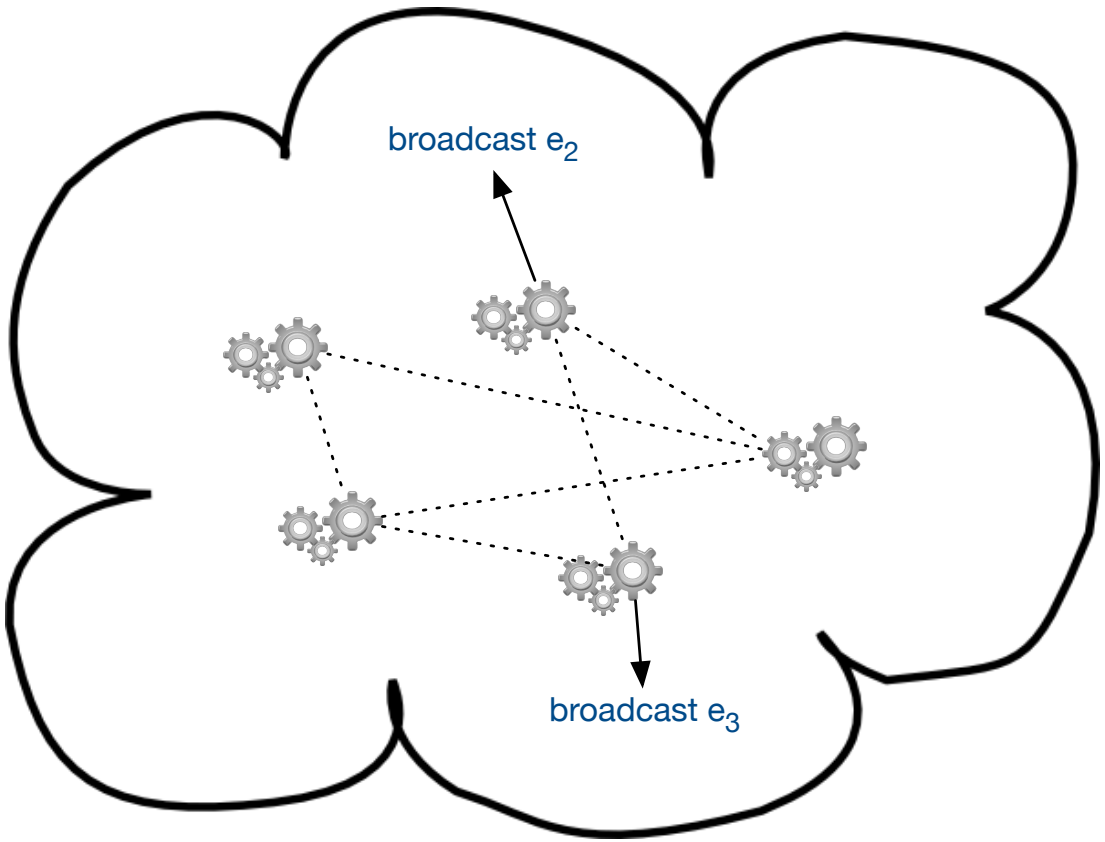
A very simple problem ...

- A set of processes in a network broadcast events, and each process must deliver every event to the application in total order.



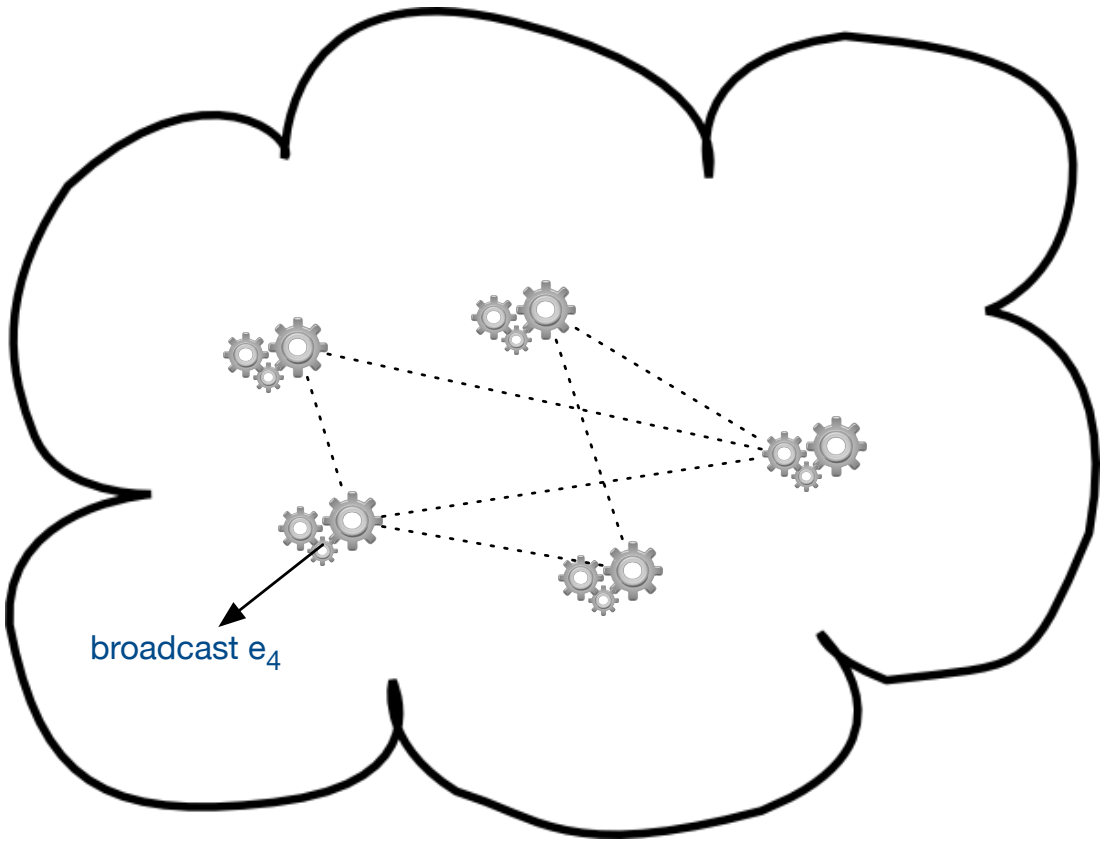
A very simple problem ...

