

Department of Computer Science

**PURDUE**  
UNIVERSITY

# CS505: Distributed Systems

Lecture 15: Probabilistic Channels,  
Broadcast, and Multicast

# Outline

- ▶ **From Reliable to Probabilistic Broadcast**
- ▶ **Practical Probabilistic Broadcast Algorithms**
- ▶ **Multicast**

# Issues

- ▶ **Deterministic group communication**
  - Sometimes impossible (scale...)
  - Sometimes over-constrained
  
- ▶ **Randomization**
  - E.g., alternative to failure detectors
  - Leads to probabilistic properties/specifications
  
- ▶ **Probability proliferation**
  - Probabilities “sum up”
  - Correlation

# Reliable Broadcast

## I. No duplication

- No message is delivered more than once

## II. No creation

- No message is delivered unless it was broadcast

## III. Validity

- If  $p$  is correct, then a message broadcast by  $p$  is eventually delivered by  $p$

## IV. Agreement

- If a correct process delivers a message  $m$ , every correct process eventually delivers  $m$

# Simple Algorithm

upon  $\text{broadcast}_R(m)$  :

$\text{send}_R(m)$  to self

upon  $\text{receive}_R(m)$  :

    if (see  $m$  for first time)

$\text{send}_R(m)$  to every process

$\text{deliver}_R(m)$

# Assessment

## ► Complexity

- $O(n^2)$  messages

## ► Prerequisites

- Reliable channels (r-channels)
- Global views of size  $n$

# Probabilistic Broadcast: An Intuition

upon  $\text{broadcast}_P(m)$  :

$\text{send}_R(m)$  to self

upon  $\text{receive}_R(m)$  :

    if (see  $m$  for first time)

        send( $m$ ) to random set of  $F$  processes

$\text{deliver}_P(m)$

► Guarantees?

# Probabilistic Reliable Broadcast

## I. No duplication

- No message is delivered more than once

## II. No creation

- No message is delivered unless it was broadcast

## III. Validity

- If  $p$  is correct, then a message broadcast by  $p$  is eventually delivered by  $p$

## V. Probabilistic agreement

- If a correct process delivers a message  $m$ , then with known probability  $\beta$  all correct processes eventually deliver  $m$

# Assessment

- ▶  $\beta$  tends to  $e^{e^{-c}}$  with  $F(n) = (\ln n + c + 1)$  [Kermarrec et al.'03]
  - $O(n \ln n)$  messages
- ▶ With probability of process failure  $\psi$ 
  - $F'(n) = 1/(1-\psi) F((1-\psi)n)$
- ▶  $\beta = 1?$
- ▶ Channels?
- ▶ Views?

# Probabilistic Channels

## *p*-channel

### I. No duplication

- No message is *received* unless some process did *send* it

### II. No creation

- No message is *received* unless some process did *send* it

### III. Probabilistic validity

- If a *correct* process  $p_i$  sends a message  $m$  to a *correct* process  $p_j$ , then  $p_j$  *eventually receives*  $m$  with known probability  $\alpha$

# In Perspective

## ▶ Unreliable channels (u-channels)

- Weaker than probabilistic channels

## ▶ Reliable channels (r-channels)

### IV. Validity

- If a *correct* process  $p_i$  sends a message  $m$  to a *correct* process  $p_j$ , then  $p_j$  *eventually* receives  $m$

- Stronger than p-channels

## ▶ Fair-lossy channels (f-channels)

### V. Validity

- ▶ If a *correct* process  $p_i$  sends a message  $m$  to a *correct* process  $p_j$  an infinite number of times, then  $m$  is received by  $p_j$  an infinite number of times

- Stronger than p-channels

- Maximum losses  $X_{max}$

# Stubborn Probabilistic Channels

## ▶ *s-channel*

- Keep resending

### VI. 1-Validity

- If a *correct* process  $p_i$  sends a message  $m$  to a *correct* process  $p_j$   $x$  times, then  $p_j$  receives  $m$  with probability  $P(x) = 1 - (1 - \alpha)^x$

### VI. Still weaker than fair-lossy channel

- $P(x \leq x_{max}) = \dots$
- $P(x > x_{max}) = 1$
- $x_{max}$  *unknown, but existing*
- With  $x \rightarrow \infty$  (and independent runs) a *s-channel* converges towards an *r-channel*, but never becomes one
  - “Probability 1”

# Reliable Broadcast with p-/s-channels?

## I. No duplication

- No message is delivered more than once

## II. No creation

- No message is delivered unless it was broadcast

## III. Validity

- If  $p$  is correct, then a message broadcast by  $p$  is eventually delivered by  $p$

## VI. 1-Agreement

- If a correct process delivers a message  $m$ , all correct processes eventually deliver  $m$  with probability 1

# Further

## ▶ Probabilistic Reliable Broadcast with p-channels?

- $F'(n) = F(n)/\alpha = (\ln n + c + 1)/\alpha$

## ▶ Reliable Broadcast = Stubborn Probabilistic Reliable Broadcast?

- Keep broadcasting until delivered?

