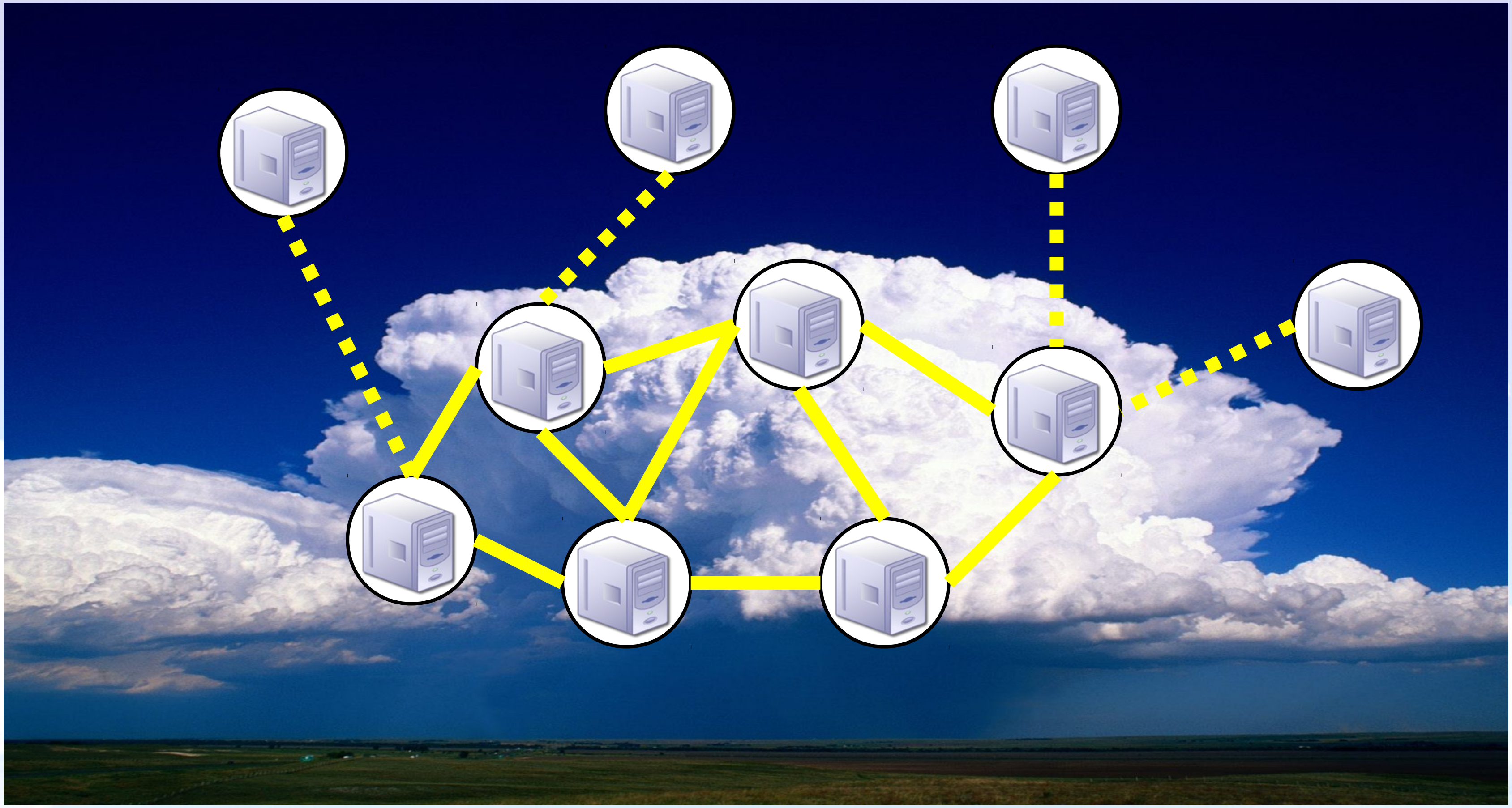
Design and Implementation of a P2P Cloud System

Ozalp Babaoglu, Moreno Marzolla, Michele Tamburini
{babaoglu, marzolla}@cs.unibo.it, miketambu@gmail.com

Our Goal

- Assemble a Cloud out of individual devices
  - E.g. PC, but also low-power devices such as set-top boxes…
  - Business model to harness the computational power of otherwise idle devices
- Individual devices leave and join, but the Cloud keeps a coherent structure
  - <u>No central controller</u>

