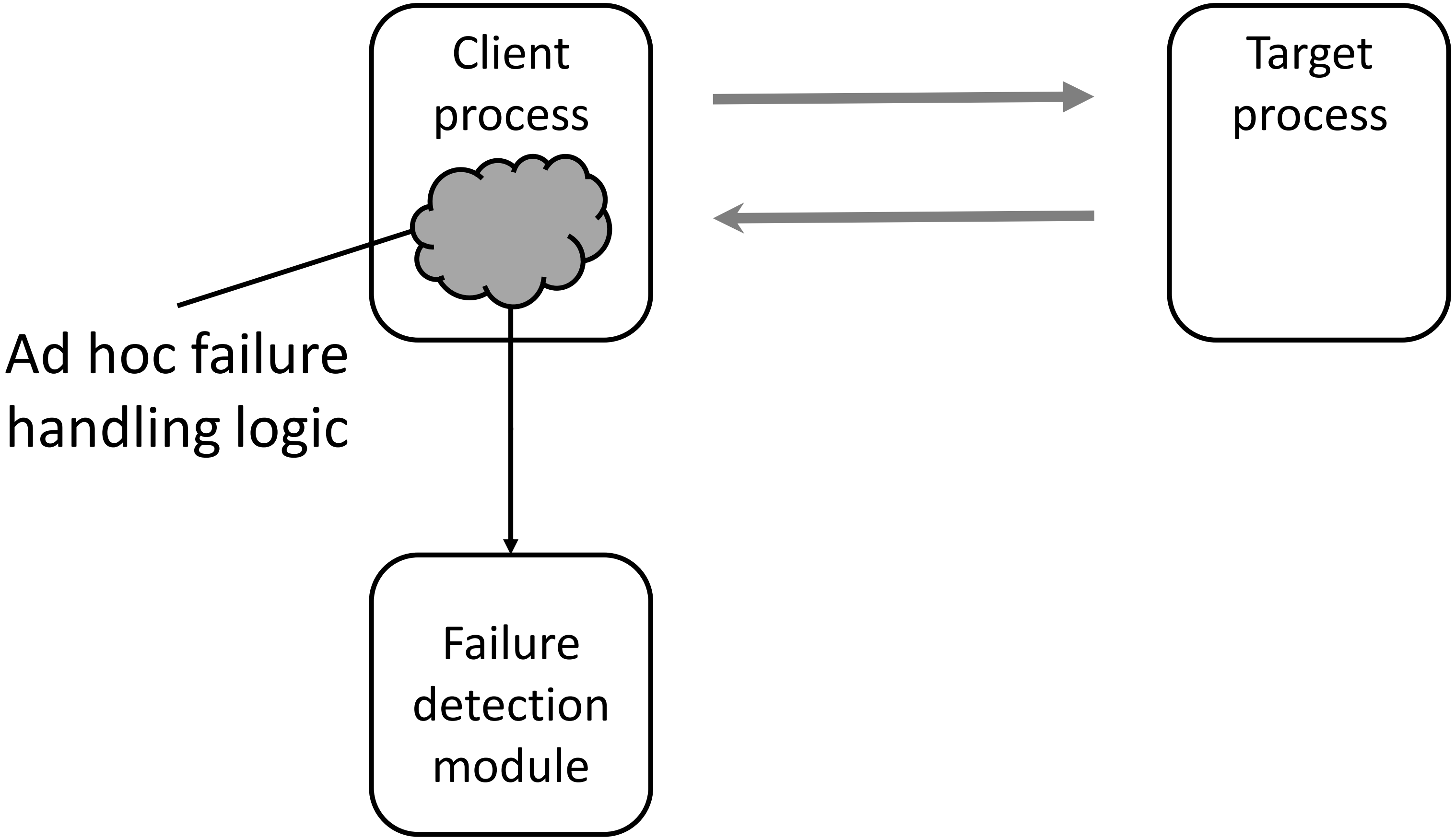
A new approach to reporting failures in distributed systems

