Cloudy Weather for P2P

with a Chance of Gossip

Alberto Montresor – Luca Abeni

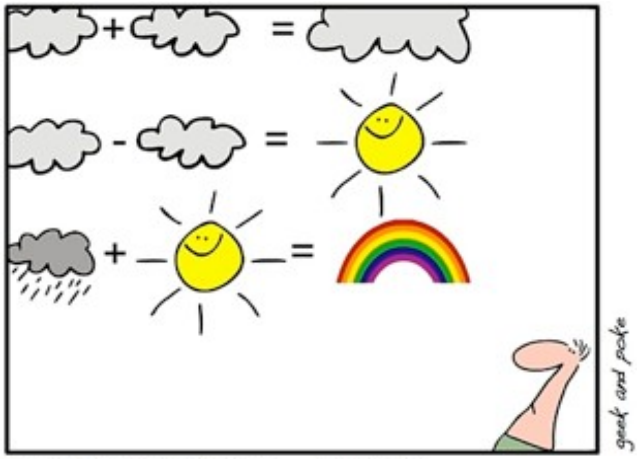
Best paper award in P2P'11

Presented by: Amir H. Payberah
amir@sics.se

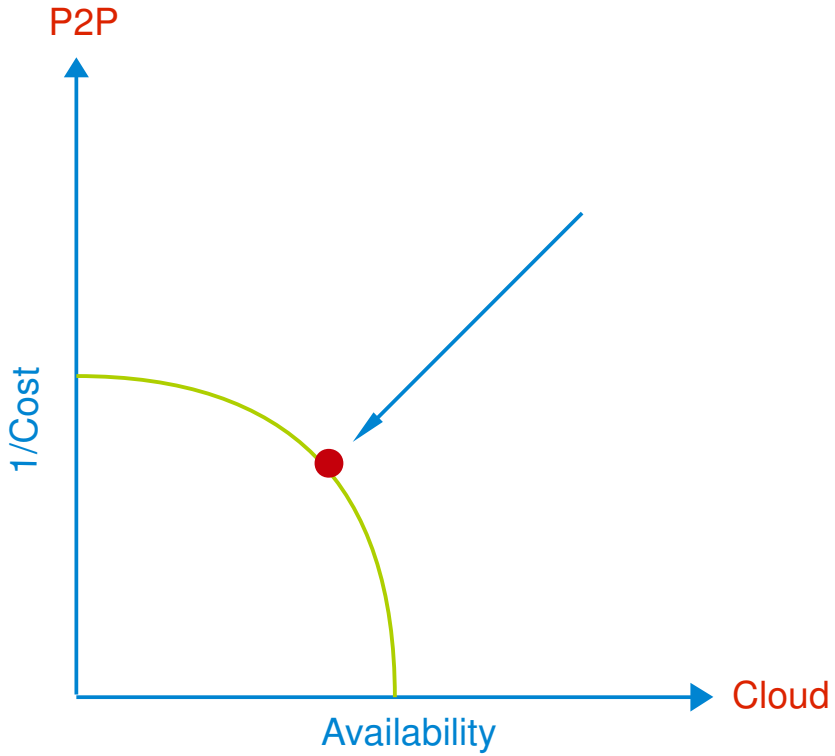
Introduction

Cloud Computing vs. P2P

- Similarity:
  - Providing the infinite availability of computing and storage resources.
- Difference:
  - The cloud provides highly-available resources, but at a cost.
  - P2P resources are for free, but their availability is shaky.



Cloud Computing vs. P2P



Related Work

- Several projects have tried to mix the two to get the advantages of both: high-availability and low cost.

- First approach: adding peers to an existing centralized solution.
  - Off-loading the services to peers, if does not affect service availability.

- Second approach: augmenting P2P systems with cloud angels.
  - Adding elastic computing nodes with the specific task of satisfying requirements beyond the reach of a P2P network.

Third Approach: The Paper Contribution - Cloudcast

- The target application: content distribution.
- All functionalities of the content distribution application are supported by a passive storage service such:
  - Topology bootstrap and membership management.
  - Information dissemination.
- No active elastic computing instances are required.

The Costs

- Storage and bandwidth costs.
- To preserve the durability, the storage consumption cannot be reduced.
- So, let's focus on the bandwidth cost. How?

How?

- Cloudcast is based on two gossip protocols:
  - Topology bootstrap and membership management.
  - Information dissemination.

- How: keep the number of cloud accesses 1 per gossip cycle, independently of the number of clients.

Background

Information Dissemination

- Epidemic protocols – a classical solution

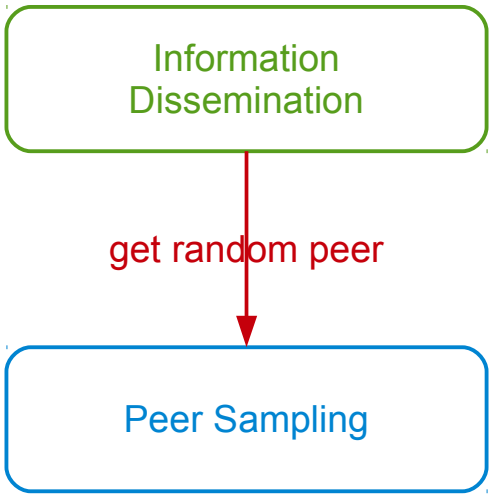
- Nodes gossip about news in random fashion:
  - Rumor mongering: to quickly push the rumors toward on-line nodes.
  - Anti entropy: to make sure that everybody gets all the news (push-pull).

- Epidemic protocols require a list of nodes forming the network. Where do we get it from? Peer sampling.

Peer Sampling

- Peer sampling service provides each peer with continuously up-to-date random samples of the entire population of peers.

- Such samples fulfil two purposes:
  - Used by the epidemic broadcast service to obtain random peers.
  - Maintain a random topology connecting all peers.



A Generic Gossip Protocol

- A generic gossip protocol – executed by node p.

Active thread

```
do once every δ time units
   q = getPeer(state)
   Sp = prepareMsg(state, q)
   send(REQ, Sp) to q
```

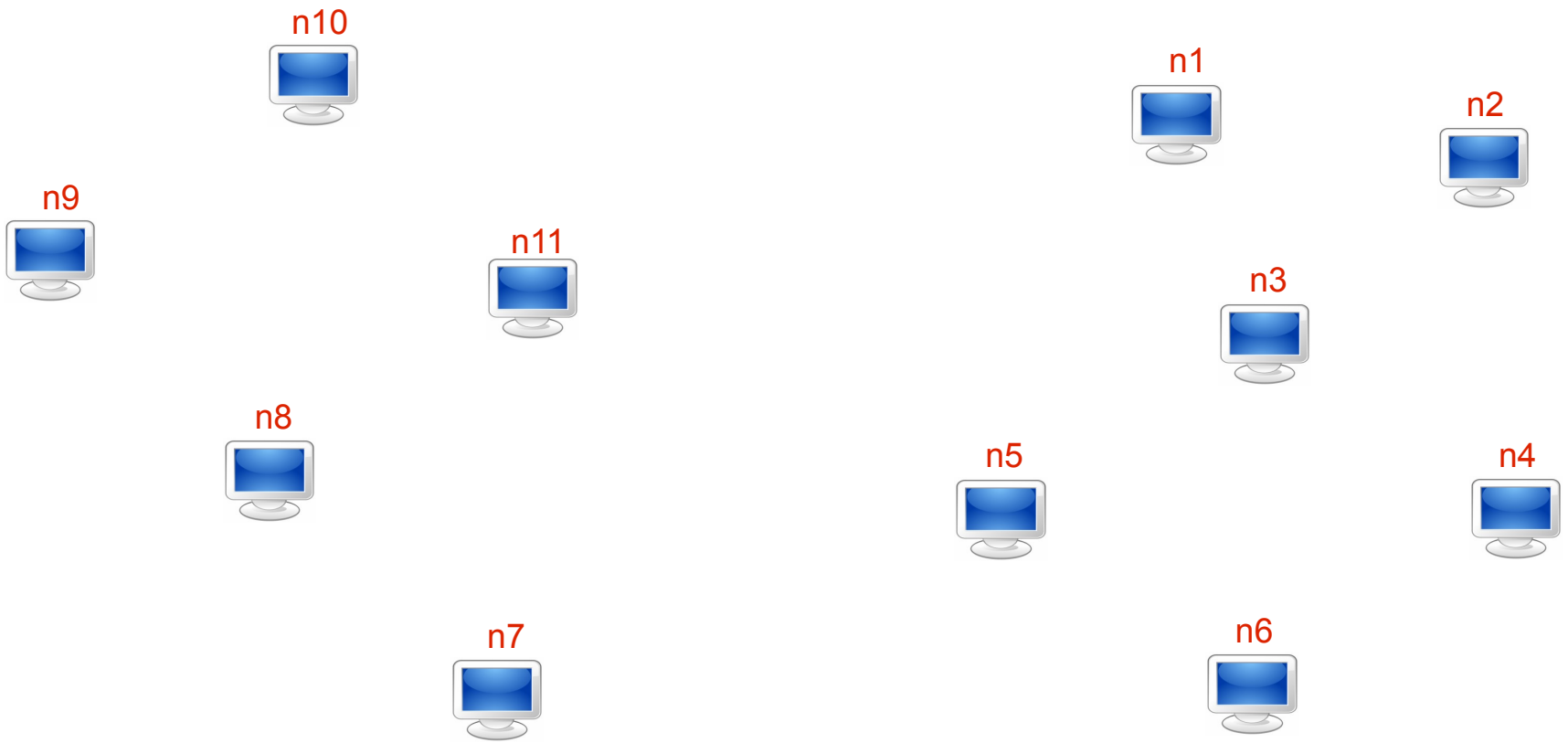
Passive thread

```
do forever
   receive(t, Sq) from *
   if (t = REQ) then
      Sp = prepareMsg(state, q)
      send(REP, Sp) to q
  state = update(state, Sq)
```