Our vision

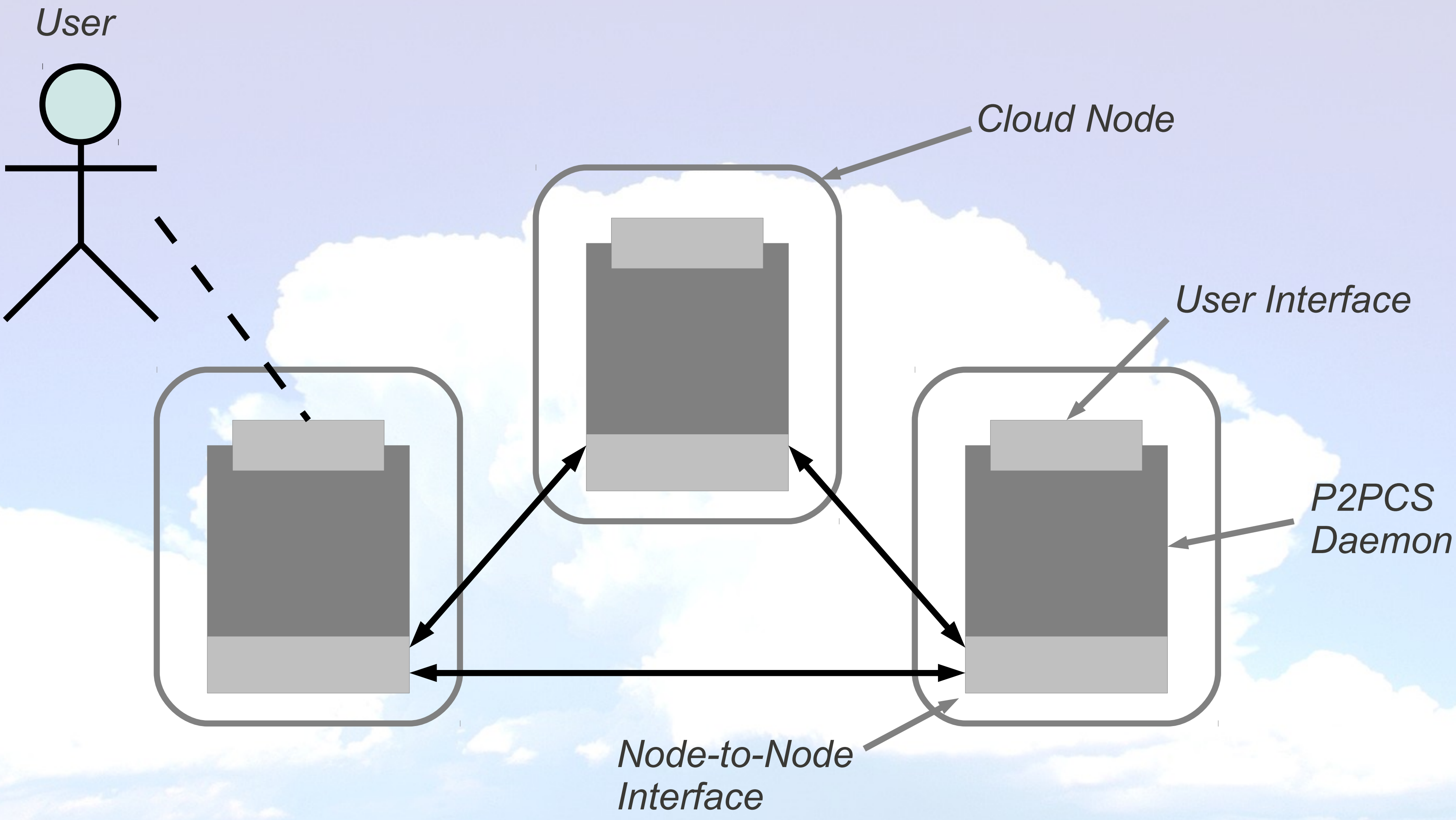


O. Babaoglu, M. Jelasity, A-M Kermarrec, A. Montresor, M. van Steen, *Managing clouds: a case for a fresh look at large unreliable dynamic networks* ACM SIGOPS Operating Systems Review, 2006

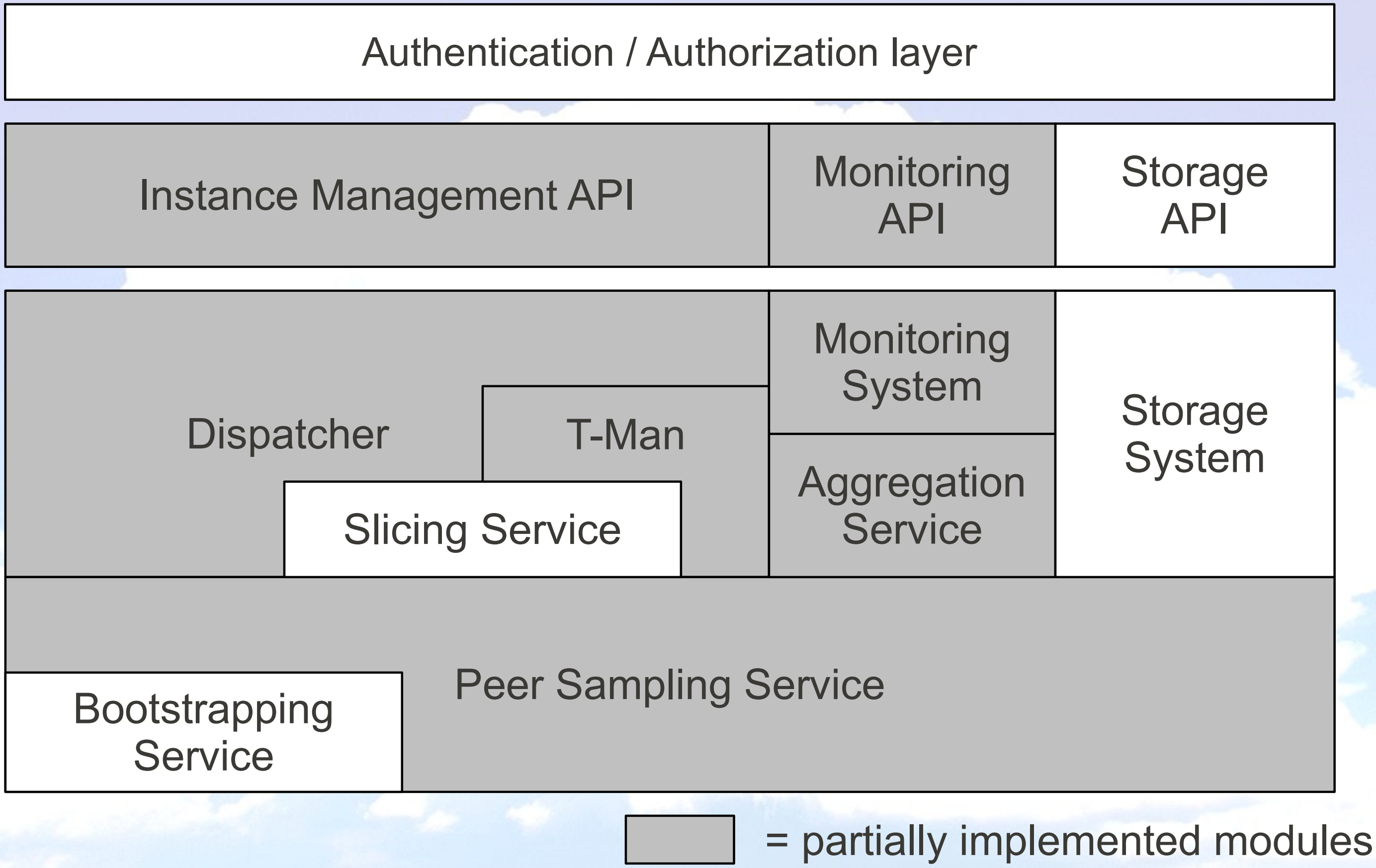
P2P Cloud—Goals

- Implement fully decentralized monitoring and management capabilities
    - *“Allocate x% of available nodes for a given task”*
    - *“Allocate at least n node for a given task”*
    - *“How many nodes are currently busy?”*
    - *“How many CPU hours have been consumed by user X?”*