Joshua B. Leners*, Trinabh Gupta*,

Marcos K. Aguilera¶, and Michael Walfish*

*The University of Texas at Austin

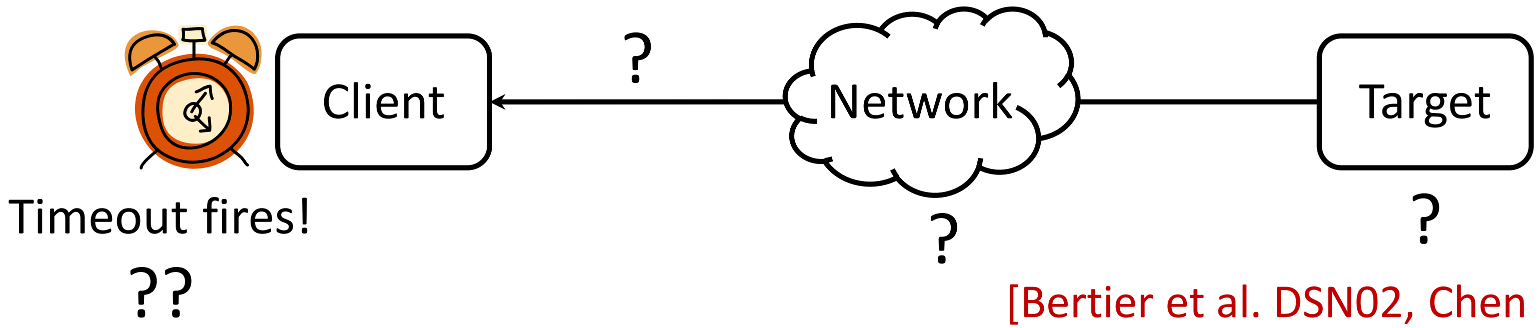
¶Microsoft Research Silicon Valley


Client process

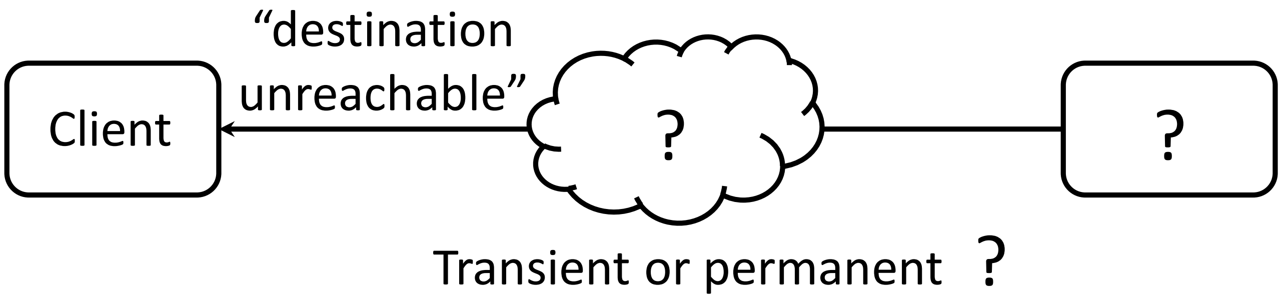
Target process

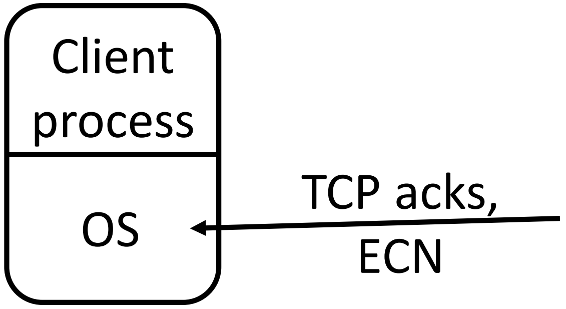
Ad hoc failure handling logic

Failure detection module



[Bertier et al. DSN02, Chen et al. 2002, ϕ-accrual FD SRDS04, sqrt(S) EuroSys07]





The network was designed to hide fine-grained failure information

“The architecture was to mask completely any transient failure.”

“ There are a number of services which are explicitly not assumed from the network [including] internal knowledge of failures ...”

[D.D. Clark. The design philosophy of the DARPA Internet protocols. SIGCOMM 1988]