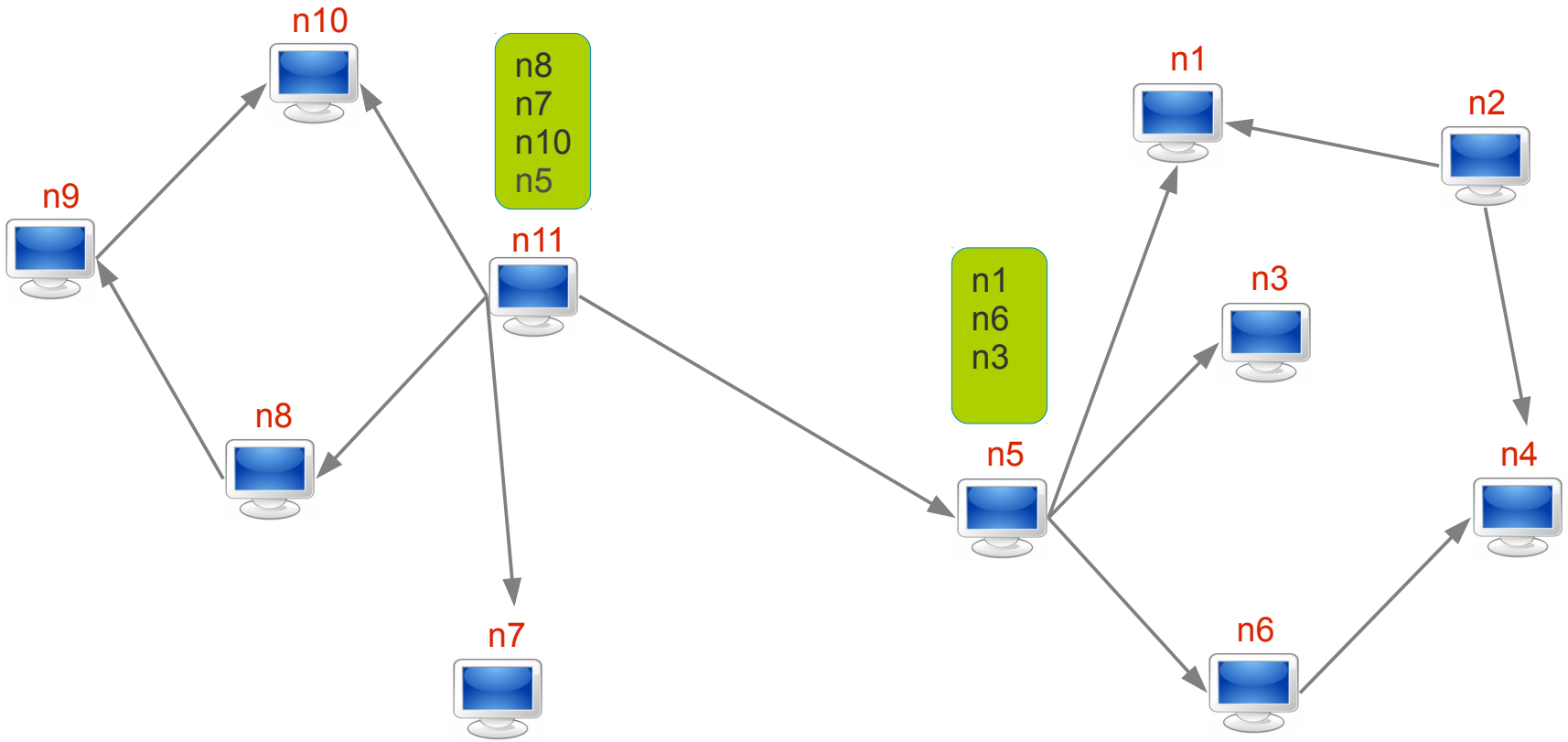
Cyclon

- State at each node: A partial view containing c neighbour descriptors (c = view size).
- Node descriptor: address + timestamp (age)
- getPeer()
  - select the oldest descriptor in the view
  - remove it from the view
- prepareMsg(view, q)
  - active thread: returns a subset of t-1 random nodes, plus a fresh local identifier of itself
  - passive thread: returns a subset of t random nodes
- update(view, Sq)
  - discard entries in Sq that p already knows
  - add Sq , removing entries sent to q

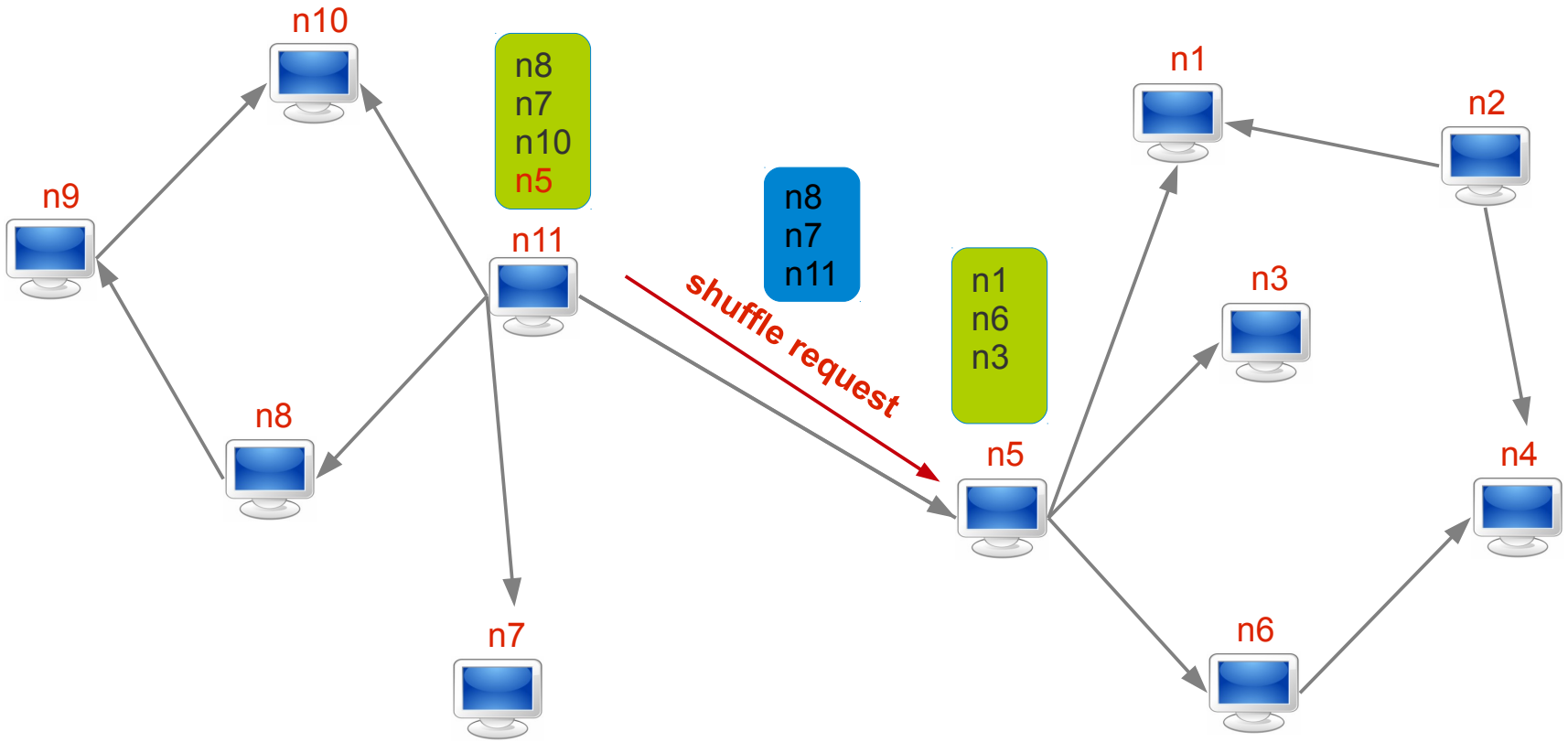
Cyclon Peer Sampling Protocol (1/7)



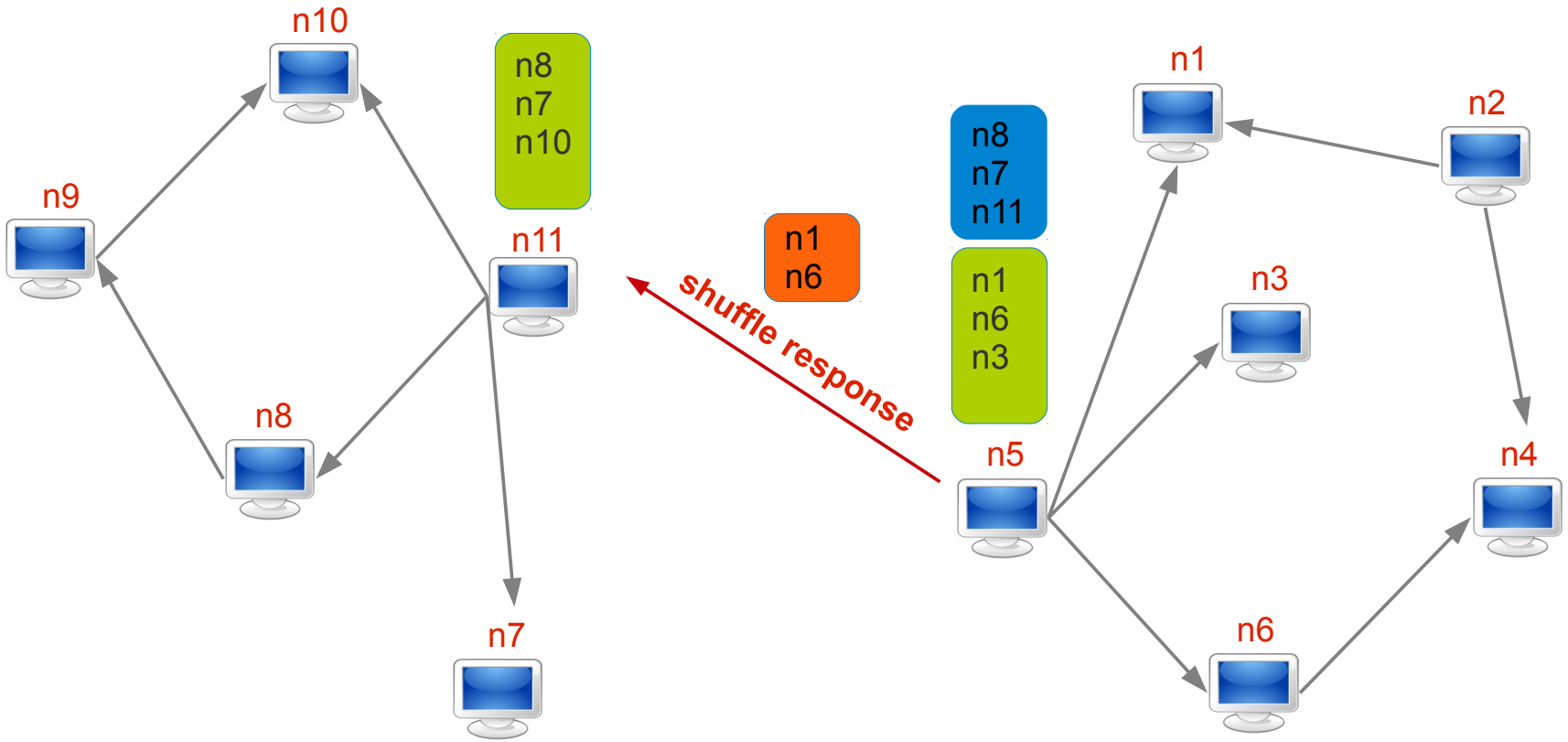
Cyclon Peer Sampling Protocol (2/7)