P2P Cloud—Architecture



P2P Cloud—Architecture



Authentication / Authorization layer

Instance Management API

Monitoring API

Storage API

Dispatcher

T-Man

Slicing Service

Monitoring System

Aggregation Service

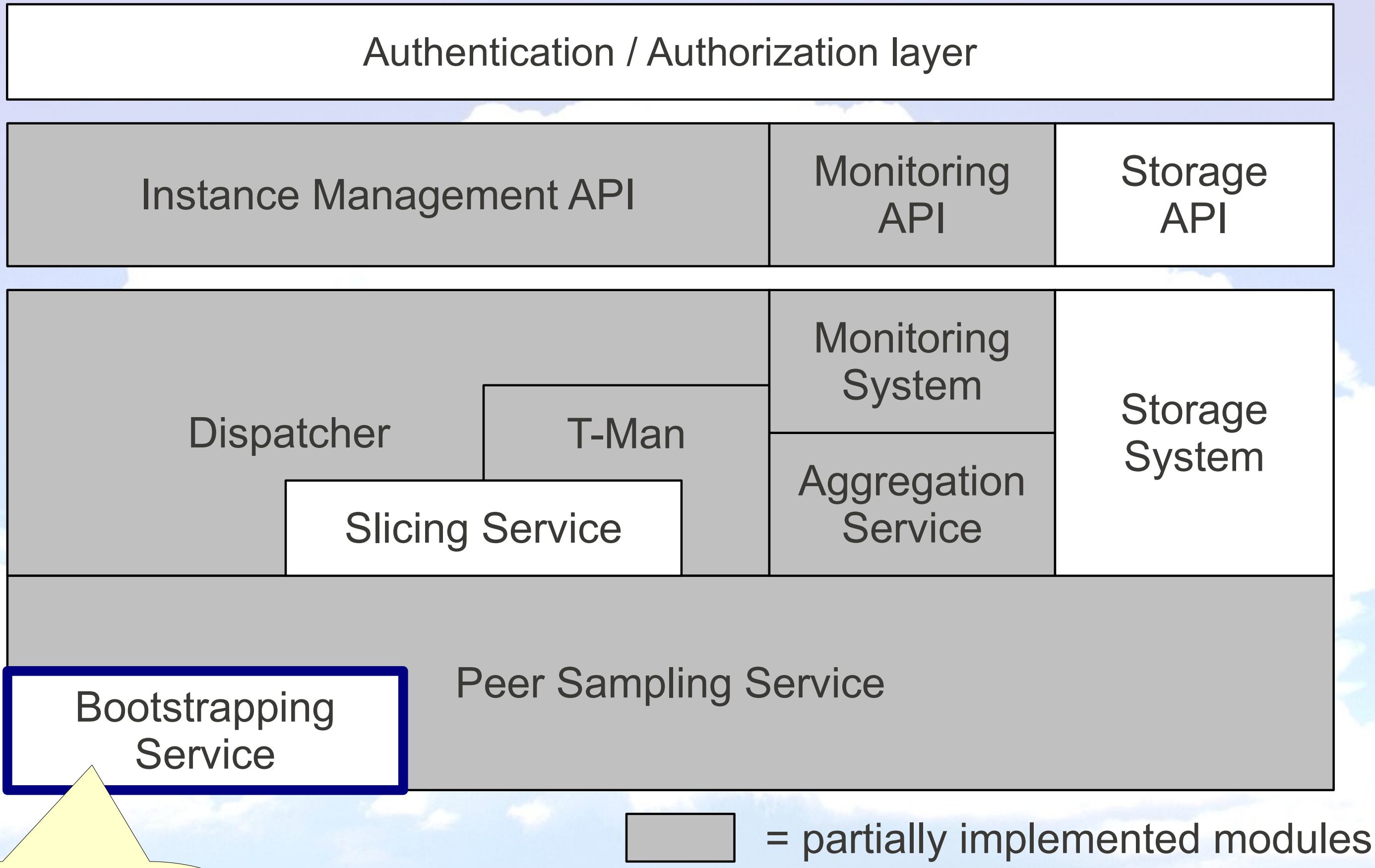
Storage System

Peer Sampling Service

Bootstrapping Service

▨ = partially implemented modules

P2P Cloud—Architecture



Authentication / Authorization layer

Instance Management API

Monitoring API

Storage API

Dispatcher

T-Man