- A set of processes in a network broadcast events, and each process must deliver every event to the application in total order.



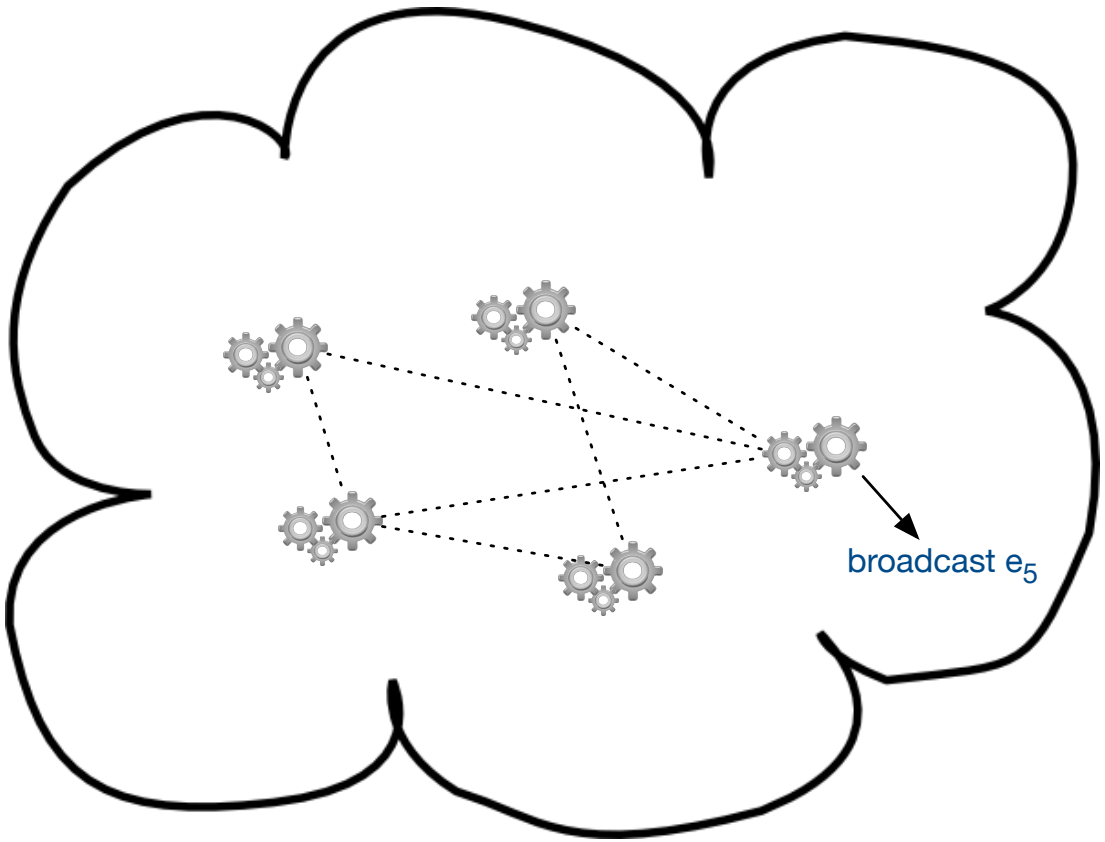
A very simple problem ...

- A set of processes in a network broadcast events, and each process must deliver every event to the application in total order.



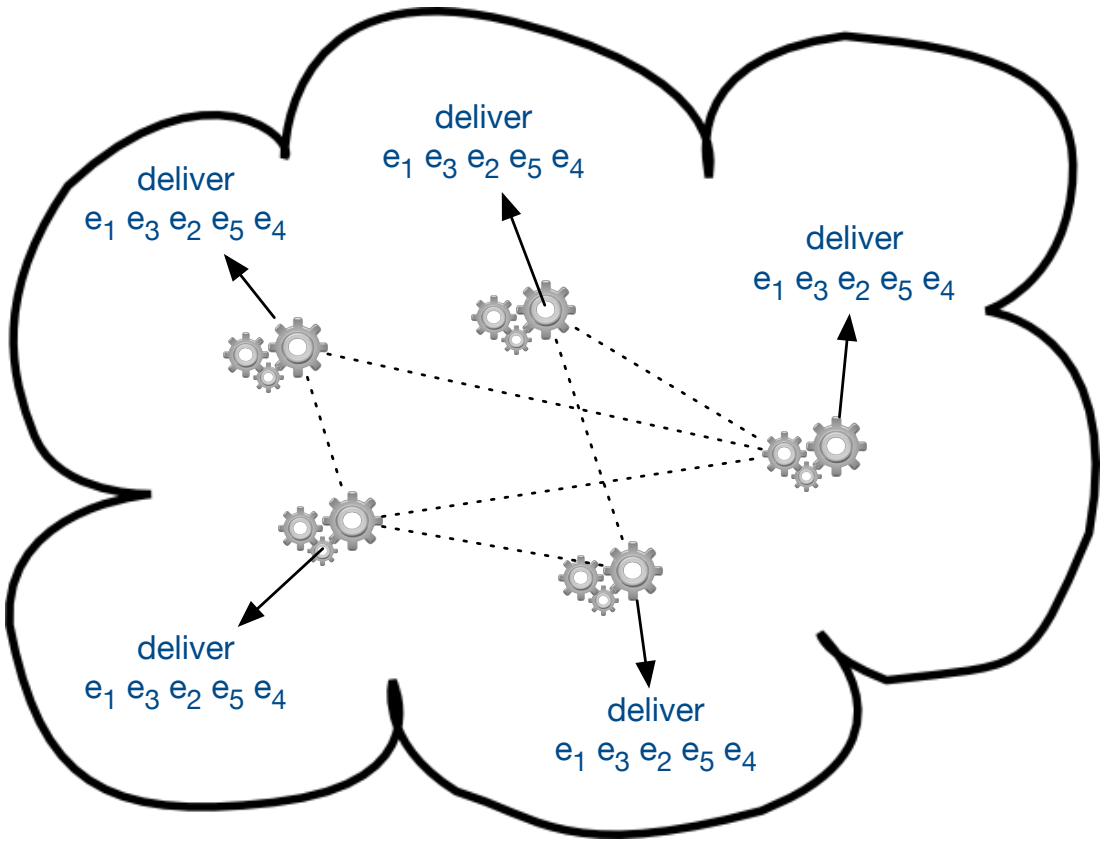
A very simple problem ...