## ▶ Consensus?

# Practical (“Gossip-based”) Algorithms

## ▶ TTL

- *Forwards  $g$* : a same process sends same information  $g$  times
- *Hops  $h$* : same information is forwarded at most  $h$  times (causal chain of length  $h$ )

## ▶ Information

- Messages: payload
- Digests: identifiers

## ▶ Interaction

- Push: process sends information to arbitrary processes/digest sender
- Pull: process asks for information
- Mix, cf.[Karp et al.'00]

# Example: *pbcast*

## ▶ Probabilistic Broadcast (*pbcast*) [Birman et al.'99]

- Seminal work

## ▶ Two phases

1. Best-effort broadcast based on spanning tree

2. Gossip-based propagation of histories and according updates

- Periodically every process chooses a random set of  $F$  processes and sends digest
- Missing messages are sent

## ▶ Based on complete membership $O(n)$

- $O(n \ln n)$  messages for a broadcast
- p-channels assumed

# Example: *lpbcast*

- ▶ **Lightweight Probabilistic Broadcast (*lpbcast*) [Eugster et al.'03]**
- ▶ **Based on partial membership**
  - Assumes *uniform* views of size  $l$  for analysis
- ▶ **Messages convey**
  1. Application messages
  2. Digests
  3. Membership subsets
- ▶ **Requires  $O(n \ln n)$  messages for a broadcast**

# Analysis (Markov)

- ▶ Probability that a given gossip message “infects” a given (“uninfected”) process:

$$p = (l/n)(F/l)(1 - \alpha)(1 - \psi)$$

$$= (F/n)(1 - \alpha)(1 - \psi)$$

$$q = 1 - p$$

- ▶ Probability of stepping from  $i$  infected processes to  $j$  infected processes at the next round:

$$p_{ij} = B(n-i, j-i)(1-q)^i j^{j-i} (q^i)^{n-j} \quad \text{with } B(x,y) = \binom{x}{y} = \frac{x!}{y! (x-y)!}$$

- ▶  $P(j \text{ infected at round } r) = \sum_{i \leq j} P(i \text{ infected at round } r-1) p_{ij}$

- ▶ Independent of  $l$

- Provided that views are *uniformly* distributed

# Membership Stability

## ► Probability of partitioning

- Partition of size  $i > l$

$$B(n,i) \left( B(i,l) / B(n,l) \right)^i \left( B(n-i,l) / B(n,l) \right)^{n-i}$$

- Upper bound

- Several partitions can be seen as recursive partitions

- Decreases with increasing  $l$

- But also  $n$

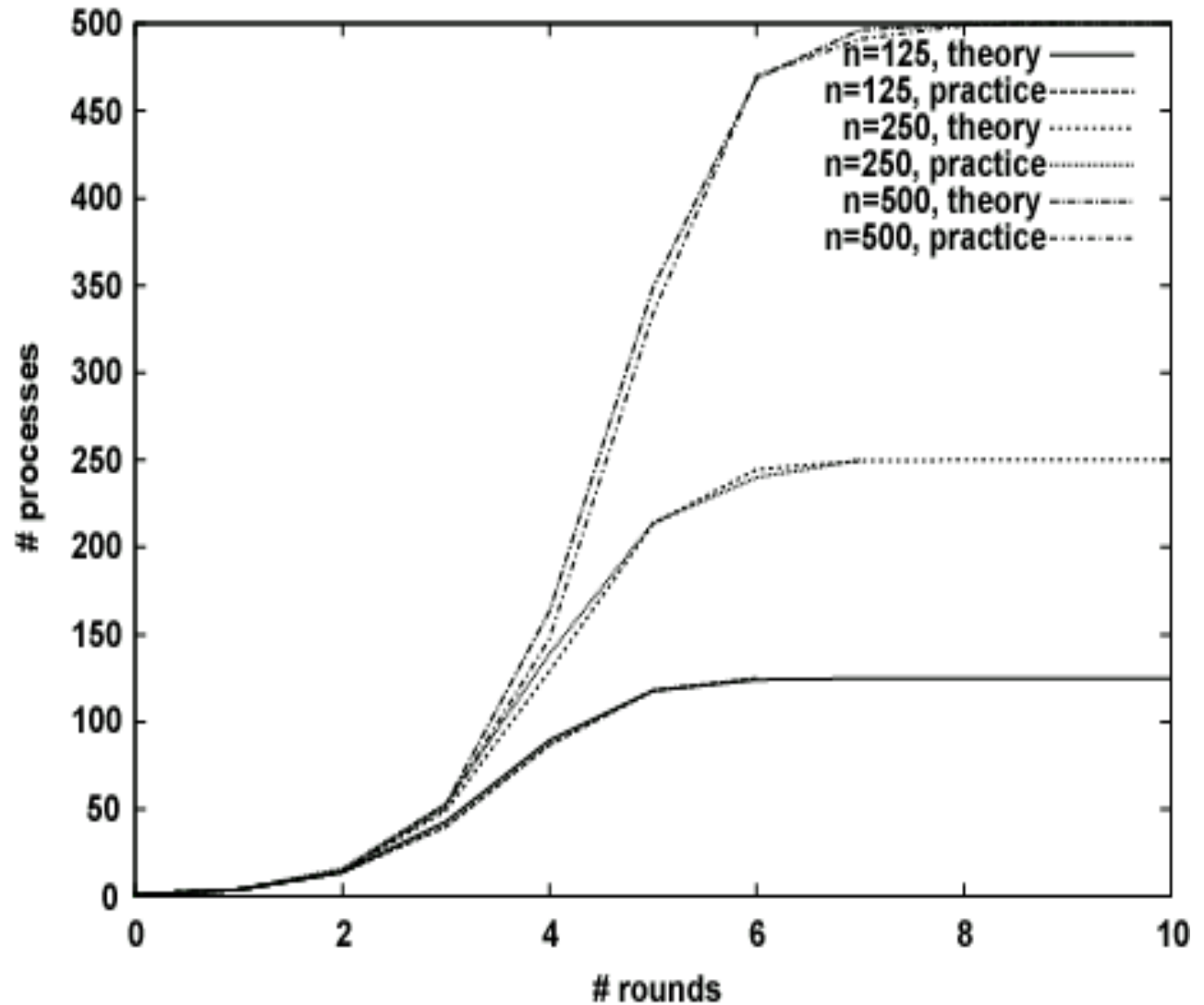
- Becomes more stable with increasing system size
- Total amount of membership information in the system increases

