Failure detection as a network abstraction for end-host applications

Michael Walfish,*
under the influence of
Marcos K. Aguilera,† Trinabh Gupta,* and Joshua B. Leners*

*The University of Texas at Austin
†Microsoft Research Silicon Valley

Failure detection as a network abstraction for end-host applications

Michael Walfish,*
under the influence of
Marcos K. Aguilera,† Trinabh Gupta,* and Joshua B. Leners*

*The University of Texas at Austin
†Microsoft Research Silicon Valley

The opportunity

Failures are common and diverse, but ...

... the current Internet hides failures.

Today's interfaces to failures

What could we gain by exposing failures?

Questions and challenges

What should the interface be?

Research questions

Costs and deployment

Prior work

Application-specific networking

Network monitoring

We organize this area according to two axes:
- Who is the intended recipient?
- What information is gathered?

Failure detection and recovery

The opportunity

Failures are common and diverse, but ...

Networks and end-hosts are subject to a rich pathology of failures.

The possibilities include hardware malfunctions, software bugs, configuration errors, excess load and more.

The effects include end-host crashes, network partitions, degraded performance, incorrect routing state, and more.

... the current Internet hides failures.

There is much prior research. However, none of it exposes failure information to applications.

application-specific networking

network monitoring

failure detection and recovery

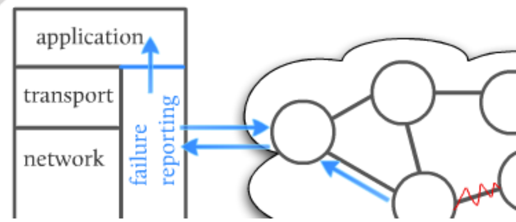
Today's interfaces to failures ...

lack coverage, are coarse, and incur delay.

What could we gain by exposing failures?

If an application has better intelligence about failures, it can make better decisions about whether and how to recover.

Consider replication services, client-server systems, and storage systems.



In April 2011, servers in Amazon's EBS wrongly inferred that backups had crashed.

The traffic from re-replication congested the network, leading to more false suspicions.

The result was a "re-mirroring storm" that contributed to a twelve-hour outage.
["Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region", Amazon Web Services Team.]

Moral: the recovery action should match the actual failure.

Moral: the recovery action should match the actual failure.

The opportunity

Failures are common and diverse, but ...

Networks and end-hosts are subject to a rich pathology of failures.

The possibilities include hardware malfunctions, software bugs, configuration errors, excess load and more.

The effects include end-host crashes, network partitions, degraded performance, incorrect routing state, and more.

... the current Internet hides failures.

"[At] the top of transport, there is only one failure, and it is total partition. The architecture was to mask completely any transient failure ... [The] Internet ... detect[s] network failures using Internet level mechanisms, with the potential for a slower and less specific error detection."

–D. D. Clark, the design philosophy of the DARPA Internet protocols, SIGCOMM 1988.

There is much prior research. However, none of it exposes failure information to applications.

- application-specific networking
- network monitoring
- failure detection and recovery

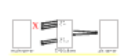
Today's interfaces to failures ...

lack coverage, are coarse, and incur delay.

What could we gain by exposing failures?

If an application has better intelligence about failures, it can make better decisions about whether and how to recover.

Consider replication services, client-server systems, and storage systems.



“[At] the top of transport, there is only one failure, and it is total partition. The architecture was to mask completely any transient failure … [The] Internet … detect[s] network failures using Internet level mechanisms, with the potential for a slower and less specific error detection.”

–D. D. Clark, the design philosophy of the DARPA Internet protocols, SIGCOMM 1988.

Today's interfaces to failures ...

lack coverage, are coarse, and incur delay.

For example, applications can receive a TCP "connection reset" through the sockets interface.

This signal indicates a remote process exit but not other problems (host crash, network partition, etc.)

Consider application-level end-to-end timeouts.