- A set of processes in a network broadcast events, and each process must deliver every event to the application in total order.



A very simple problem ...

- A set of processes in a network broadcast events, and each process must deliver every event to the application in total order.



... actually, maybe not so simple

- The ordering of events is one of the most fundamental problems in distributed systems.
- A large body of research has been dedicated to the design of ordering abstractions with different guarantees and tradeoffs.

In real-life dynamic large-scale settings, deterministic algorithms are prohibitively expensive and/or non-scalable.

Probabilistic dissemination protocols are highly scalable and resilient to adversarial network conditions, but often overlook stronger properties such as ordering.

State of the art

- Deterministic total order algorithms [Défago et al. 2004]
- Optimistic total order algorithms [Pedone & Schiper 2003]
- Deterministic algorithms leveraging eventual consistency [Baldoni et al. 2006, Vogels 2009].
- Deterministic algorithms with probabilistic delivery guarantees [Ezhilchelvan 2011] (no ordering properties, not scalable without probabilistic dissemination).
- Probabilistic algorithms with no or weak ordering guarantees [Birman et al. 1999, Koldehofe 2002, Eugster et al. 2003, Carvalho et al. 2007, ...]
- Probabilistic total order algorithms assuming static and fully synchronous networks [Hayden & Birman 1996]
- ...

To the best of our knowledge, no scalable and robust total order algorithm is available.

Our contribution: EpTO

- We present and analyze EpTO, a new total order epidemic dissemination algorithm with probabilistic reliability guarantees.
- EpTO guarantees that processes eventually agree on the set of received events with high probability and deliver these events in total order to the application.

Advantages of EpTO

EpTO is well-suited for large-scale dynamic distributed systems:

- Conceptually simple and fully decentralized, not requiring any form of coordination among processes.
- Does not require sub-protocols, acknowledgments nor retransmissions.
- Scalable as the number of events and processes increases.
- Does not require a global clock nor synchronized processes.
- Highly robust even when the network suffers from large delays and significant churn and message loss.

EpTO properties

Deterministic Integrity: Every process delivers an event at most once, and only if it was previously broadcast.

Deterministic Validity: Correct processes eventually deliver the events they broadcast.

Deterministic Total Order: Process p delivers event e before e' if and only if process q delivers e before e'.

Probabilistic Agreement: If a process broadcasts an event e, then with high probability all correct processes eventually deliver it.

EpTO properties

Straightforward to prove

Deterministic Integrity: Every process delivers an event at most once, and only if it was previously broadcast.

Deterministic Validity: Correct processes eventually deliver the events they broadcast.

Deterministic Total Order: Process p delivers event e before e' if and only if process q delivers e before e'.

Probabilistic Agreement: If a process broadcasts an event e, then with high probability all correct processes eventually deliver it.

EpTO properties

Straightforward to prove

Deterministic Integrity: Every process delivers an event at most once, and only if it was previously broadcast.

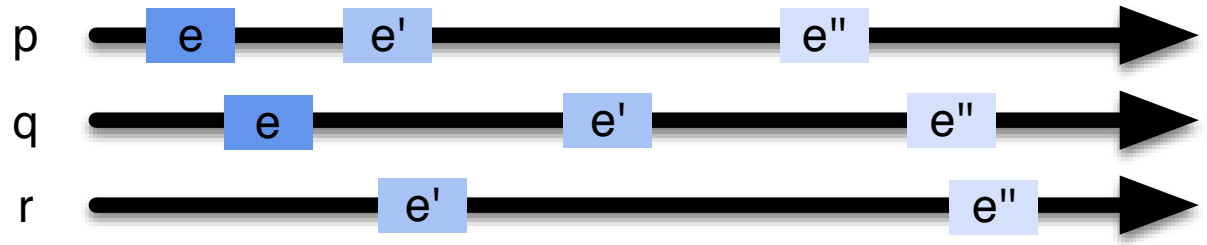
Deterministic Validity: Correct processes eventually deliver the events they broadcast.

Deterministic Total Order: Process p delivers event e before e' if and only if process q delivers e before e'.

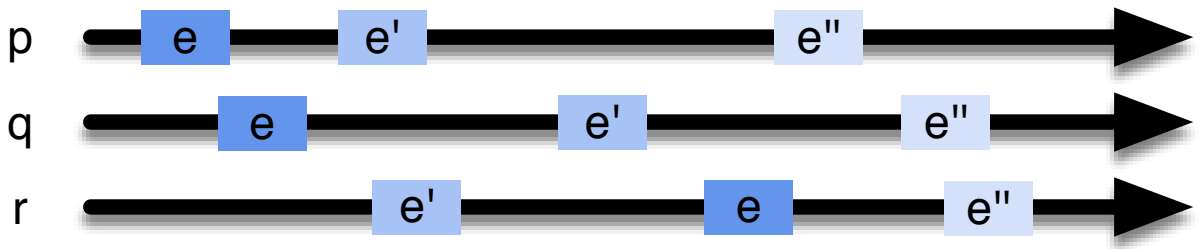
More challenging

Probabilistic Agreement: If a process broadcasts an event e, then with high probability all correct processes eventually deliver it.