Slicing Service

Monitoring System

Aggregation Service

Storage System

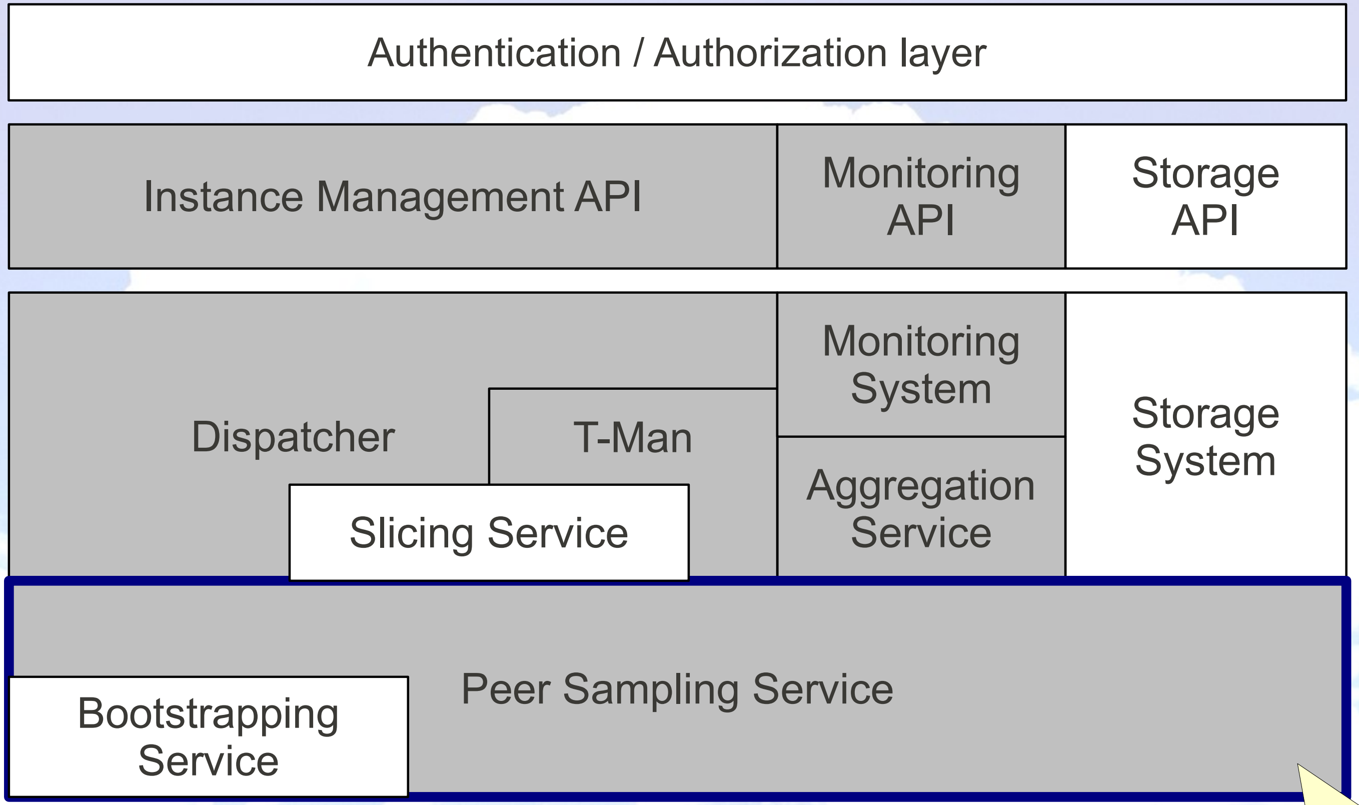
Peer Sampling Service

Bootstrapping Service

▨ = partially implemented modules

Gather an initial set of nodes to start the message exchange

P2P Cloud—Architecture



Authentication / Authorization layer

Instance Management API

Monitoring API

Storage API

Dispatcher

T-Man

Slicing Service

Monitoring System

Aggregation Service

Storage System

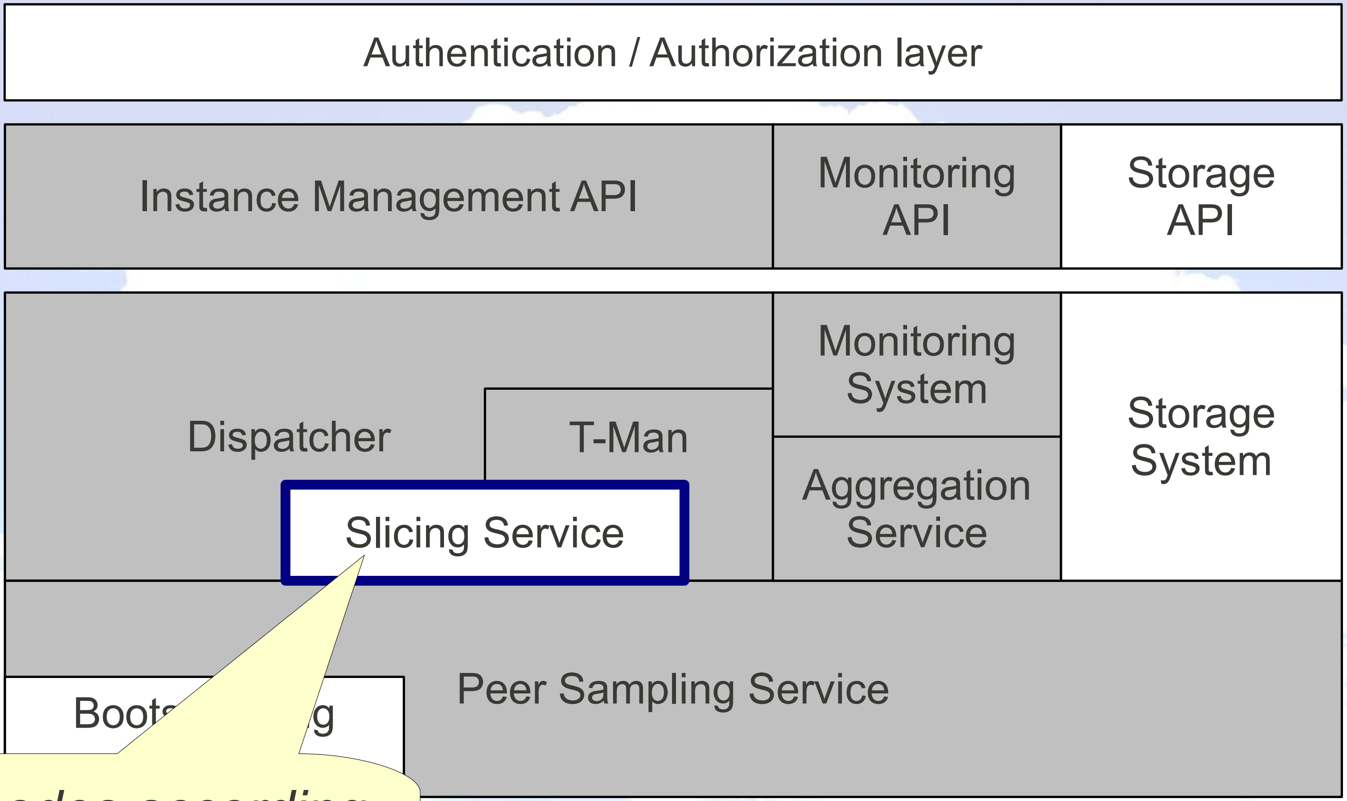
