AVMON: Consistent and Scalable Availability Monitoring Overlay

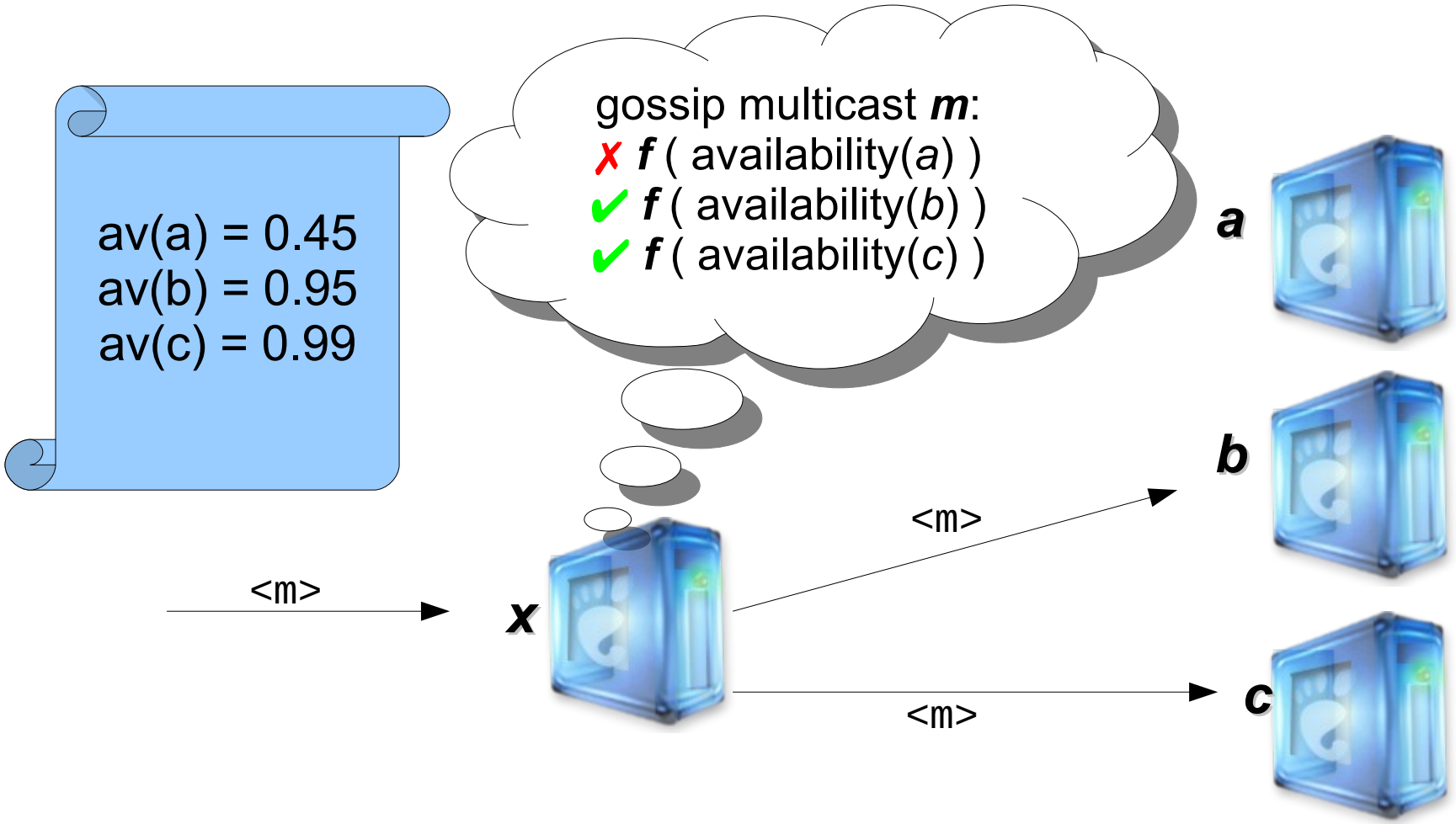
Ramsés Morales and Indranil Gupta

Dept. of Computer Science
University of Illinois at Urbana-Champaign

Motivation

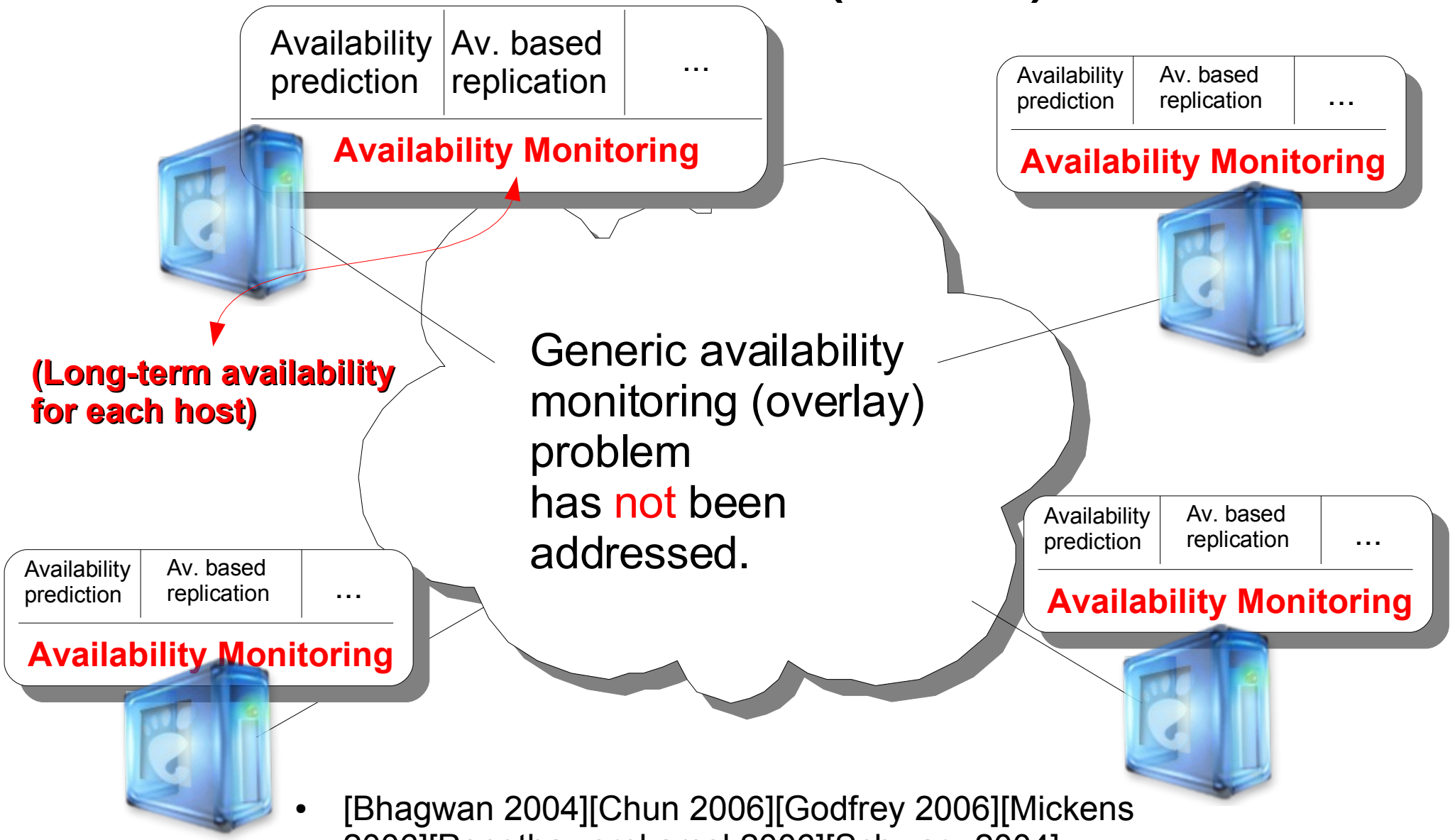
- Large scale distributed applications must deal with churn.
- Difficult to predict availability variations across nodes and time cause churn.
  - Availability: fraction of time a node is online.

Motivation (cont.)



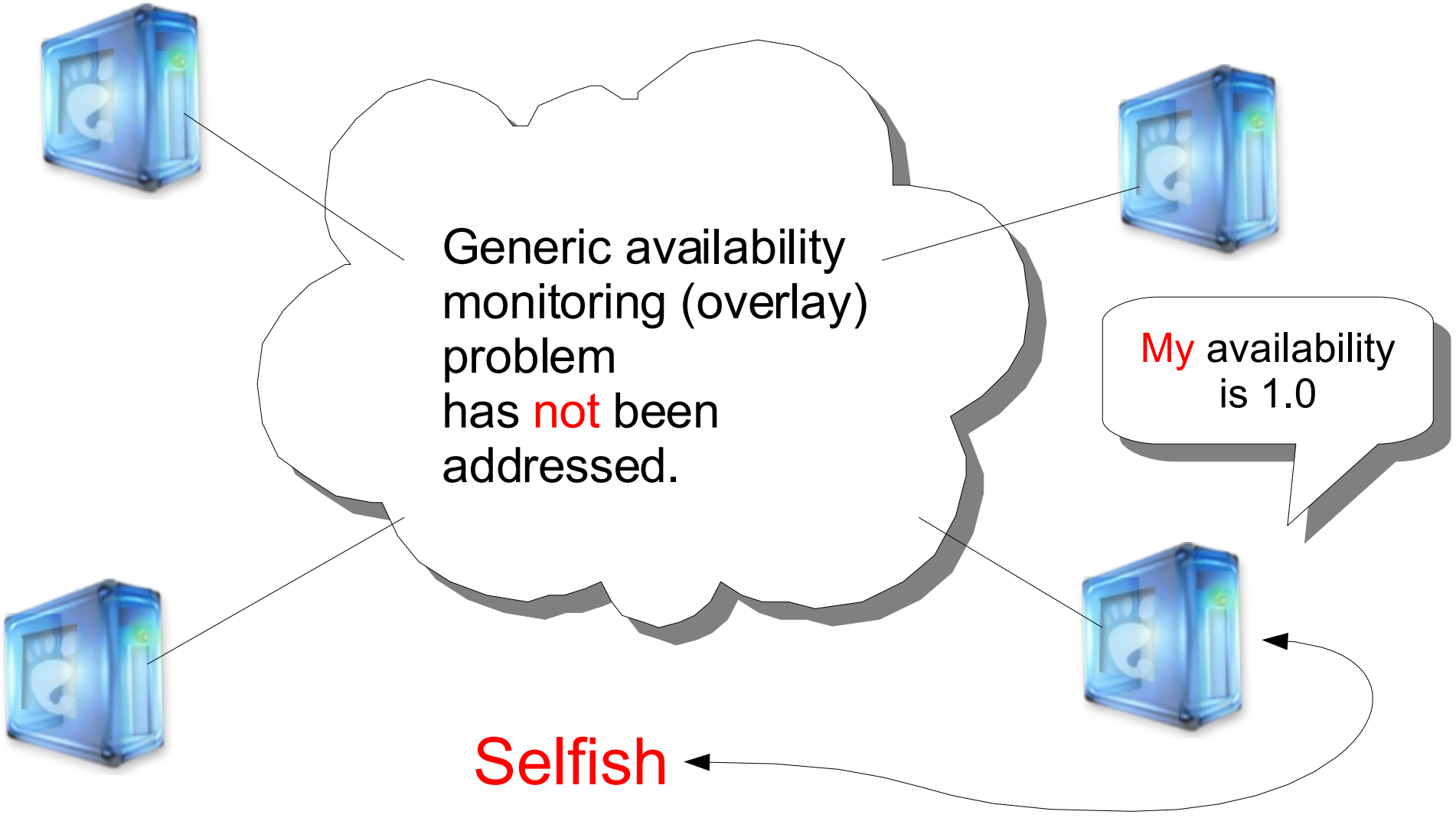
- Multicast reception reliability ∝ availability
  [Pongthawornkamol 2006]

Motivation (cont.)

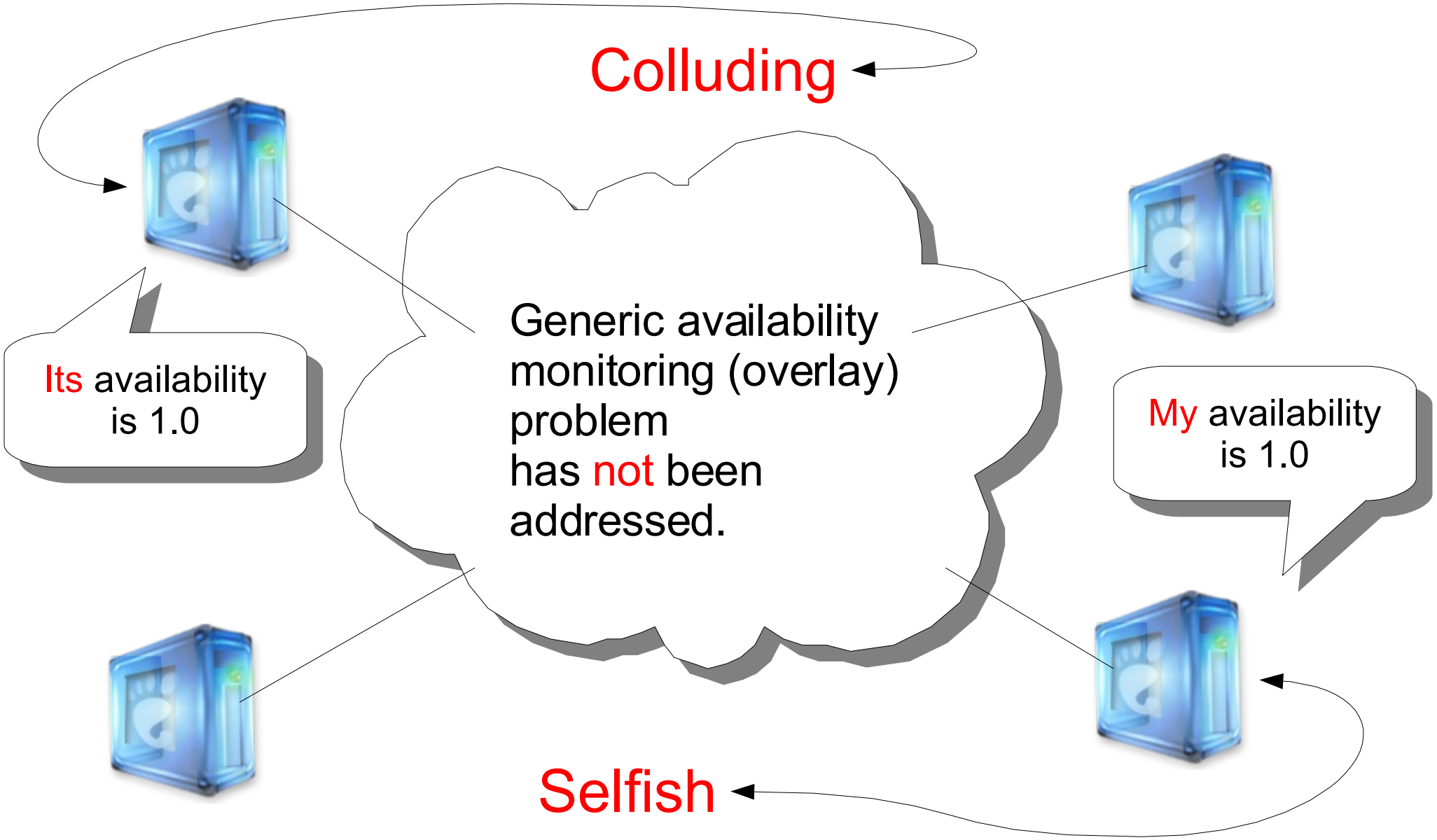


- [Bhagwan 2004][Chun 2006][Godfrey 2006][Mickens 2006][Pongthawornkamol 2006][Schwarz 2004]

Motivation (cont.)



Motivation (cont.)



Availability Monitoring Problem

I. Availability Monitoring Overlay [our focus]

II. Availability History Maintenance

- [Bhagwan 2004][Mickens 2006]

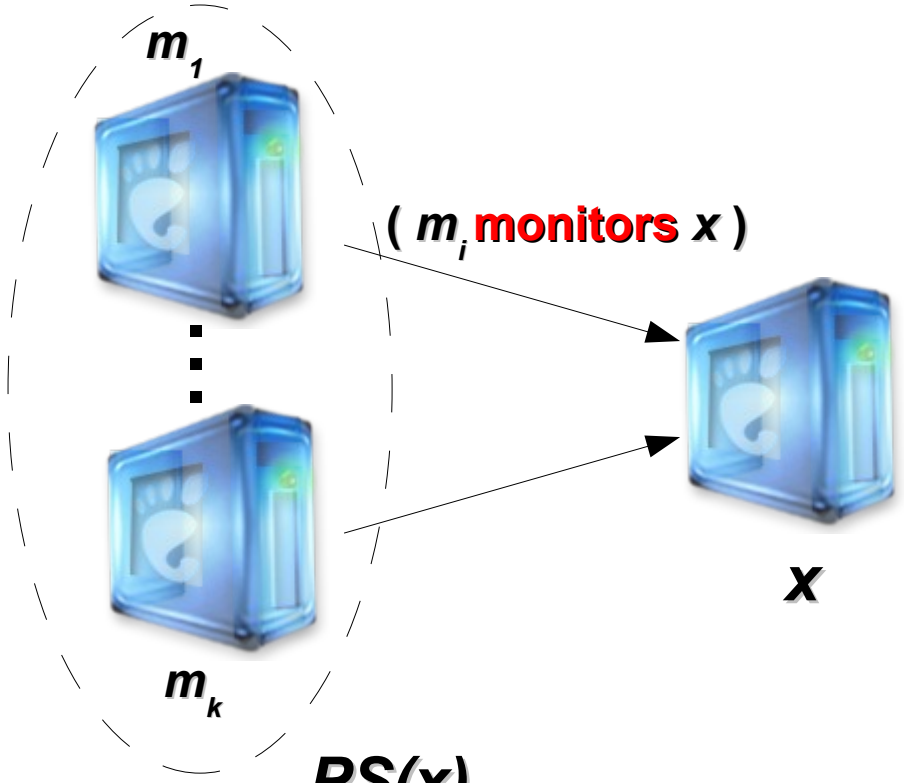
Availability Monitoring Overlay Problem

- *For each node* **x**, *select and discover a small subset of nodes to monitor* **x**.

x

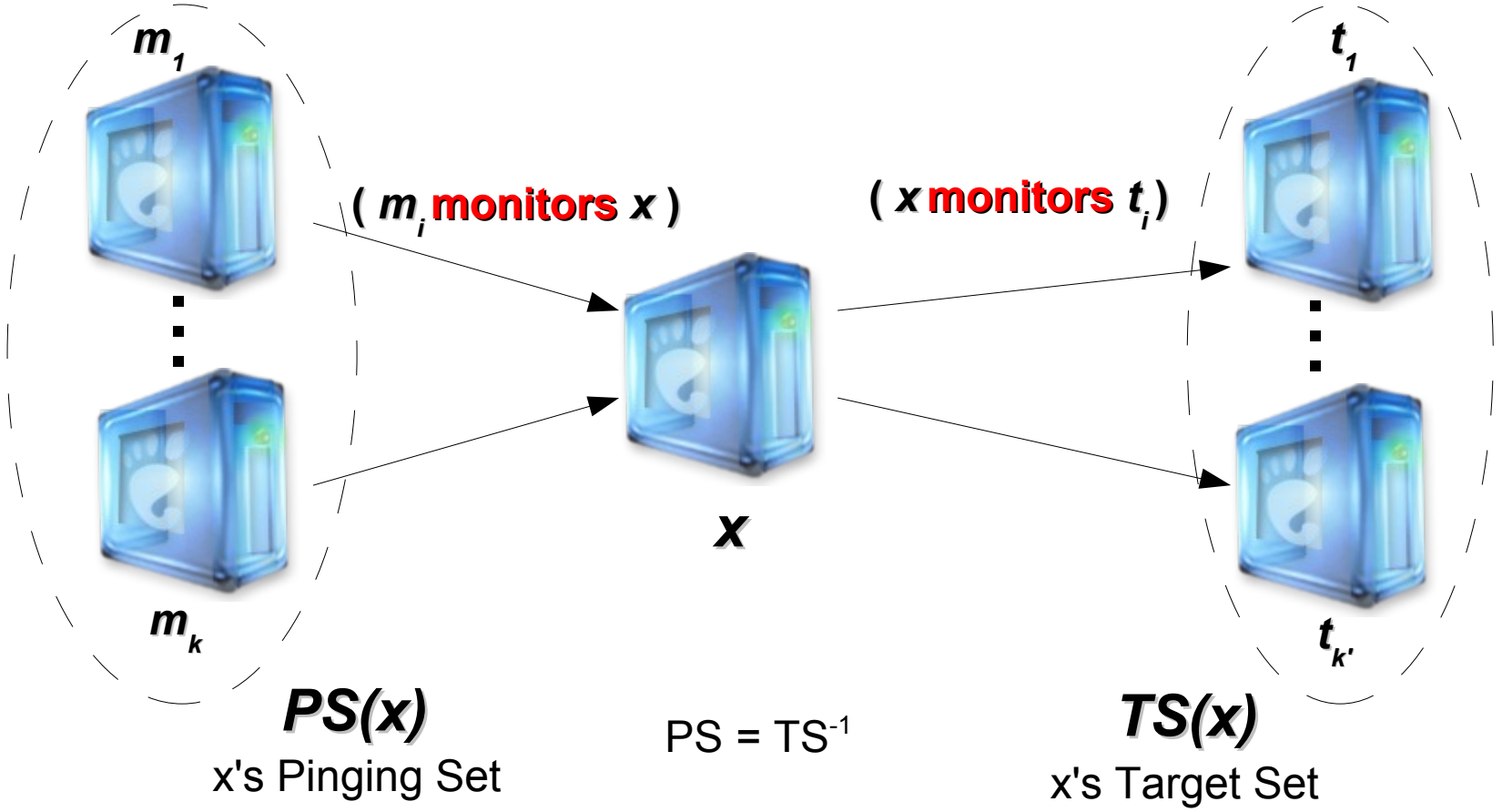
Availability Monitoring Overlay Problem

- *For each node* **x**, *select and discover a small subset of nodes to monitor* **x**.