Probabilistic agreement may create holes in the sequence of delivered events



Unlikely but allowed

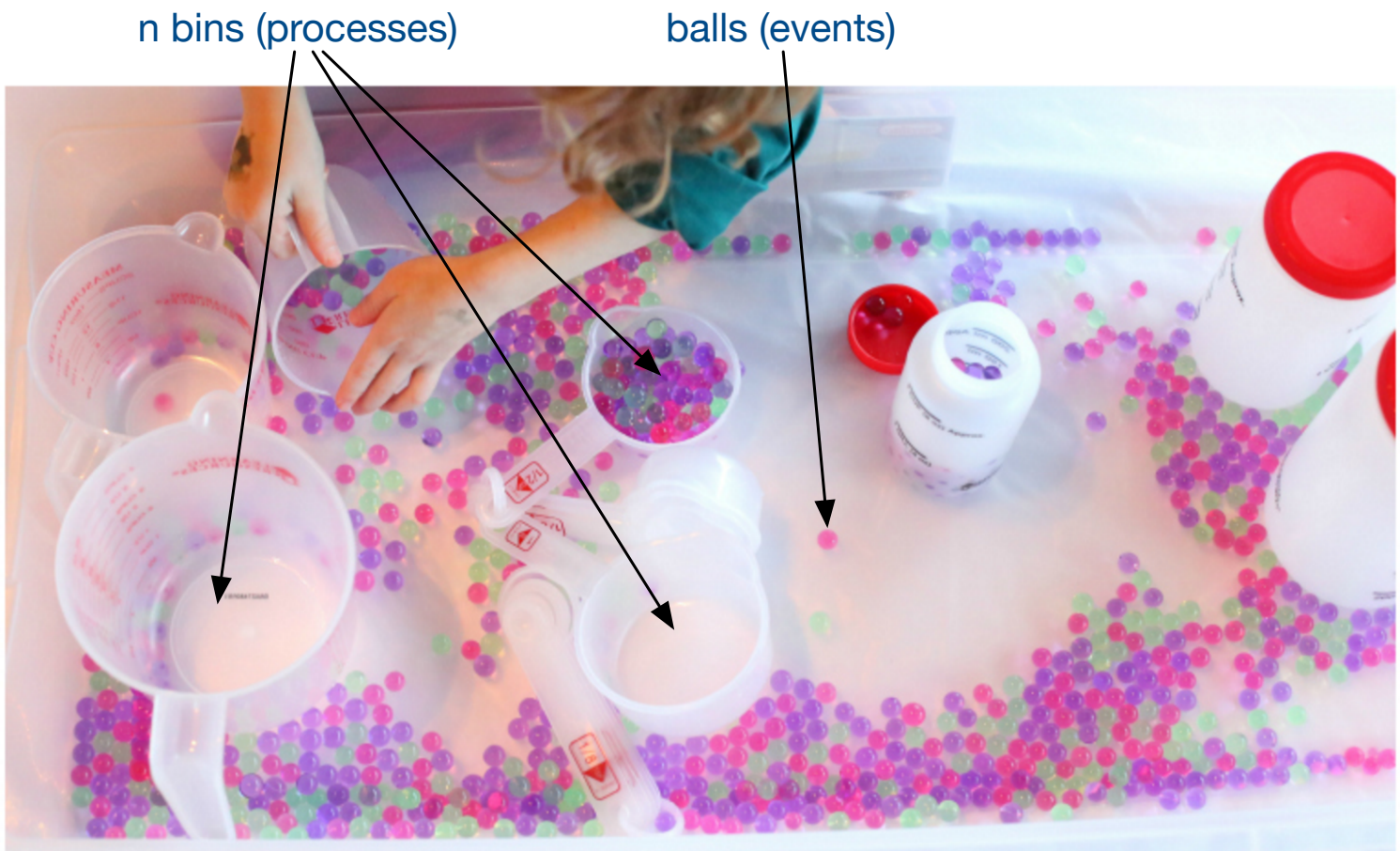


Not allowed

Gossiping with balls and bins

Gossiping with balls and bins

- The n processes are abstracted as n bins, and events (rumors) are abstracted as balls.