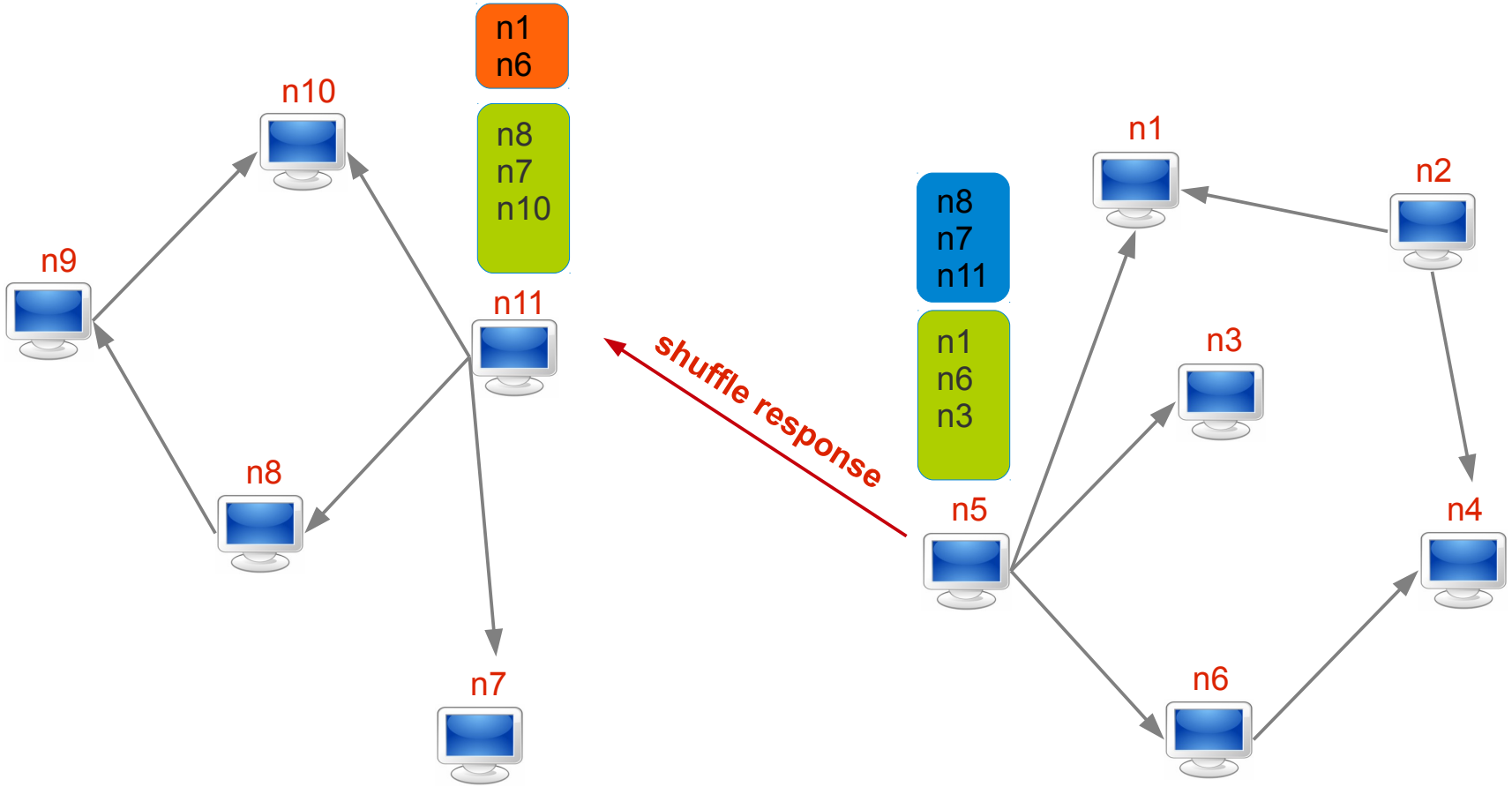
Cyclon Peer Sampling Protocol (3/7)



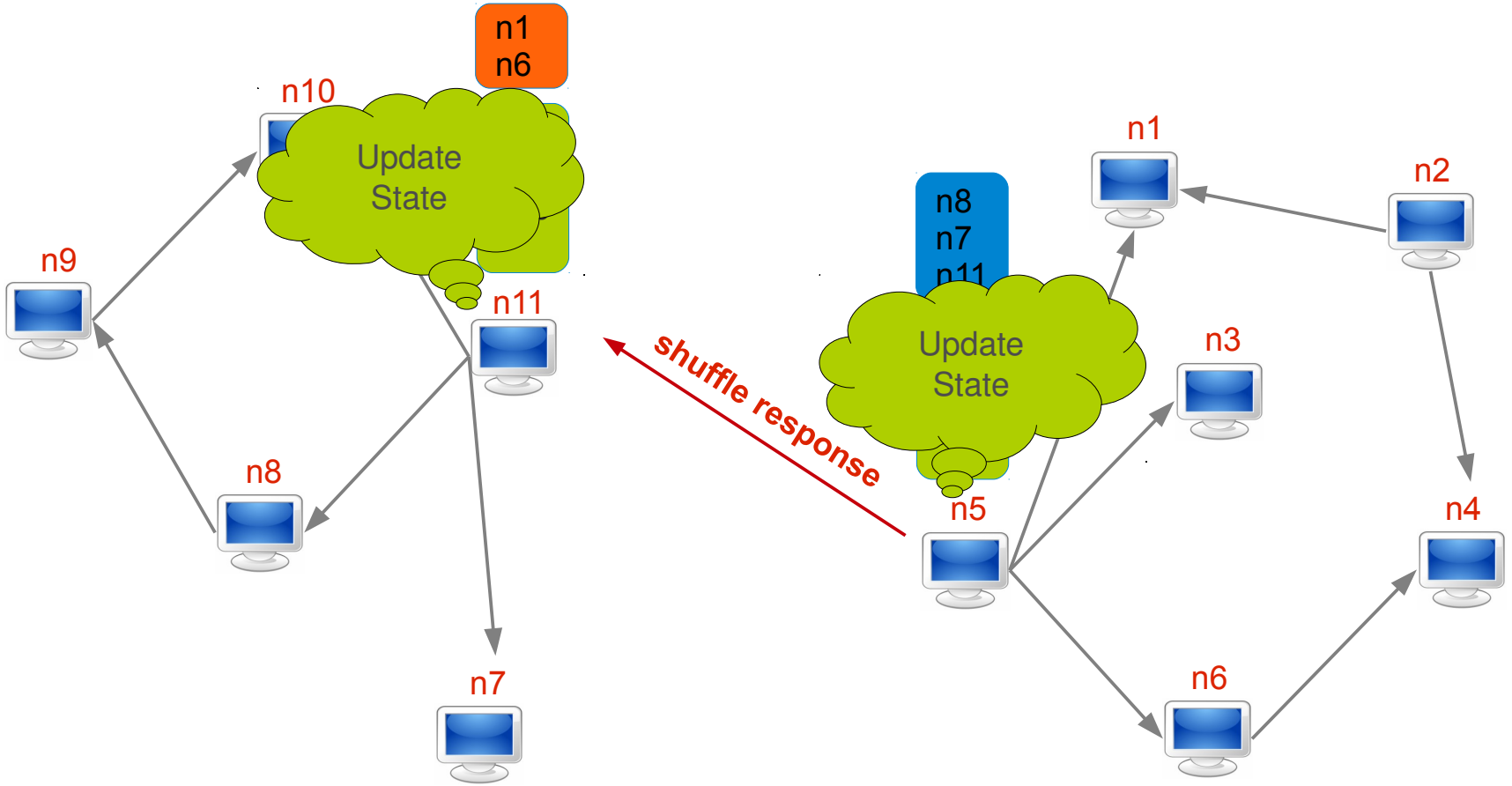
Cyclon Peer Sampling Protocol (4/7)



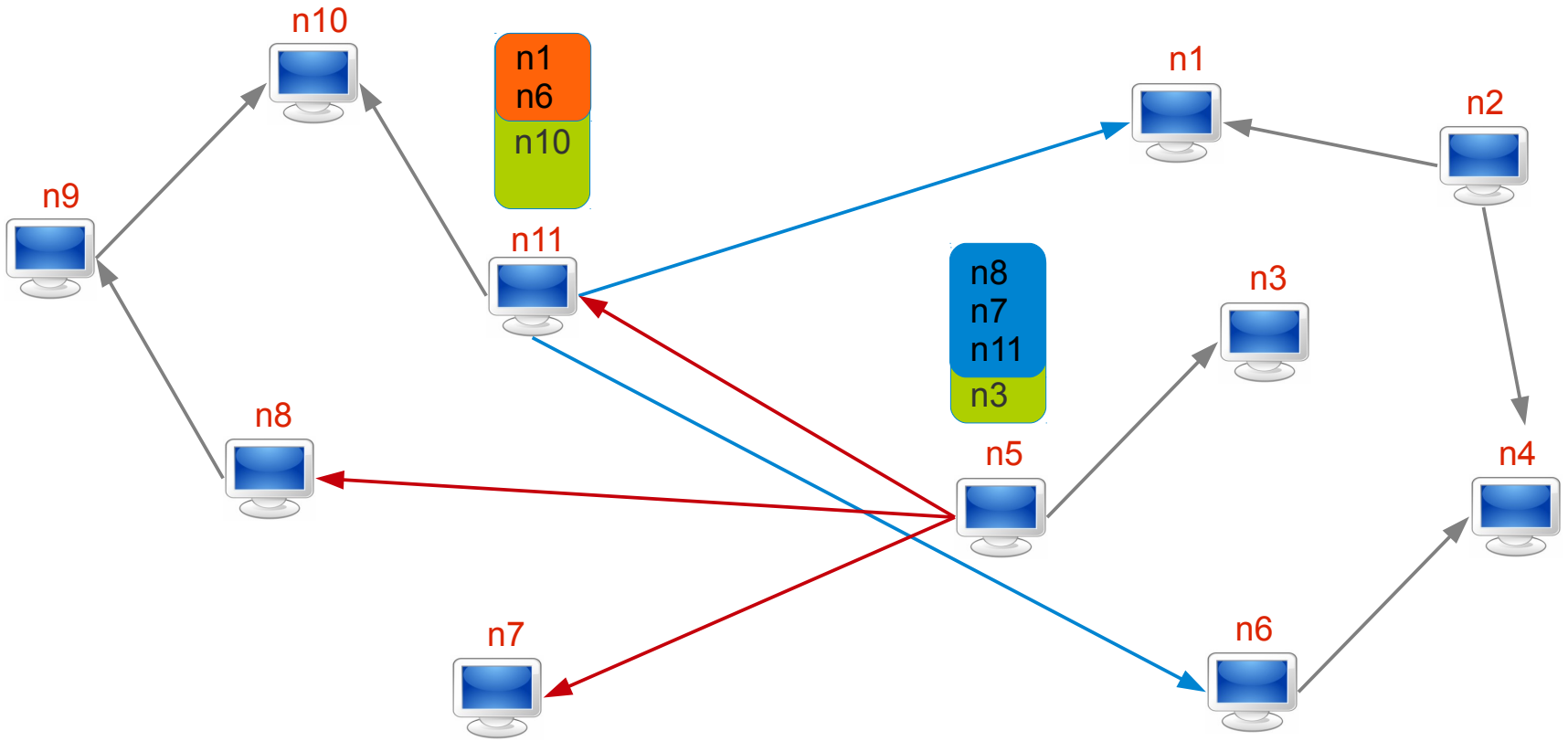
Cyclon Peer Sampling Protocol (5/7)