Availability Monitoring Overlay Problem

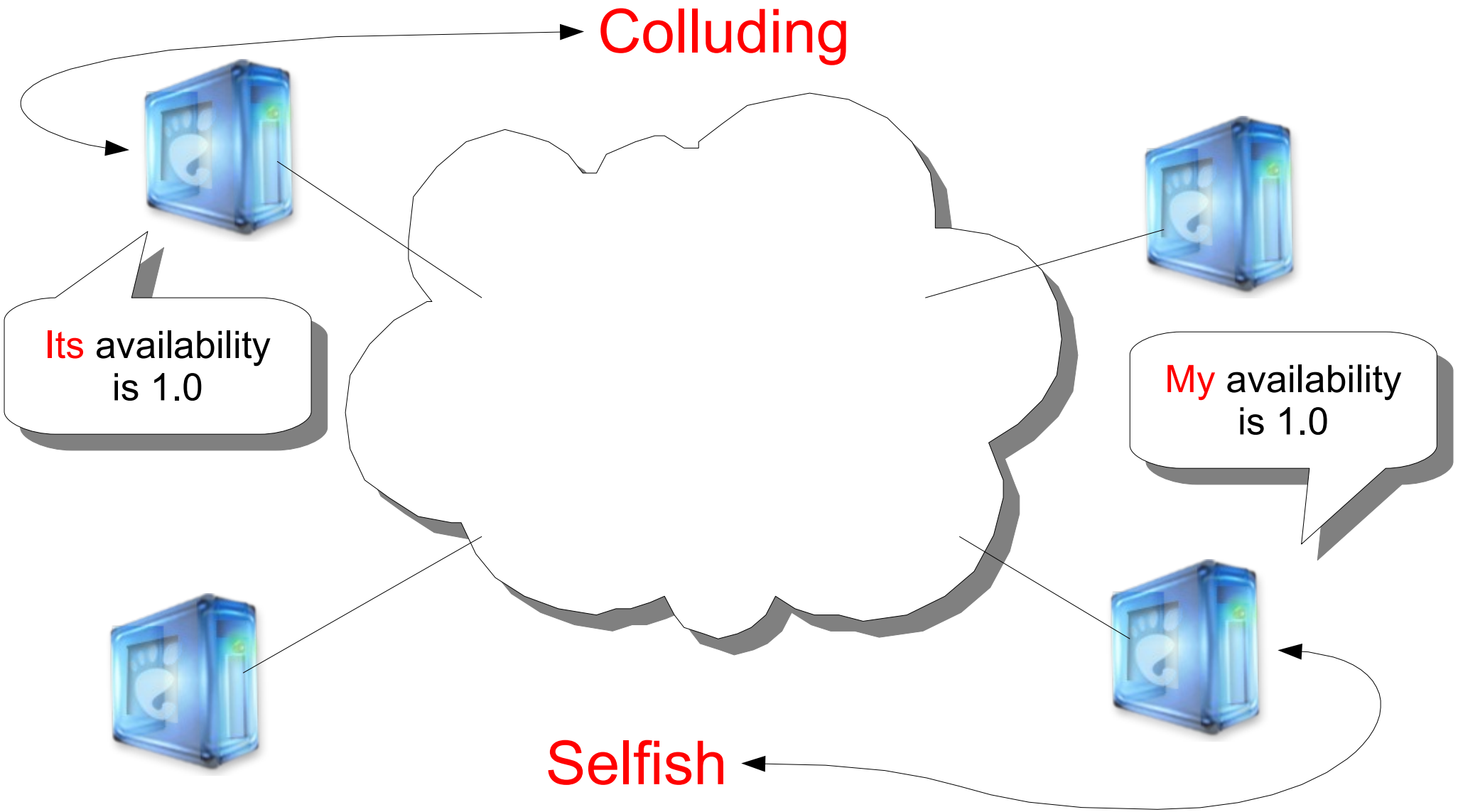
- *For each node* **x**, *select and discover a small subset of nodes to monitor* **x**.



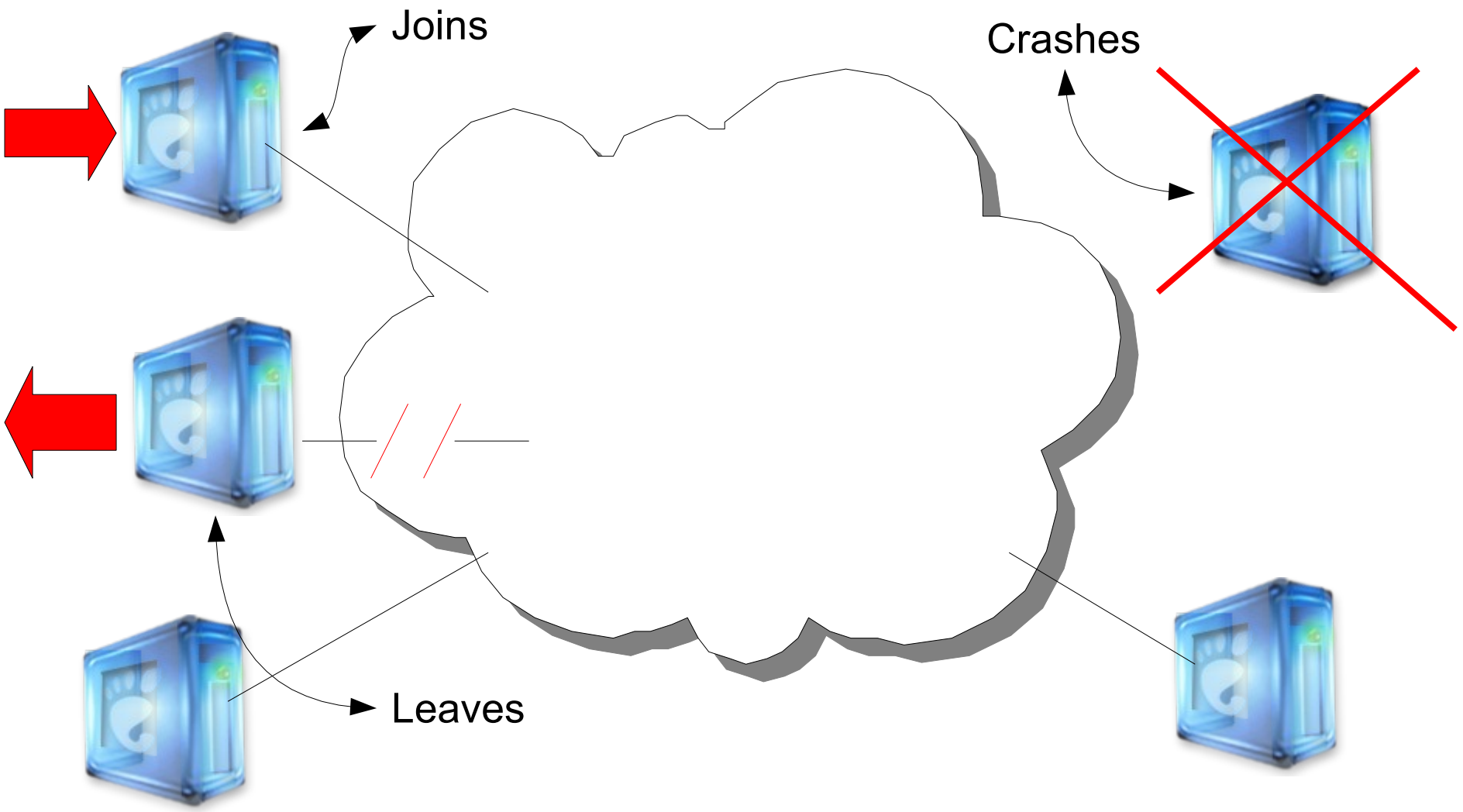
Outline

- Motivation
- Availability Monitoring Problem
- System Model
- Design Goals
- AVMON System
- Experimental Results

System Model



System Model (cont.)



System Model (cont.)

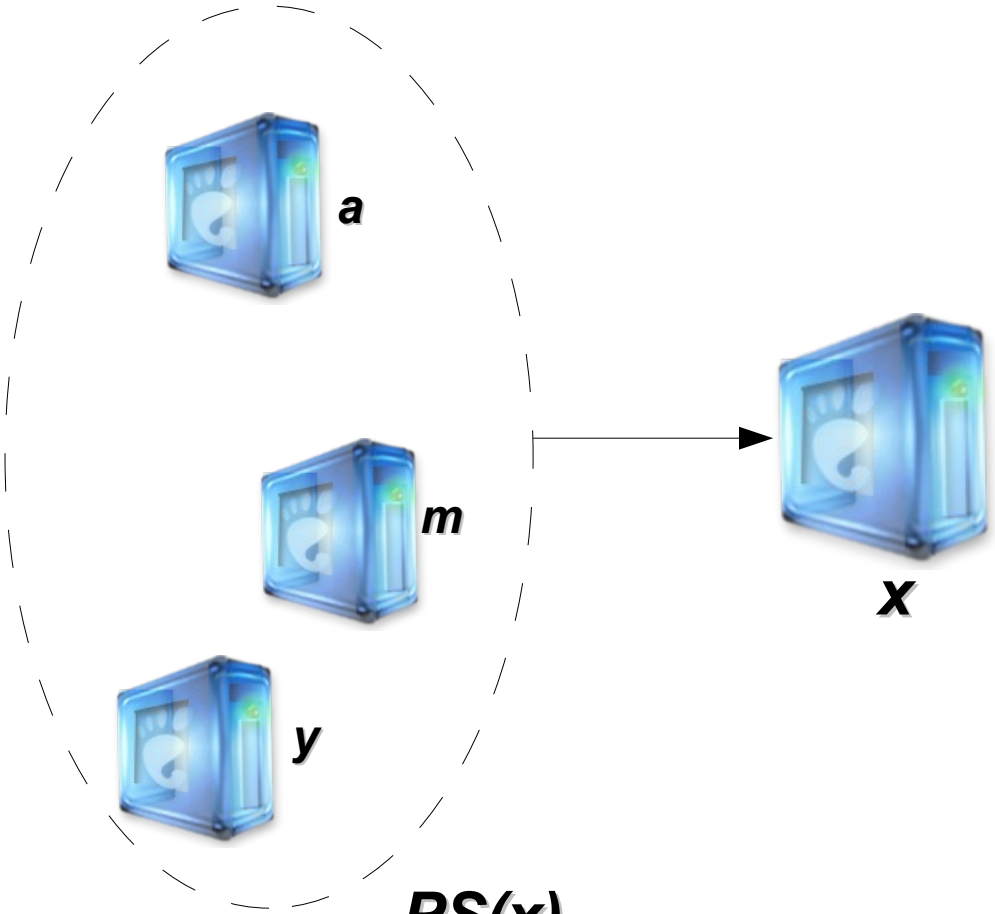
- Stable system size, ***N***.
  - The number of ***online*** nodes in the system varies within a constant factor of ***N***.
    - Valid for:
      - P2P systems [Bhagwan 2003]
      - PlanetLab based overlays
      - Grid systems

Outline

- Motivation
- Availability Monitoring Problem
- System Model
- Design Goals
- AVMON System
- Experimental Results

- *Consistency*, _____, _____, _____, _____, _____.

Design Goal: Consistency

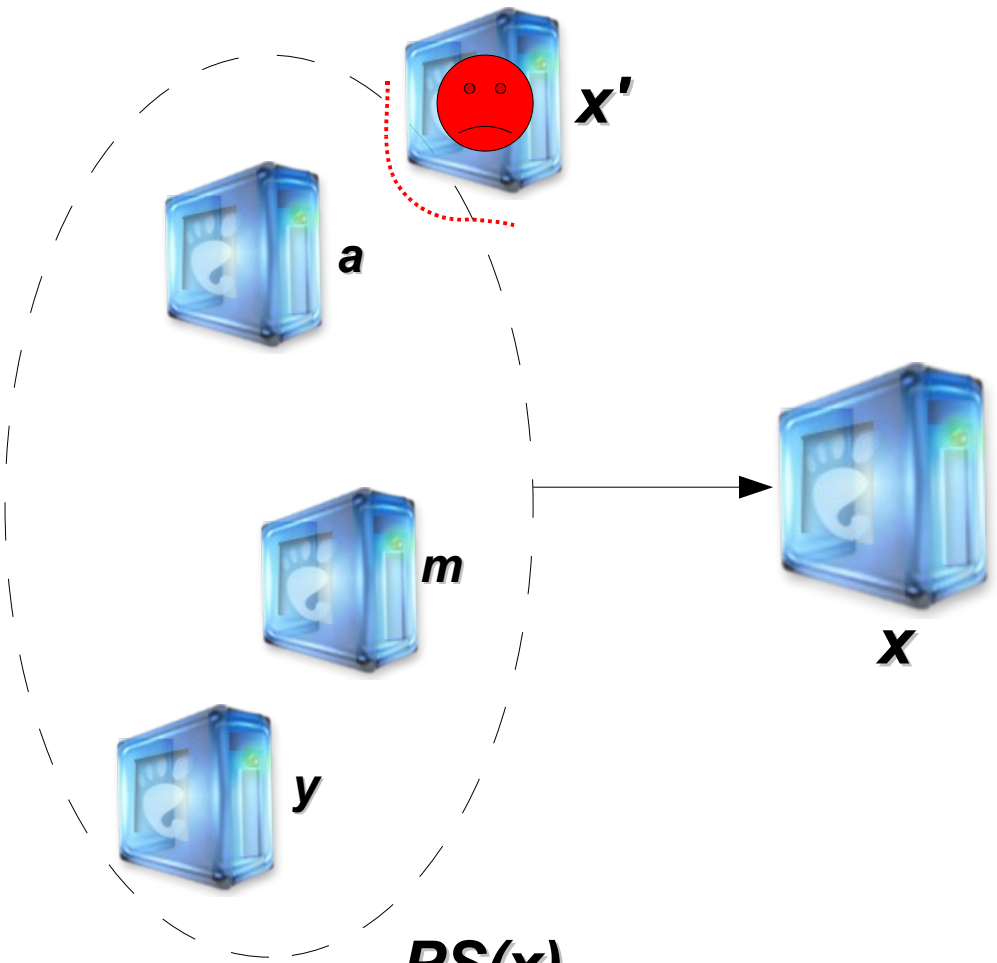


PS(x)
x's Pinging Set

- ***y ∈ PS(x)***, must be consistent, i.e.,
  - won't change regardless of system changes.

- *Consistency*, _____, _____, _____, _____, _____.

Design Goal: Consistency (cont.)

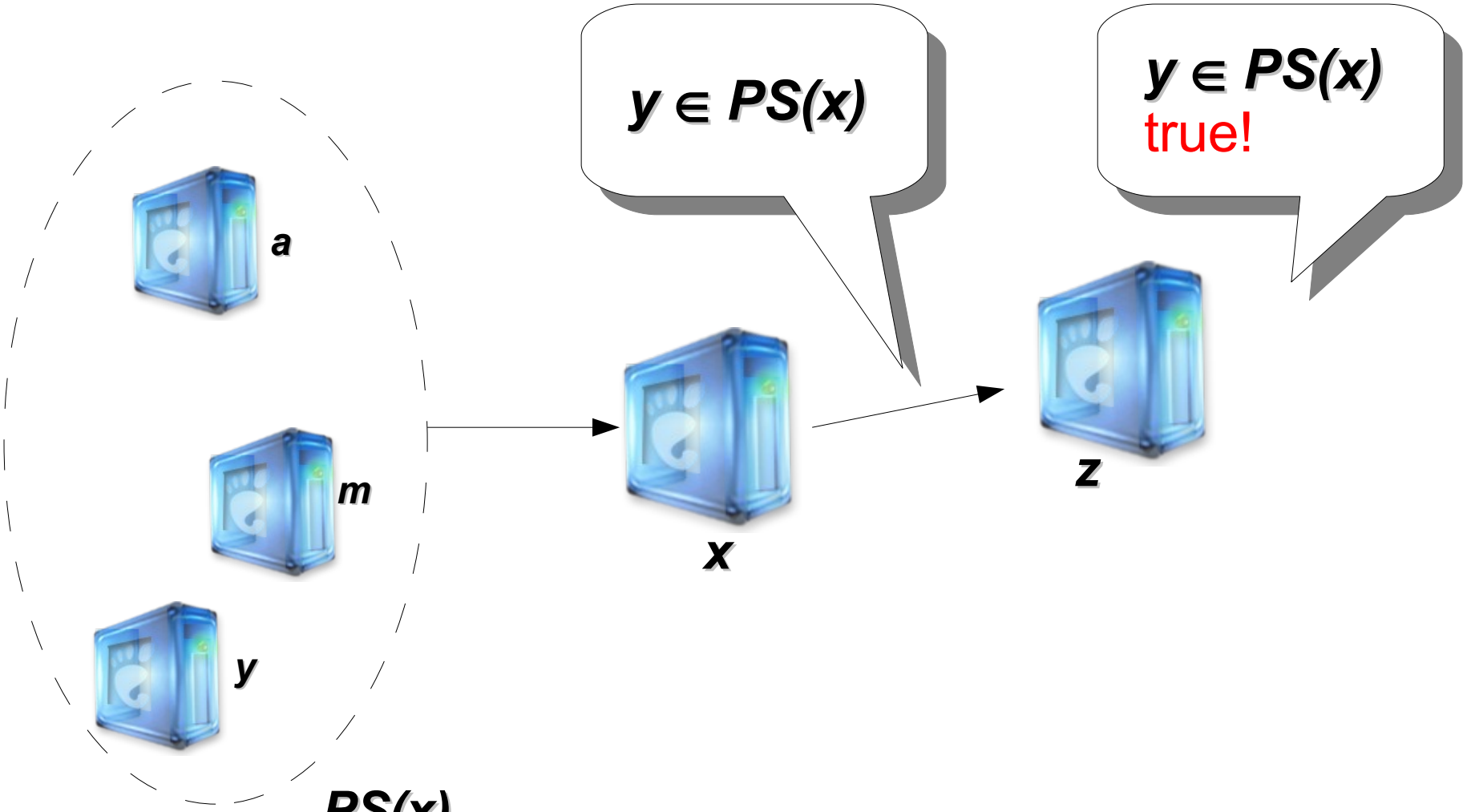


PS(x)
x's Pinging Set

- $y \in PS(x)$ must be consistent.
- Useful for
  - long term availability history maintenance,
  - no history transferral under churn,
  - avoid pollution with colluders.