If the timeout fires, that indicates that something may have failed, but not what.

Moreover, an end-to-end timeout is hard to set. Setting it too low risks inaccuracy and ...

.... an end-to-end timeout set too large delays recovery.

Also, none of the aforementioned detects latent failures.

	OS crash	overloaded network	single link failure	multiple link failures
`connect`	21s	⊖	3.1s	3.6s
`send`	⊖	⊖	⊖	⊖
`epoll`	⊖	⊖	⊖	⊖
`epoll`, error Q	⊖	⊖	3.2s	3.6s
`sendto`	⊖	⊖	⊖	⊖
`sendto`, error Q	18s	⊖	⊖	3.5s
ICMP	20s	⊖	⊖	3.5s

⊖ means that the mechanism does not detect the failure.

lack coverage,

For example, applications can receive a TCP "connection reset" through the sockets interface.

This signal indicates a remote process exit but not other problems (host crash, network partition, etc.)

, are coarse, and i

Consider application-level end-to-end timeouts.

If the timeout fires, that indicates that something may have failed, but not what.

Moreover, an end-to-end timeout is hard to set. Setting it too low risks inaccuracy and ...

l incur delay.

.... an end-to-end timeout set too large delays recovery.

Also, none of the aforementioned detects latent failures.

	OS crash	overloaded network	single link failure	multiple link failures
`connect`	21s	⊖	3.1s	3.6s
`send`	⊖	⊖	⊖	⊖
`epoll`	⊖	⊖	⊖	⊖
`epoll`, error Q	⊖	⊖	3.2s	3.6s
`sendto`	⊖	⊖	⊖	⊖
`sendto`, error Q	18s	⊖	⊖	3.5s
ICMP	20s	⊖	⊖	3.5s

⊖ means that the mechanism does not detect the failure.

... the current Internet hides failures.

"[At] the top of transport, there is only one failure, and it is total partition. The architecture was to mask completely any transient failure ... [The] Internet ... detect[s] network failures using Internet level mechanisms, with the potential for a slower and less specific error detection."

–D. D. Clark, the design philosophy of the DARPA Internet protocols, SIGCOMM 1988.

Today's interfaces to failures ...

lack coverage, are coarse, and incur delay.

There is much prior research. However, none of it exposes failure information to applications.

- application-specific networking
- network monitoring
- failure detection and recovery

d we gain

There is much prior research. However, none of it exposes failure information to applications.

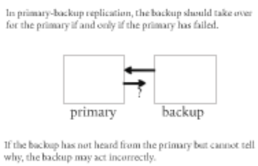
- application-specific networking
- network monitoring
- failure detection and recovery

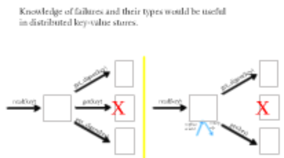
ould we gain sing failures?

What could we gain by exposing failures?

If an application has better intelligence about failures, it can make better decisions about whether and how to recover.

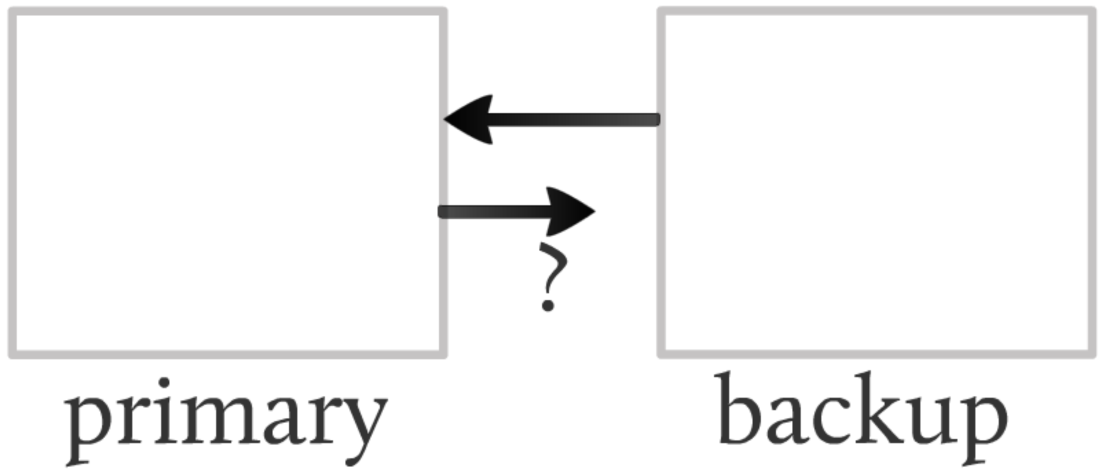
Consider replication services, client-server systems, and storage systems.