# Uniform Views

- ▶ **Every process has the same probability of appearing in any other process' view**
  - On average  $l$  appearances; probability of  $p_i$  being in  $p_j$ 's view:  $l/n$
- ▶ **Performance *does* depend on  $l$  in practice**
  - **Dependency**
    - Causal dependencies between messages and destinations
  - **Reliability**
    - Latency increases, and buffers are limited
- ▶ **Random walk, e.g., [Jelasity et al.'07]**
  - $O(\log n)$  messages,  $O(\log n)$  latency
  - Still approximation (with very high probability)

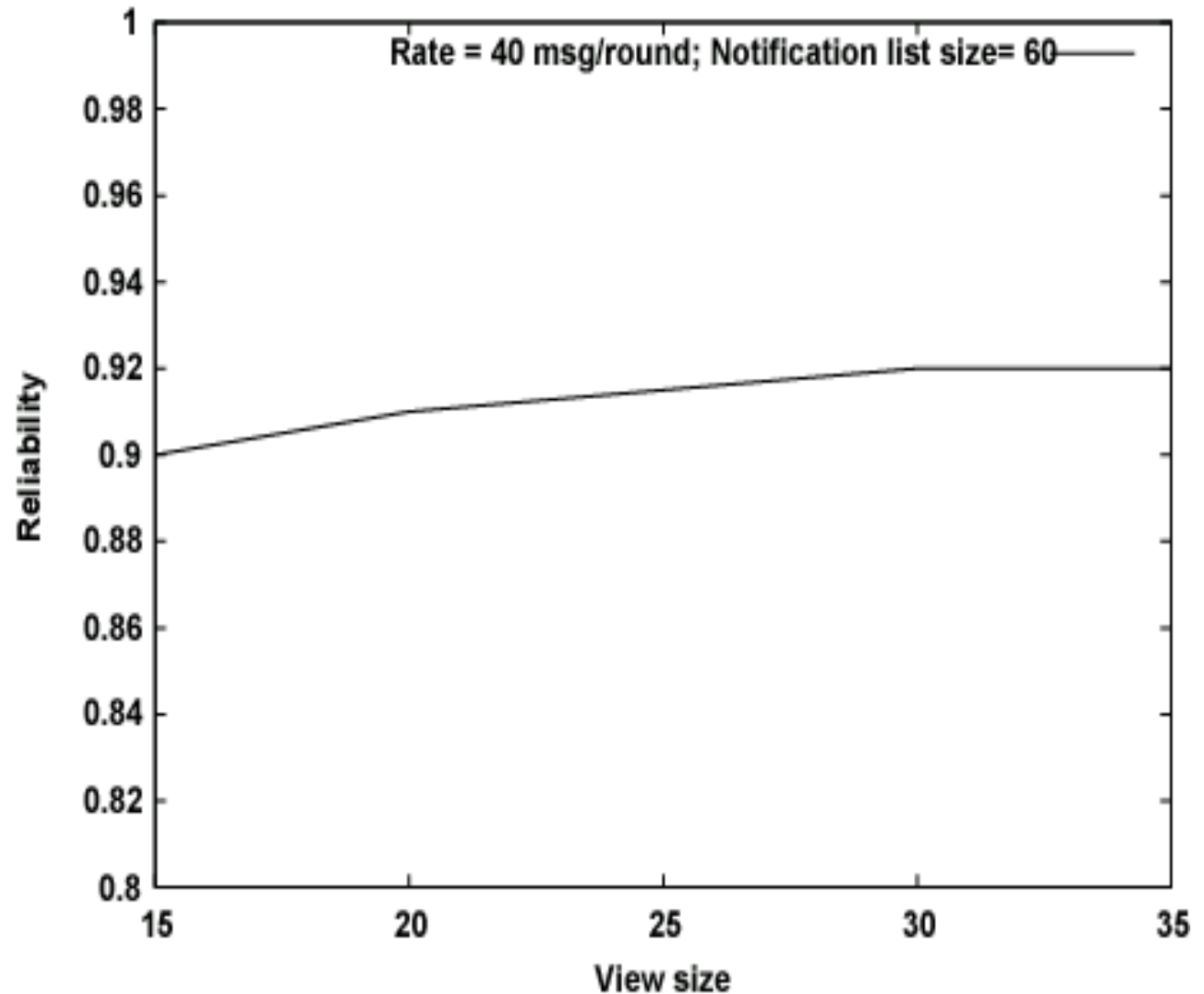
# Analysis vs Simulation

- ▶ Fanout 3
- ▶ 1 msg injected
- ▶ Varying system size



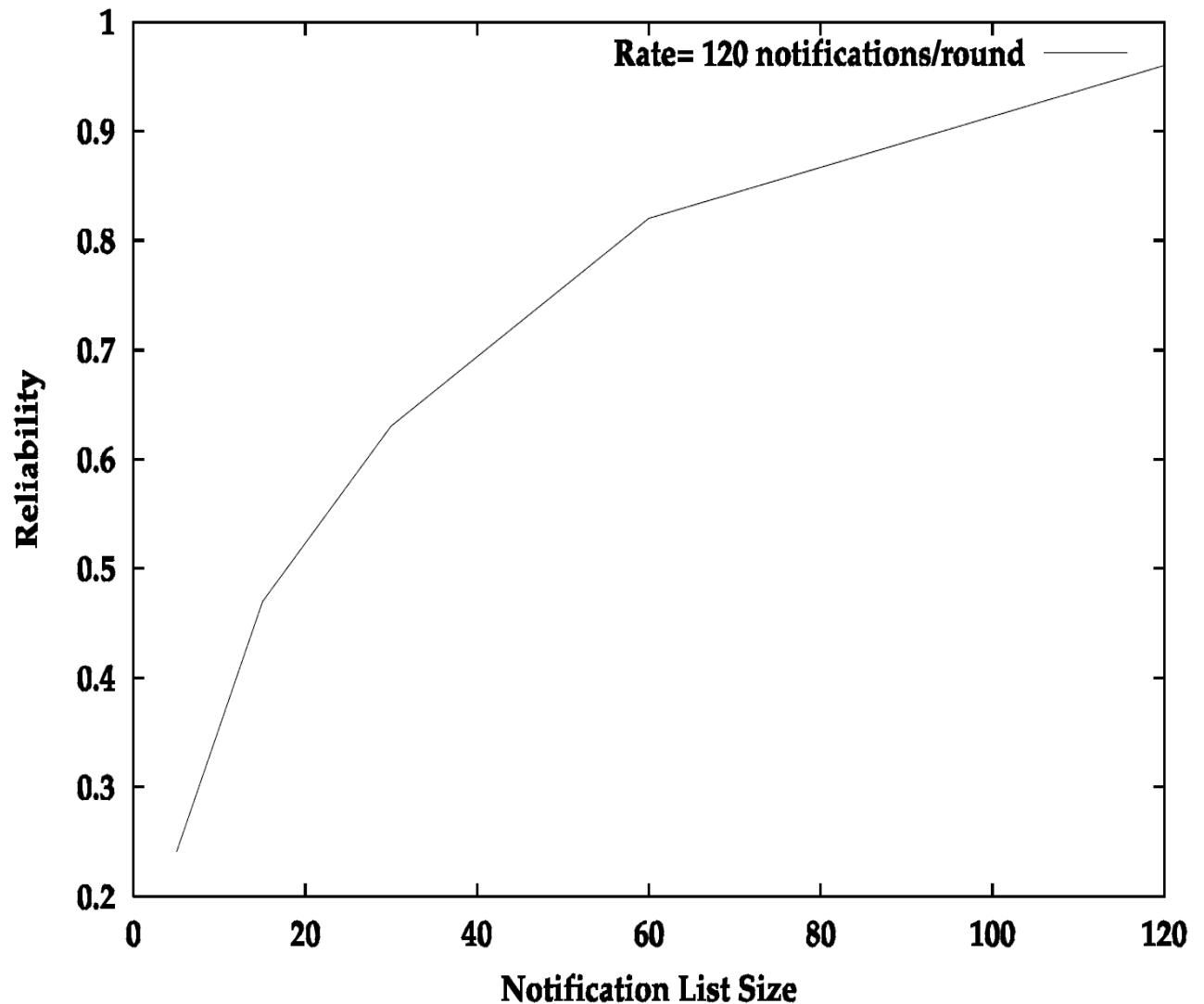
# View Size and Reliability

- ▶ System size of 125
- ▶ Fanout 3
- ▶ 40 msgs/round are injected
- ▶ 60 msgs are in buffer
- ▶ Varying view size



# Buffer Size and Reliability

- ▶ System size of 125
- ▶ Fanout 3
- ▶ 120 msgs/round are injected
- ▶ Varying buffer size



# Guarantees

## I. No duplication

- No message is delivered more than once

## II. No creation

- No message is delivered unless it was broadcast

## III. Validity

- If  $p$  is correct, then a message broadcast by  $p$  is eventually delivered by  $p$

## VII. Probabilistic agreement'

- If a correct process delivers a message  $m$ , then any given correct process eventually delivers  $m$  with known probability  $\beta'$