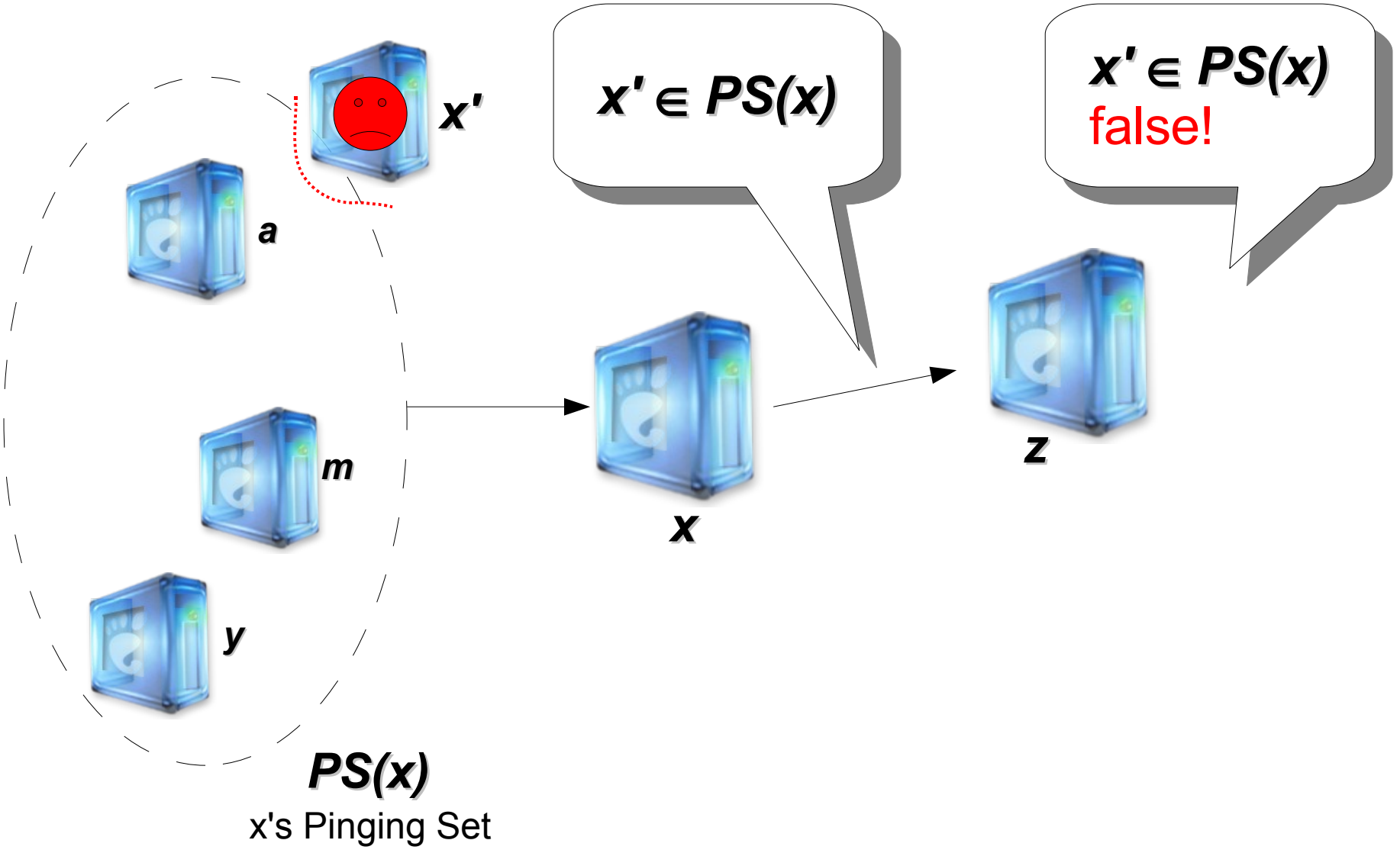
- *Consistency, Verifiability, _____, _____, _____, _____.*

Design Goal: Verifiability



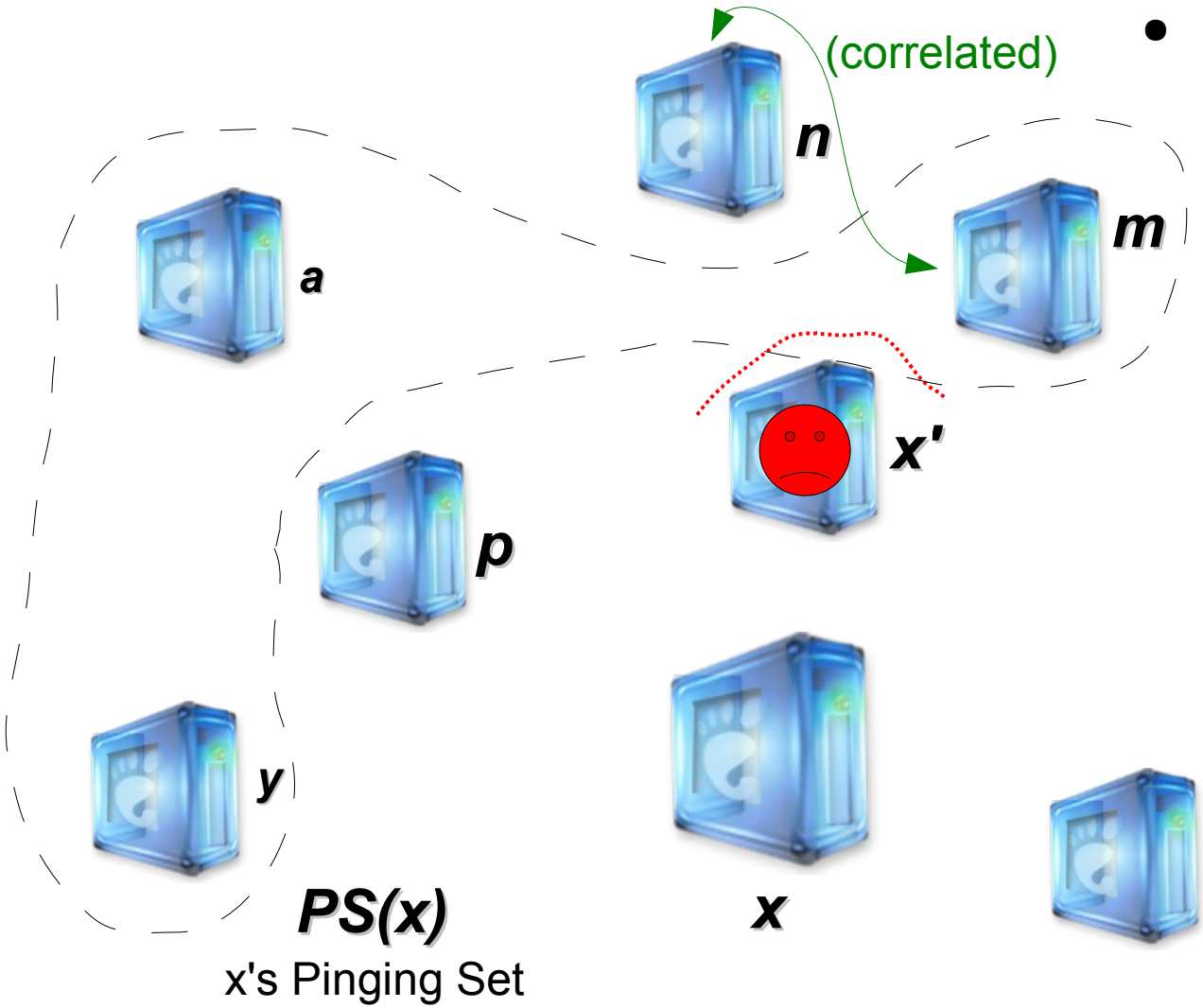
- *Consistency, Verifiability, _____, _____, _____, _____.*

Design Goal: Verifiability (cont.)



- *Consistency, Verifiability, Randomness, _____, _____, _____.*

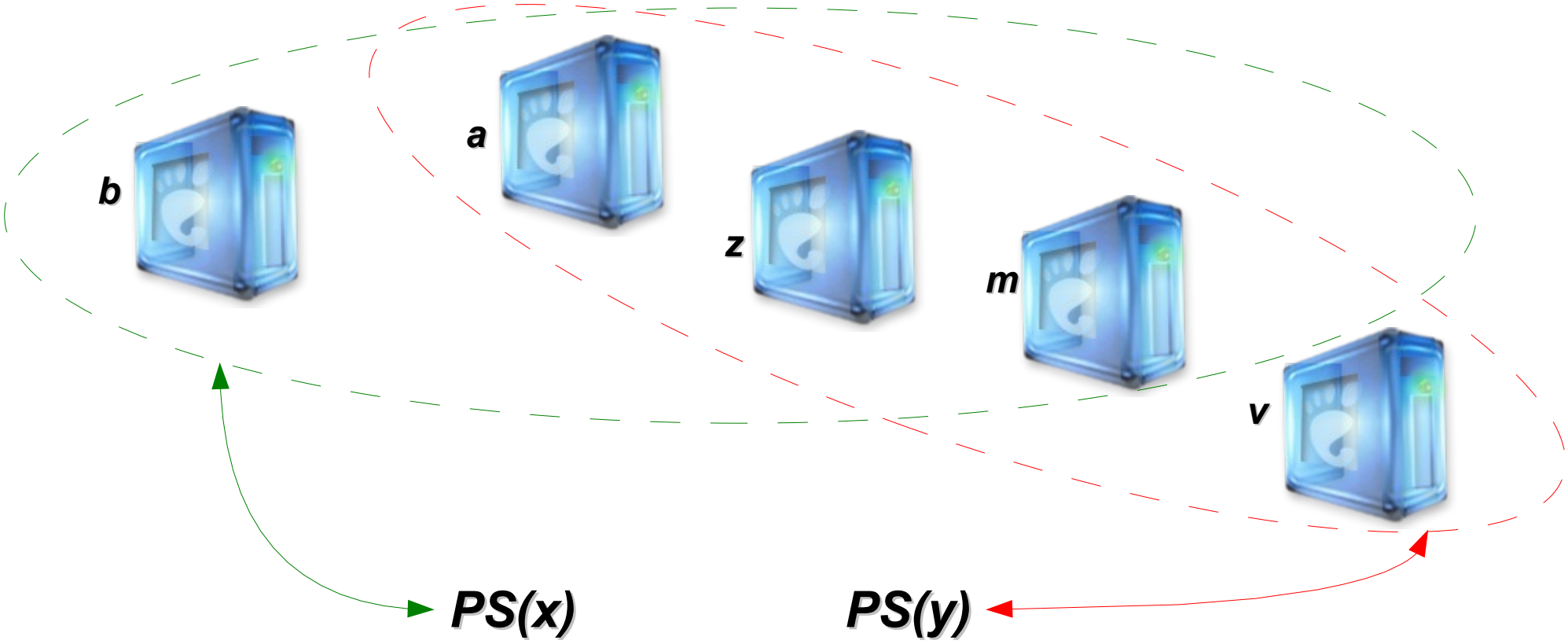
Design Goal: Randomness



- ***PS(x)*** populated uniformly at random,
  - identically distributed fashion, and
  - no correlation among nodes.

- *Consistency, Verifiability, Randomness, _____, _____, _____.*

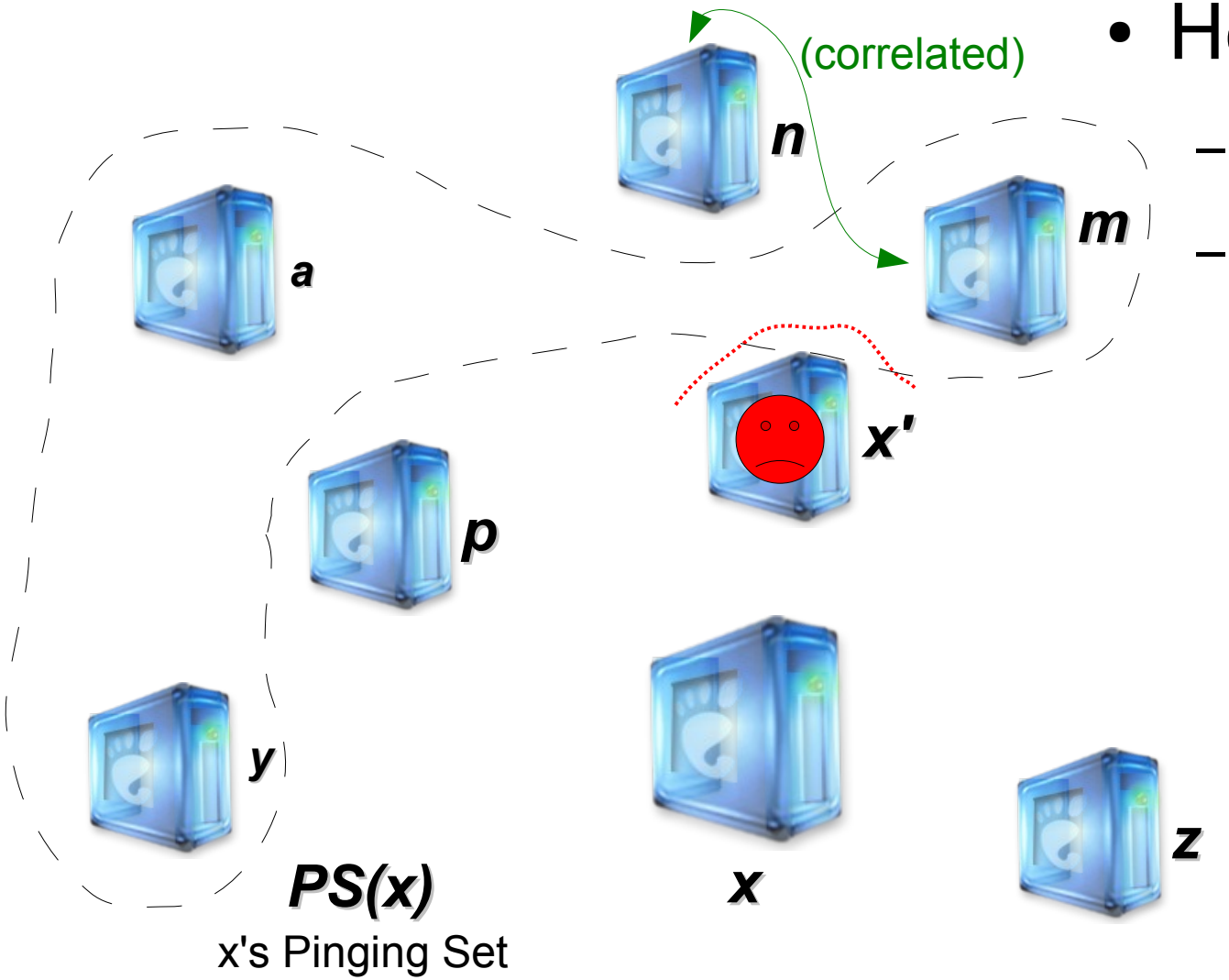
Design Goal: Randomness (cont.)



✗Correlated ping sets

- *Consistency, Verifiability, Randomness, _____, _____, _____.*

Design Goal: Randomness (cont.)

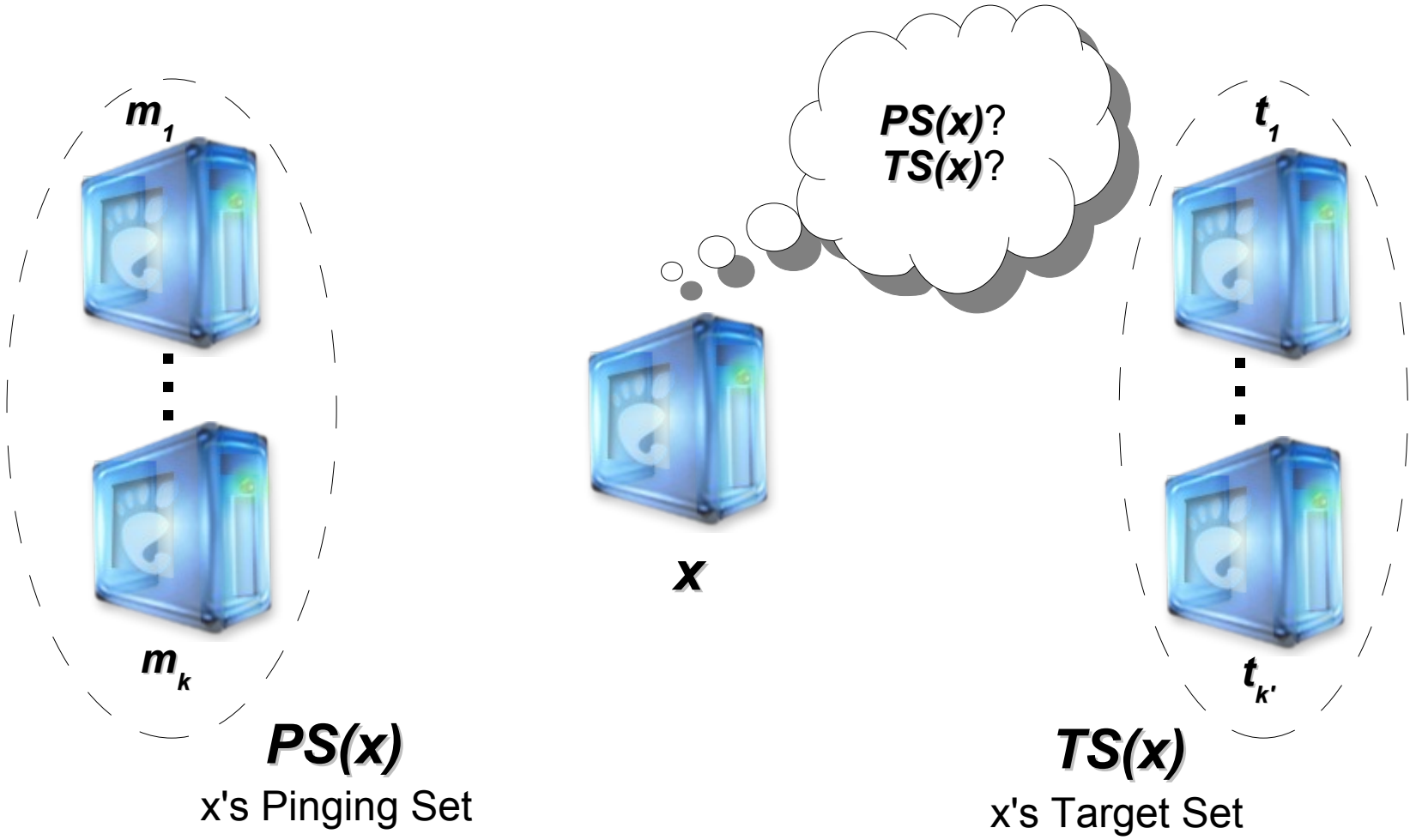


- Helps with
  - scalability,
  - load balancing.

- *Consistency, Verifiability, Randomness, Discoverability, _____, _____.*

Design Goal: Discoverability

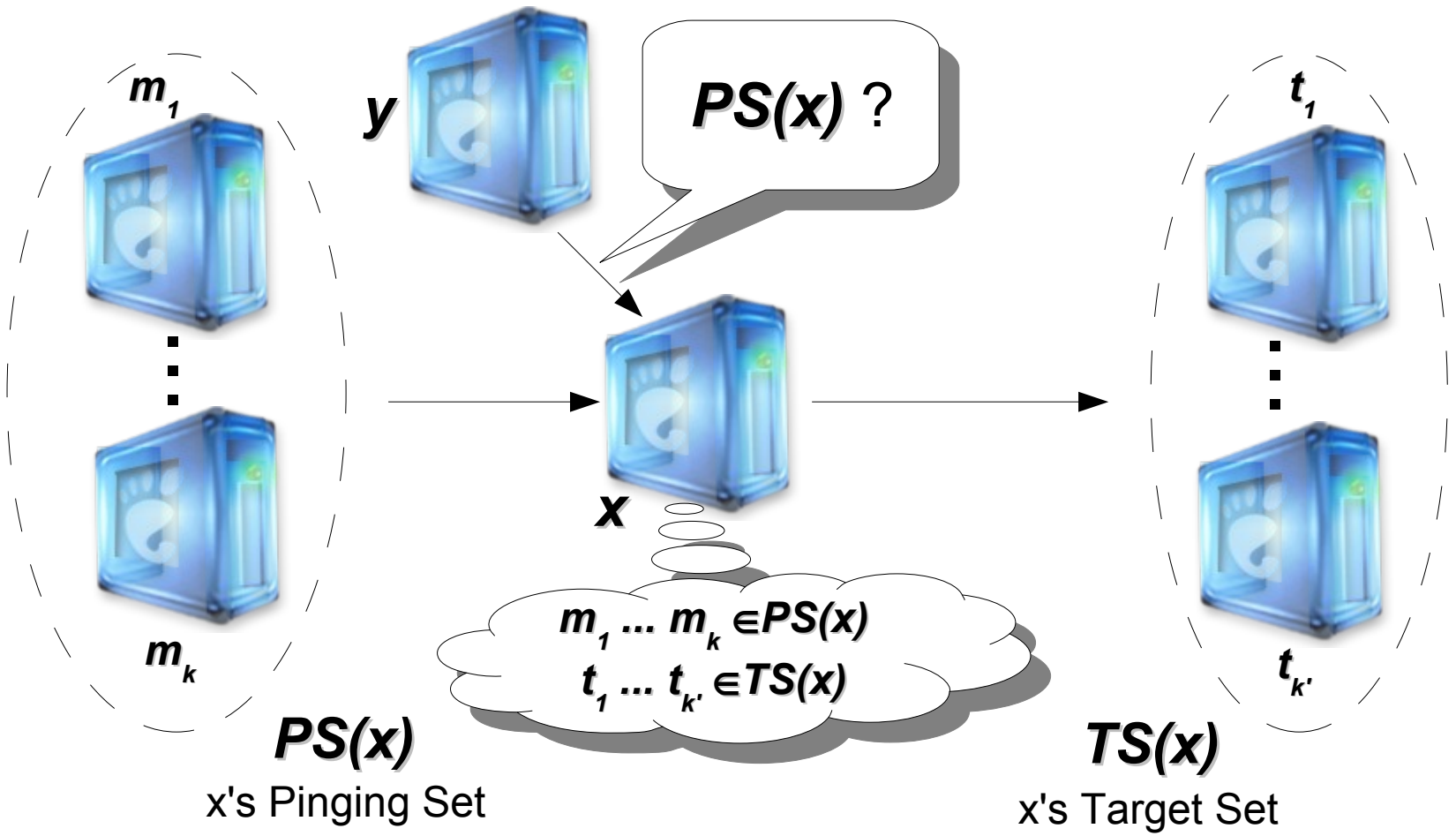
- ***x*** should learn quickly its ***PS(x)*** and ***TS(x)***



- *Consistency, Verifiability, Randomness, Discoverability, _____, _____.*

Design Goal: Discoverability (cont.)