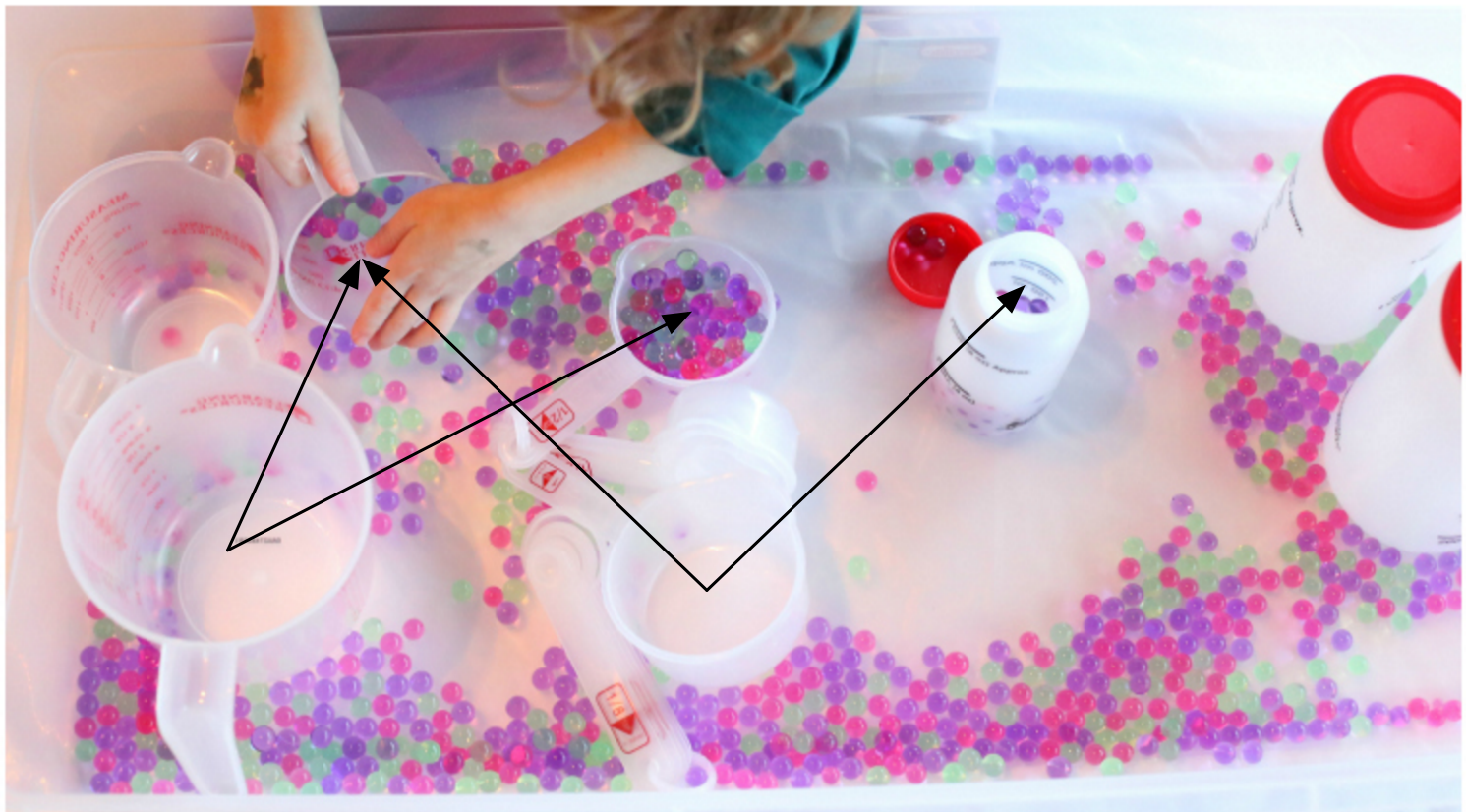


Gossiping with balls and bins

- The process starting a rumor sends a ball to K other processes chosen uniformly at random.

Gossiping with balls and bins

- For each round that follows, the processes which received one or more balls in the previous round a send a ball to K other processes chosen uniformly at random.
- The protocol terminates after *TTL* rounds.



Gossiping with balls and bins

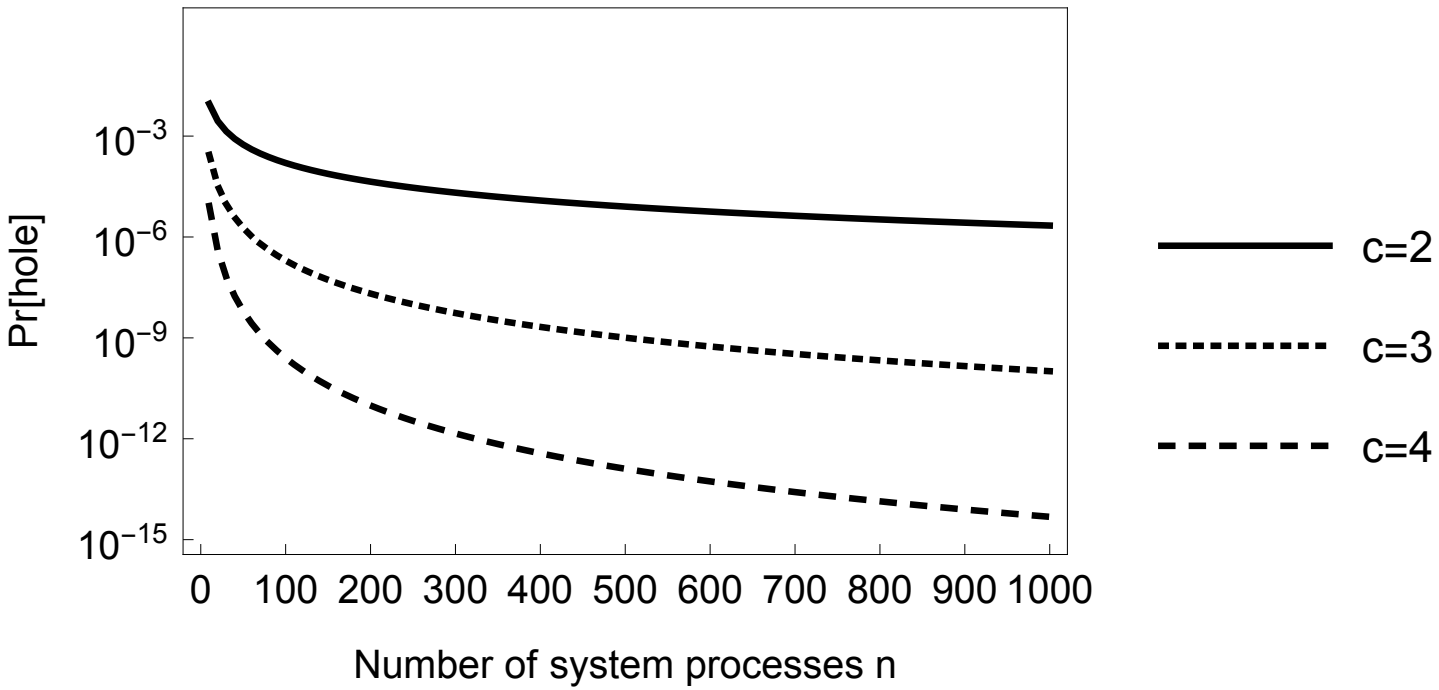
Theorem (Koldehofe 2002)

If the gossip protocol uses $K = \lceil \frac{2e \ln n}{\ln \ln n} \rceil$ *balls per process and runs for* $TTL = (c+1) \log_2 n$ *rounds, where* $c > 1$ *is a constant, then each process learns the rumor with probability* $1 - O\left(n^{-(c+1)}\right)$.

Proof idea.