# Comparison

## ▶ Atomicity

- A process has no means of distinguishing (unilaterally) “good” or “bad” run
  - What is “good” or “bad” run? Who are *all* the processes?
- Would require Reliable Broadcast primitive, global views

## ▶ Thus information is limited

- $\beta$  is lower bound for  $\beta'$ 
  - Reflects case where with probability  $\beta$  *all* processes `deliver`, and with probability  $(1-\beta)$  *no single* process `delivers`

## ▶ Application-specific

# (Explicit) Reliable Multicast

## I. No duplication

- No message is delivered more than once

## II. No creation

- No message is delivered unless it was multicast

## III. Validity

- If  $p_i$  is correct, then a message multicast by  $p_i$  is eventually delivered by some correct  $p_j \in \text{Dest}(m)$  if exists

## IV. Agreement

- If a correct process delivers a message  $m$ , every correct process in  $\text{Dest}(m)$  eventually delivers  $m$

# Simple Algorithm

```
upon multicastR(m, Dest) :
```

```
    forward(m, Dest)
```

```
upon receiveR(m, Dest) :
```

```
    if (see m for first time)
```

```
        forward(m, Dest)
```

```
        deliverR(m)
```

```
forward(m, Dest) :
```

```
    sendR(m) to every process in Dest
```

# Implicit Multicast

## ► Remember

- Main motivation for probabilistic approach is (system) scalability
- Main motivation for multicast is (data) scalability
- How about Dest?