Cyclon Peer Sampling Protocol (6/7)



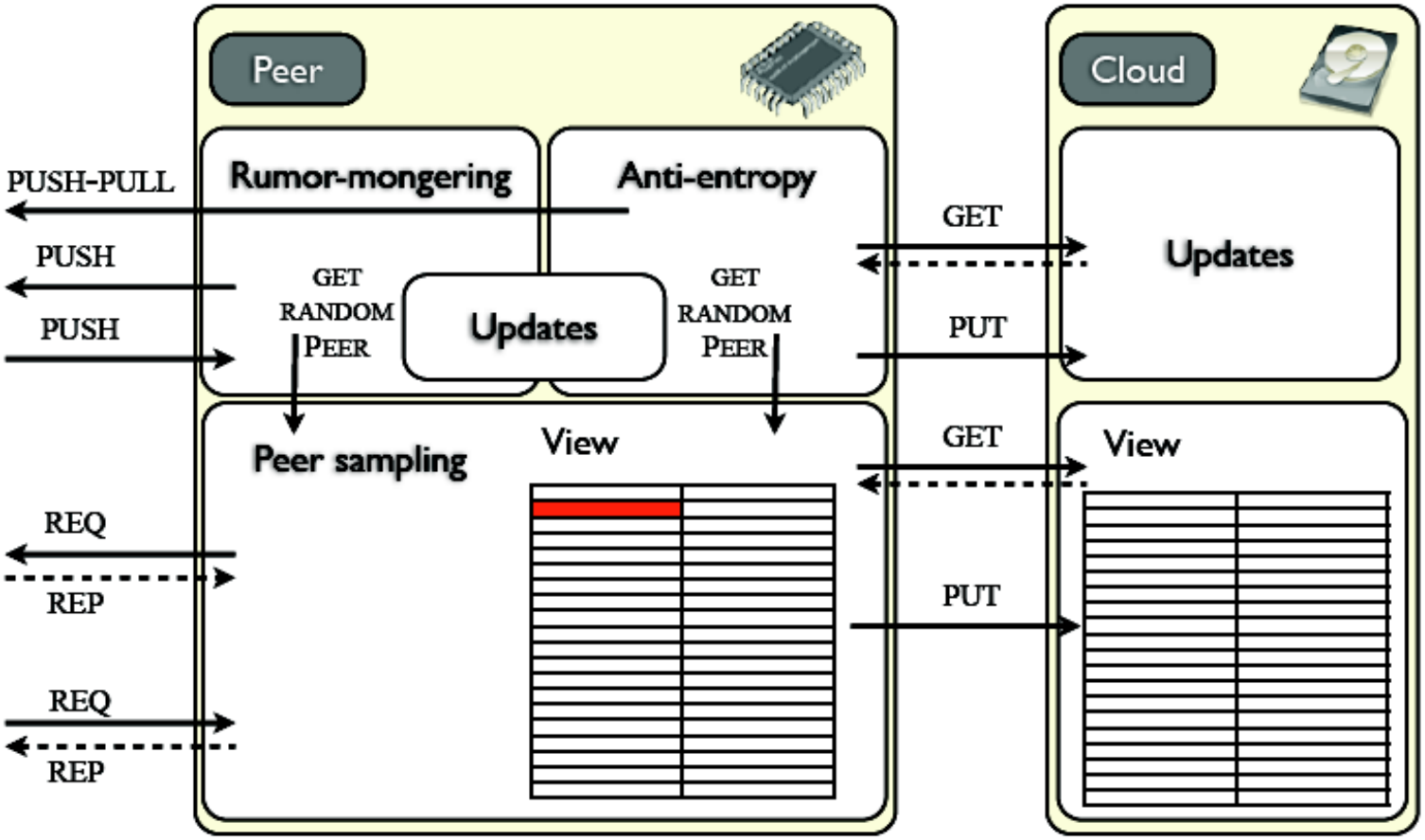
Cyclon Peer Sampling Protocol (7/7)



Cloudcast

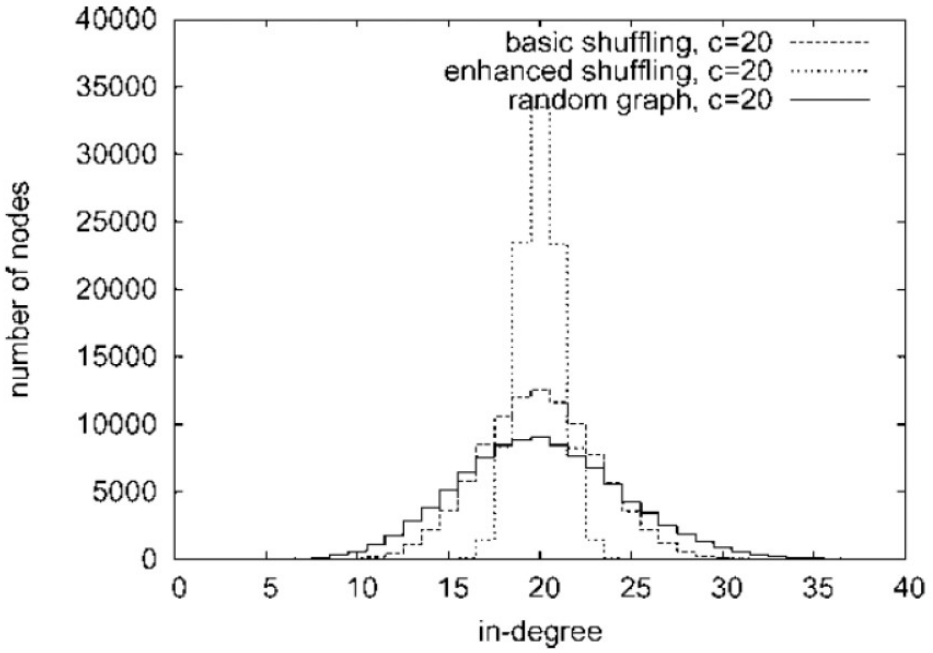
Cloudcast Architecture

- The cloud join the group - so there will be always one member.



Storage Clouds vs. Epidemic Protocols

- Reducing the number of accesses to the cloud is one of the most important requirements in Cloudcast.
- Cyclon ensures that the in-degree of each peer p tends to stay around c (view size).
- If number of nodes n > c, the cloud in-degree is c and if n < c, then cloud in-degree is n.
  - Proving that Cloudcast scales down as well.



Cloudcast Peer Sampling

- Cloud is a passive key/value storage, and can not perform computation.

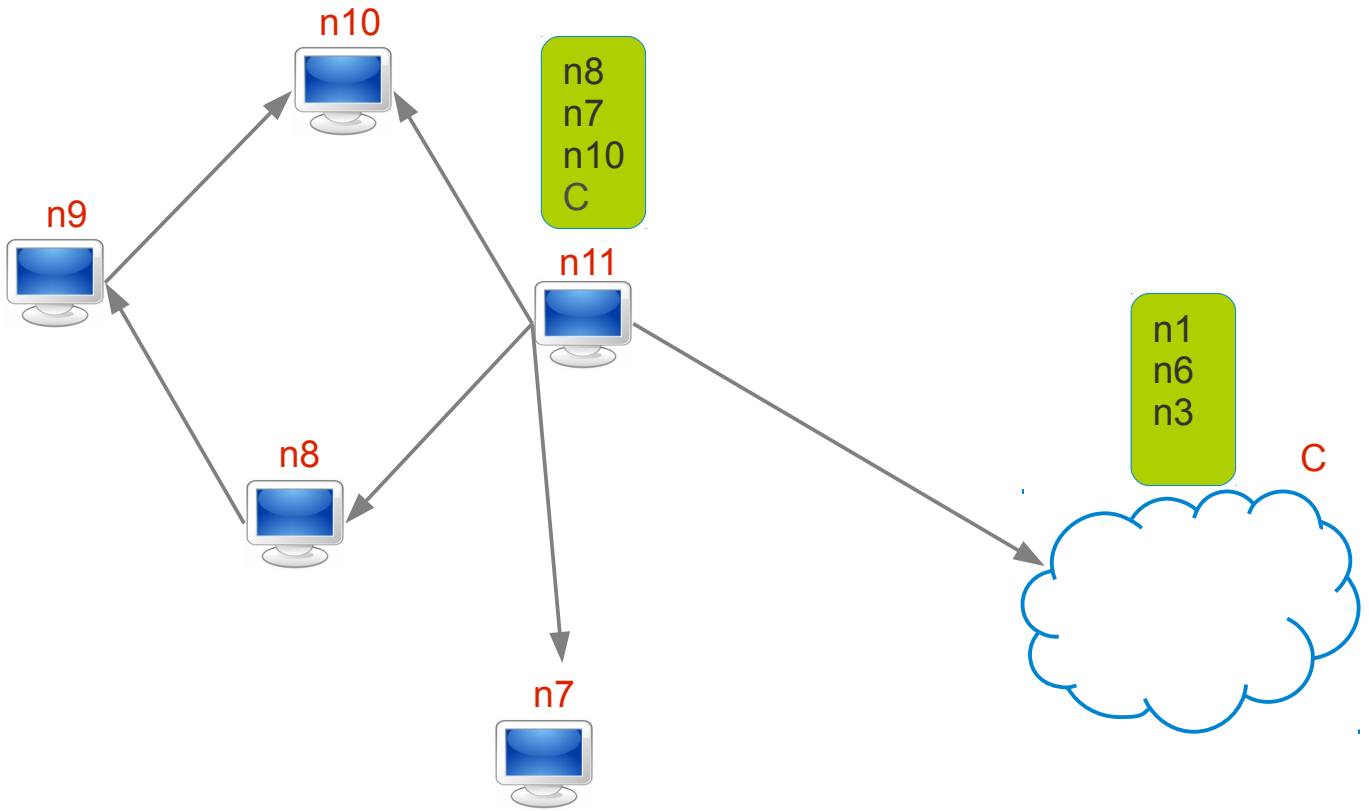
Active thread

```
do once every δ time units
   q = getPeer(state)
   if (q is cloud) then
      send(GET, VIEW)
   else
      Sp = prepareMsg(state, q)
      send(REQ, Sp) to q
```