Peer Sampling Service

Bootstrapping Service

▨ = partially implemented modu

Provide each node with a list of peers to exchange messages with

P2P Cloud—Architecture



Authentication / Authorization layer

Instance Management API

Monitoring API

Storage API

Dispatcher

T-Man

Slicing Service

Monitoring System

Aggregation Service

Storage System

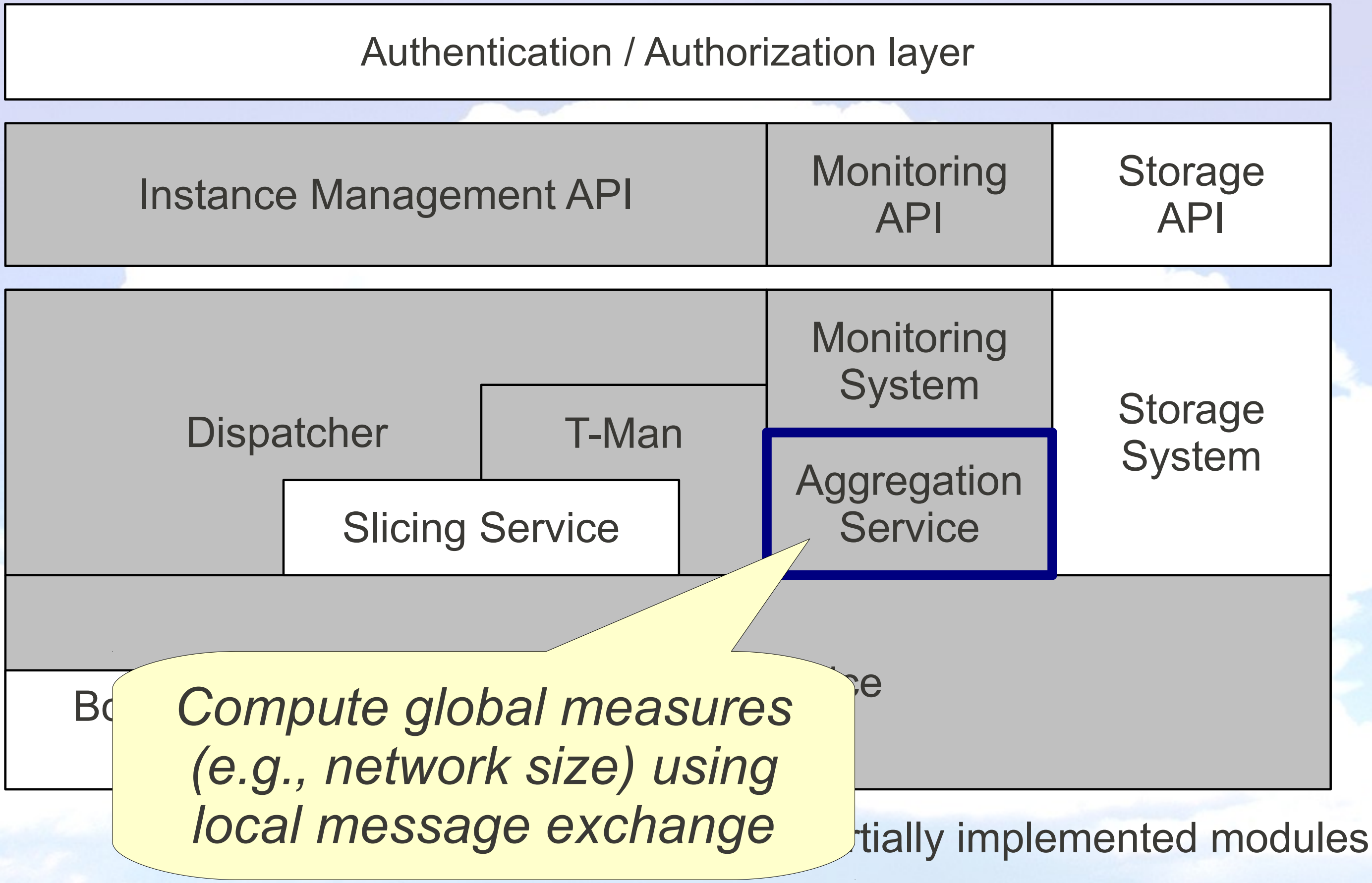
Boots g

Peer Sampling Service

Rank the nodes according to one attribute (e.g., top 5% of fastest nodes)