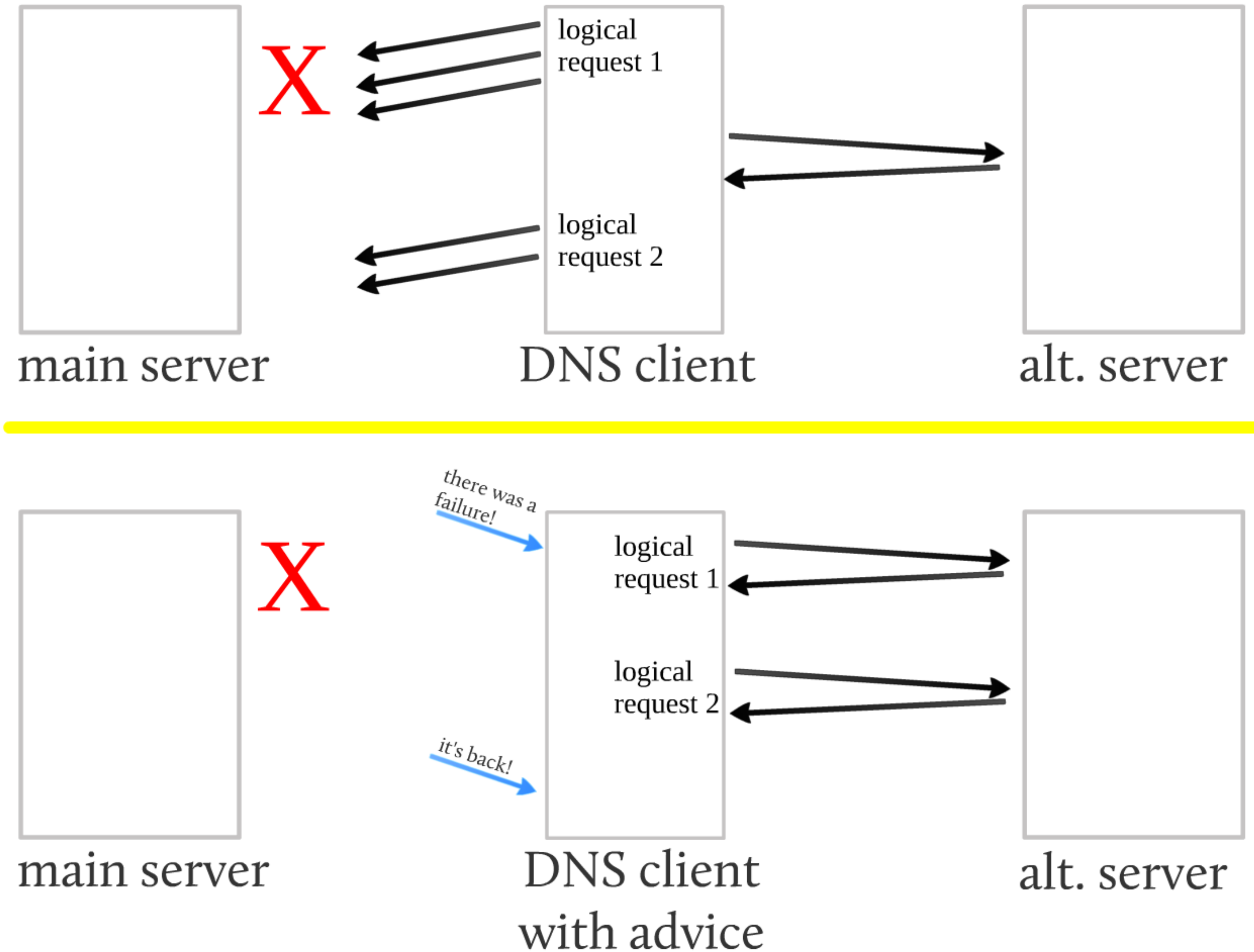


application	host failure	congestion	route failure	partition
DNS	use alternate server	change primary probabilistically	use alternate server	use alternate server
NFS (soft mount)	return error	wait	wait	return error
Paxos	immediate leader election	invoke election if majority report persistent congestion	wait	invoke leader election if majority report partition
Primary-backup	immediately failover	use slow failover	wait	use slower failover
RAMCloud	start recovery	wait	wait	start recovery
Cassandra	skip replica, report in op.	choose alt. primary replica	choose alt. primary replica	skip replica, report in op.

In primary-backup replication, the backup should take over for the primary if and only if the primary has failed.