## ► Cf. *publish/subscribe*

- Subscribe  $\equiv$  join multicast group
- Publish  $\equiv$  broadcast
- *Content-based* publish/subscribe
  - Interest predicate for process  $p_i$   $P_i: M \rightarrow bool$

# (Implicit) Reliable Multicast

## I. No duplication

- No message is delivered more than once

## II. No creation

- No message is delivered unless it was multicast

## III. Validity

- If  $p_i$  is correct, then a message multicast by  $p_i$  is eventually delivered by some correct  $p_j \mid P_j(m) = true$  if exists

## IV. Agreement

- If a correct process delivers a message  $m$ , then any correct process  $p_i \mid P_i(m) = true$  eventually delivers  $m$

# Sender-filtering Implicit Multicast

upon  $\text{multicast}_R(m)$  :

    forward(m)

upon  $\text{receive}_R(m)$  :

    if (see m for first time)

        forward(m)

$\text{deliver}_R(m)$

forward(m) :

$\text{send}_R(m)$  to every process  $p_i$  s.t.  $P_i(m)$

# Assessment

- ▶ **Message complexity**
- ▶ **Membership complexity**
- ▶ **Fault tolerance**
- ▶ **Parasitism**

# Assessment

## ► Complexity

- Messages:  $O(l^2)$  messages, where  $l = |\{p_i \mid P_i(m)\}|$
- Memory:  $O(n \max_i (|P_i|))$
- No parasite messages

## ► Variant: receiver-filtering

- Messages:  $O(n l)$  messages, where  $l = |\{p_i \mid P_i(m)\}|$
- Memory:  $O(n)$
- $O((n-l) l)$  parasite messages

## ► Observations

- $\{p_i \mid P_i(m)\}$  can be small
- $|P_i|$  become large

# Probabilistic Multicast: Simple Algorithm?

upon  $\text{multicast}_P(m)$  :

$\text{forward}(m)$

upon  $\text{receive}_P(m)$  :

    if (see  $m$  for first time)

$\text{forward}(m)$

$\text{deliver}_P(m)$

$\text{forward}(m)$  :

$\text{send}_P(m)$  to  $F$  processes  $p_{i \in [1..F]}$  s.t.  $P_i(m)$

# (Implicit) Probabilistic Multicast

## I. No duplication

- No message is delivered more than once

## II. No creation

- No message is delivered unless it was multicast

## III. Validity

- If  $p_i$  is correct, then a message multicast by  $p_i$  is eventually delivered by some correct  $p_j \mid P_j(m) = true$  if exists

## V. Probabilistic Agreement

- If a correct process delivers a message  $m$ , then any correct process  $p_i \mid P_i(m) = true$  eventually delivers  $m$  with known probability  $\omega$

# Gossip-based

## ► Complexity

- Messages:  $O(l \ln l)$  messages, where  $l = |\{p_i \mid P_i(m)\}|$
- Memory:  $O(n \max_i (|P_i|))$
- No parasite messages

## ► Variant: receiver-filtering

- Messages:  $O(l \ln n)$  messages, where  $l = |\{p_i \mid P_i(m)\}|$
- Memory:  $O(n)$
- $O(l \ln (n/l))$  parasite messages

# Intuition: Hybrid Approach

- ▶ **Need to balance between determinism (structure) and randomization (chaos)**
- ▶ **Observations**
  1. In large-scale broadcasts only few broadcasters
  2. Internet is not fully symmetric. Built of interconnected “clouds”
- ▶ **Hierarchical approach**