- ***x*** should learn quickly its ***PS(x)*** and ***TS(x)***



- *Consistency, Verifiability, Randomness, Discoverability, Load Balancing, _____.*

Design Goal: Load Balancing

- Discovery overhead should be uniformly distributed.
  - Messages.
  - Memory.
  - Computation.

- *Consistency, Verifiability, Randomness, Discoverability, Load Balancing, Scalability.*

Design Goal: Scalability

- Discovery overhead should be low and scalable.
  - Messages.
  - Memory.
  - Computation.

Design Goals

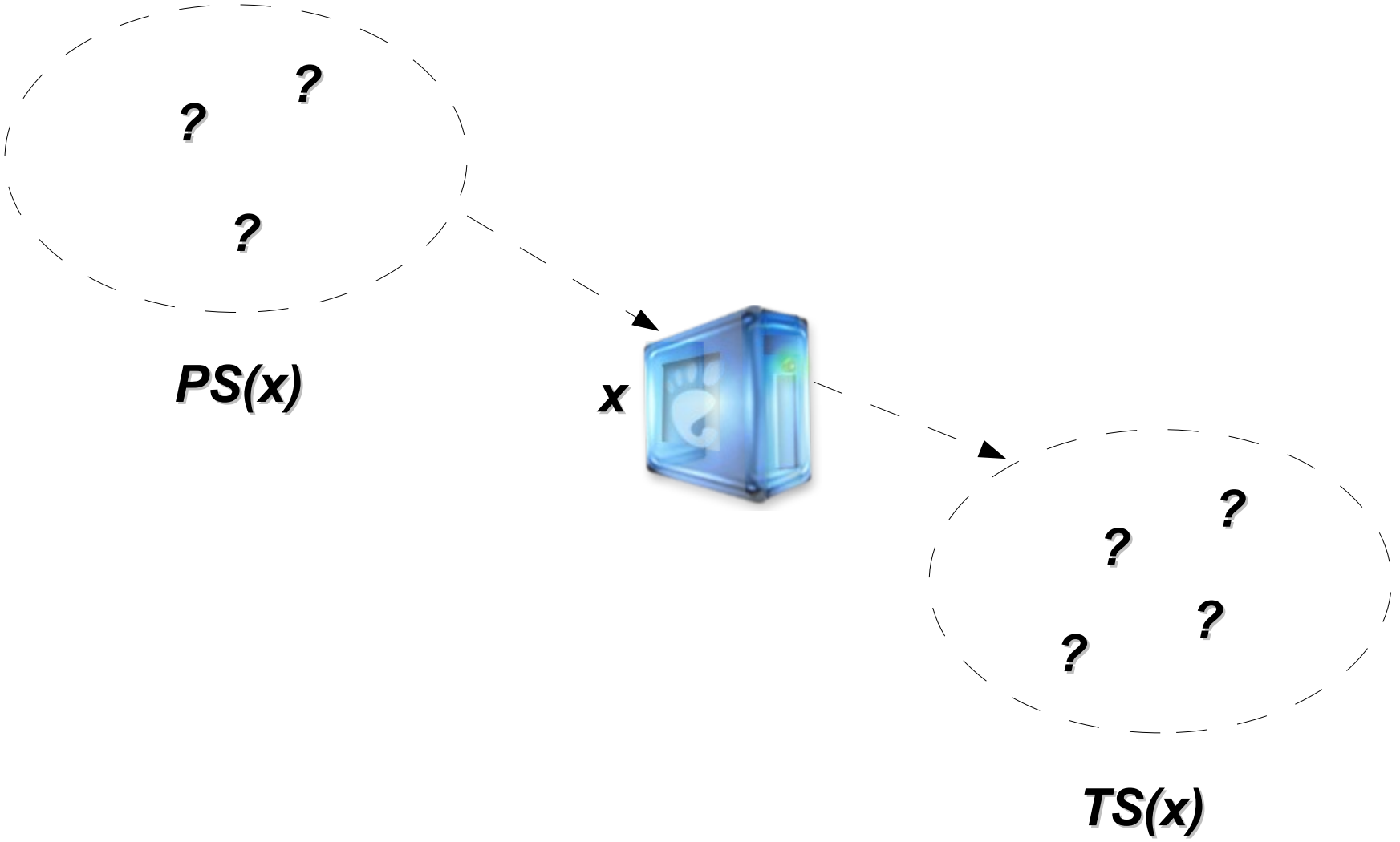
- Consistency
- Verifiability
- Randomness
- Discoverability
- Load Balancing
- Scalability

} Selection of PS(.) & TS(.)

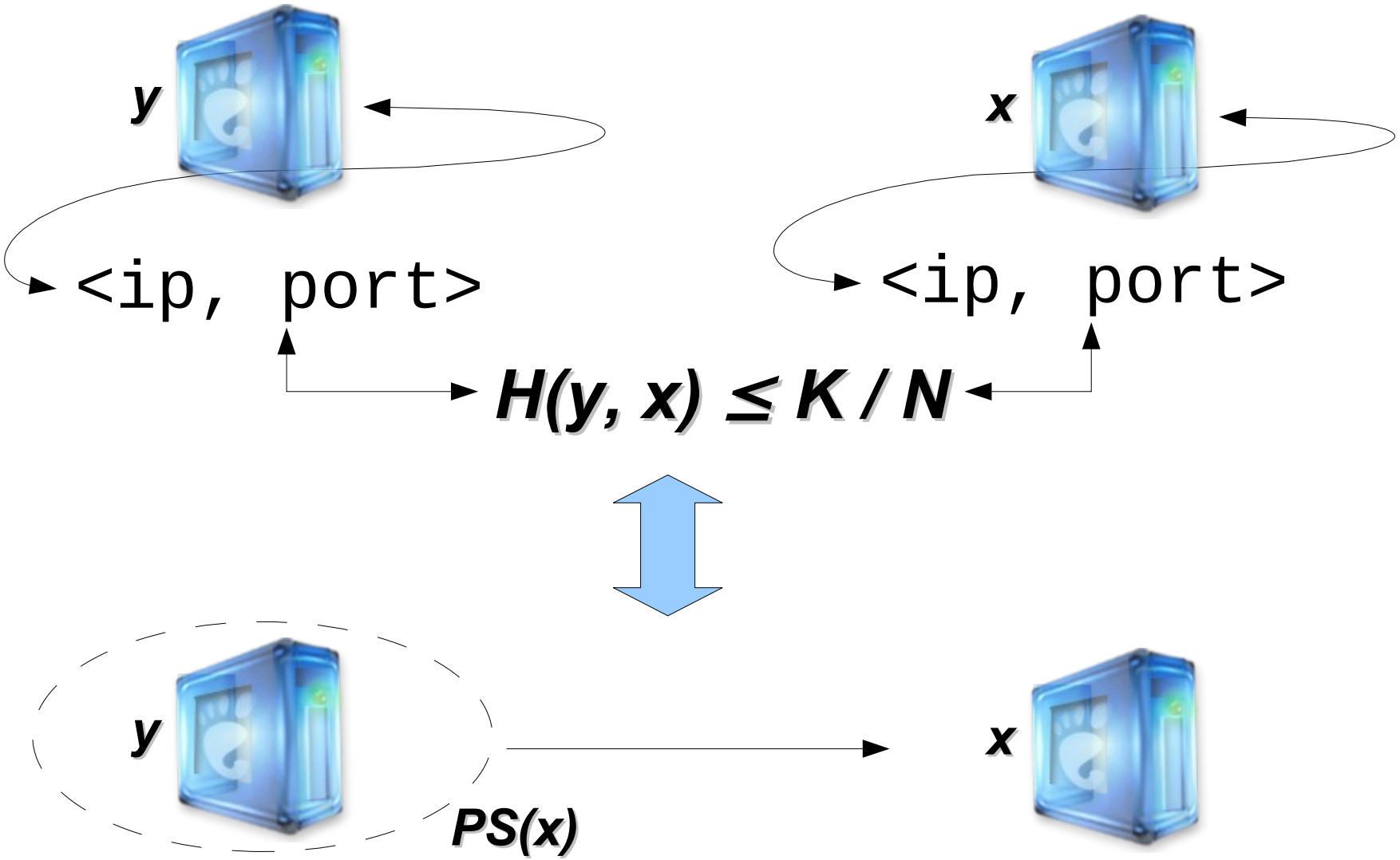
Outline

- Motivation
- Availability Monitoring Problem
- System Model
- Design Goals
- AVMON System
  - Selection and Discovery
- Experimental Results