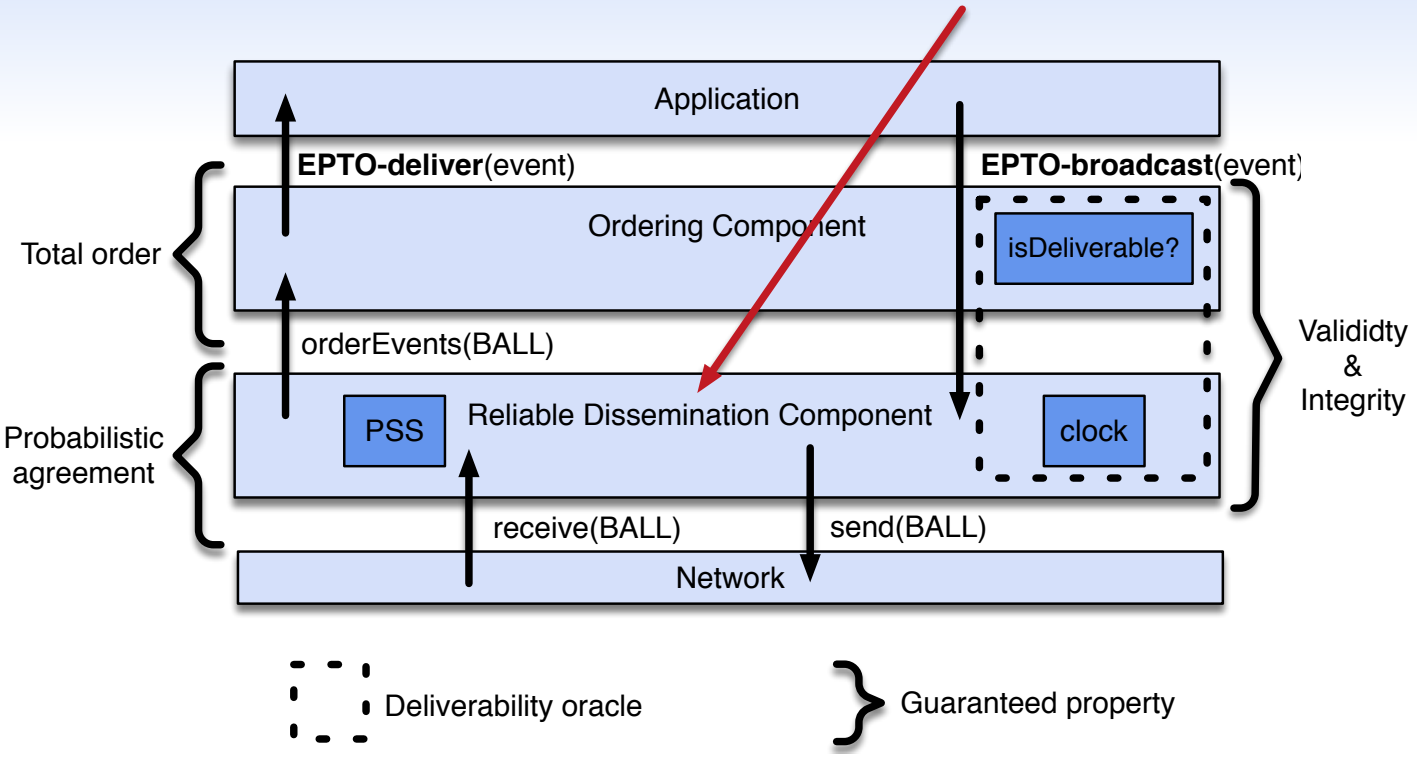
- During the first $\log_2 n$ rounds, the number of balls disseminated doubles at each round until at least n balls are transmitted per round.
- The last $c \log_2 n$ rounds create at least $cn \log_2 n$ balls, which is sufficient to conclude that each process has received at least one ball with high probability.

The probability of hole can be made orders of magnitude smaller than the probability of a catastrophic hardware or network failure



Probability that event e has a hole for at least one process.

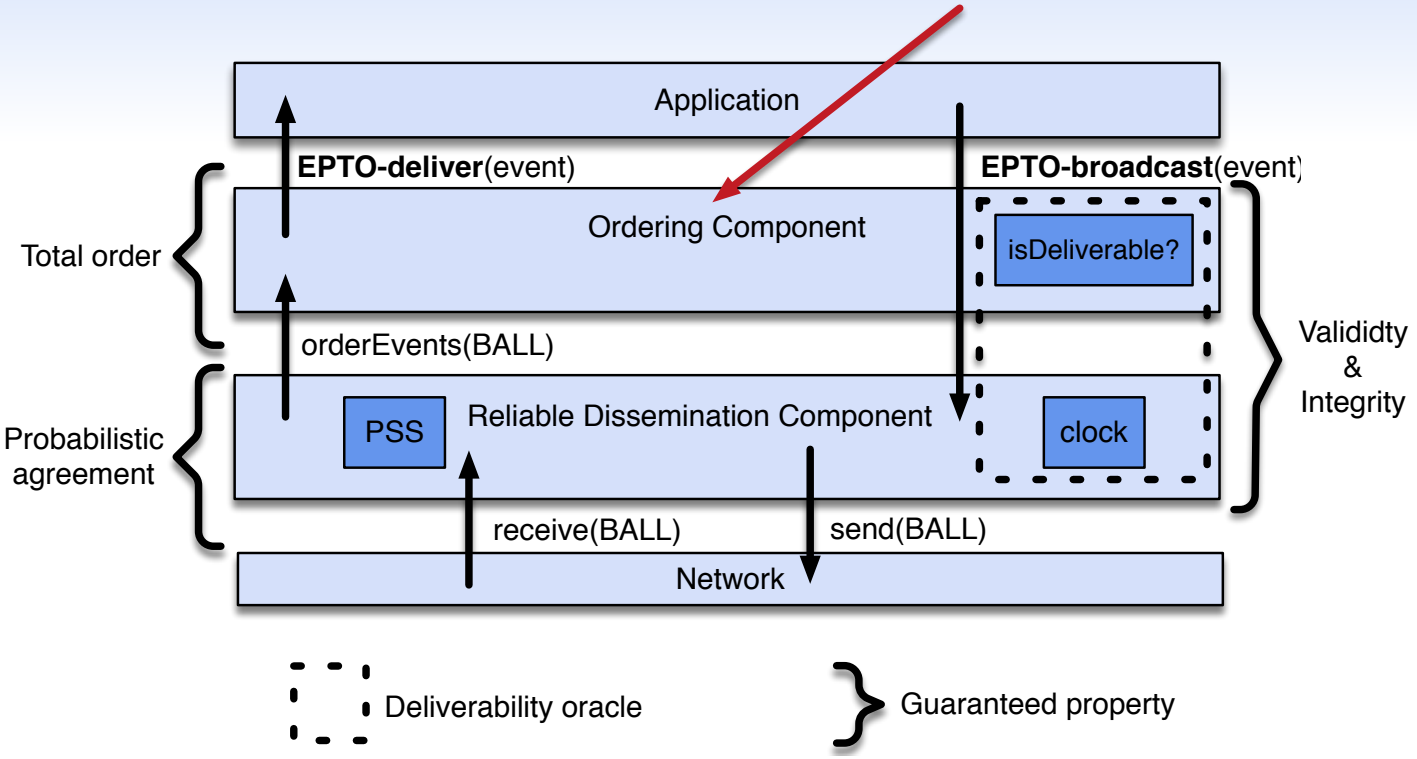
EpTO architecture - dissemination component



The dissemination component handles the reception and retransmission of events.