# Example: *pmcast* [Eugster et al.'02]

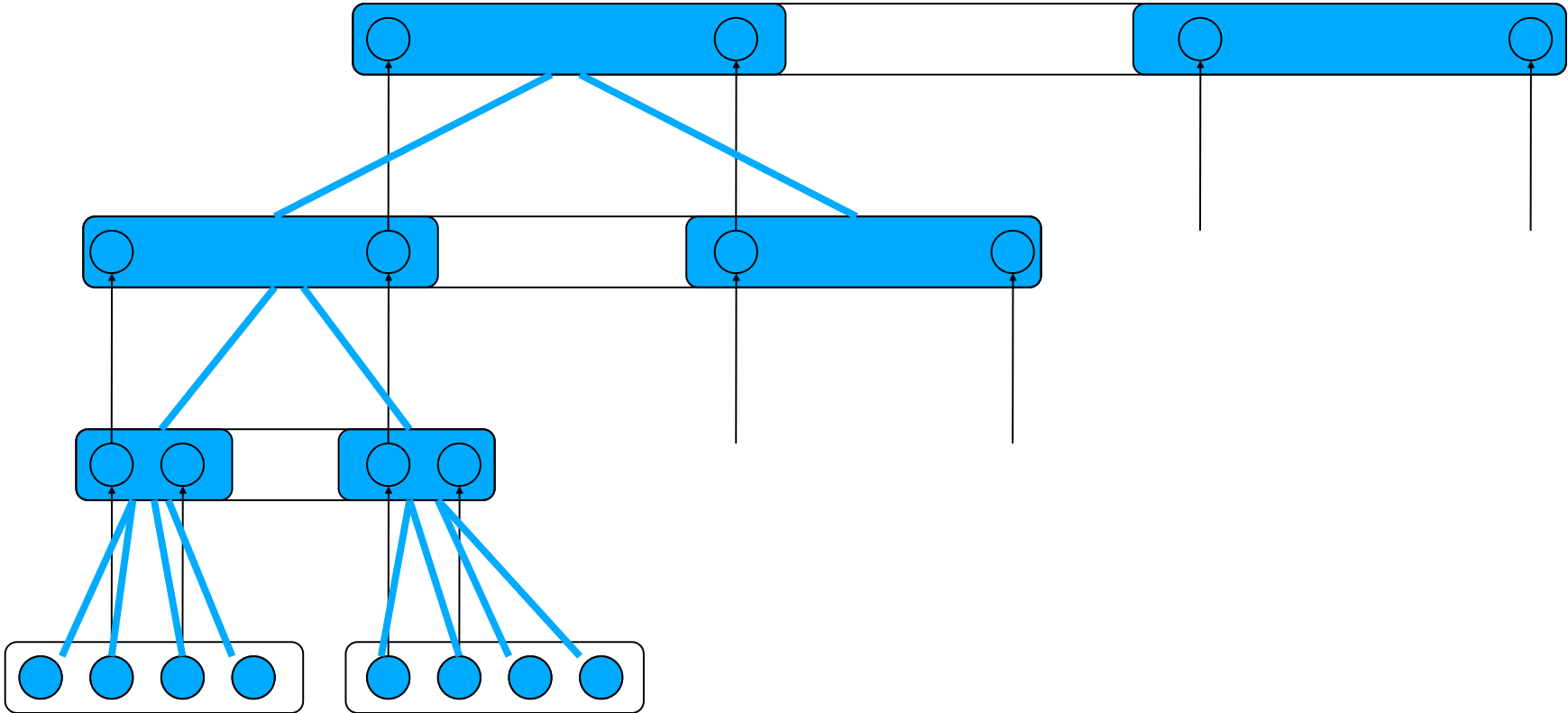
## ▶ Bottom-up

- Processes have precise knowledge about „close” neighbors
- Knowledge decreases with the „distance”
- Processes are orchestrated in subgroups, according to distance
- $r$  delegates are chosen for each subgroup, manifest interests of all represented processes
- Each process knows the  $r$  delegates of each (known) subgroup

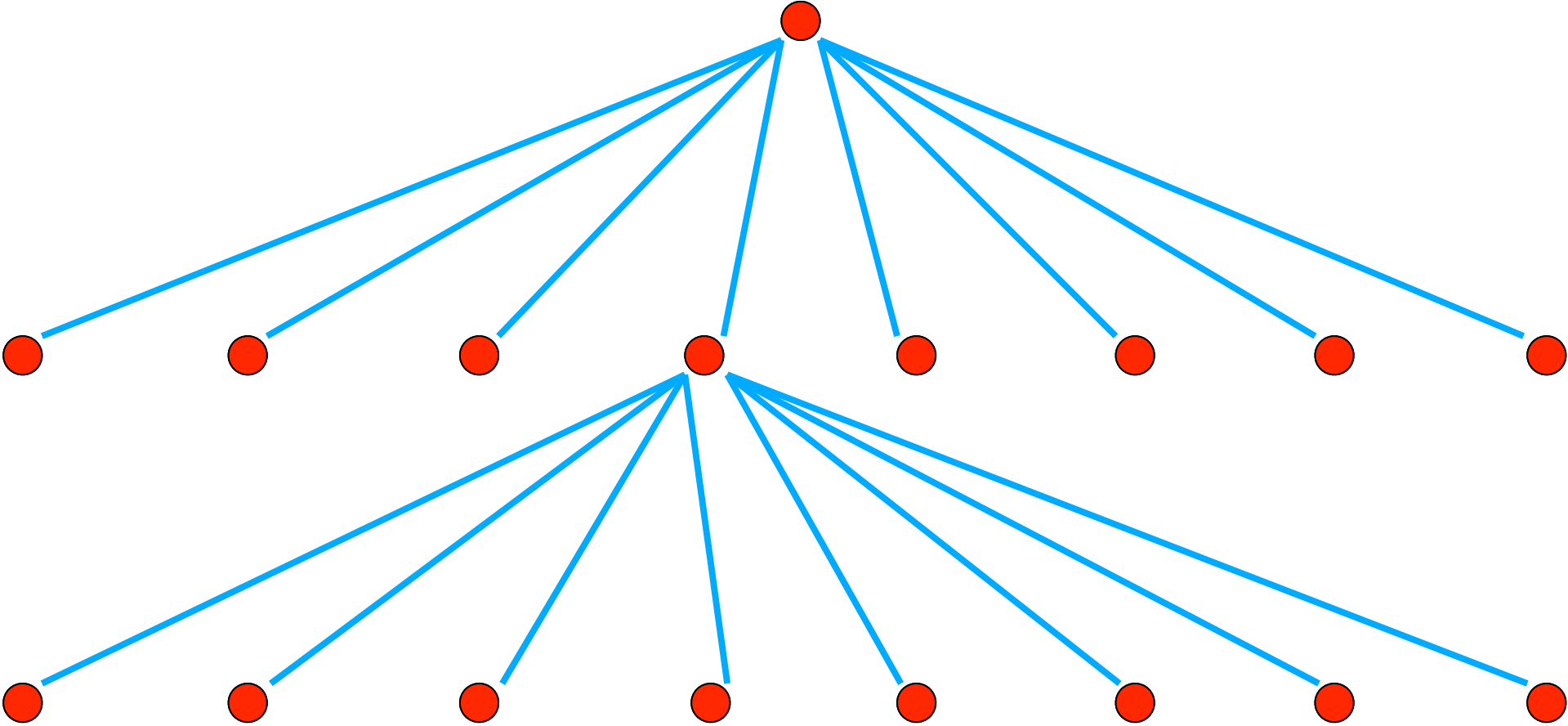
## ▶ Recursively

- Put several subgroups of level 1 together: form a group of level 2
- Elect  $r$  delegates for every such supergroup
- etc.