This design choice is mismatched to today's requirements

Process crash

Network partition

A FAILURE

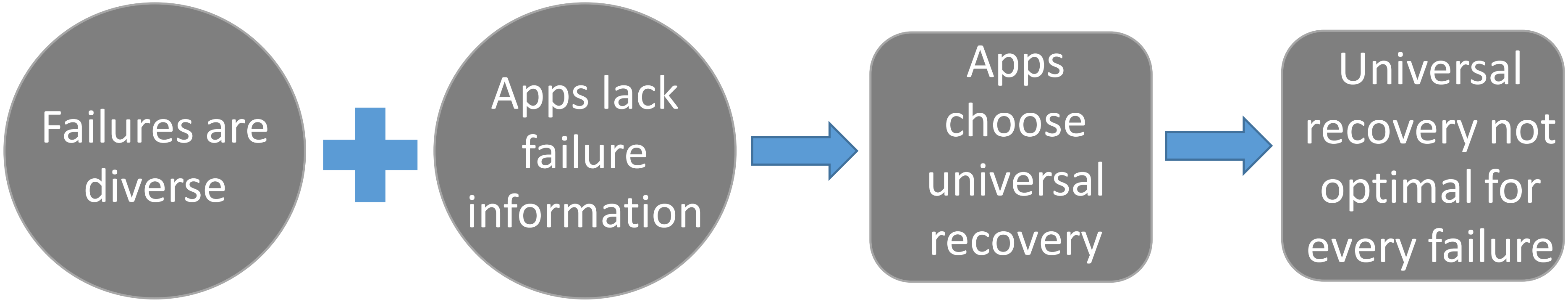
Stale routing state

Network congestion

Host crash

Misconfigured network

Transmission errors



Failures are diverse

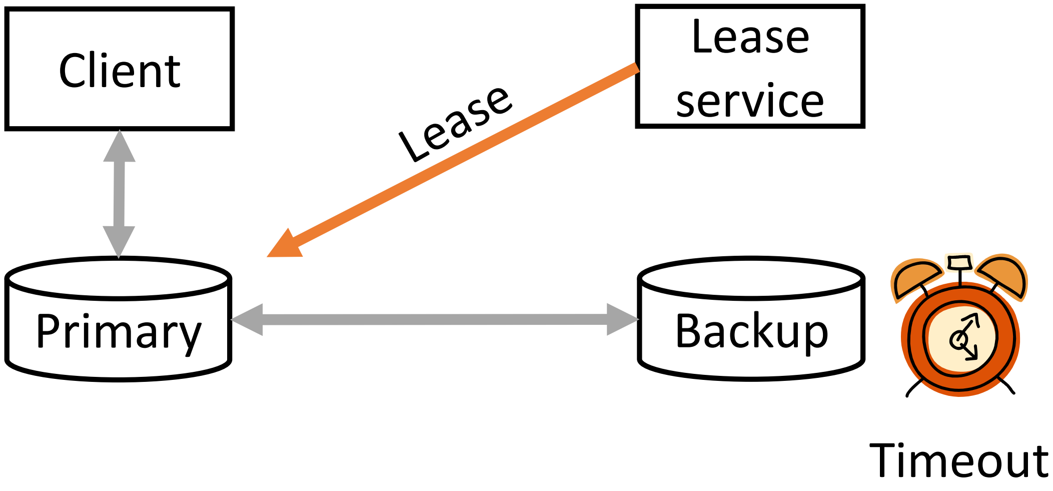
Apps lack failure information

Apps choose universal recovery

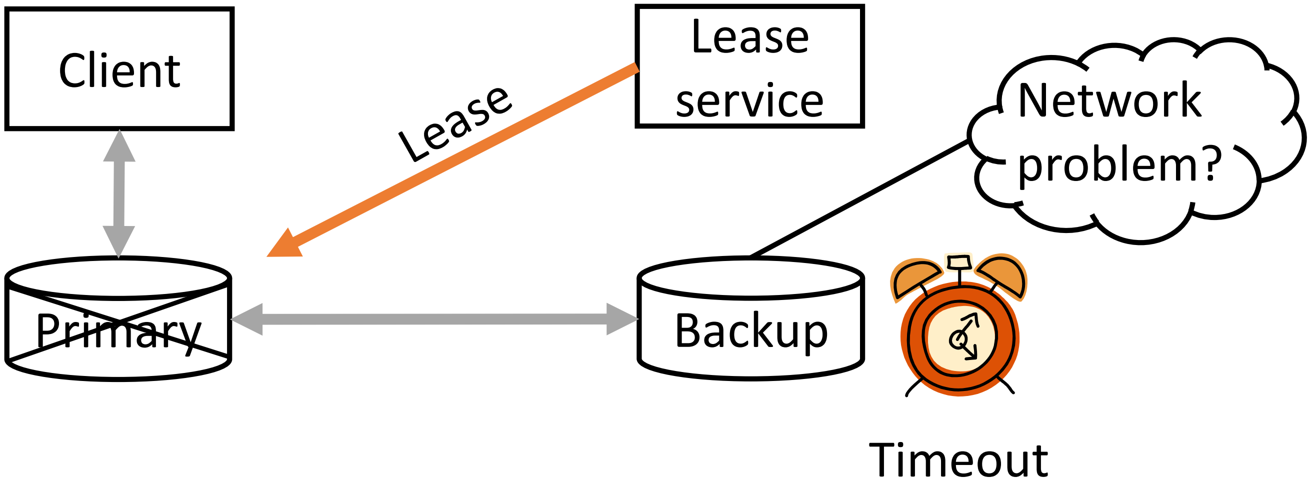
Universal recovery not optimal for every failure

Lower application availability

Applications choose a recovery action that must work for all failures

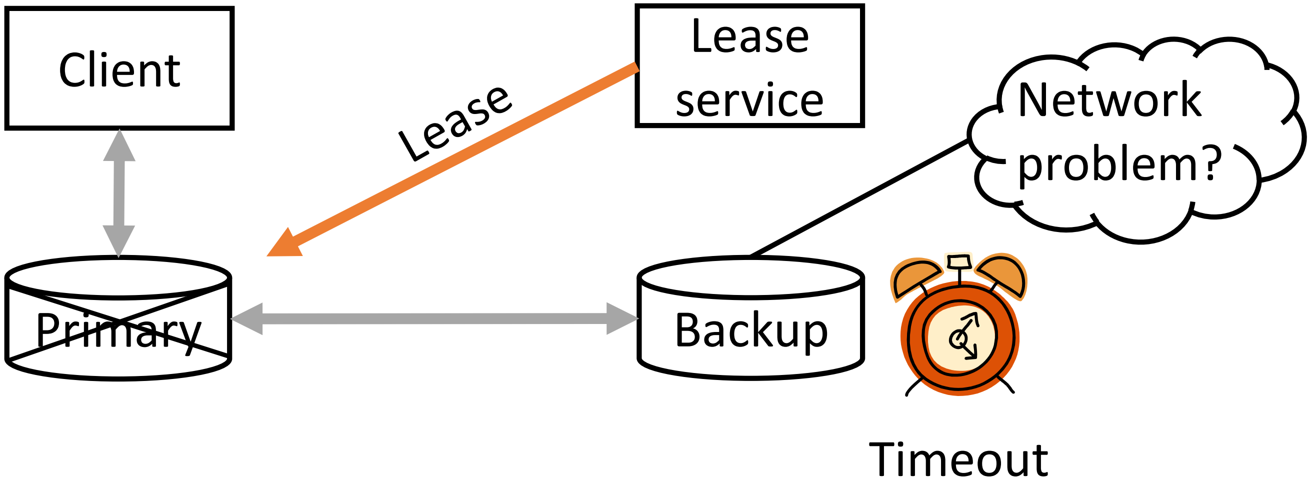


Applications choose a recovery action that must work for all failures



If primary crashes, the backup must wait for the lease to expire.

Improve recovery and availability using more information about failures



The backup must wait for the lease to expire.

If the backup knew that the primary had crashed then it could immediately take over.

We argue:

The network interface should expose information about failures

Outline

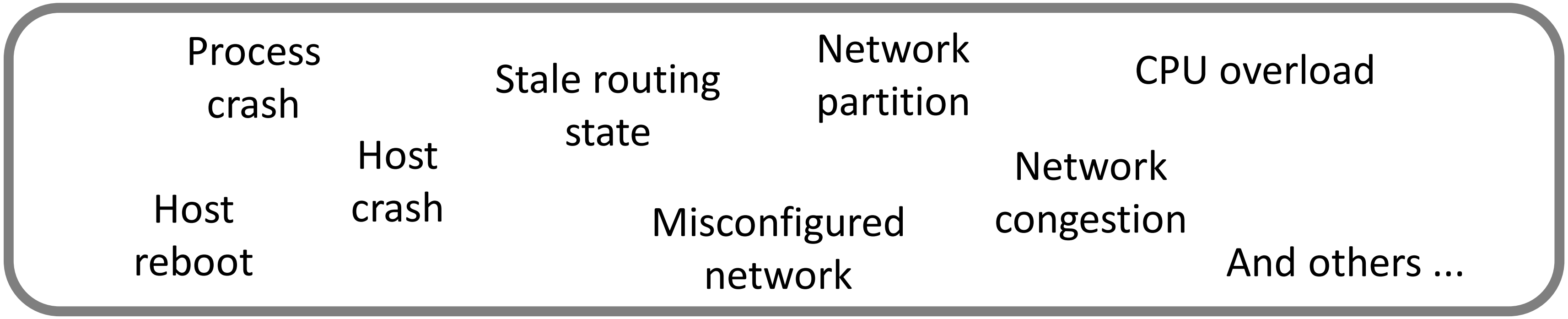
- An Interface to failures and its implementation, Pigeon.

- Benefits of this Interface to real-world applications.

The network interface should expose information about failures

But what failure information should it expose?

Strawman: expose individual failure type to application



Process crash

Host reboot

Host crash

Stale routing state

Misconfigured network

Network partition

Network congestion

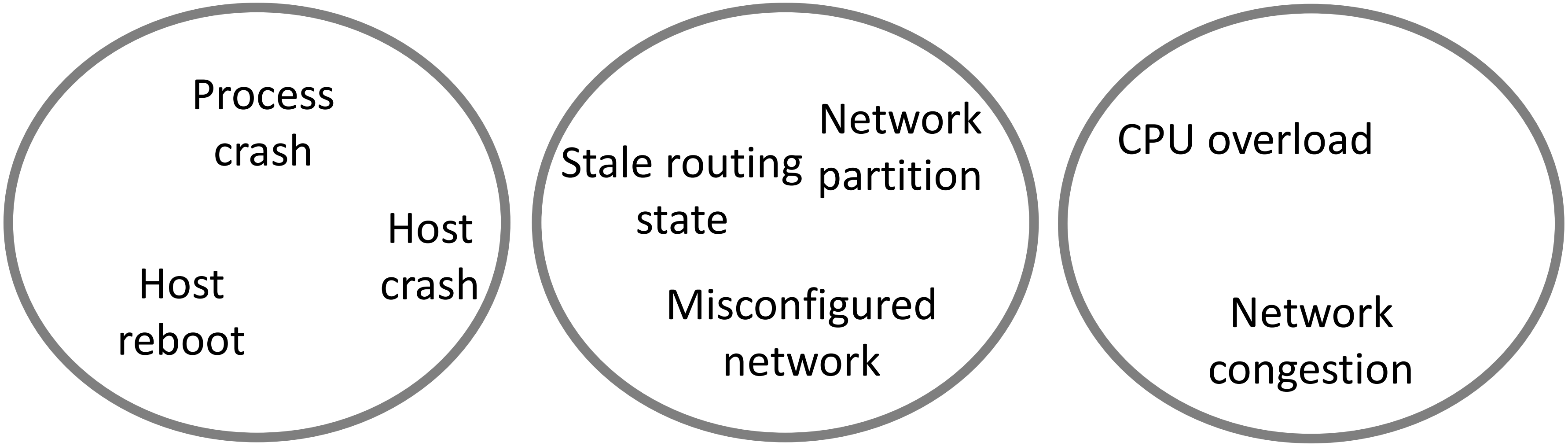
CPU overload

And others ...

- Exposing too many failure types is burdensome.
  - Understand semantics of each failure
  - New failure type requires updating code

- We need something simpler yet similarly effective.