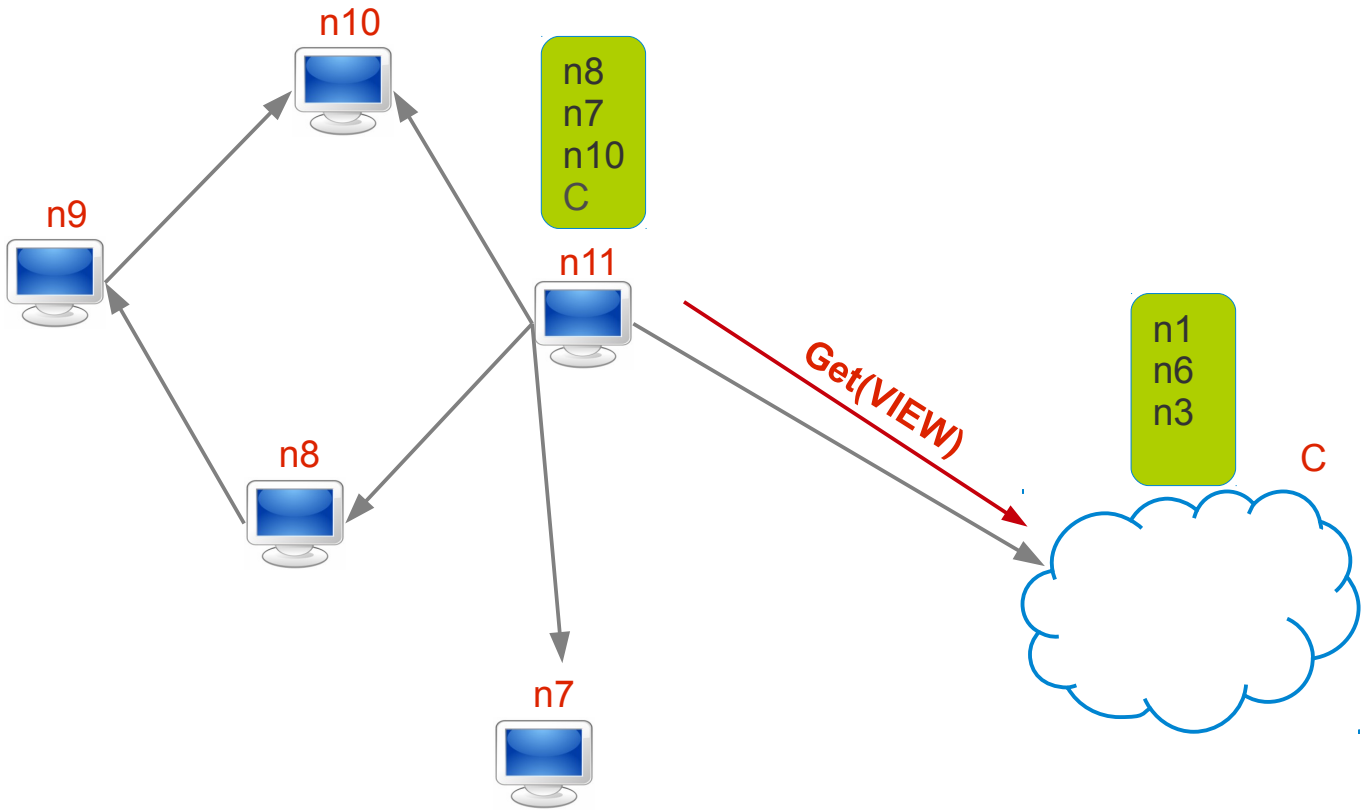
Passive thread

```
do forever
   receive(t, Sq) from *
   if (t = VIEW) then
      Sp = emulateRequestMsg()
      Sq = emulateReplyMsg()
      V = emulateView(Sp)
      send(PUT, VIEW, V) to q
   else if (t = REQ) then
      Sp = prepareMsg(state, q)
      send(REP, Sp) to q
   State = update(state, Sq)
```

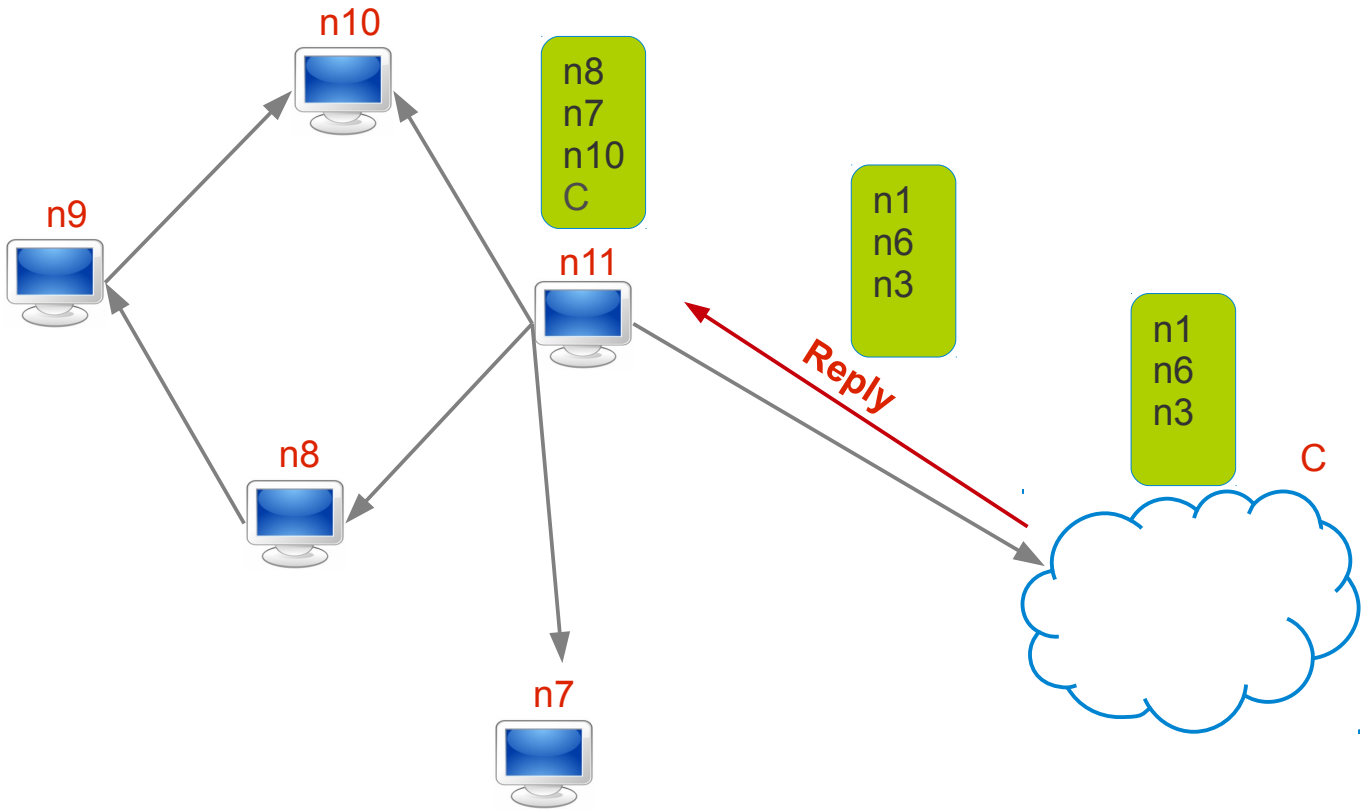
Cloudcast Peer Sampling Protocol (1/9)



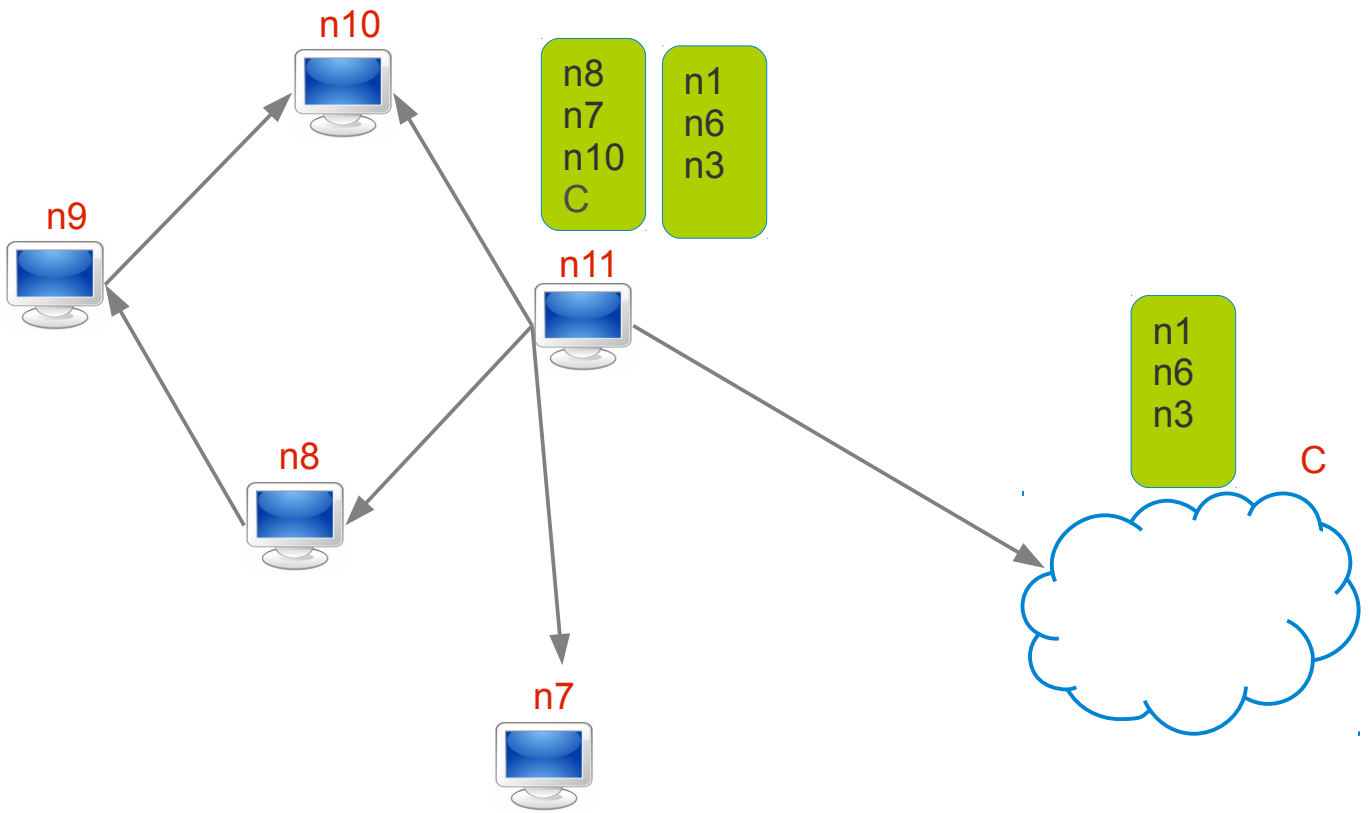
Cloudcast Peer Sampling Protocol (2/9)



