Scalability with a bit of uncertainty

Paul Ezhilchelvan
Newcastle University

Talk Structure

Theme:

- Building large-scale applications in Clusters
  - Premise:
    - Worth looking into some ‘simple’ techniques
  - Investigation by Example
    - What property (ies) can be lost?
    - How to restore? How costly?
  - Proposed: Design Variants and Trade-offs
  - What we envisage

An Example

- Say, *a* sends *m* to *b* by TCP/IP
  - Assume no failures
  - When *a* completes its send operation
    - it knows that *m* reaches its destination
- Say, *a* needs to send *m* to *b1, b2, ..., b20*
  - 20 TCP Connections give *a* the same knowledge
- What if *a* has to send to *b1, b2, ...,* $b10^4$
  - Should *a make* 10000 TCP connections?
  - Need a scalable dissemination protocol

Gossip

- *a* sends *m* to a small, *randomly-selected* subset of *b*'s
- So does every *b* that receives *m*
- For all 10^4 to receive *m*, *expected* number of connections needed:
  - 10001 × 20
  - $n \times 2[(\ln(n) + 0.5772)]$ (Ezhilchelvan, Mitrani 2006)
- 10001 × 20 shared among 10001 processes
- Incorporates growing parallelisation

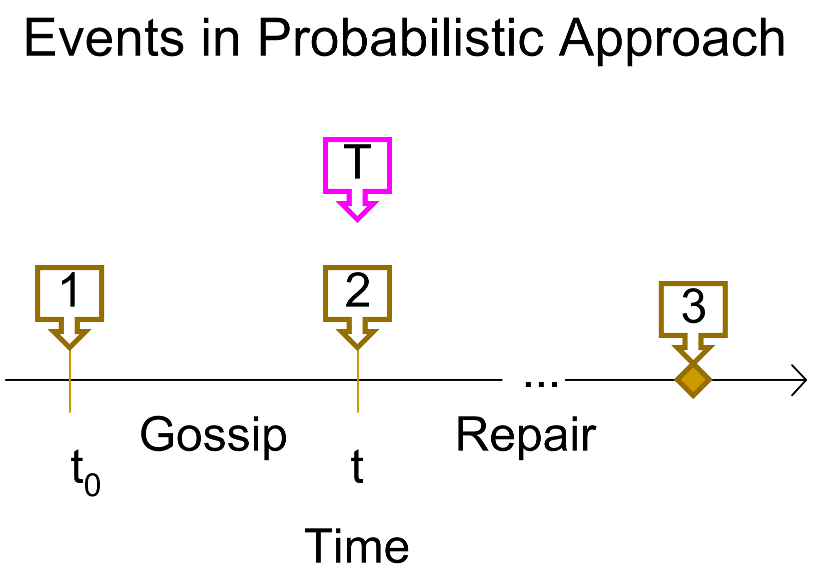
Near Certainty to Certainty

- What is lost through gossiping?
  - certainty on the outcome when gossip terminates
    - 2 is the *expected* number of coin tosses for a desirable outcome (e.g., head)
  - A process that gossiped *m* knows:
    - all *b* processes receive *m* with a high probability
- Say, future gossips carry *m* in their history
- At some time in future (eventually, ◊)
  - all *b* processes receive *m* with probability = 1
  - Certainty is feasible albeit at a cost

Certainty, Cost and Termination

- Gossip of *m* Starts (1)
- Gossip of *m* terminates (2)
- All omissions of *m* repaired (3)

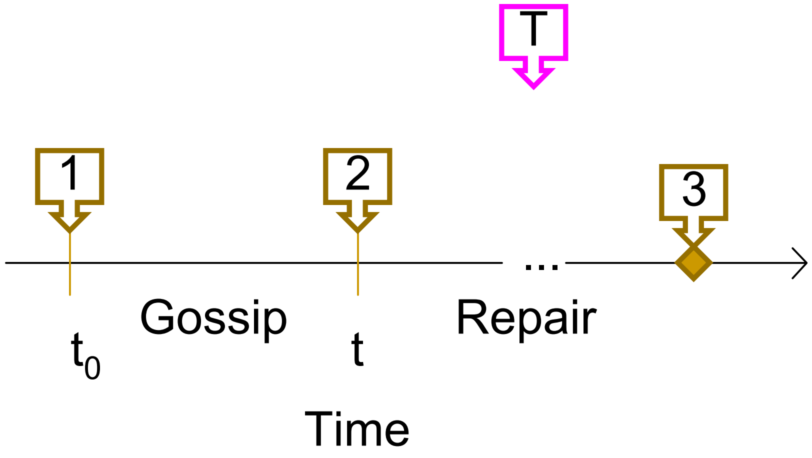
- A dissemination protocol can be designed to have:
  - Just gossip and no repair