Our approach: group remote failures that applications handle similarly



Process crash

Host reboot

Host crash

Stale routing state

Network partition

Misconfigured network

CPU overload

Network congestion

Stop: Target stopped permanently

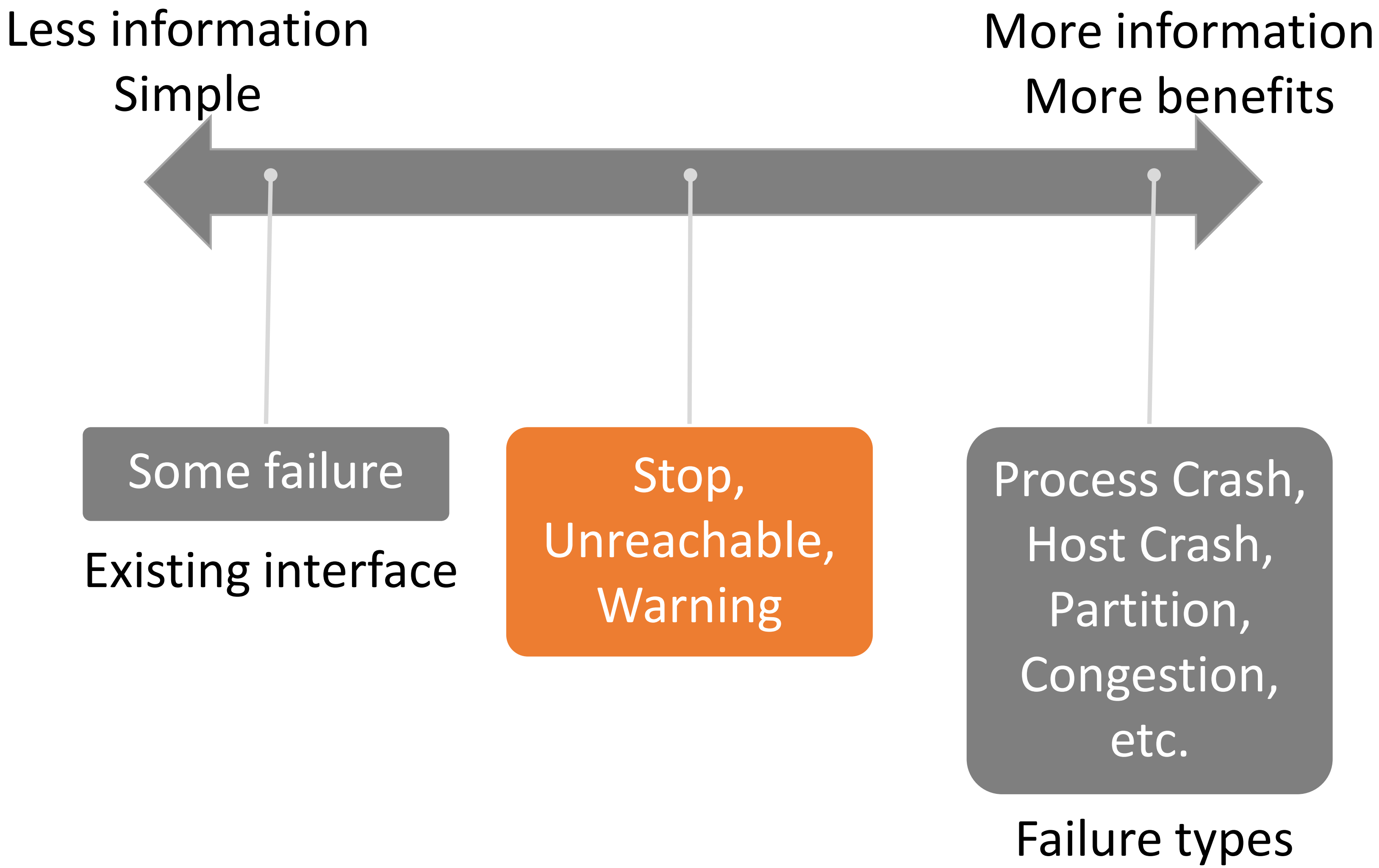
Unreachable: Target is unreachable

Expected Duration

Warning: Some resource is depleted

Depleted Resource

A simple yet effective abstraction



Less information
Simple

More information
More benefits

Some failure

Existing interface

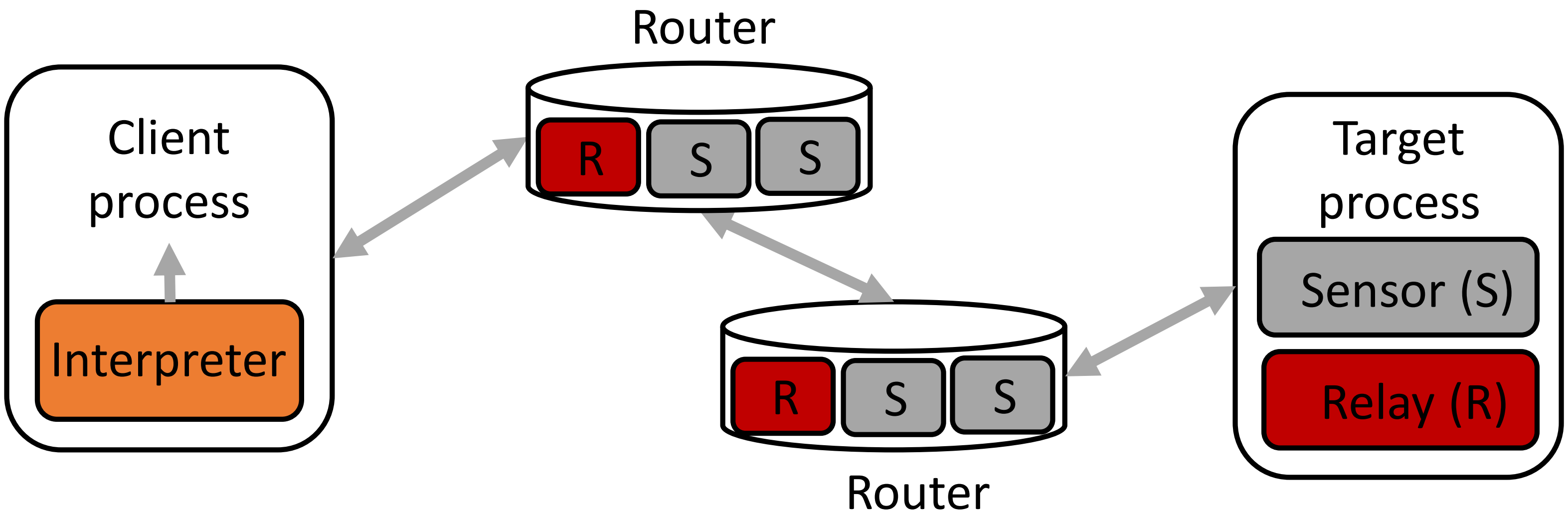
Stop,
Unreachable,
Warning

Process Crash,
Host Crash,
Partition,
Congestion,
etc.

Failure types

Bird's eye view of Pigeon

- Sensors gather local information; extends [Falcon SOSP11].
- Relays transport information to end-hosts.
- Interpreter presents the API to the applications.



Implementing the interface is easy but ...

- Pigeon must detect all possible remote failures and tolerate its own failures.

  Response: Use end-to-end timeouts.

- Pigeon must report duration of unreachability.

  Response: Use previously collected failure data.

Outline

✓ An Interface to failures and its implementation, Pigeon.