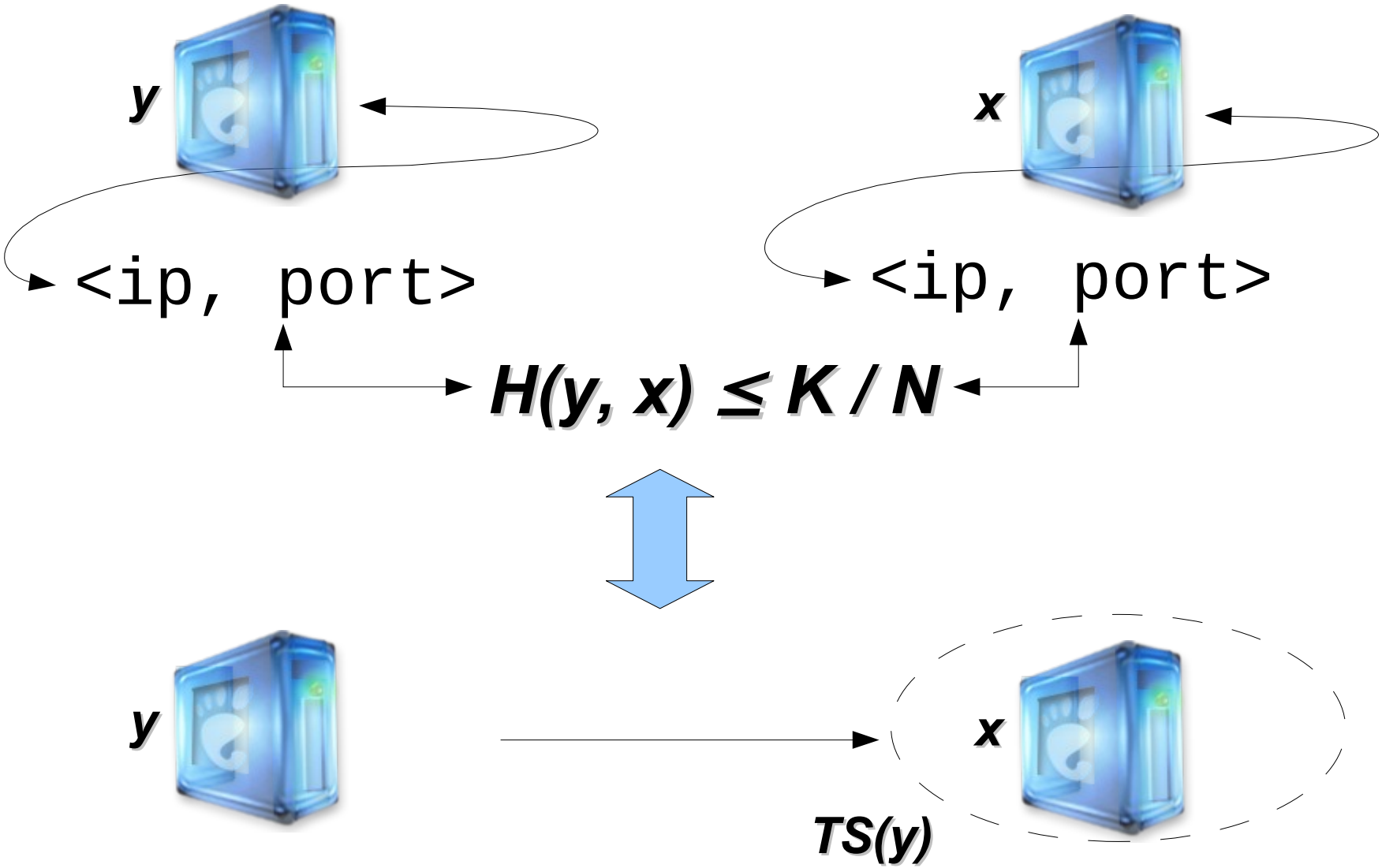
AVMON's Monitor Selection



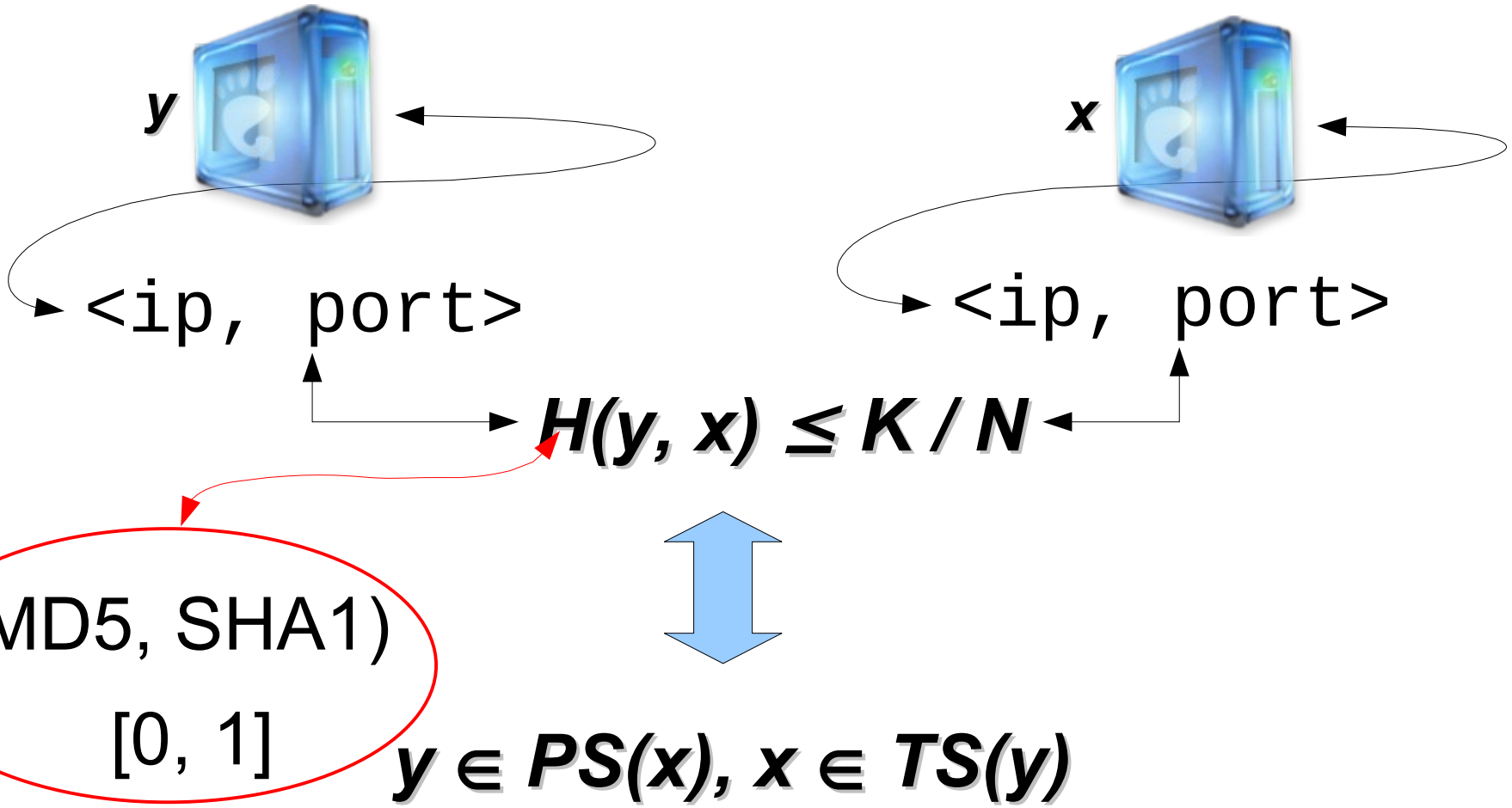
AVMON's Monitor Selection



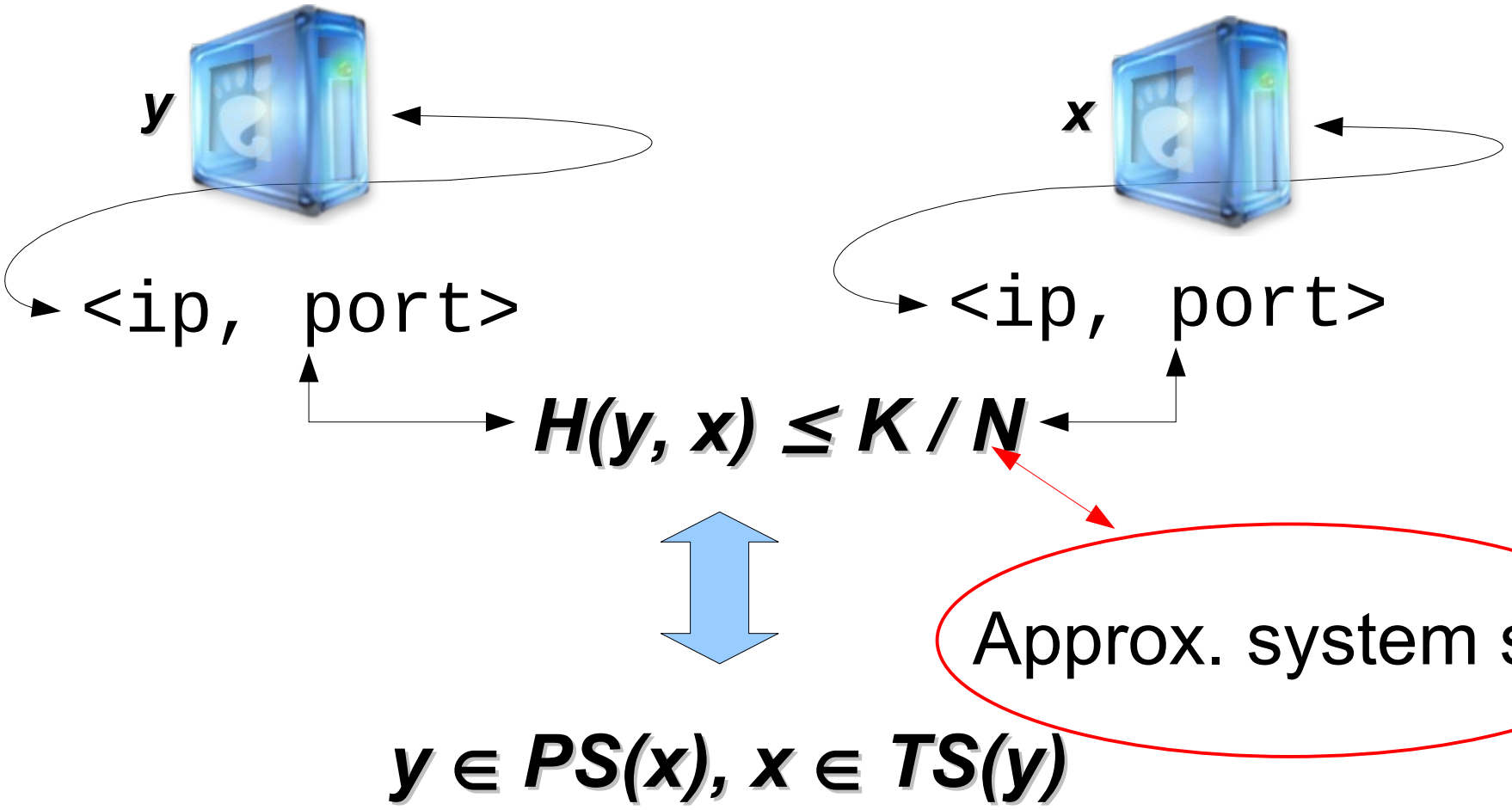
AVMON's Monitor Selection



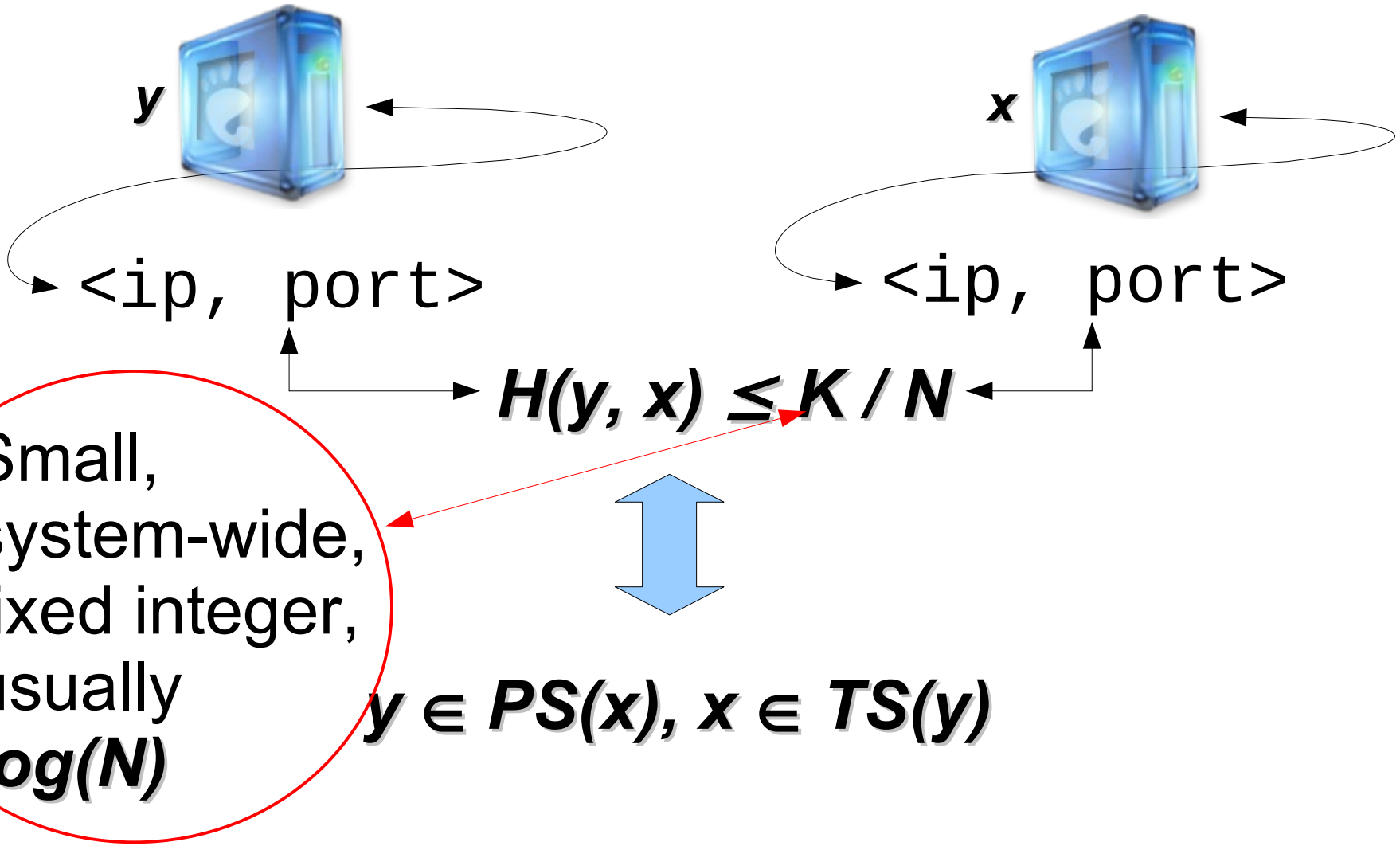
AVMON's Monitor Selection



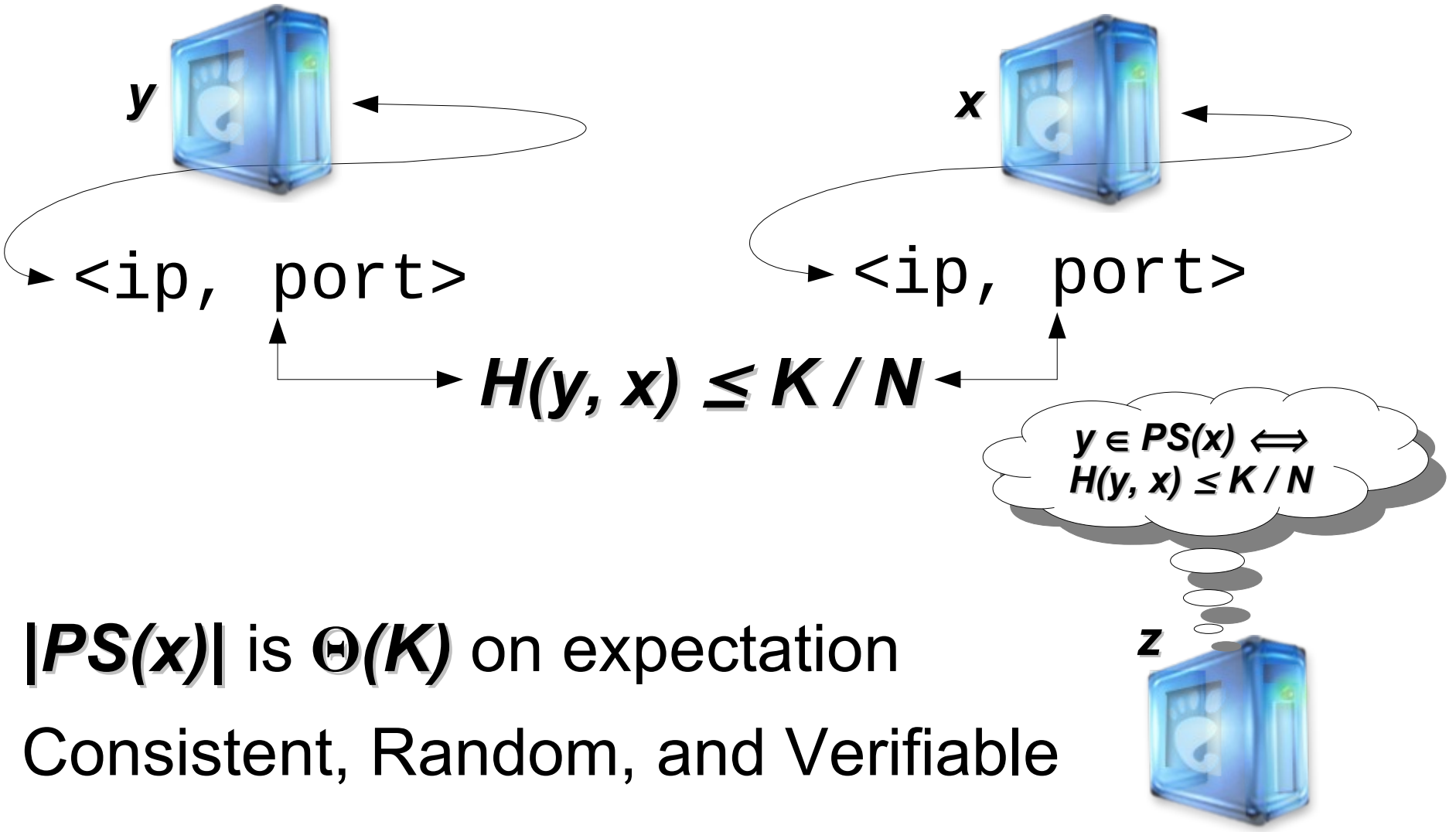
AVMON's Monitor Selection



AVMON's Monitor Selection

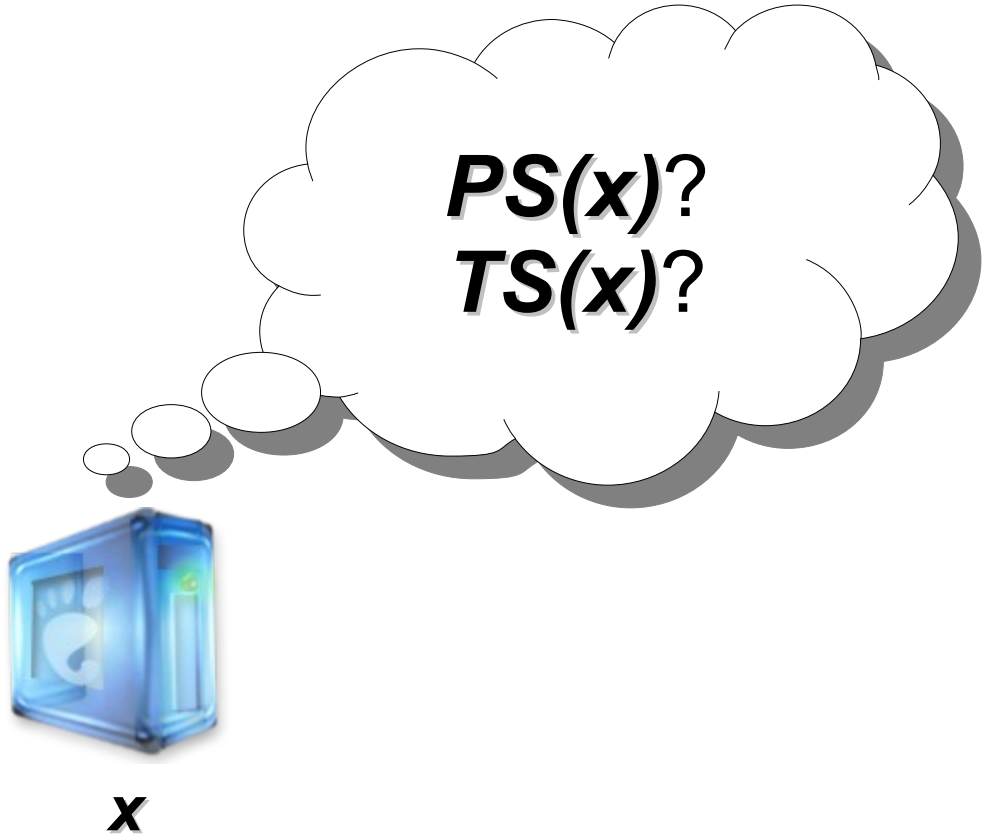


AVMON's Monitor Selection



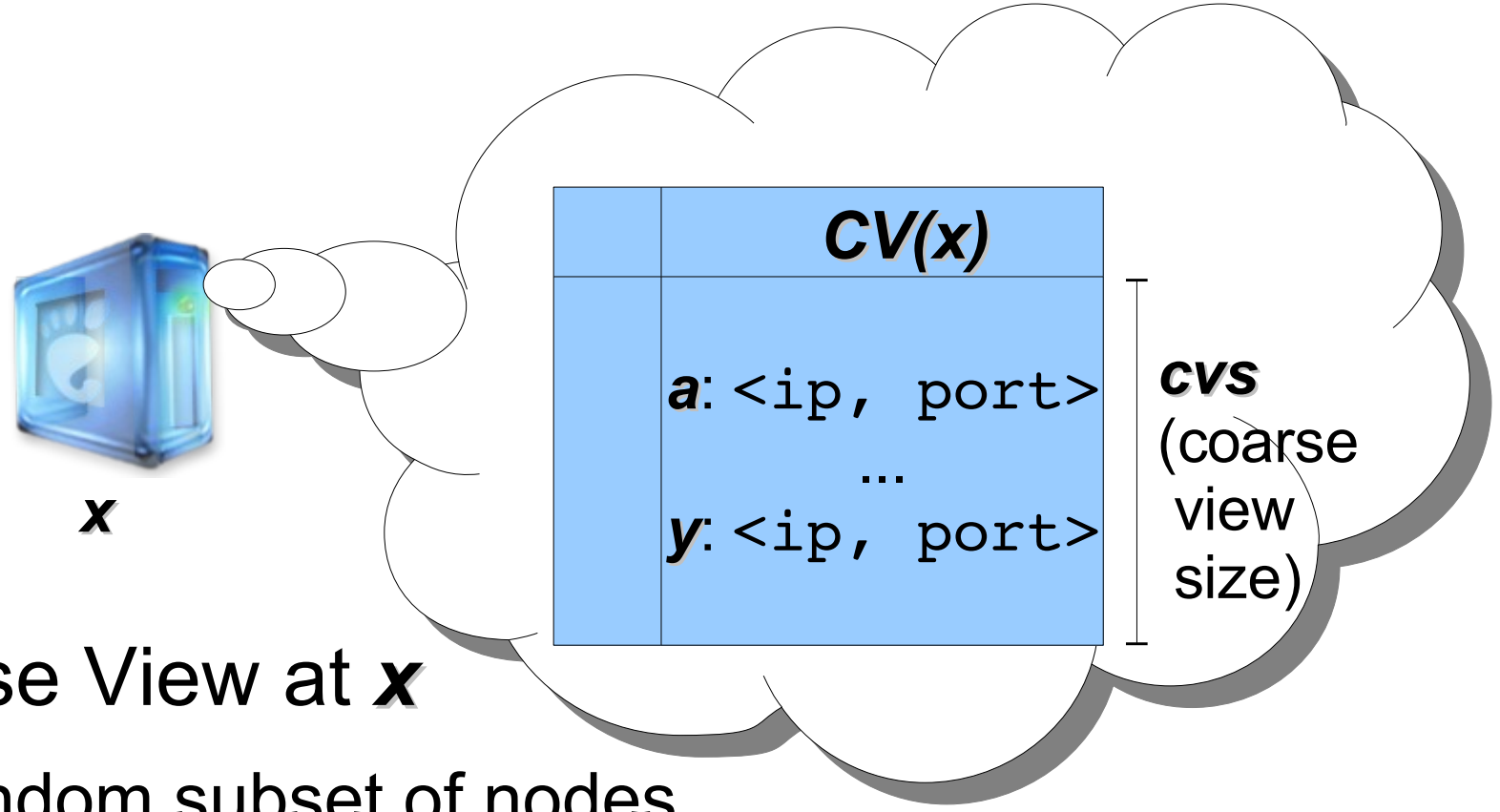
- $|PS(x)|$ is $\Theta(K)$ on expectation
- Consistent, Random, and Verifiable

AVMON's Monitor Discovery



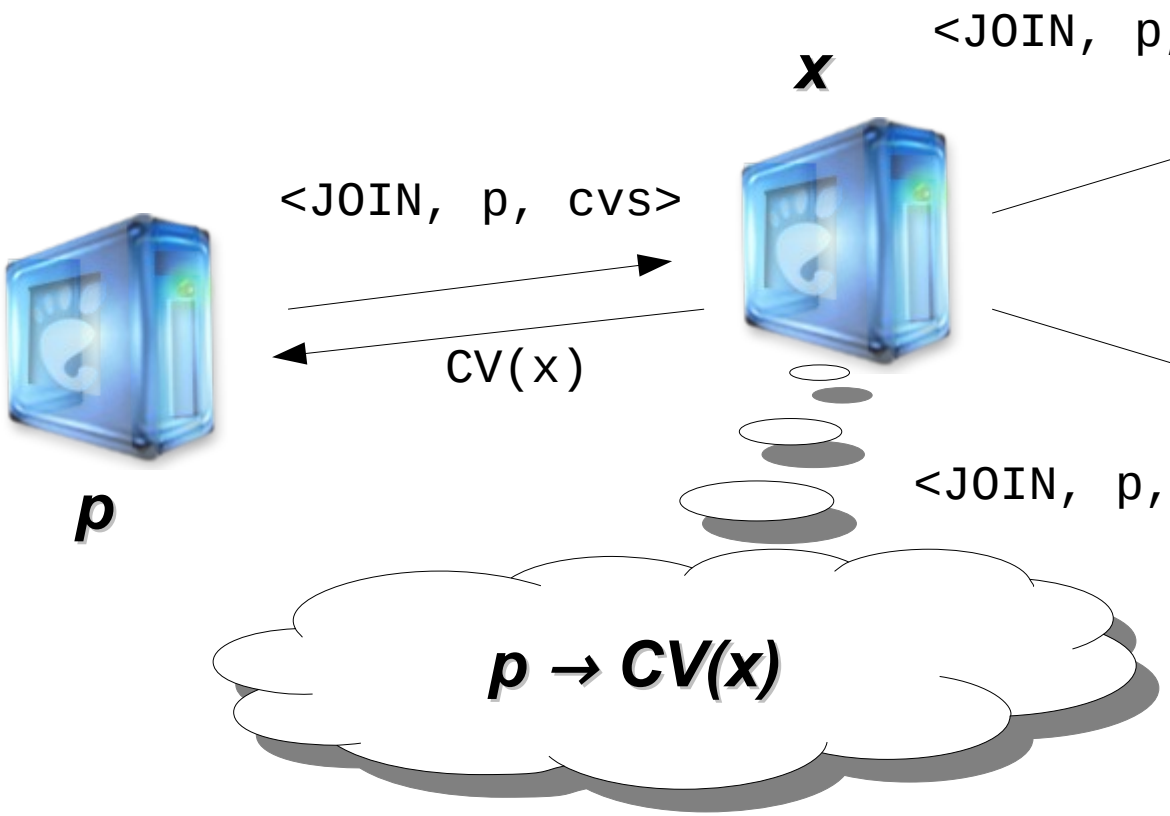
- For any monitor selection scheme that is *consistent* and *verifiable*

AVMON's Monitor Discovery (cont.)



- Coarse View at ***x***
  - Random subset of nodes
- Managed by
  - join subprotocol, and
  - maintenance and discovery subprotocol.

AVMON's Monitor Discovery: CV Join Subprotocol



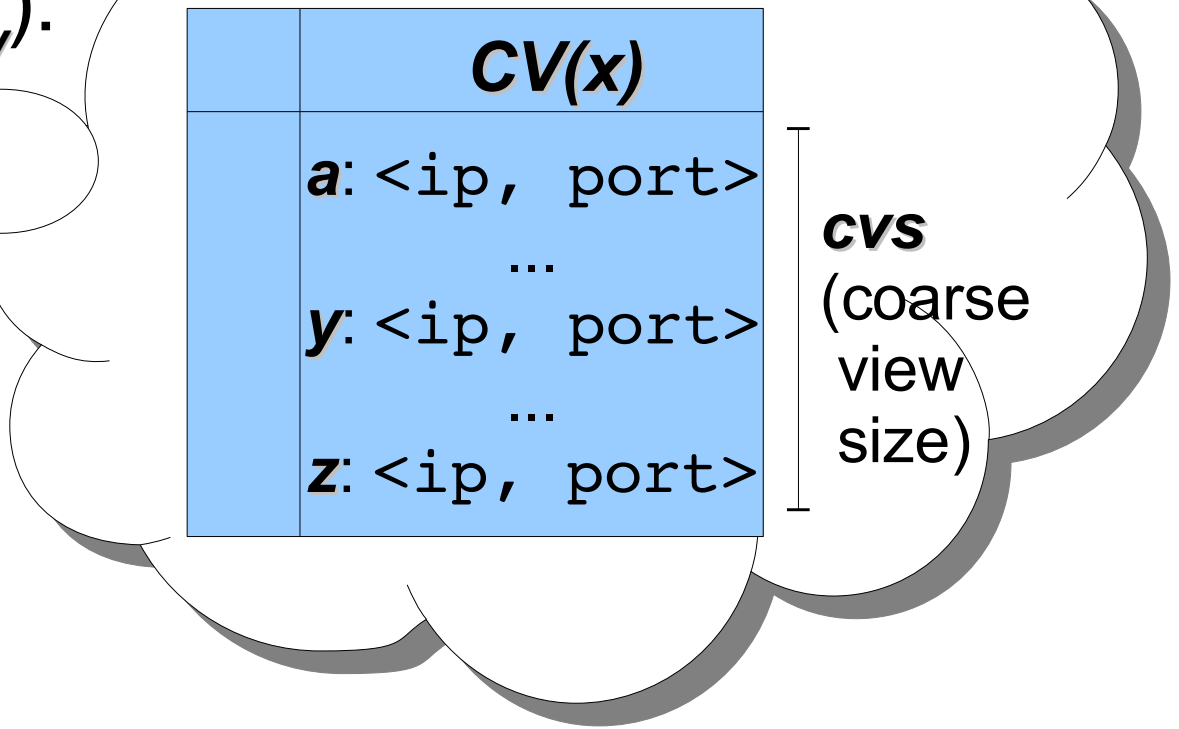
Random targets taken from ***CV(x)***

- ***p*** will be included in a total of ***cvs*** number of coarse views.