# Illustration



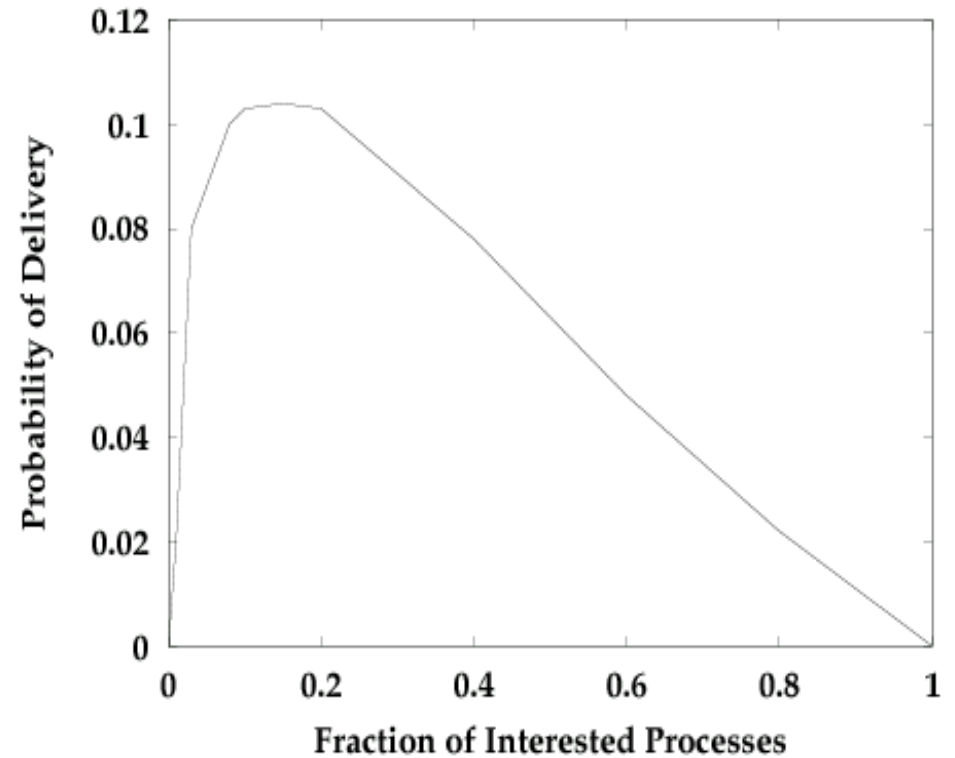
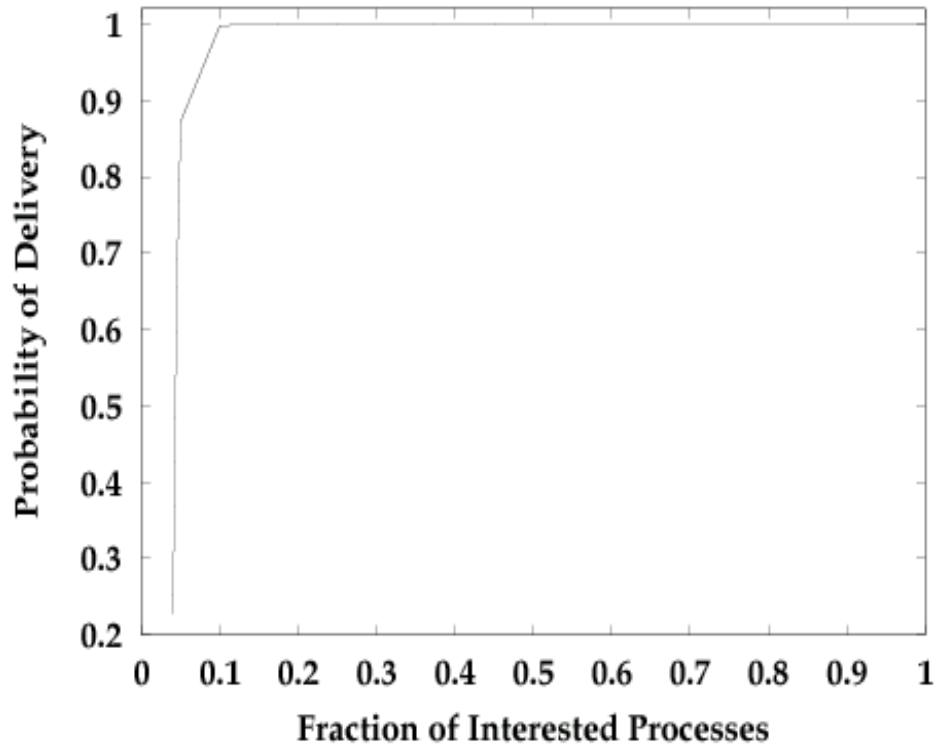
# Illustration



# Analysis (Diff')

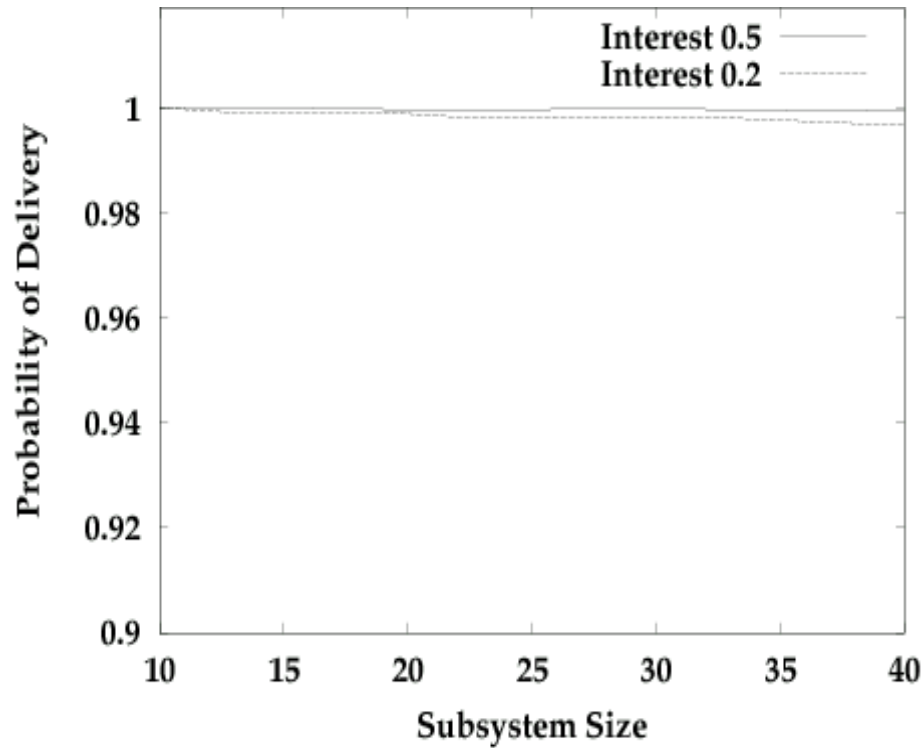
- ▶ **Number of processes that a given process knows**
  - $O(\ln n)$
- ▶  **$I(t)$ : infected entities at time  $t$  [Diekmann&Heesterbeek'00]**
  - $dl/dt = F/n I(n - I)$ , and hence  $I = n / (1 + ne^{-Ft})$
  - Limited number of hops
- ▶ **„Worst“ case: broadcast**
  - Same „time“ to complete as when every process knows every other process:  $O(\ln n)$
  - Same message complexity:  $O(n \ln n)$
- ▶ **Multicast**
  - Probability  $\omega$  of `pmdelivering` a message to an *interested* process
  - Comes very close to 1
  - Small reception probability for non-interested processes