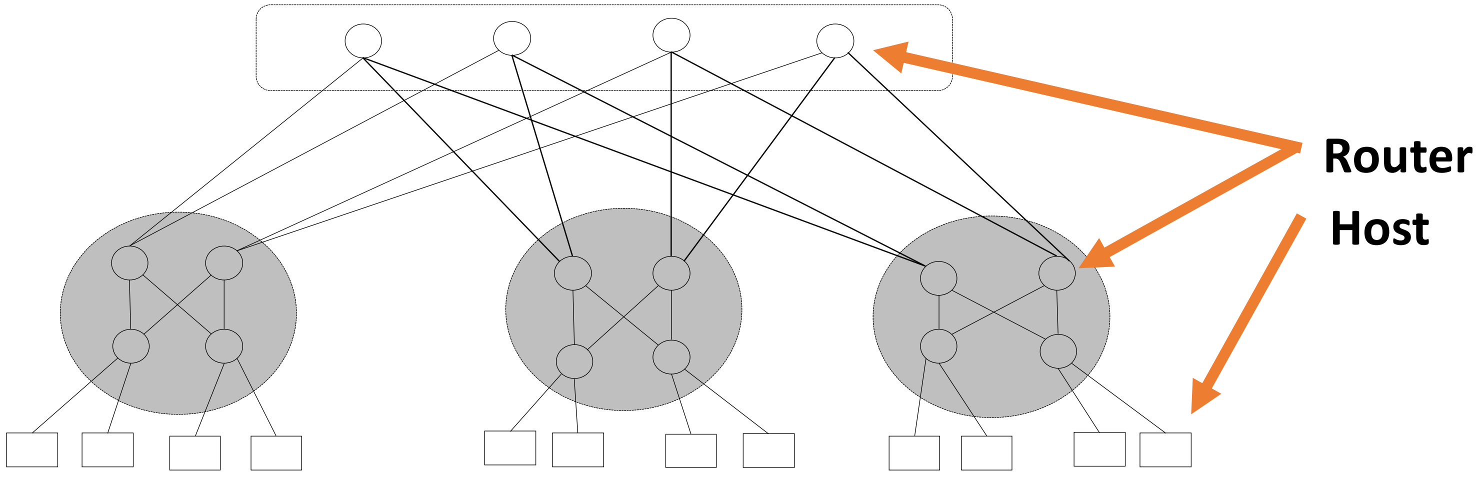
- Benefits of this Interface to real-world applications.

Evaluation questions

- Does using the interface improve application availability?
- Does using the interface let applications take optimal recovery actions?
- How much do all of these benefits cost?

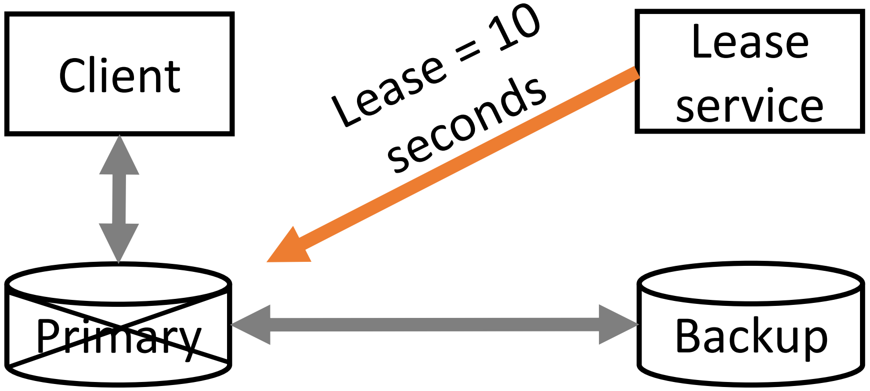
Our test network

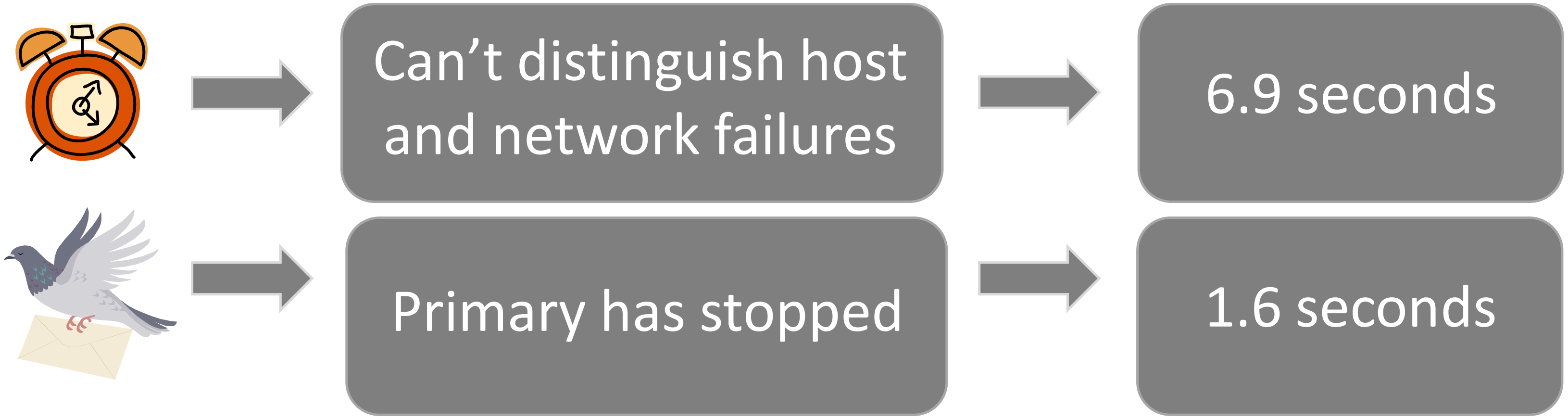
- 16 physical routers running OSPF and connected in a 4-port fat-tree topology.
- 12 hosts connected to this network.



Does distinguishing between host and network failures improve application availability?