AVMON's Monitor Discovery: CV Maintenance and Discovery

- Periodically (T_{cv}):

x



<CV_PING>

(z)

AVMON's Monitor Discovery: CV Maintenance and Discovery

- Periodically (T_{cv}):

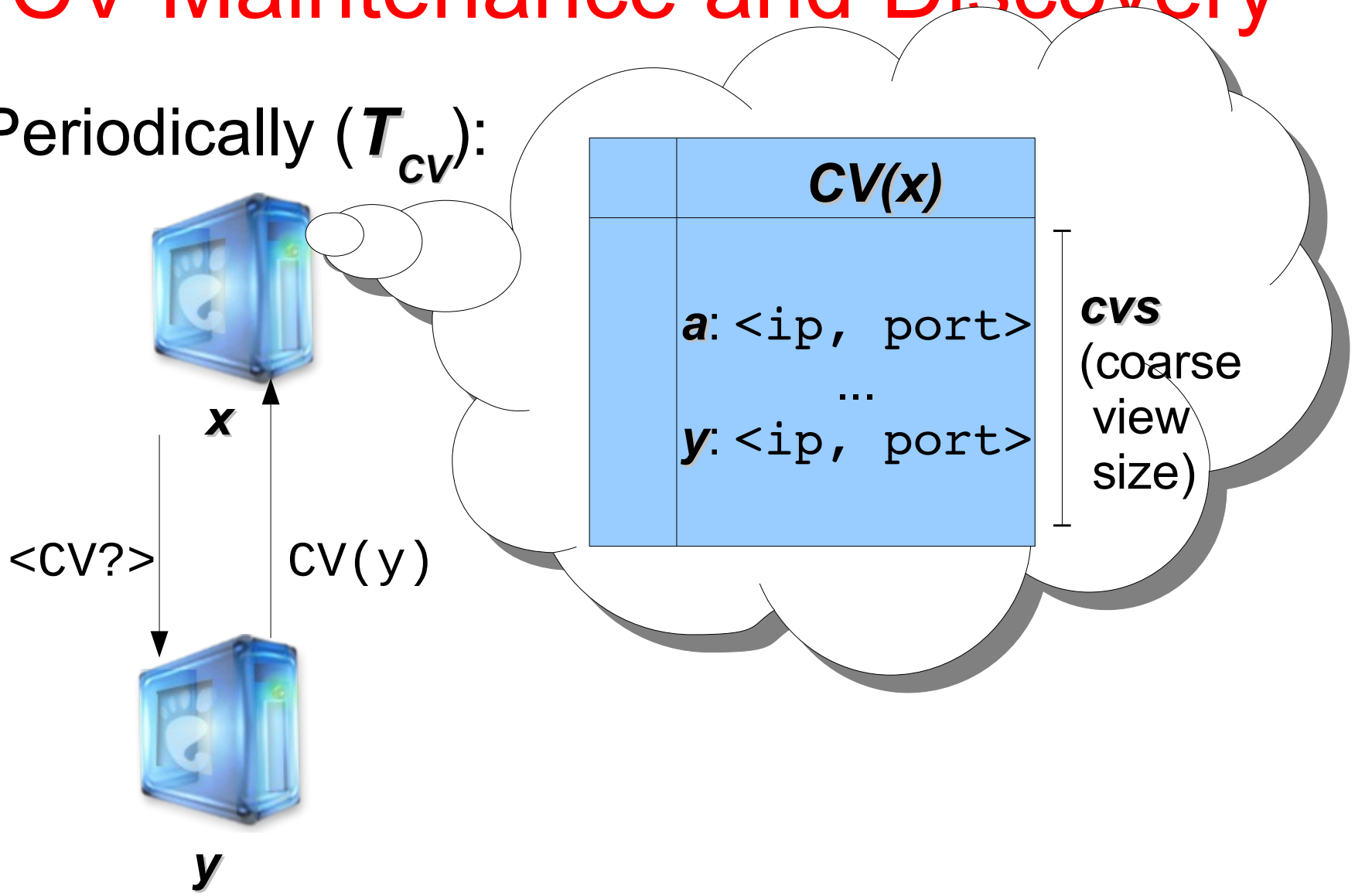
x

CV(x)
a: <ip, port>
…
y: <ip, port>
…
~~***z***: <ip, port>~~

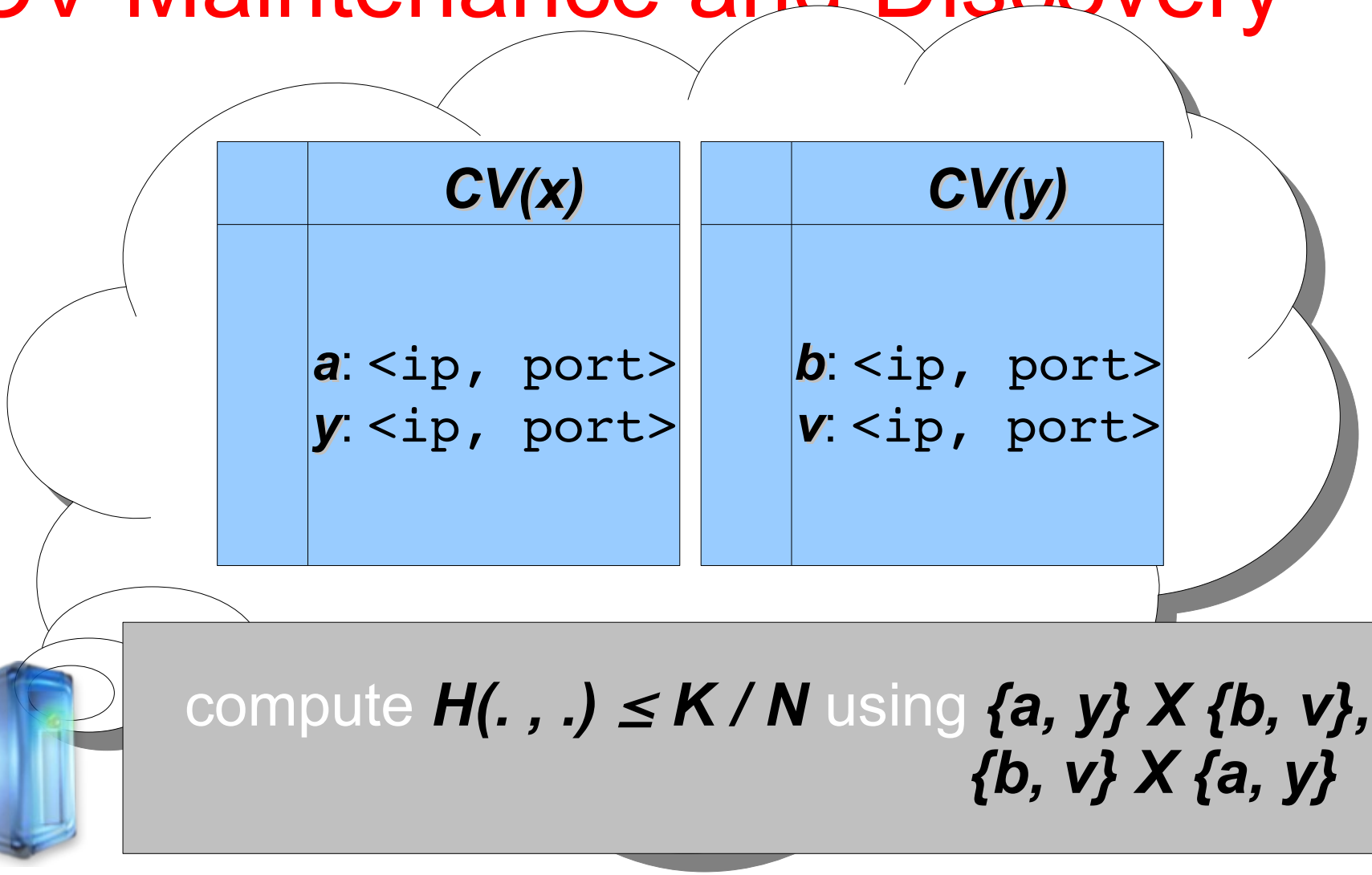
cvs (coarse view size)

AVMON's Monitor Discovery: CV Maintenance and Discovery

- Periodically (T_{cv}):

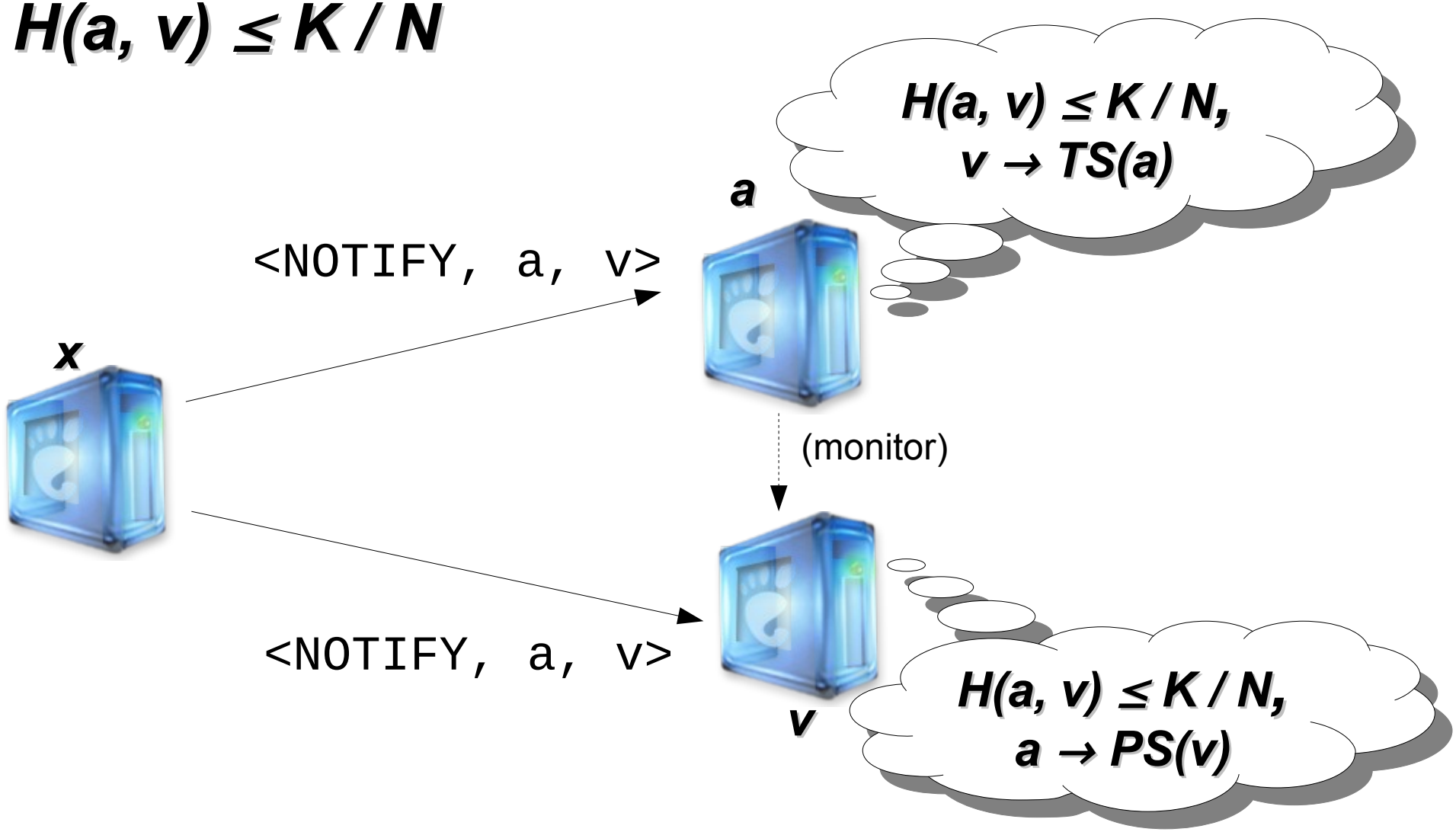


AVMON's Monitor Discovery:
CV Maintenance and Discovery

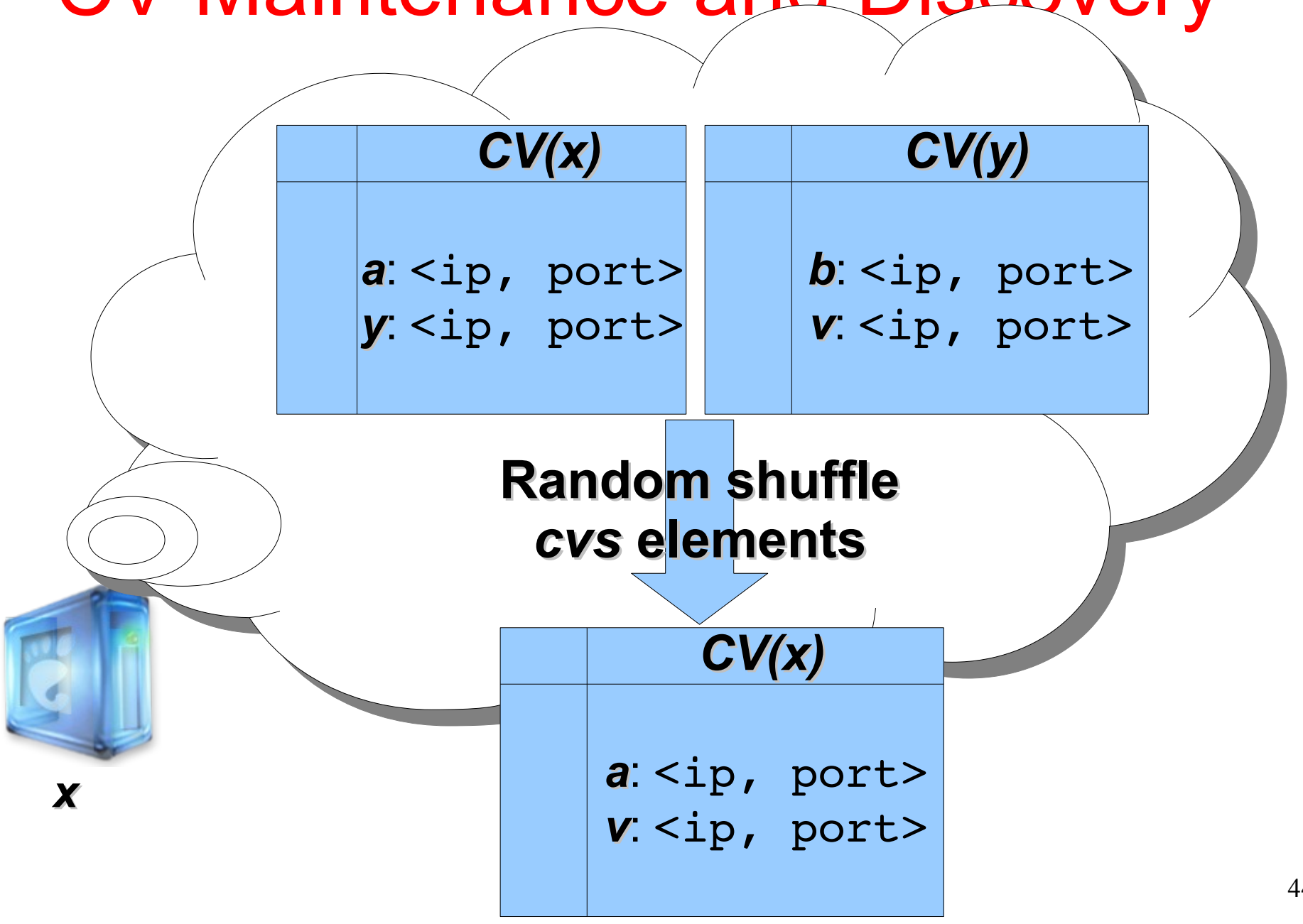


AVMON's Monitor Discovery: CV Maintenance and Discovery

- ***H(a, v) ≤ K / N***



AVMON's Monitor Discovery:
CV Maintenance and Discovery



Outline

- Motivation
- Availability Monitoring Problem
- System Model
- Design Goals
- AVMON System
  - Analysis
- Experimental Results

Optimal Memory, Discovery, Computation

- Minimize: f(cvs) = M + C + E[D]
- f(cvs) = cvs + cvs^2 + N / cvs^2

(Derived in the paper)

- d(f(cvs))/d(cvs) = 1 + 2cvs - 2N / cvs^3 = 0
- ***cvs*** for Optimal MDC ≈ $N^{1/4}$, e.g.,
  - ***N*** = 1 Million
  - ***cvs*** = 32
  - ***CV(.)*** is 192 Bytes
  - ***K*** = $\log_2(N)$ = 20
  - Computation overhead (***cvs^2***), 0.384ms (per min. on a PC)
  - Discover 1 monitor every 20 rounds on average

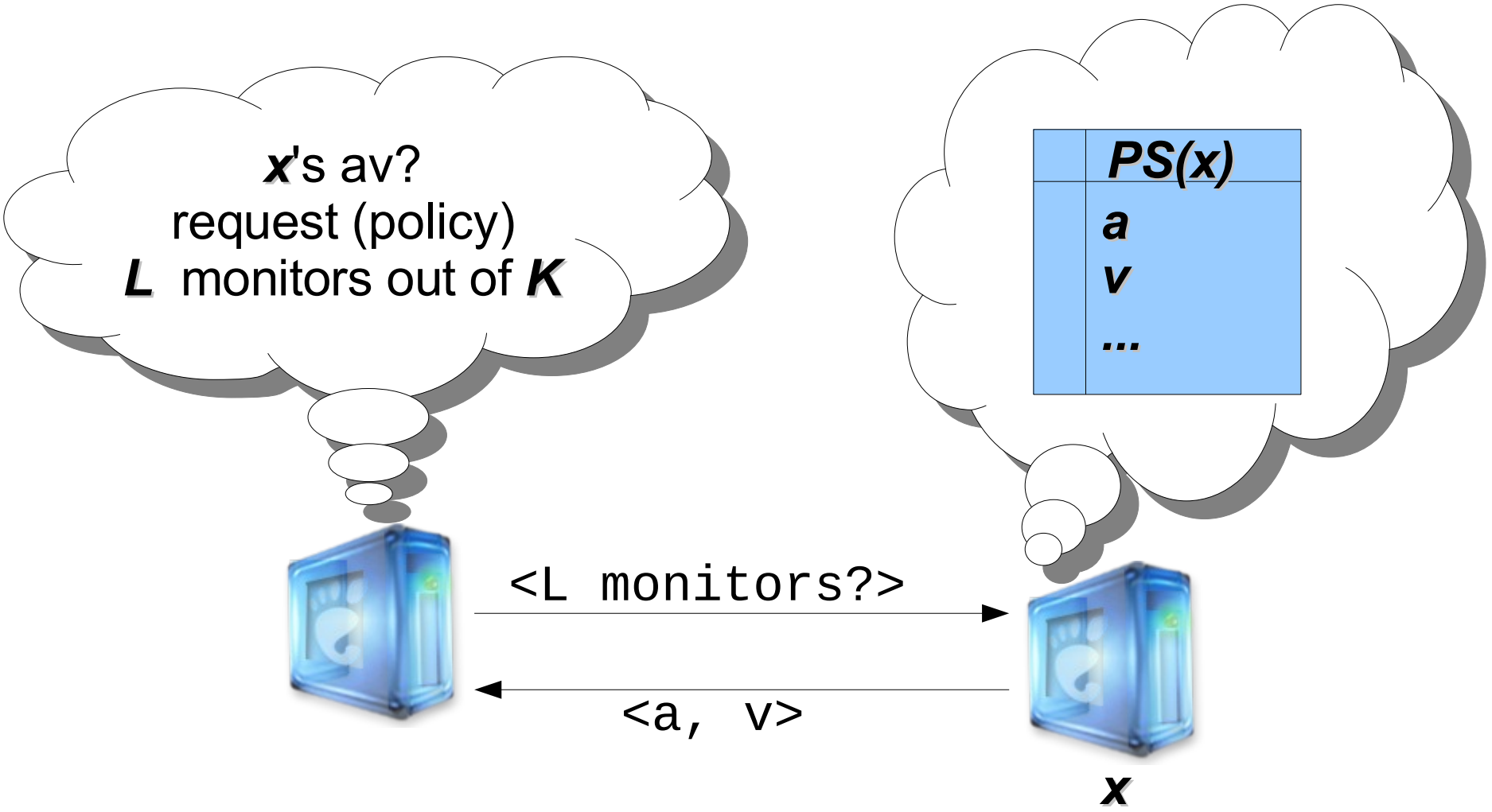
Collusion Resilience

- ***x*** colludes with C nodes
  - C=o(N / log(N))
- ***K*** = Θ(log(N))
- Probability no such colluders appear in ***PS(x)***
  - $(1 - K / N)^C \cong (1 - CK / N) \rightarrow 1$ as $N \rightarrow \infty$