▨ = partially implemented modules

P2P Cloud—Architecture



Authentication / Authorization layer

Instance Management API

Monitoring API

Storage API

Dispatcher

T-Man

Slicing Service

Monitoring System

Aggregation Service

Storage System

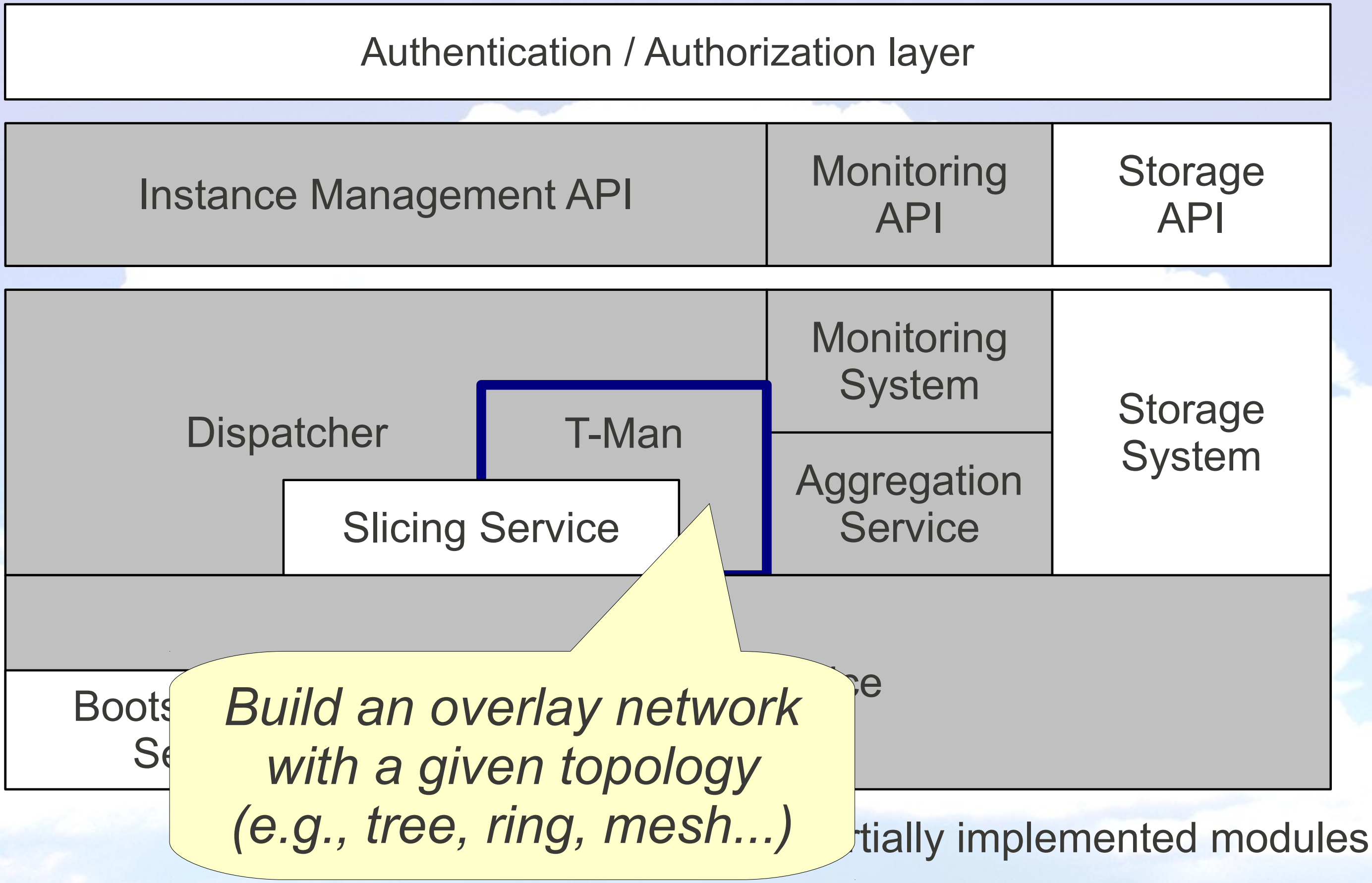
Bo

ce

rtially implemented modules

Compute global measures (e.g., network size) using local message exchange

P2P Cloud—Architecture



Authentication / Authorization layer

Instance Management API

Monitoring API

Storage API

Dispatcher

T-Man

Slicing Service

Monitoring System

Aggregation Service

Storage System

Boots
Se

ce

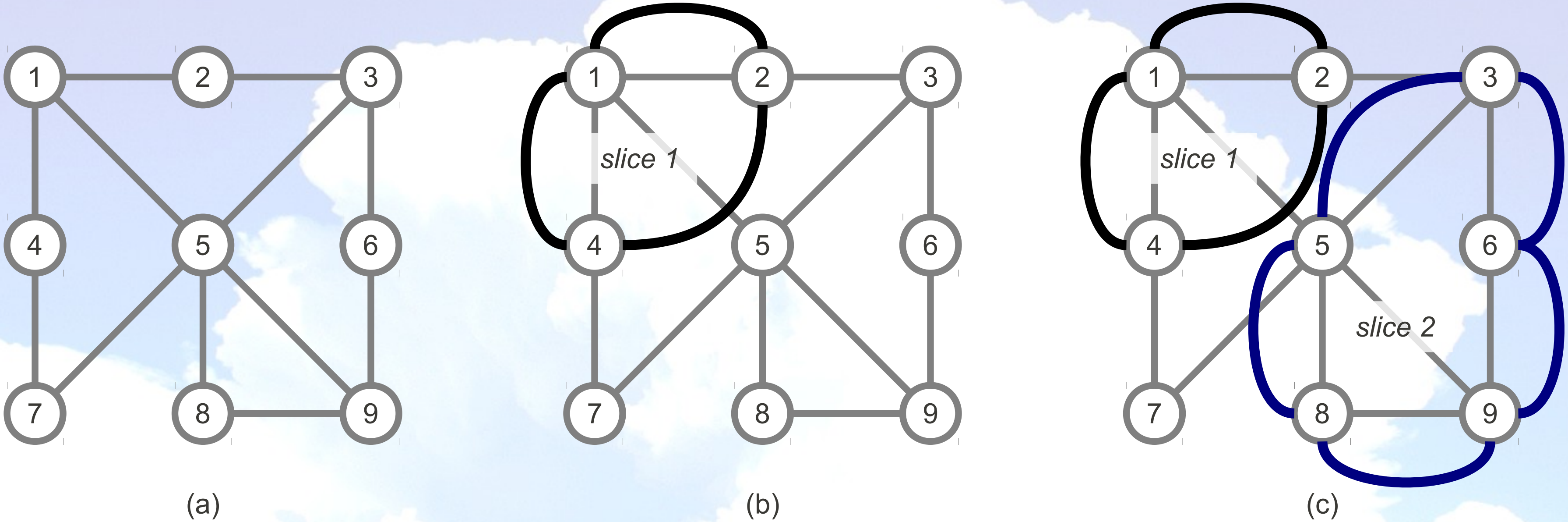
rtially implemented modules

Build an overlay network with a given topology (e.g., tree, ring, mesh...)