Distinguishing host and network failures reduces unavailability by 4x



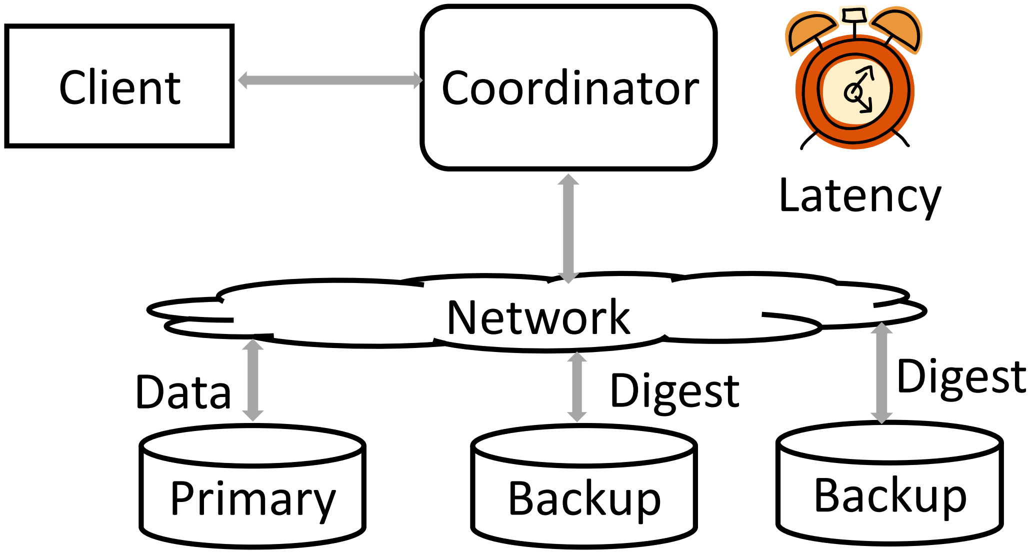


Can't distinguish host and network failures → 6.9 seconds

Primary has stopped → 1.6 seconds

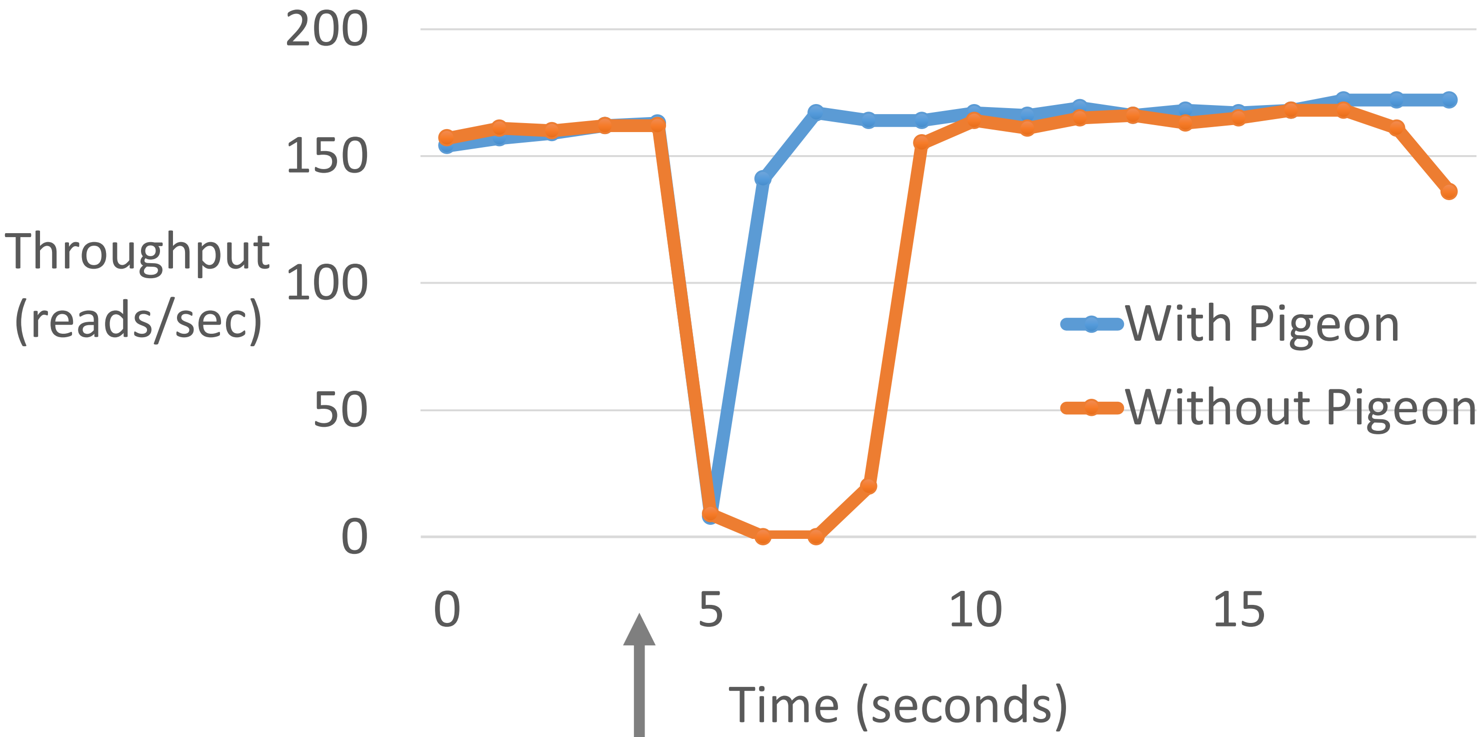
Does reporting network failure information improve application availability?

Cassandra



- Coordinator uses smallest average latency of past requests to select primary.