# Delivery and Unvoluntary Delivery

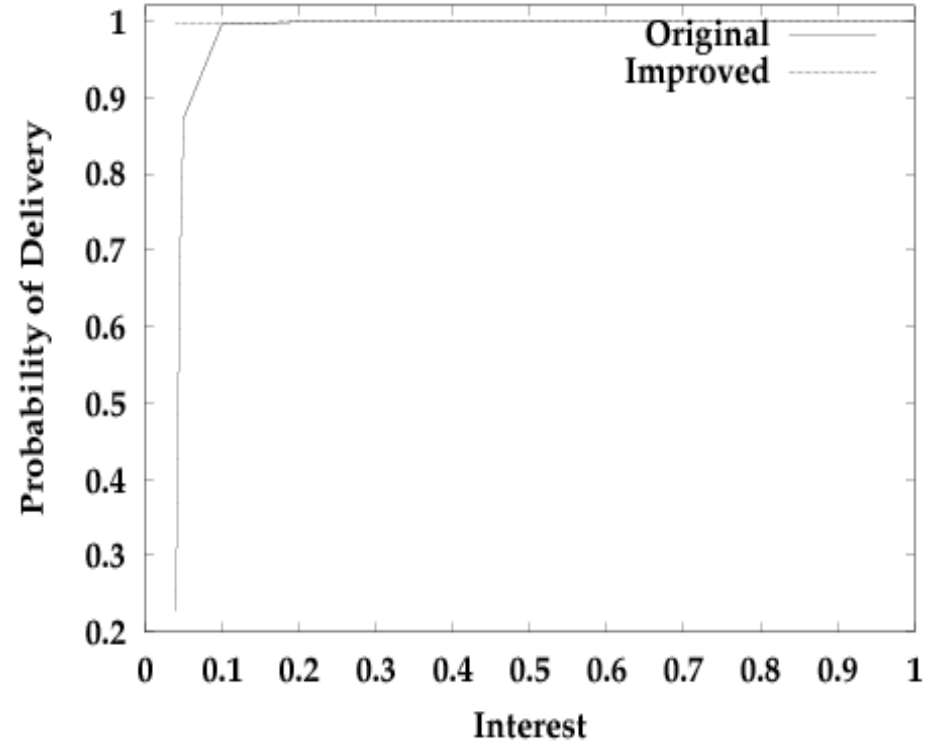


10000 processes, 3 levels, redundancy and fanout of 3

# Scalability and Tuning



3 levels, redundancy 4, fanout 3



≈10000 processes, 3 levels,  
redundancy and fanout of 3

# Conclusions

## ▶ „Natural“ tradeoffs

- Increasing the number of levels
  - Reduces the membership knowledge each process has
  - Increases the average filtering load, etc.
- Approximations for the number of rounds at each level
  - Small vs large fractions of interested processes

## ▶ Root processes

- Require more computing power
  - Use approximate matching to reduce filtering load
- Require more memory
  - Hash functions to compact space for „interests“

# References

- ▶ ***Bimodal Multicast.*** K. Birman, M. Hayden, O.Ozkasap, Z. Xiao, M. Budiu, Y. Minsky. ACM TOCS, 17(2): 41-88, 1999.
- ▶ ***Randomized Rumor Spreading.*** R. Karp and C. Schindelhauer and S. Shenker and B. Vöcking. FOCS 565-574, 2000.
- ▶ ***O. Diekmann and J.A.P. Heesterbeek.*** Mathematical Epidemiology of Infectious Diseases: Model Building, Analysis and Interpretation. Wiley, 2000.
- ▶ ***Lightweight Probabilistic Broadcast.*** P. Eugster, R. Guerraoui, S. Handurukande, A.-M. Kermarrec, P. Kouznetsov, ACM TOCS, 21(4): 341-374 2003.
- ▶ ***Probabilistic Multicast.*** P. Eugster, R. Guerraoui, 313-323, DSN'02.
- ▶ ***Probabilistic Reliable Dissemination in Large-Scale Systems.*** A.-M. Kermarrec, L. Massoulie, A.J. Ganesh, IEEE TPDS 14(3): 248-58, 2003.
- ▶ ***Gossip-based Peer Sampling.*** M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, M. van Steen. ACM TOCS 25(3) 2007.