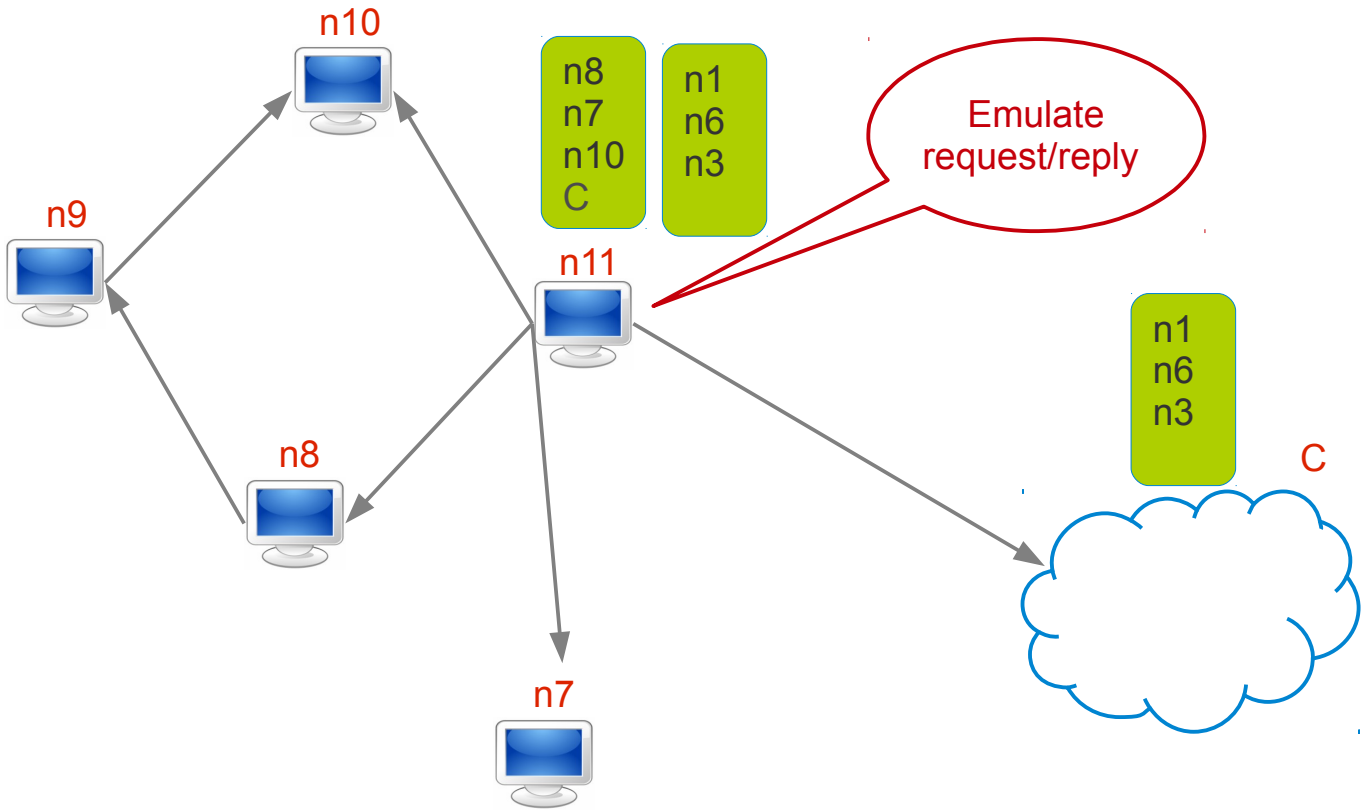
Cloudcast Peer Sampling Protocol (3/9)



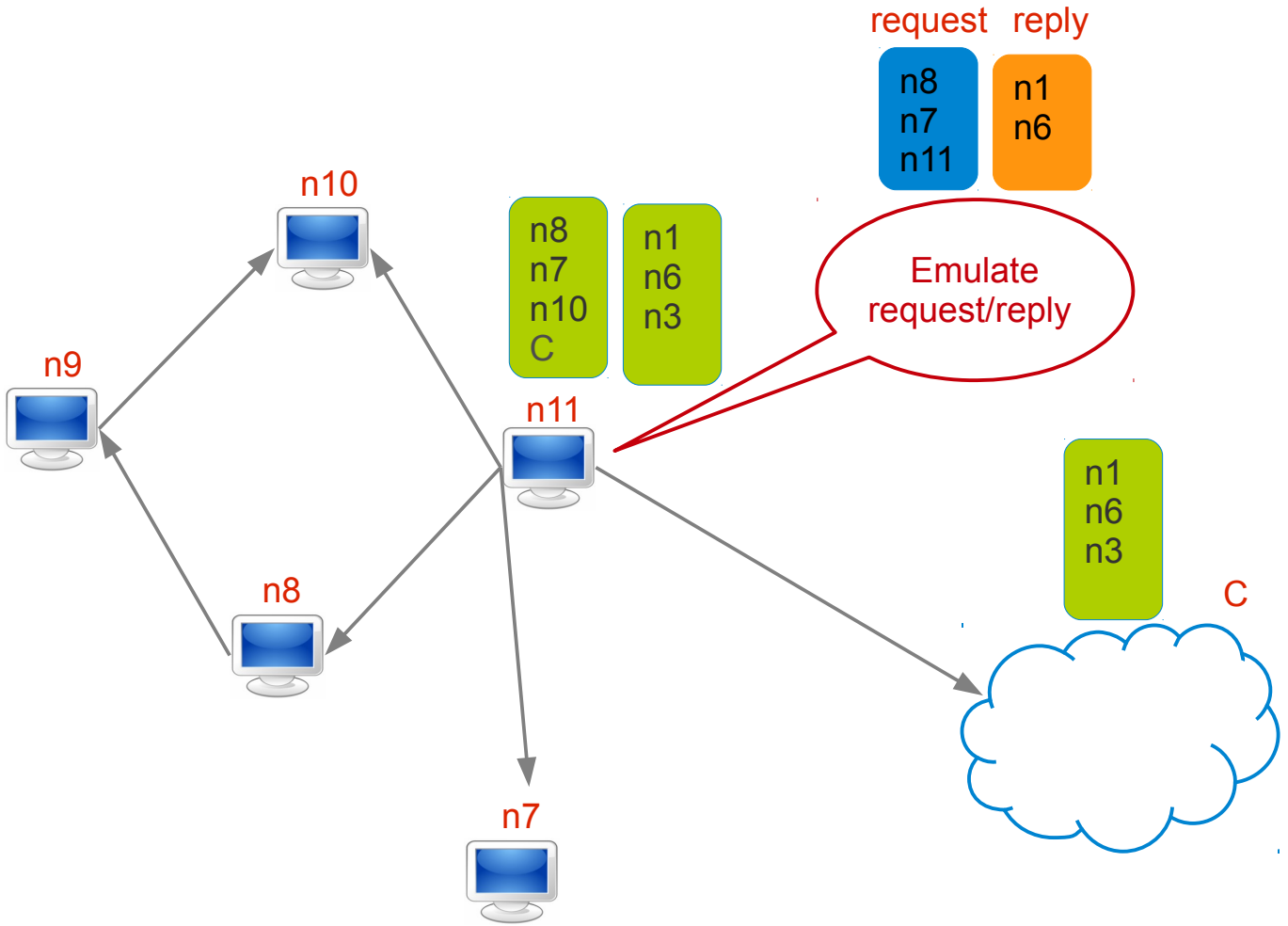
Cloudcast Peer Sampling Protocol (4/9)



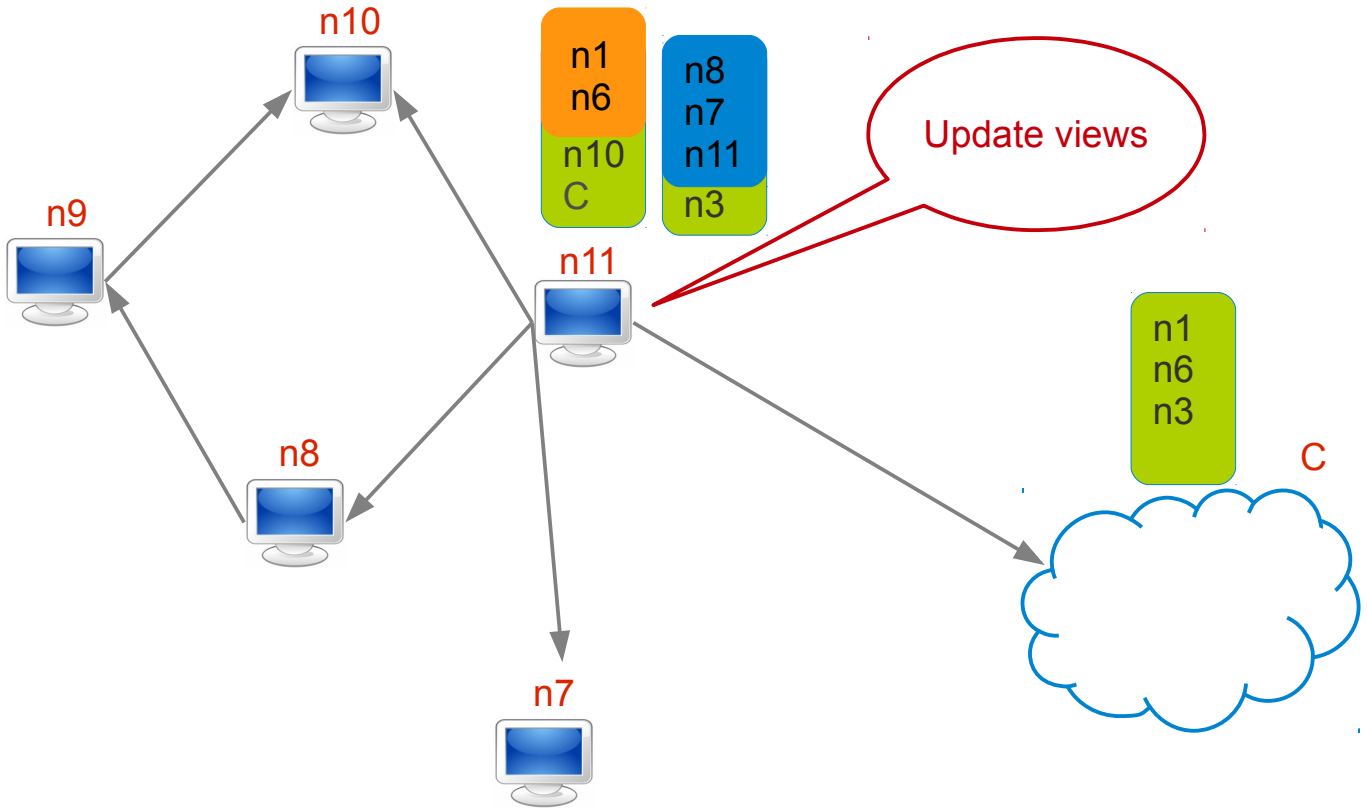
Cloudcast Peer Sampling Protocol (5/9)