P2PCS API

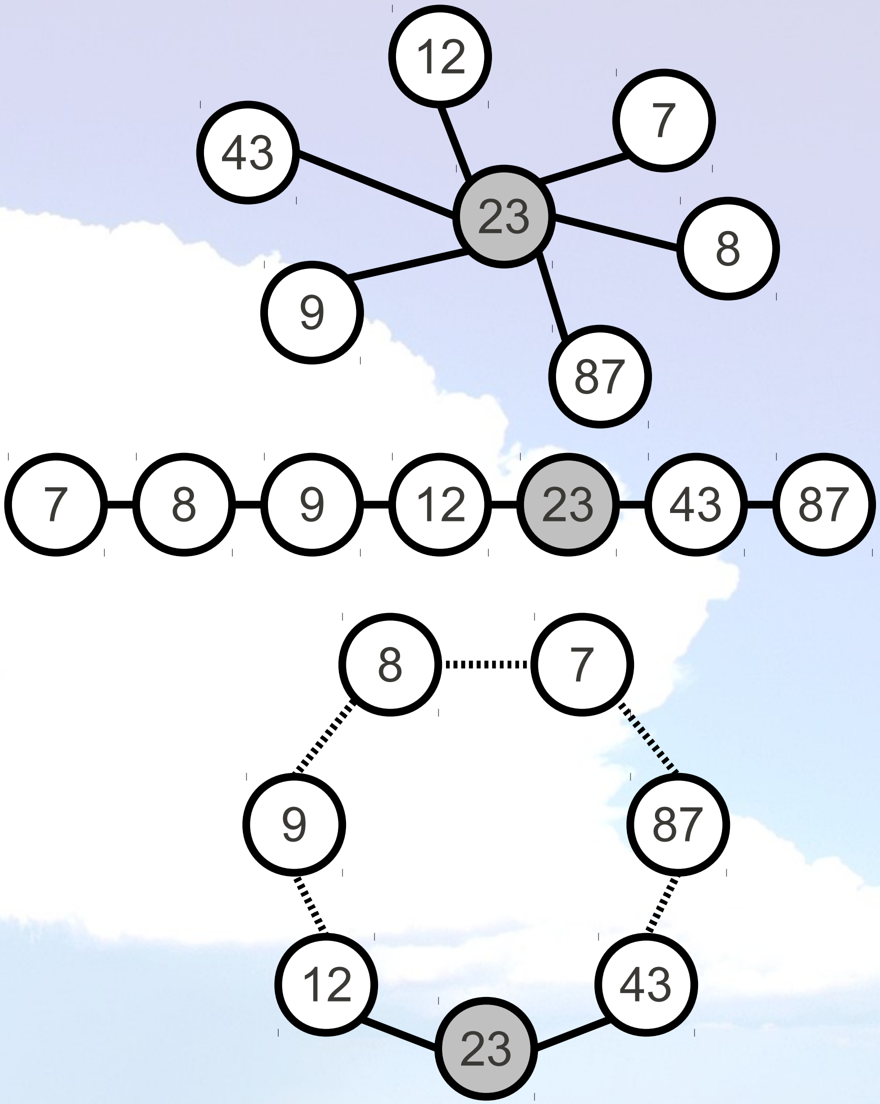
- **`run-nodes`** ***`subcloud_id number`***
  - Creates a subcloud with *number* nodes; *subcloud_id* is set as the name of the newly created subcloud
- **`terminate-nodes`** ***`subcloud_id nodename1 … nodenameN`***
  - Removes the named nodes from the subcloud with given id
- **`add-new_nodes`** ***`subcloud_id number`***
  - Adds *number* nodes to the subcloud identified by *subcloud_id*. The new nodes are chosen without any particular criteria
- **`describe-instances`** ***`nodename`***
  - Prints a human-readable description of the given node
- **`monitor-instances`**
  - Return the global size of the Cloud using the aggregation service
- **`unmonitor-instances`**
  - Stops printing the global size of the Cloud