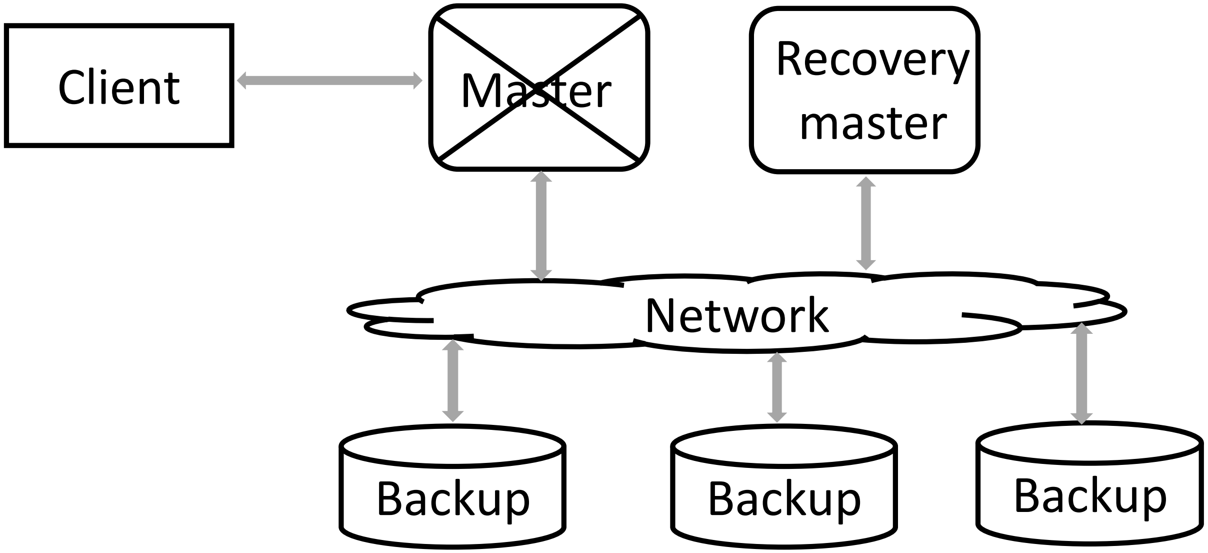
Detecting failures quickly helps Cassandra optimize replica selection



Transient routing problem

Does reporting the expected failure duration improve recovery decisions?

RAMCloud



- RAMCloud may recover even during transient problems.

Distinguishing transient and long-lasting failures helps avoid unnecessary recovery

For a transient routing problem

	# of times RAMCloud recovered	System downtime
Using timeout	10/10	2.8 seconds
With Pigeon	0/10	2.6 seconds

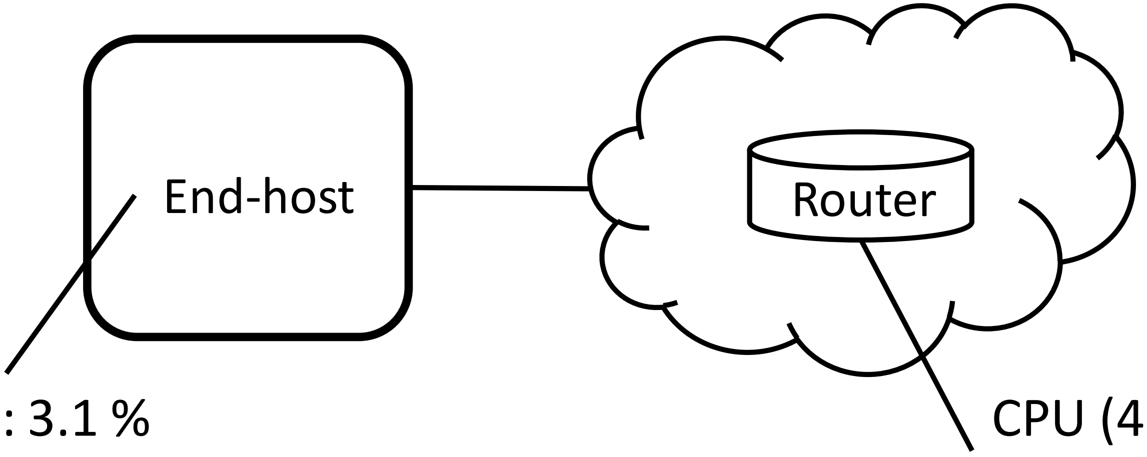
The interface is generally applicable

For a transient routing problem

Application	Action taken under Pigeon
RAMCloud	wait for problem to resolve
Cassandra	pick another primary replica

How much does Pigeon cost?

Pigeon provides benefits at low costs



CPU (2.4 GHz): 3.1 %
Network: 2.3 kbps

CPU (480 MHz): 0.3 %
Network: 2.1 kbps

	LOC
Pigeon (C++, Java)	5400
RAMCloud changes	68
Cassandra changes	414

Other related work: network monitoring

- For operators: ([Shaikh & Greenberg NSDI04, Kompella et al. NSDI05, Zhao et al. SIGCOMM06, Goldberg et al. SIGMETRICS 08])
- For transport: ([Krishnan et al. Computer Networks 2004, Stone & Partridge SIGCOMM00, NEH SIGCOMM08])
- For end-host app.
  - Latency: ([King IMW02, Meridian SIGCOMM05])
  - State of the network: ([CHHMR WORLDS04, Knowledge Plane SIGCOMM03, NetQuery SIGCOMM11, Sophia HotNets03, iPlane OSDI06])
  - Loss: ([Packet Obituaries HotNets04])
  - Path anomalies: ([PlanetSeer OSDI04])

Take-away points

- The network interface should expose failure information to applications.

- The interface should expose host and network failures, transient and long-lasting failures.

- This interface can be implemented at low cost (and with simple design) and benefits a variety of applications.