Certainty, Cost and Termination

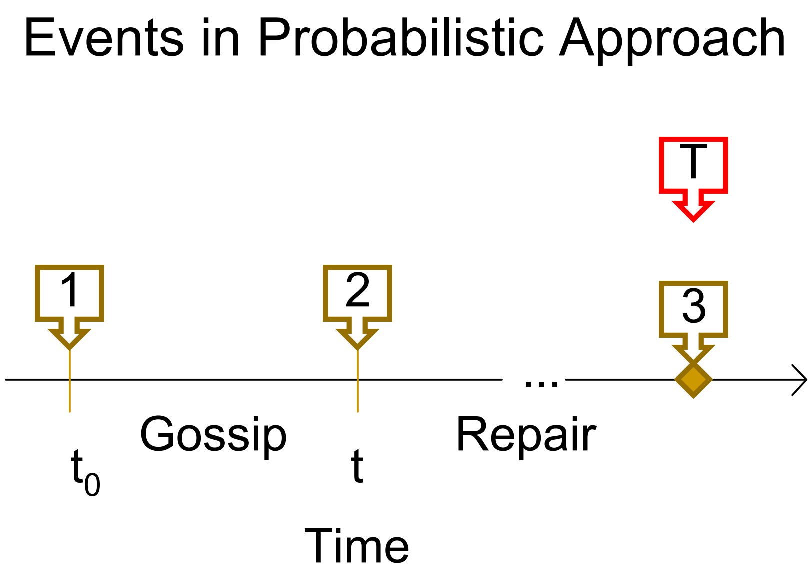
- Gossip Starts (1)
- Gossip terminates (2)
- All omissions repaired (3)
- A dissemination protocol can be designed to have:
  - Just gossip and no repair, or
  - Gossip + 'some' repair
- as (T-t) increases
  - Outcome is more certain
  - Cost also increases

Events in Probabilistic Approach



Certainty, Cost and Termination

- Gossip Starts (1)
- Gossip terminates (2)
- All omissions repaired (3)
- A dissemination protocol can be designed to have:
  - Just gossip and no repair, or
  - Gossip + ‘some’ repair, or
  - Gossip + full repair
- as T becomes ◊
  - Outcome is certain
  - Cost Maximum
  - Harder to estimate T



Probabilistic vs. Randomized

- Protocol is randomized (R-type) if T is ◊
- It is probabilistic (P-Type), otherwise
- When gossip is effective, say, coverage = 90%
  - Median or upper quartile latency is identical for all P and R
  - Average latency among those received *m*
    - increases as T is delayed in P
    - the largest in R
  - During Repair
    - 90% observation, 10% work
      - Like the Security in Superstores
    - Computational complexity is not much
    - Ensuring full repair in R is the hard part
  - P and R offer
    - Low average cost, low average latency, high throughput for large *n*

Deconstructing an Application

- Technology use
- Optimisation
- Interfacing
- Crash-tolerant distributed Computing Problems
  - With well-known solutions/impossibilities

Deconstructing Solutions

- They all build Common Knowledge CK on termination
  - Say, ϕ is a fact
  - CK(ϕ): (*knows*) × (*knows* ϕ)
  - everyone knows that everyone knows ϕ
- Examples
  - Multicasting: CK(ϕ), ϕ = contents of *m*
  - Clock Synchronisation: CK(ϕ), ϕ: Current Time = $T \pm \varepsilon$
  - Transaction Commit: CK(ϕ), ϕ = *decision* $\in$ {*abort*, *commit*}
  - Consensus: CK(ϕ), ϕ = *decision* $\in \{v_1, v_2, .., v_n\}$
  - Atomic Multicast: CK(ϕ), ϕ = *m* is 10^{th} in order
  - Group membership: CK(ϕ), ϕ = *decision*: p_i is crashed
- P offers probabilistic CK and R CK with full certainty

Computer Clusters

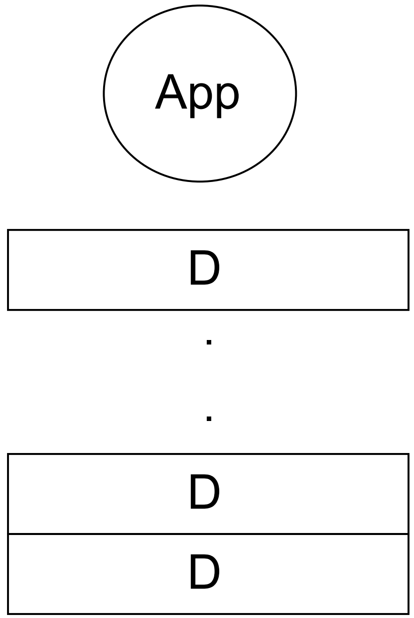
Almost all known solutions appropriate to clusters:

- Deterministic (D-type)
- Offer CK with full certainty
- Can only have T = ◊
- Complex
- Scalable??