- Each event in a ball contains its *id*, *timestamp*, and age (*ttl*).
- When a ball is received, the age of its events is incremented.
- At each round, each process groups all the received events in the same ball for the next round.

EpTO architecture - ordering component



The ordering component is responsible of the total order property.

- It orders received events based on their timestamp.
- It ages the received but not yet delivered events.
- It delivers stable (old enough) events to the application.

Probabilistic agreement property - synchronous EpTO

Completely unrealistic fairy-tale assumptions:
no churn, access to global time, no lost messages, synchronous rounds and no network latency!!!

Lemma

If $K \geq \lceil \frac{2e \ln n}{\ln \ln n} \rceil$ *and* $TTL \geq \lceil (c+1) \log_2 n \rceil$ *where* $c > 1$, *then synchronous* EpTO *satisfies the Probabilistic Agreement property.*

Proof idea.

- Disseminating events with EpTO corresponds to *aging* them.
- When an event *ttl* reaches *TTL*, a process locally knows that this event has been in the system long enough to reach all other processes with high probability and can be delivered.
- Every event is transmitted $\sim cn \log_2 n$ times.

EpTO works when all these assumptions are lifted

- Logical clocks ✔
- Process drift ✔
- Latency ✔
- Churn and message loss ✔

In all cases, we slightly increase the number of rounds and/or the fan-out K and formally prove that every process still receives every event with high probability.

Probabilistic agreement property - churn and message loss

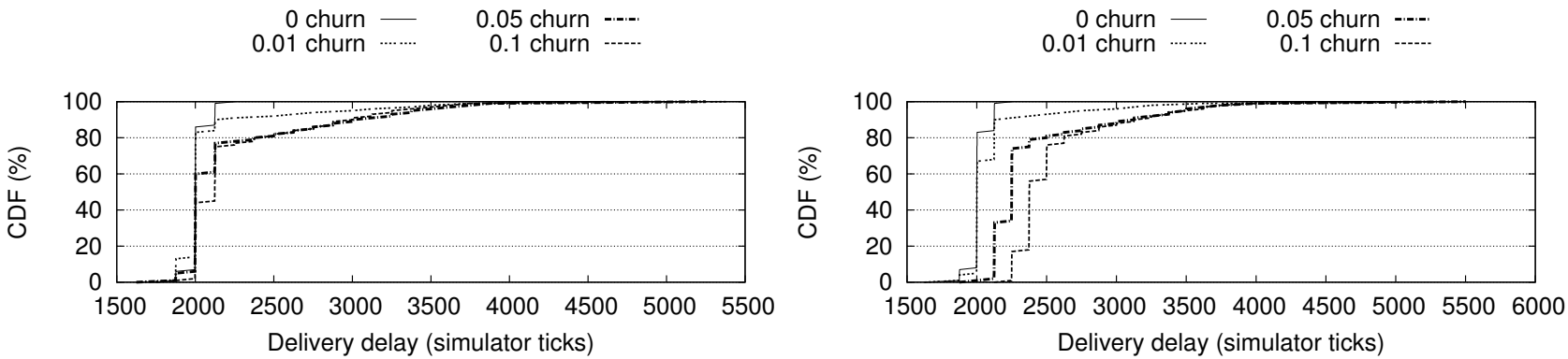
Lemma

Let α and ϵ define the churn and message loss rate of the network, and let $K = \left\lceil \frac{2e \ln n}{\ln \ln n} \cdot \frac{n}{n-\alpha} \cdot \frac{1}{1-\epsilon} \right\rceil$. *If* $TTL \geq \lceil (c+1) \log_2 n \rceil$, *then* EpTO *satisfies the Probabilistic Agreement property.*

EpTO guarantees total ordering and good performance in the presence of significant churn and message loss for large-scale applications.

Some simulation results

In all our experiments, we have not observed a single hole in the sequence of delivered events (I don't count the bugs :-))



Delivery delay under churn for 500 processes, logical clocks and 5% broadcast rate. The end-to-end latency distribution drawn from a sample of geographically dispersed PlanetLab nodes. Left figure assumed perfect view, whereas right figure used the Cyclon peer sampling service.

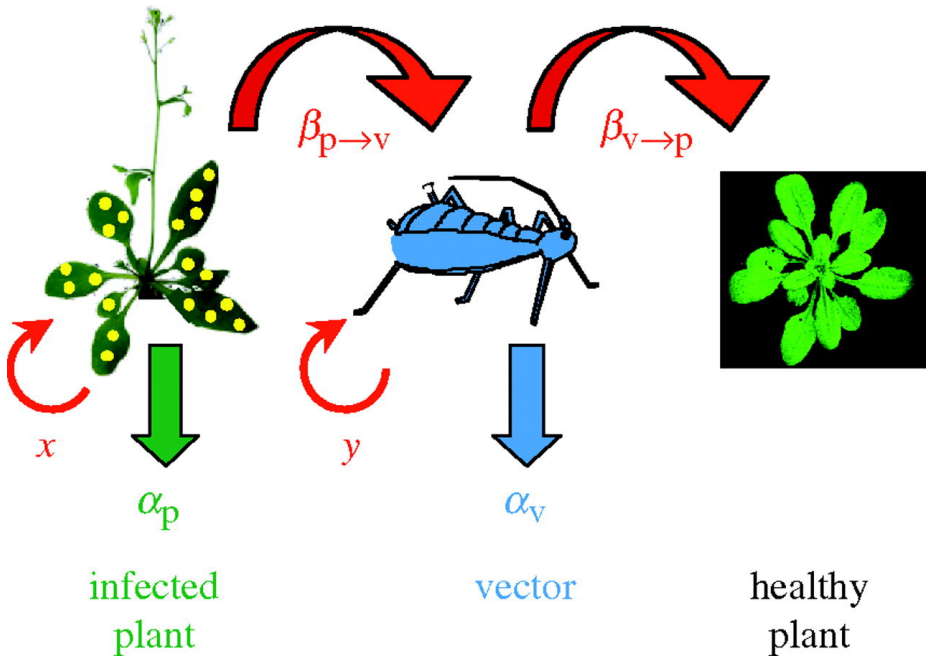
Concluding remarks

- Traditional total order algorithms do not scale well, nor do they work well under challenging network conditions.
- EpTO solves both problems at the same time.