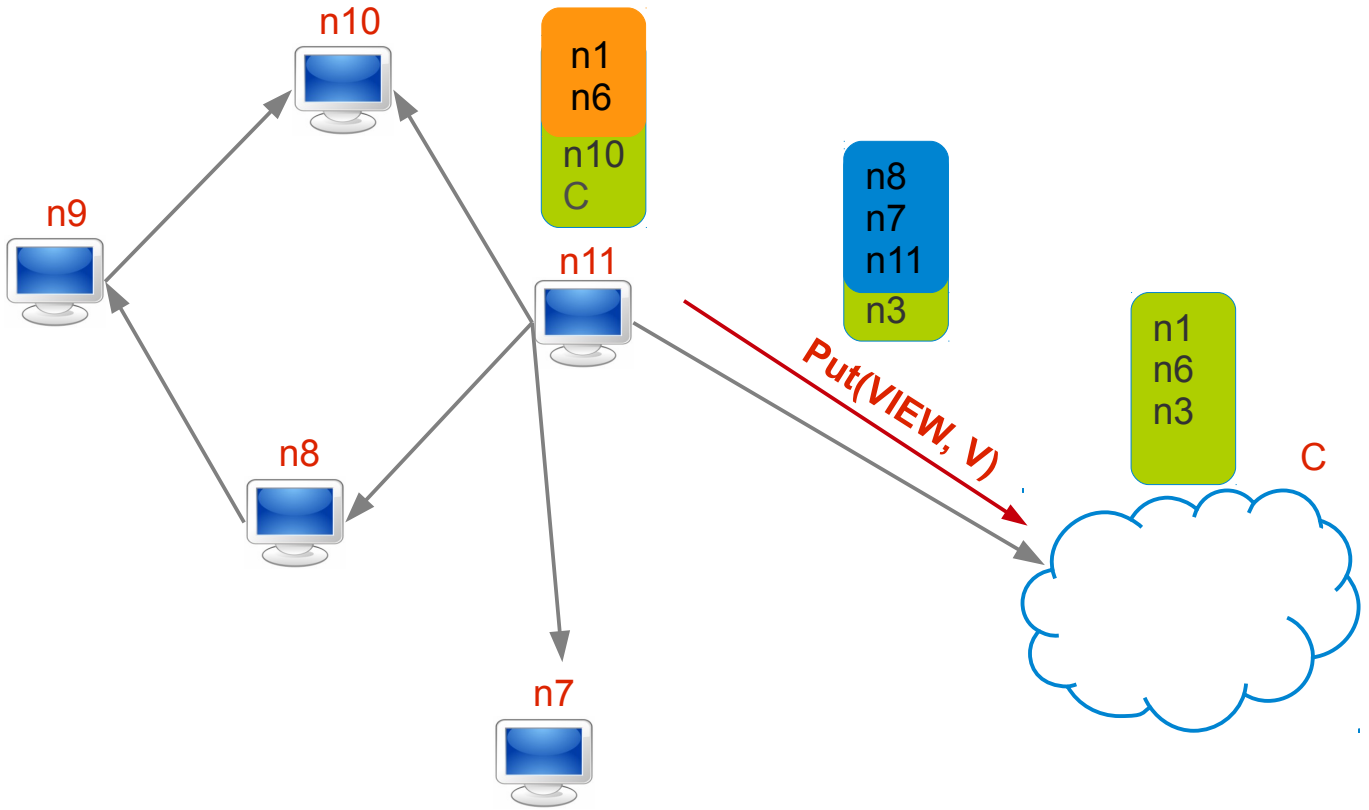
Cloudcast Peer Sampling Protocol (6/9)



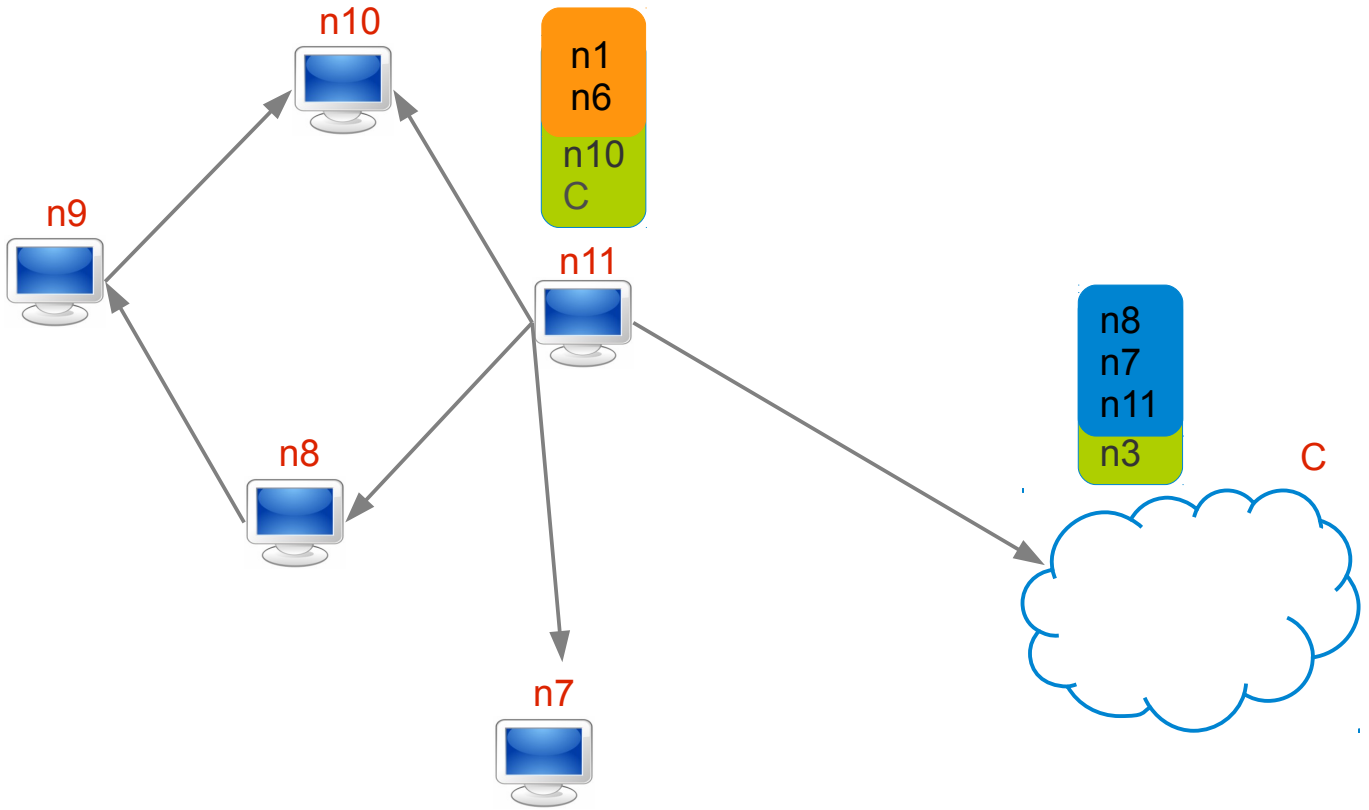
Cloudcast Peer Sampling Protocol (7/9)



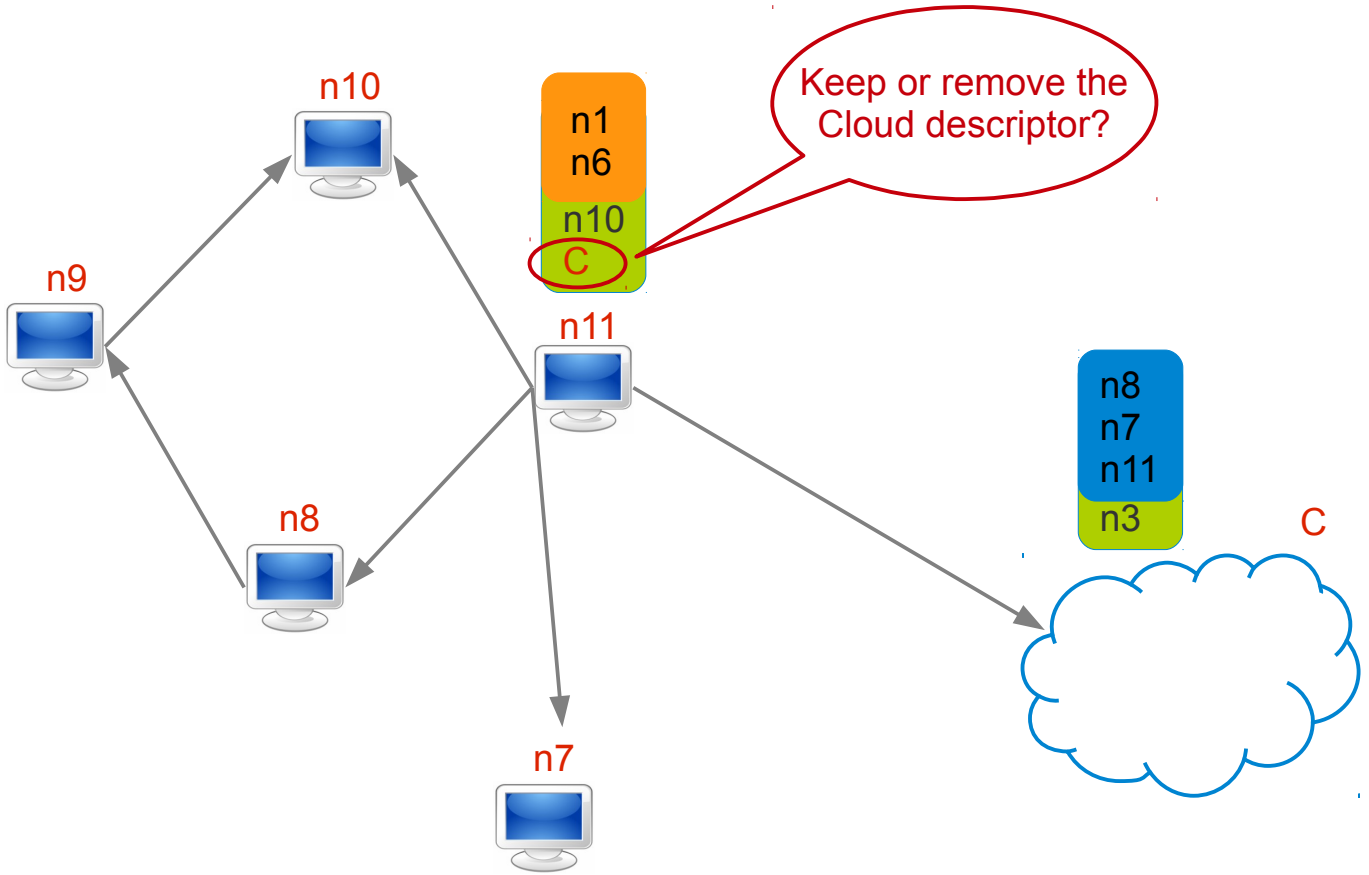
Cloudcast Peer Sampling Protocol (8/9)



Cloudcast Peer Sampling Protocol (9/9)



Keep or Remove the Cloud Descriptor?