P2PCS: Building sub-clouds



P2PCS: Building sub-clouds

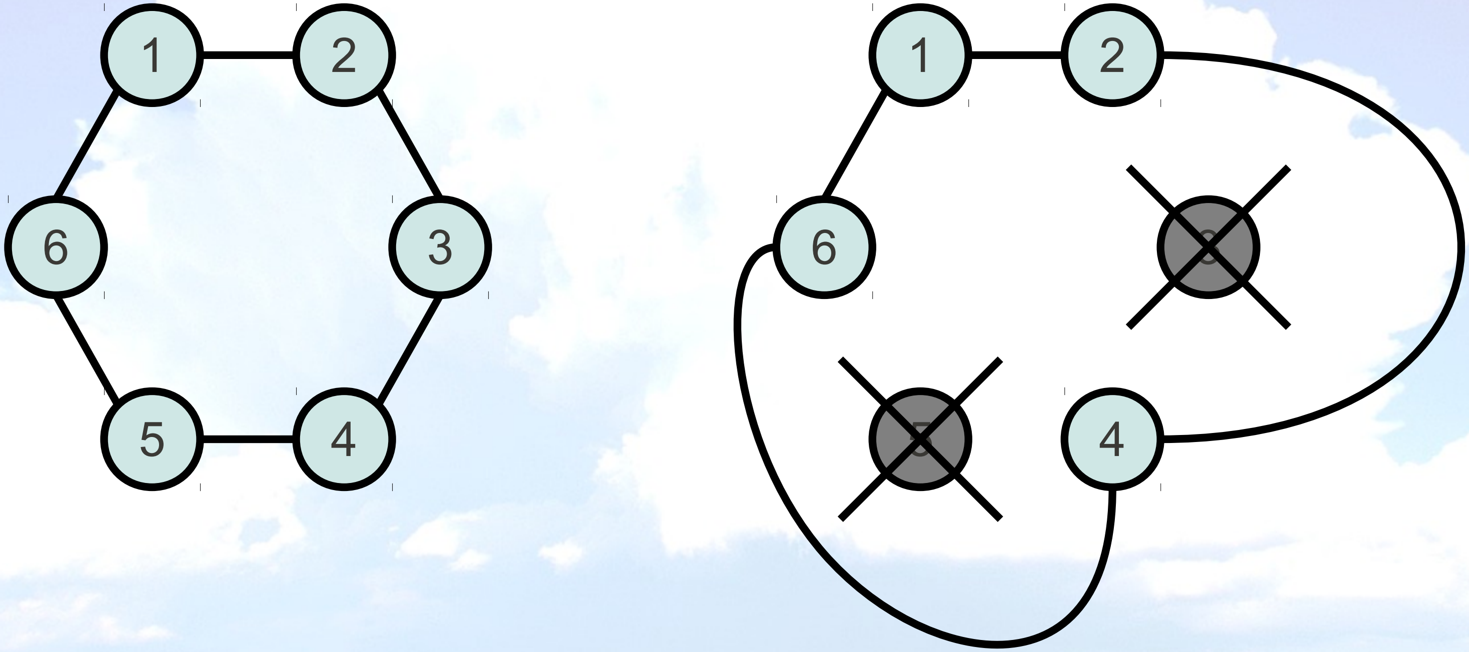
- Nodes have unique numerical IDs (e.g., hash value of their IP addresses)
- Each node collects IDs of neighbors (and of itself)
- IDs are sorted and "wrapped" as a ring
- The "distance" to each neighbor is defined as the minimum number of hops along the ring
- The node connects to the two "nearest" neighbors



O. Babaoglu, M. Jelasity, A. Montresor, *T-MAN: Gossip-based fast overlay topology construction*, Computer Networks, 53(2009), 2321—2339

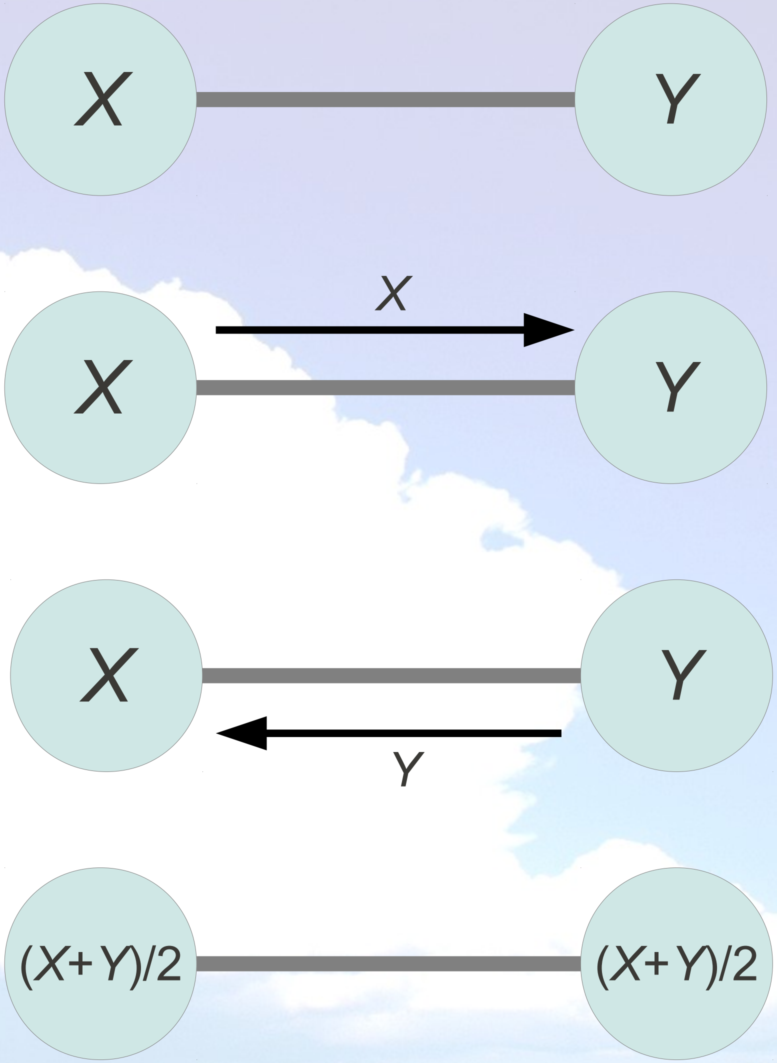
P2PCS: Building sub-clouds

- T-Man allows rings (sub-clouds) to be automatically “repaired” when one or more participating nodes fail



Estimating the Cloud size

- We compute the mean of numerical values held at each node
- Each node holds zero by default; the node on which the `monitor-instances` command originated holds one
- After a few rounds we pick the value stored at any node and invert it. This is the estimated Cloud size
- (If there are *N* nodes, the mean converges to 1/*N*)



Márk Jelasity, Alberto Montresor, Ozalp Babaoglu, *Gossip-based aggregation in large dynamic networks*, ACM Transactions on Computer Systems (TOCS), 2005

Conclusions

- P2PCS uses simple epidemic protocols as basic building blocks
  - Intrinsecally scalable and robust
- Ongoing activity
  - Transforming the prototype into a usable application
  - Deployment and testing on some large infrastructure
- Prototype available at:

http://cloudsystem.googlecode.com/

Thank you for your attention

Questions?