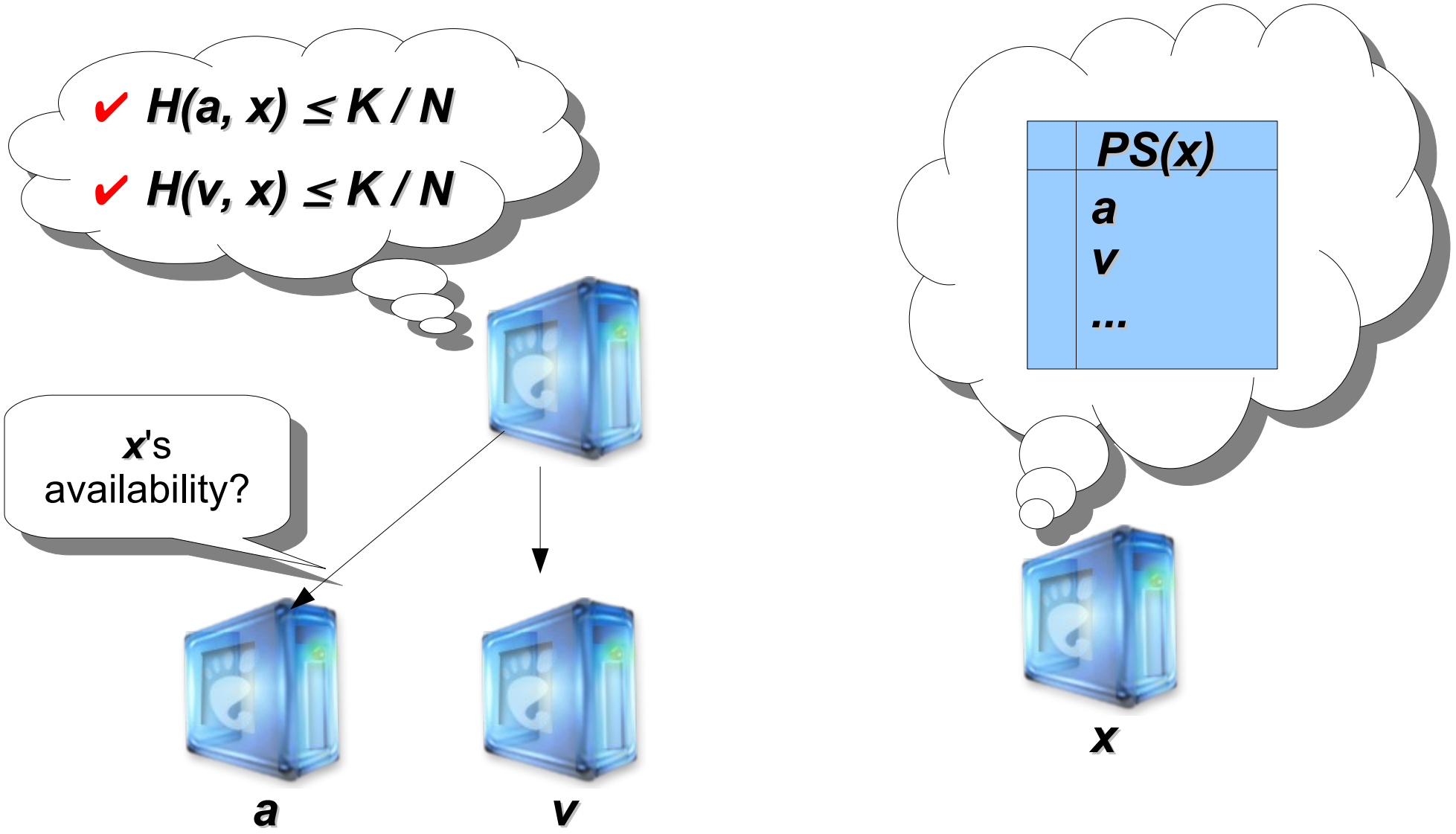
Outline

- Motivation
- Availability Monitoring Problem
- System Model
- Design Goals
- AVMON System
  - Using AVMON
- Experimental Results

Using AVMON
Querying Availability

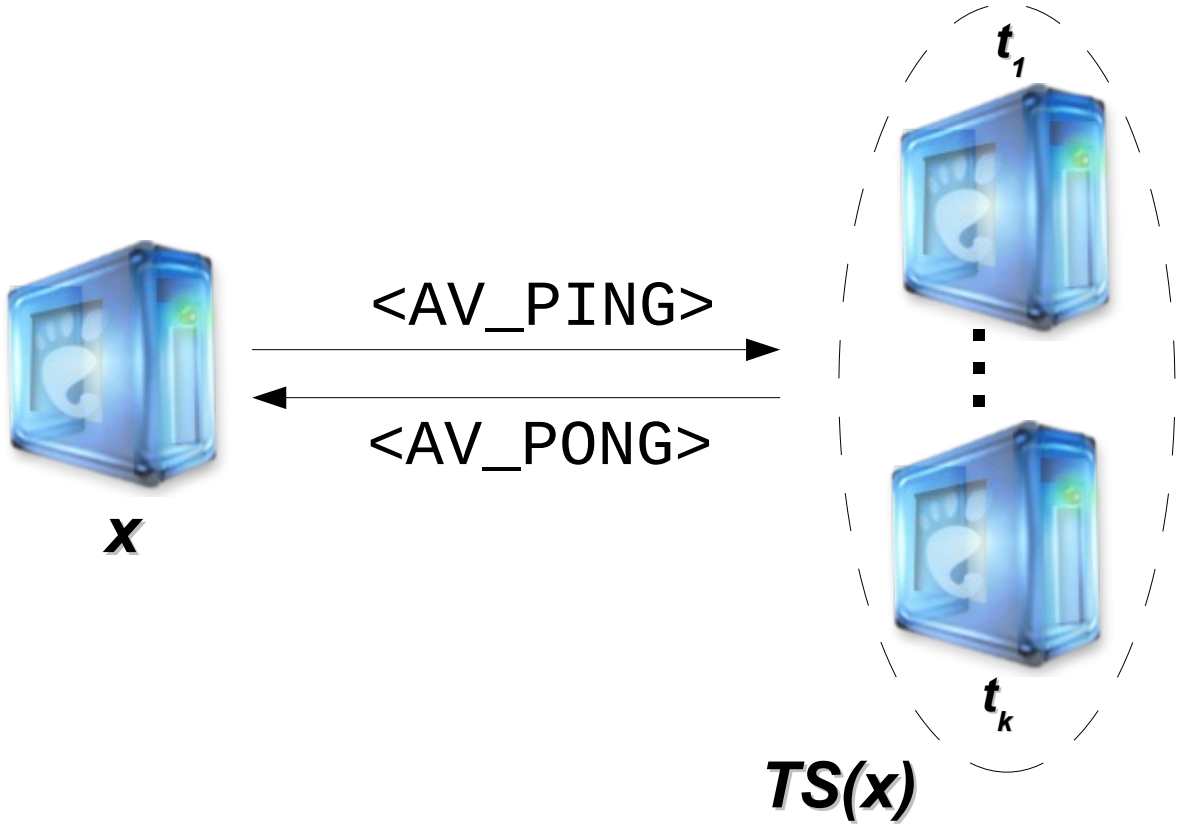


Using **AVMON**
Querying Availability



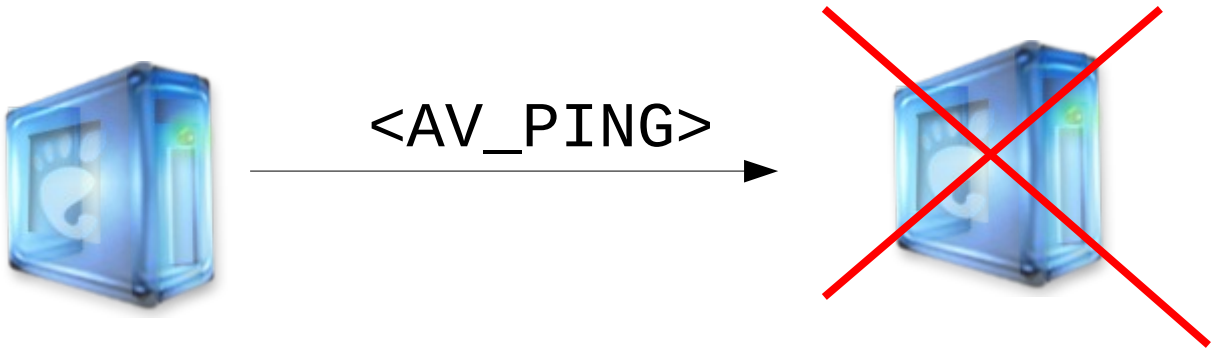
Using AVMON
Monitoring Availability

- Periodically (T_{mon}):



Monitoring Optimization: Forgetful Pinging

- Periodically (T_{mon}):
- $t_{unresponsive} > \tau$
  - Ping with probability $(c^* t_{lastUptime})/(t_{lastUptime}+t_{unresponsive})$



Outline

- Motivation
- Availability Monitoring Problem
- System Model
- Design Goals
- AVMON System
- Experimental Results

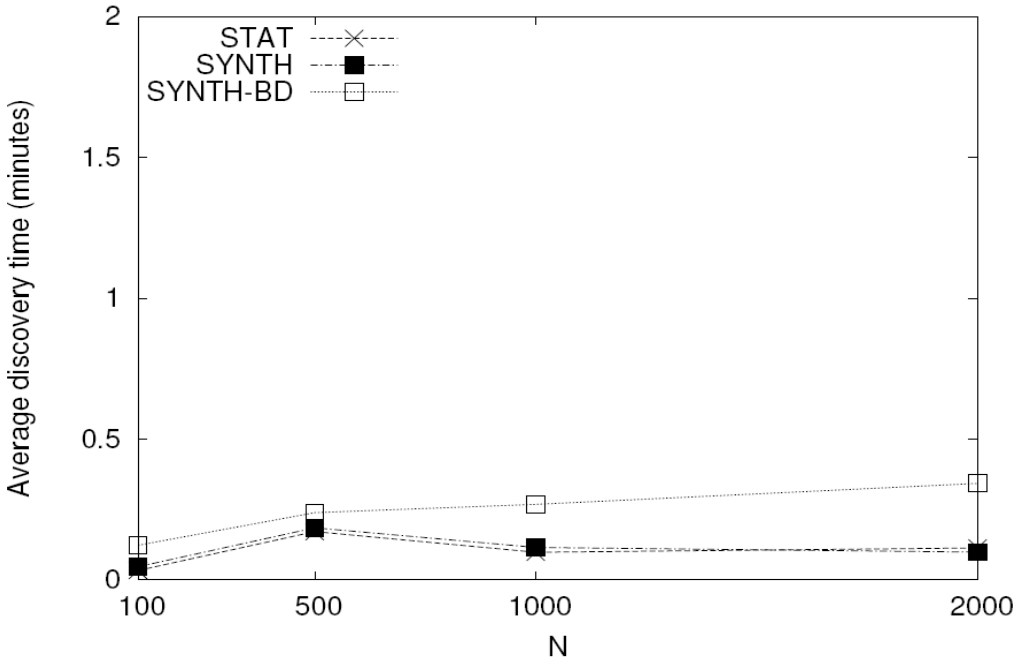
Experimental Design

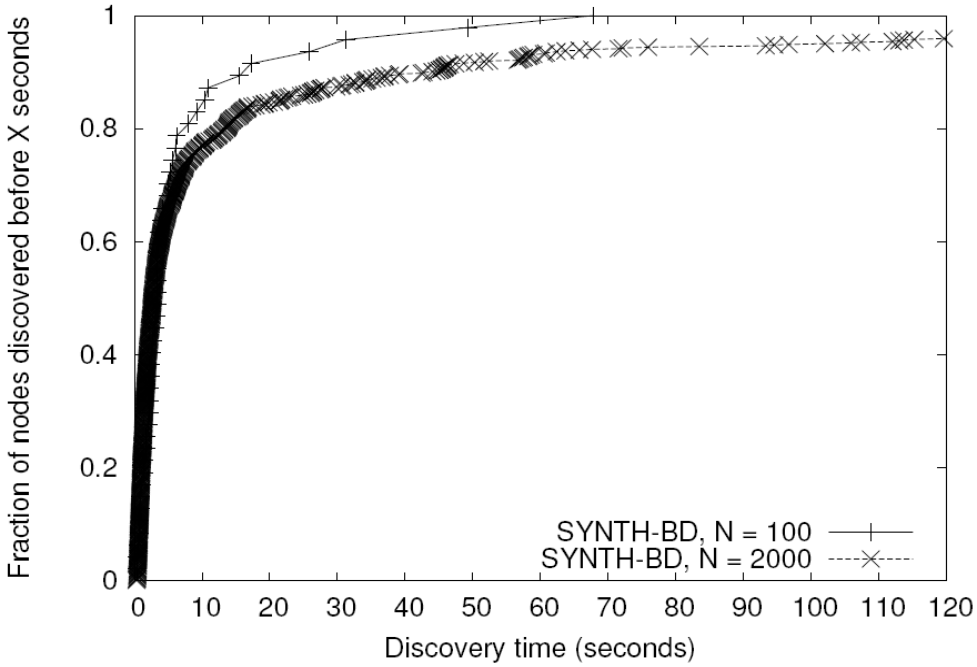
- Synthetic churn models:
  - `STAT`: no churn
  - `SYNTH`: 20% leave/join per hour
  - `SYNTH-BD`: 20% birth/death per day
- Availability Traces:
  - `OV`: Overnet system
  - `PL`: Planetlab system

Experimental Design (cont.)

	Coarse View Maintenance Period	***N***	Total Nodes	Coarse View Size $4 * N^{1/4}$	***K***= $\log_2(N)$
STAT	1 min	2000	2200	27	11
SYNTH	1 min	2000	2200	27	11
SYNTH-BD	1 min	2000	2809	27	11
OV	1 min	550	1319	19	9
PL	1 min	239	239	16	8

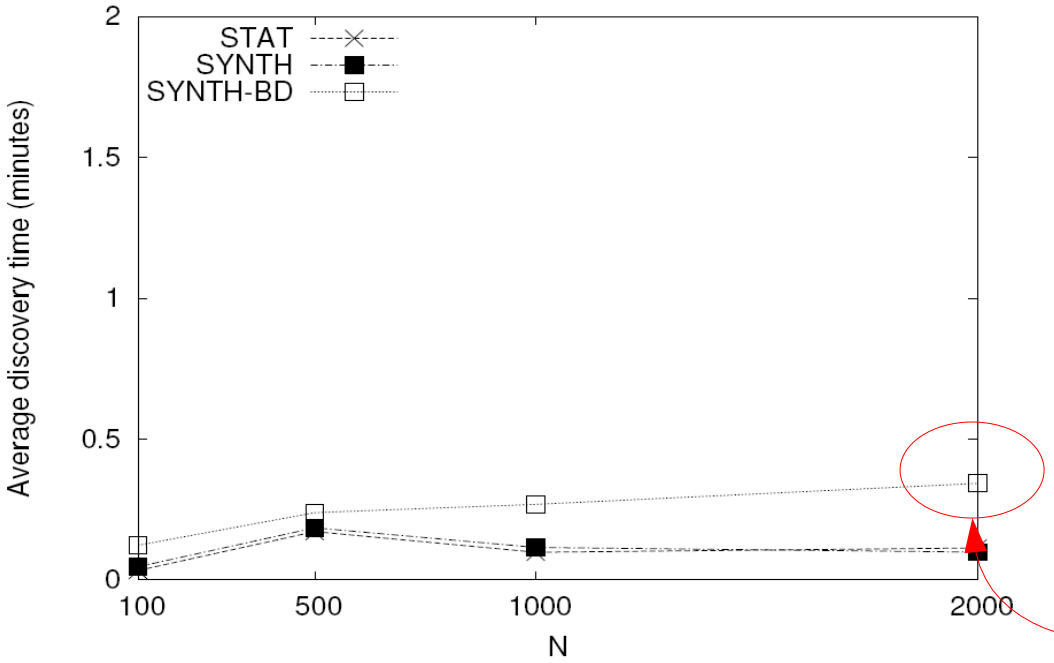
First Monitor Discovery

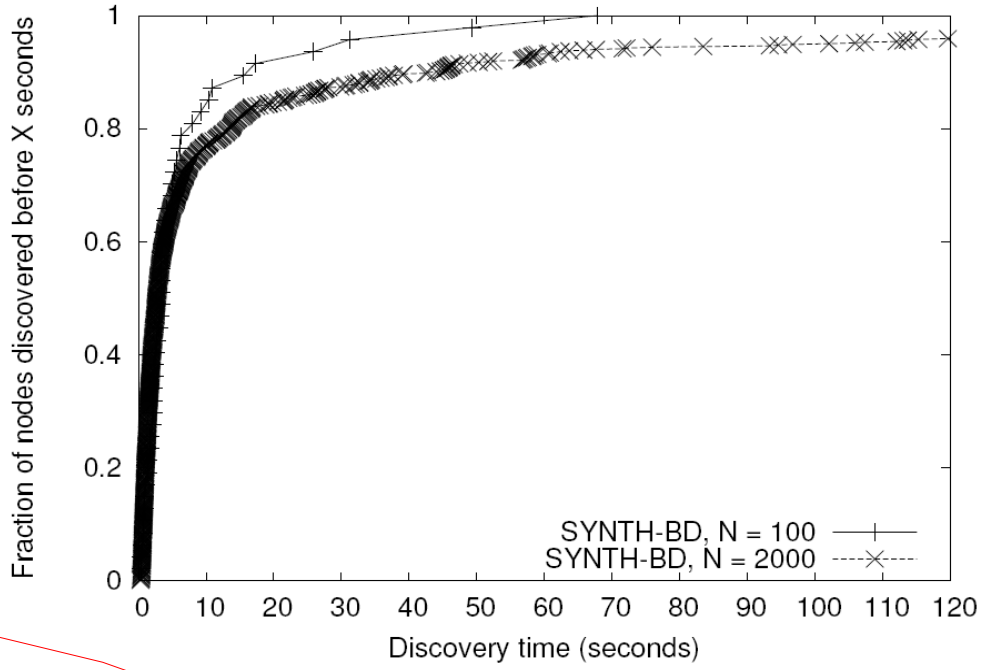




- Discovery of first monitor (for an arbitrary node) happens within one protocol period.