- D-Type dissemination:
  - Form a tree for 10001 nodes rooted at *a*
  - Parent transmits *m* to its children
  - Crashes likely as *n* increases and warrant tree re-formation
- Examples: JGroups, Chubby, Paxos, Isis, Horus, …

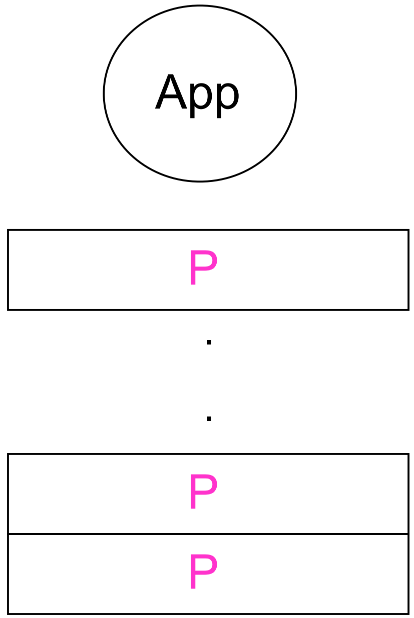
Current Design, D*

- Each layer is a solution to a problem
  - Multicast supports atomic multicast
- Claim: D* cannot scale
- Reason:
  - Multicast is the simplest of all
  - Complex solutions
    - In each step, a quorum of *n* act in synchrony
    - Several such steps before eventual termination (T is ◊)



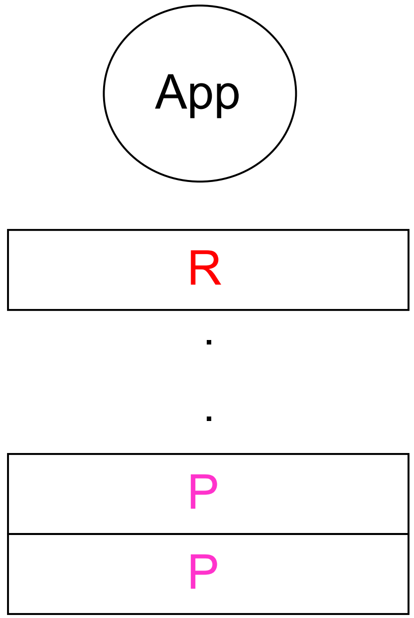
Design Option P*

- Each layer is a P-solution
- Feasible
  - For every D-solution, there is a P-solution
- CK on offer is probabilistic
- So, applications either
  - Live with low-probability events
  - Roll-back and recover
- P* highly scalable
  - No quorum, decentralised, ..

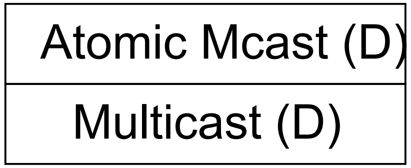


Design Option P*R

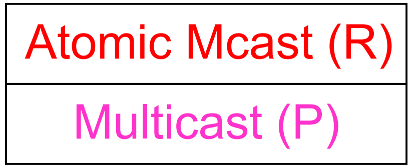
- Top layer is an R-solution
- Rest is P
- Feasible
  - Consensus is the hardest
  - Randomized solutions exist
    - Ezhilchelvan, Raynal (2002)
    - Ezhilchelvan, Alakeish (2011) for Manets
- Applications
  - Have certainty of outcome
  - ◊Termination
- P*R is also scalable at a moderate cost
  - No quorum, decentralised, repair not computationally expensive
  - Ensuring full repair is the main cost element



Current work



Chubby



Ours

- Comparative Evaluation is a 3-year project
  - Evidence to our belief that P* and P*R are better suited

Conclusions

- P*
  - A shop with security guards and no CCTV installation
  - Scalable
- P*R
  - A shop with security guards and CCTV cameras
  - CCTV images processed off-line
    - Processing takes time
    - Often reveals no prosecutable offence
- D*
  - CCTV images processed on-line
  - Coordinated with security guards on the floor
  - Every customer is under suspicion
  - Ideal for a small shop dealing with high-valued items like diamonds

Computer Clusters

- Message delay from *a* to *b* at any time, *d*
- A constant bound on *d* not possible
- Almost all known solutions:
  - Deterministic (D-type)
  - Offer CK with full certainty
  - Can only have T = ◊
  - Complex
  - Scalable??
- D-Type dissemination:
  - Form a tree for 10001 nodes rooted at *a*
  - Parent transmits *m* to its children
  - Crashes likely as *n* increases and warrant tree re-formation
- Examples: JGroups, Chubby, Paxos, Isis, Horus, …