  - In the worst case, we can set the algorithm parameters based on the well-behaving nodes (or based on good network conditions) and guarantee that the well-behaving part of the network will satisfy the Probabilistic Agreement property.
  - Bad processes can remain in the network and do not affect the performance of the algorithm. No deterministic algorithm has this property.

Future work

- *In theory there is no difference between theory and practice. But, in practice, there is.* From simulation to real-world implementation.
- The asymptotics of balls-into-bins problems for probabilistic dissemination can be improved. We found similarities with biometric problems studied in the 1960s estimating the proportion of vectors capable of transmitting viruses in natural populations.



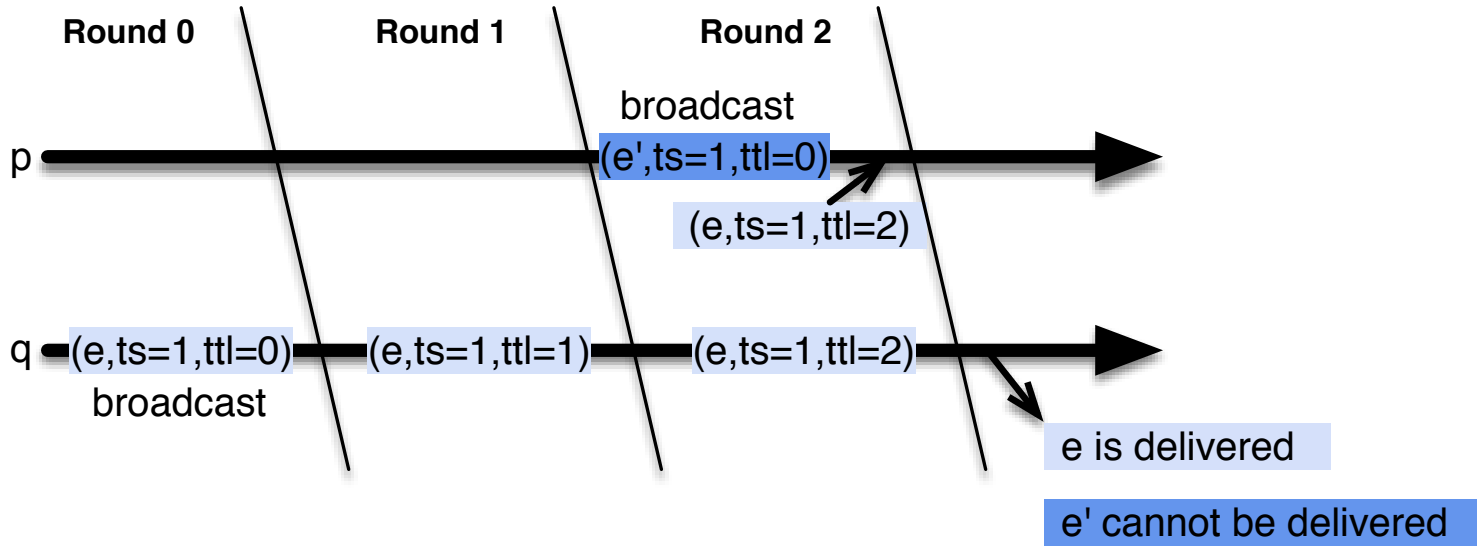
Thanks for your time. Questions?

Probabilistic agreement property - logical time EpTO

Lemma

If $K \geq \lceil \frac{2e \ln n}{\ln \ln n} \rceil$ *and* $TTL \geq 2\lceil (c+1) \log_2 n \rceil$ *where* $c > 1$, *then* EpTO *with logical time satisfies the Probabilistic Agreement property.*

Proof idea.



□

Probabilistic agreement property - process drift

Lemma

Let assume that the round duration δ of each process is always bounded by $\delta_{\min} \leq \delta \leq \delta_{\max}$, that $K \geq \left\lceil \frac{2e \ln n}{\ln \ln n} \right\rceil$, and that $c > 1$. If $TTL \geq 2\lceil (c+1) \log_2 n \rceil \cdot \frac{\delta_{\max}}{\delta_{\min}}$, then EpTO *with logical time satisfies the Probabilistic Agreement property.*

Proof idea.

The dissemination of an event in the network can be represented as a tree, where the root is the process which broadcasted the event, nodes at depth d for $d \geq 1$ are the processes which are d balls away from the root, and the leaves are the processes that delivered the event.

The depth of the leaves of the tree are at depth at least $\lceil (c+1) \log_2 n \rceil$. □

Probabilistic agreement property - network latency

Lemma

Let $N_1 \subseteq N$ be a subset of the network such that the latency within N_1 is always bounded by L_{max}. If $\delta > L_{\max}$, then EpTO *with logical time satisfies the Probability Agreement property within N_1 with $K \geq \lceil \frac{2e \ln n}{\ln \ln n} \rceil$, $c > 1$ and $TTL \geq 2\lceil (c+1) \log_2 n \rceil + 1$.*

Proof idea.

Since the latency within N_1 is smaller than the round duration, it follows that messages will always reach their destination at most one round after they were transmitted. □

- By setting the round duration based on the latency of the well-behaving nodes (or on the latency under good network conditions), we can guarantee that the well-behaving part of the network will satisfy the Probabilistic Agreement property.
- The processes with large latency can remain in the network and do not slow down the algorithm.