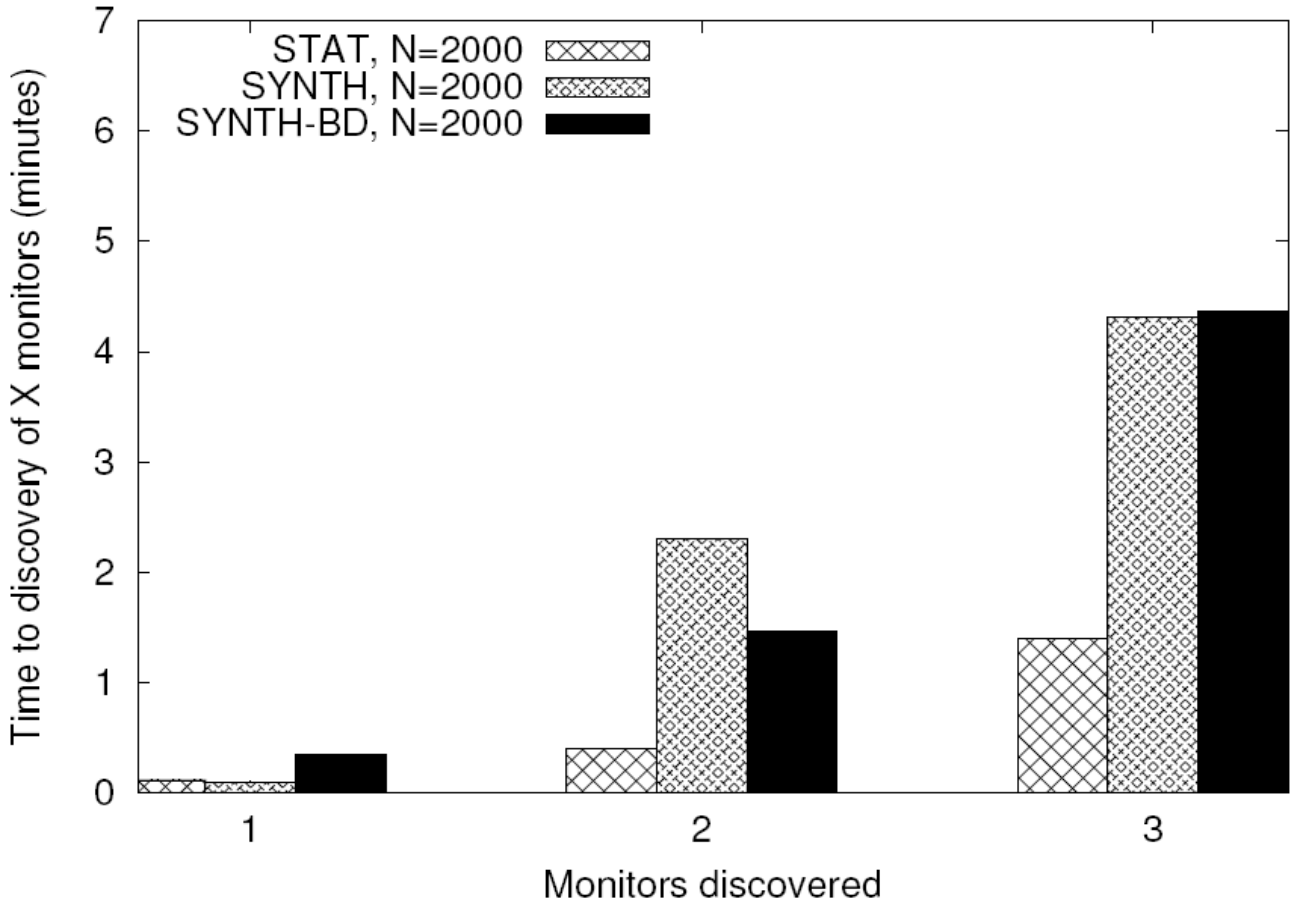
First Monitor Discovery





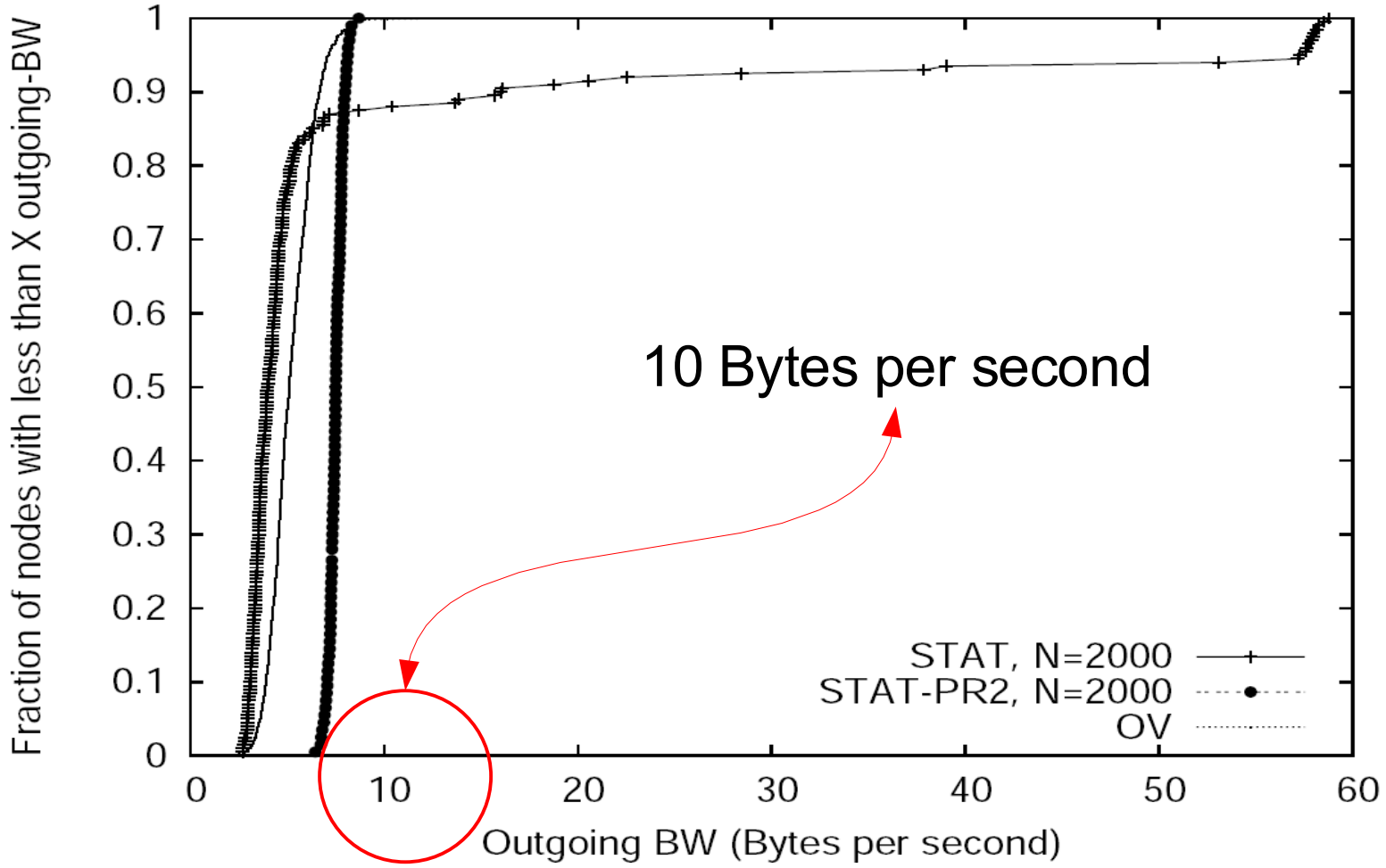
- BW: 6.81 Bps
- Mem: 52 Bytes
- Comp: 0.57ms (0.0000095% CPU) on a PC

Further Monitor Discovery



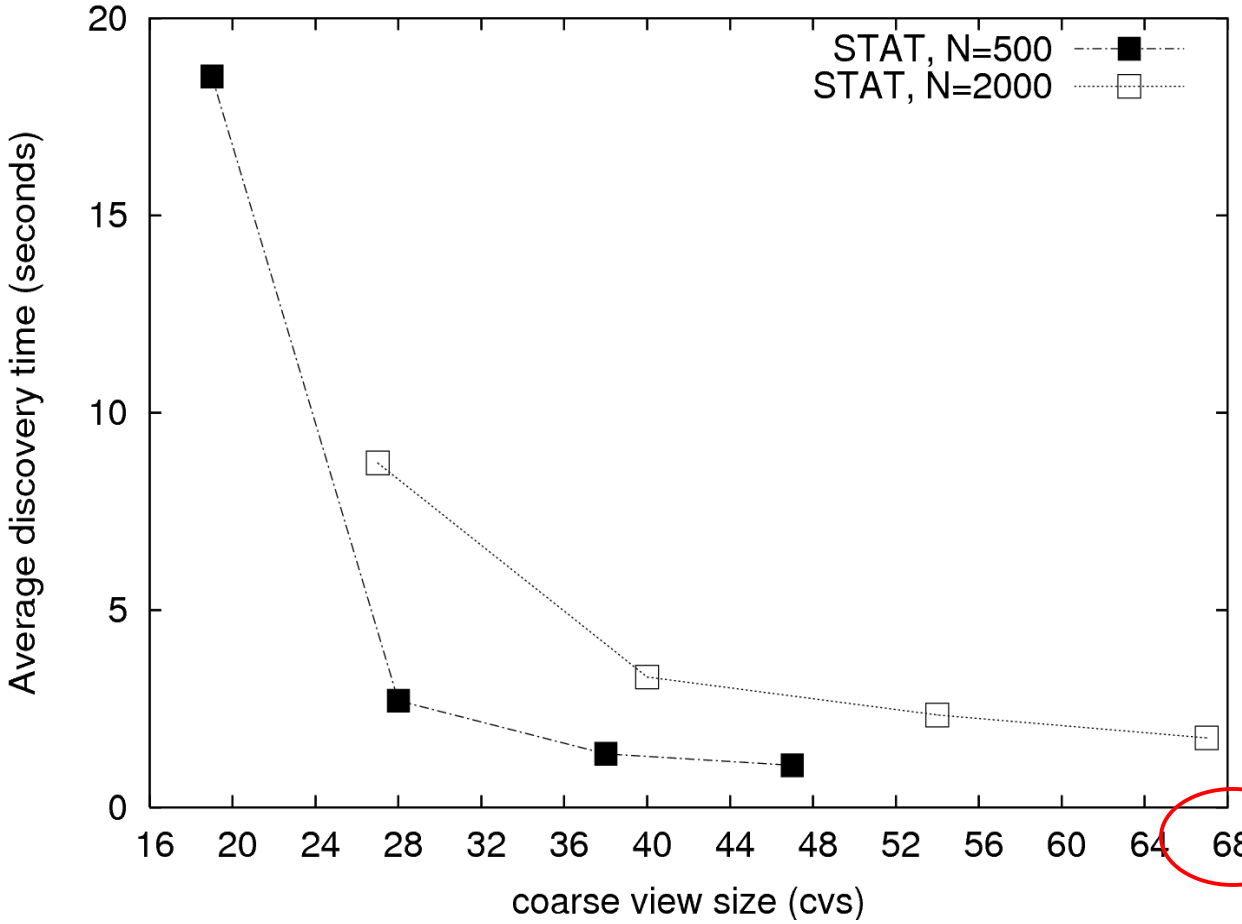
- Discovery time stays low, regardless of birth, death, join, rejoin.

Bandwidth



- BW is uniform, and low

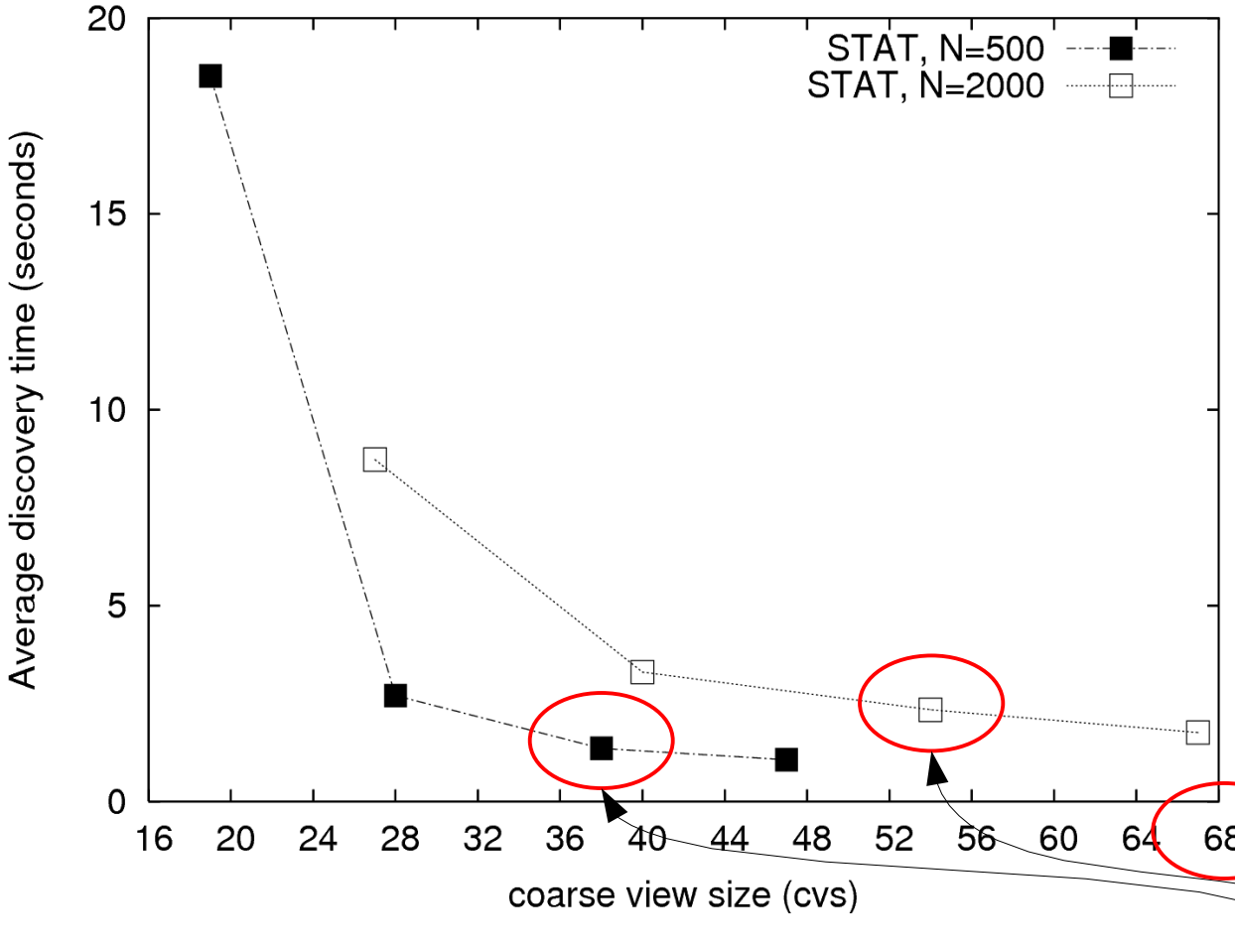
Discovery time vs. Coarse View size



- coarse view size = 68
- 9375.79 hashes
  - 3.52ms (per min)
- ***K*** = 11
- memory = 720 B
- BW = 632 Bps

- coarse view size = $4*N^{1/4}$, $6*N^{1/4}$, $8*N^{1/4}$, $10*N^{1/4}$

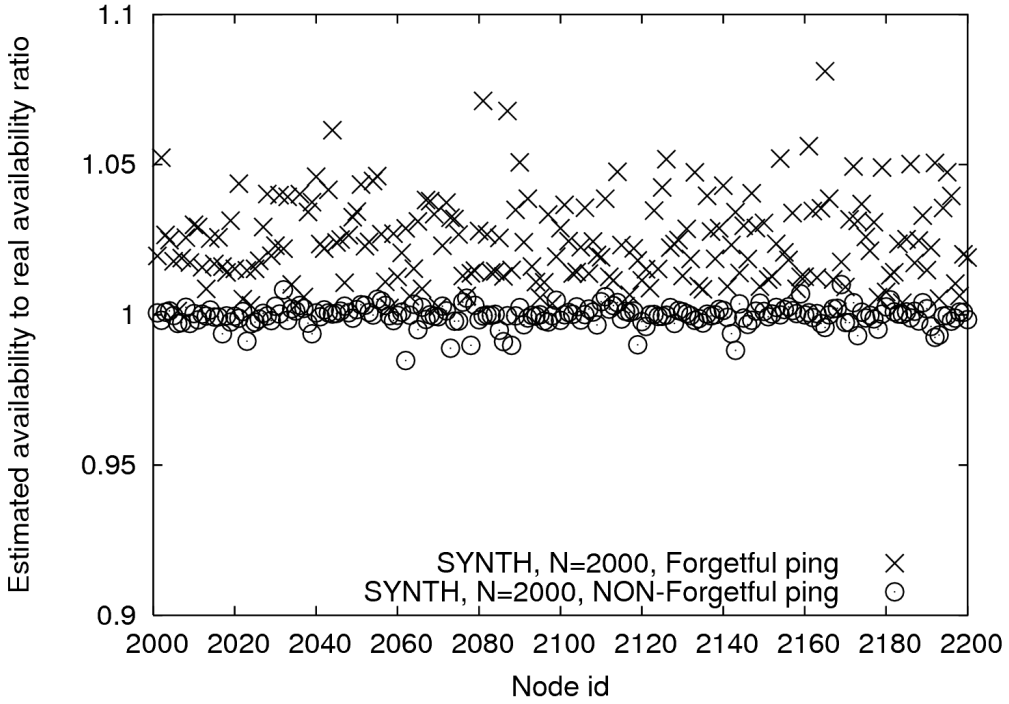
Discovery time vs. Coarse View size

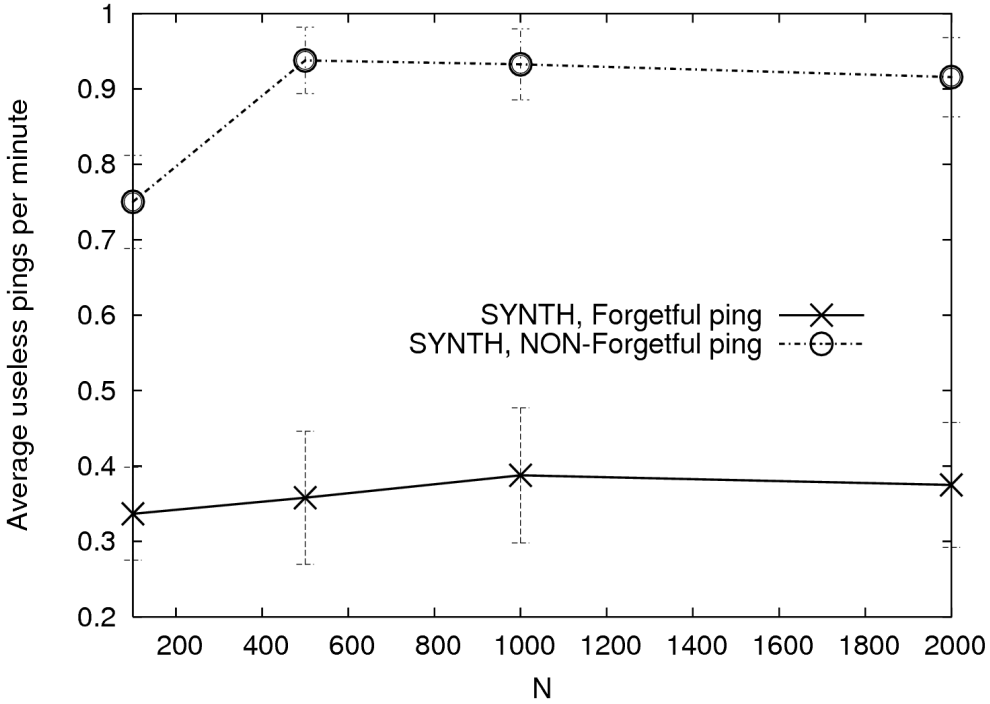


- coarse view size = 68
- 9375.79 hashes
  - 3.52ms (per min)
- ***K*** = 11
- memory = 720 B
- BW = 632 Bps

- coarse view size = $4*N^{1/4}$, $6*N^{1/4}$, $8*N^{1/4}$, $10*N^{1/4}$

Forgetful Pinging





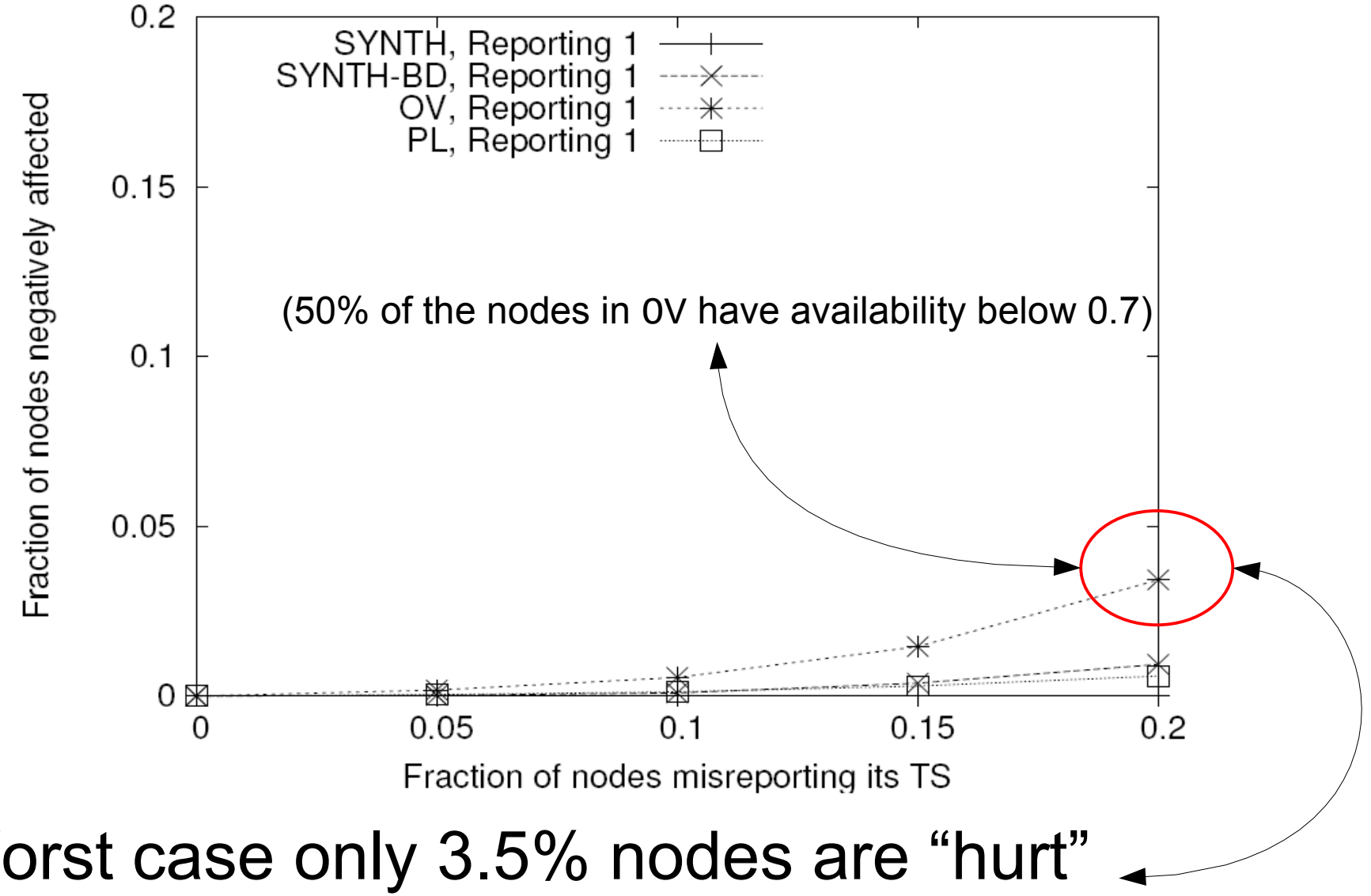
- Average error rate, <5%
- BW reduced by half

- Monitoring ping every 1 minute
- Start optimization after 2 minutes unresponsive

Overreporting Attack

- Random nodes selected as malicious
- Malicious nodes always report 100% availability for its ***TS(.)***
- Node "hurt" if its reported availability is off by at least 20% from real

Overreporting Attack



- Worst case only 3.5% nodes are “hurt”

Conclusions

- **AVMON** is the first system to address **Availability Monitoring Overlay** problem with goals:
  - Consistency, Verifiability, Randomness, Discoverability, Load Balancing, Scalability
- Efficient discovery and selection

  For N = 2000:
  - Discovery within 30s (half a protocol period)
  - Low BW: 6.81 Bps
  - Low Mem: 52 Bytes
  - Low CPU overhead: 0.57ms per min.

Thank you!

Questions?

AVMON's source code will soon be posted at:
http://kepler.cs.uiuc.edu/~rvmorale/avmon