When to add the Cloud Descriptor?

- Cloud objects have last modified information.
- When contacting a cloud:
  - If last modified is too close in time (¼ cycle), too many cloud references do not maintain the cloud reference.
  - If last modified is too far in time (4 cycles), too few cloud reference maintain the cloud reference and create a new one.

Cloudcast Bootstraping

- The new node performs a GET operation and retrieve a first view
- It can be later used to start the normal Cyclon protocol.
- If the view retrieved from the cloud is smaller than c, this means that $n < c$, and a cloud descriptor is added.

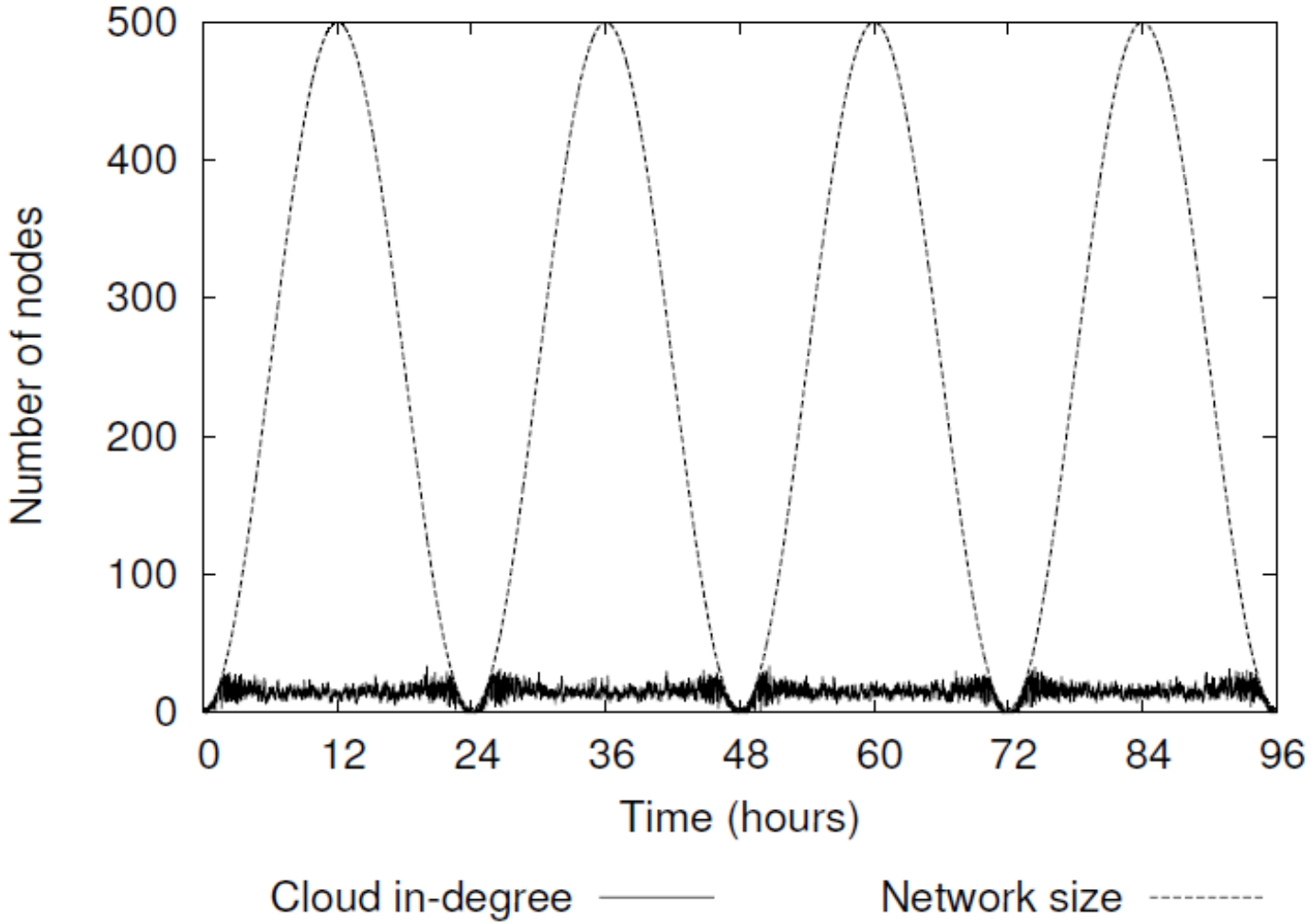
Cloudcast Information Dissemination

- The creator of a new message
  - Write the message in the cloud.
  - Update a message counter in the cloud.
  - Starts a rumor-mongering broadcasting on the new message.
- Rumor-mongering (push)
  - Every node sends the hot rumor to a random peer.
  - Stops forwarding the rumor with a given probability (0.2).
- Anti-entropy (push-pull)
  - Less frequently, node exchange summaries of the message received so far and download missing ones if needed.

Experiments

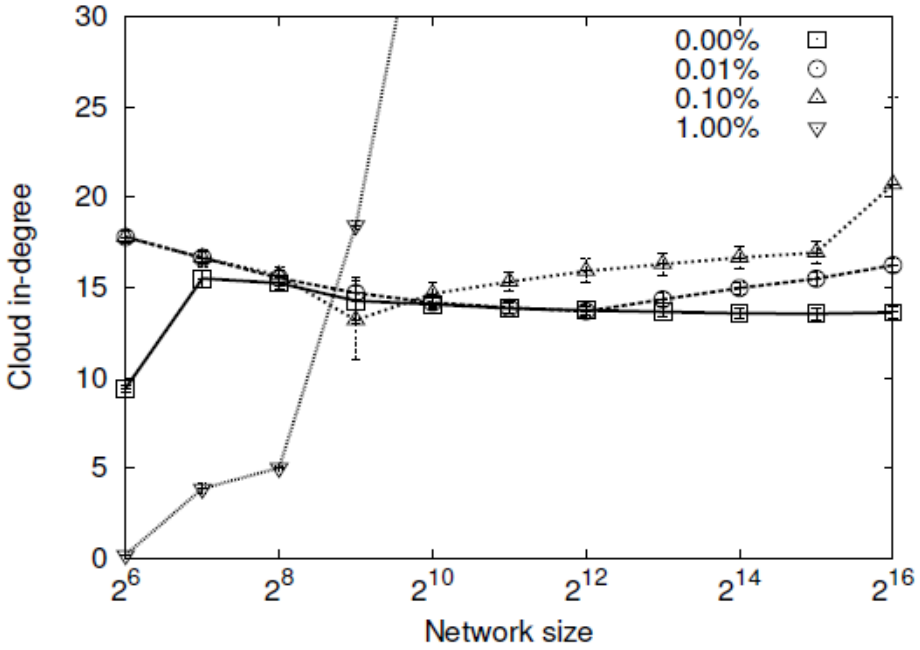
Experiments

- Simulation-based on the Peersim.

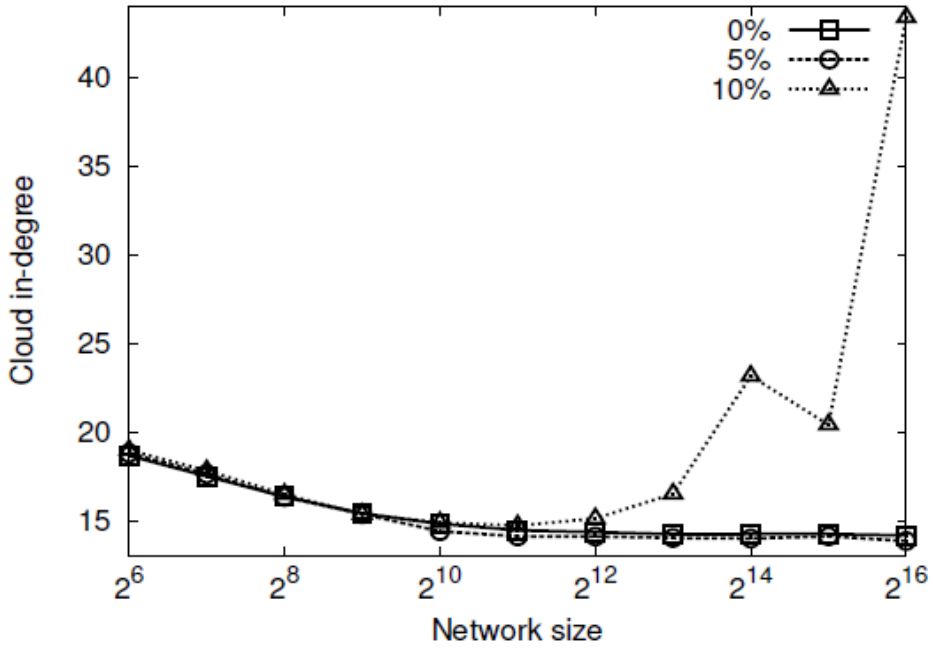


Experiments

- Simulation-based on the Peersim.



Different levels of churn



Different levels of message loss

Conclusion

Conclusion

- This paper shows a novel approach that mix the dependability of cloud computing with the low cost of P2P networks.

- Many open questions:
  - There is obviously an unbalance between peer sampling and message diffusion.
  - Avoiding so many GET/PUT requests.
  - Make the system more adaptive - if the groups grows, the cloud is used less and less (just to guarantee durability).

Question?

Acknowledgement
Some slides were derived from the slides of Alberto Montresor (University of Trento, Italy)