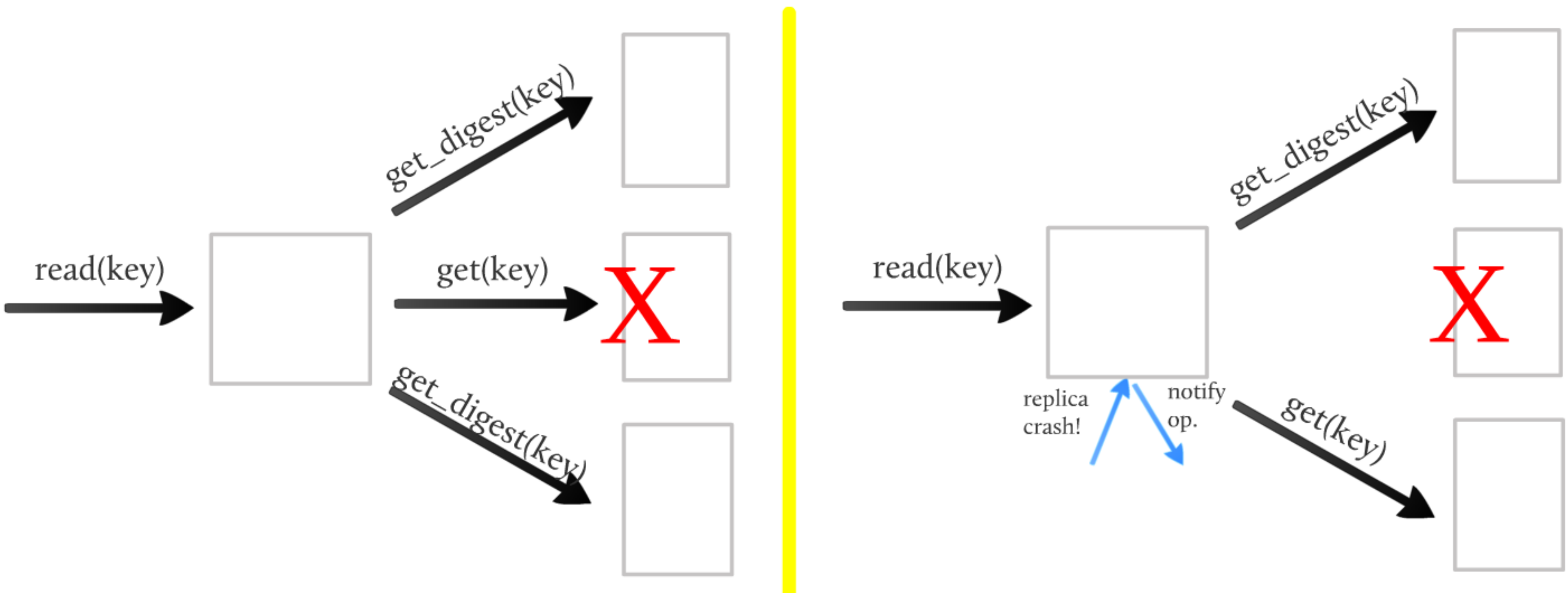


If the backup has not heard from the primary but cannot tell why, the backup may act incorrectly.

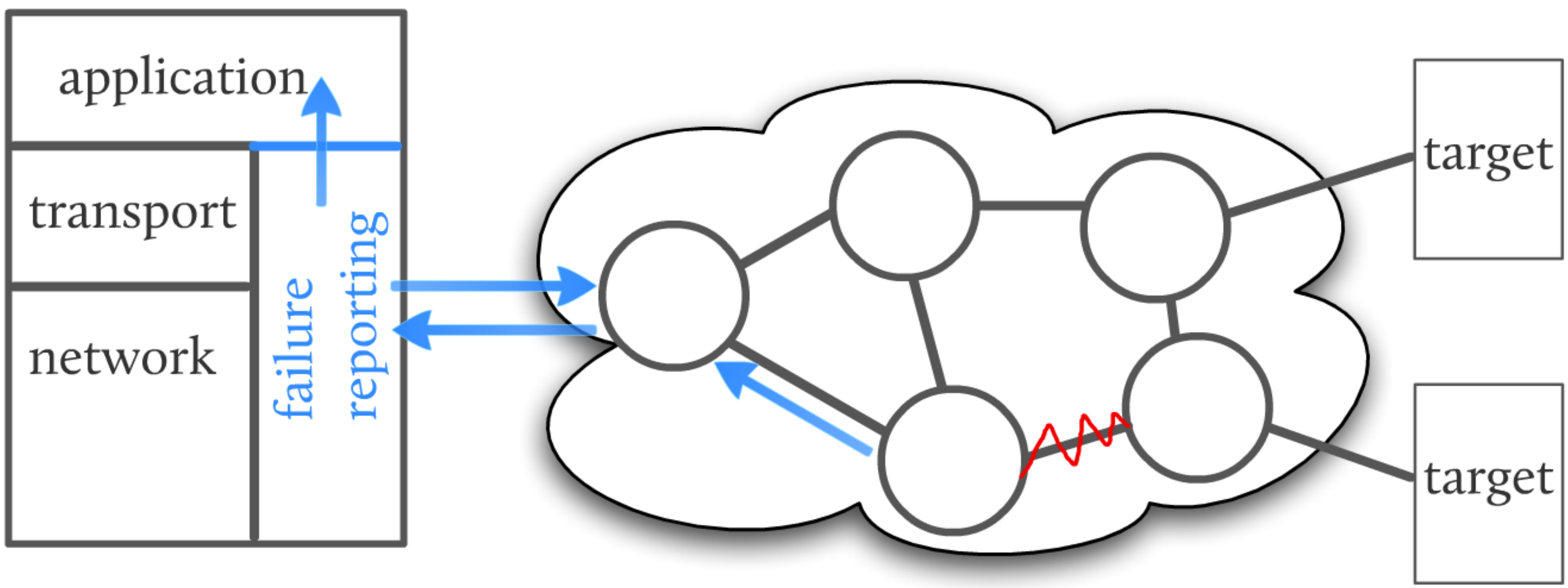


logical request 1
logical request 2
main server
DNS client
alt. server
there was a failure!
it's back!
logical request 1
logical request 2
main server
DNS client with advice
alt. server

Knowledge of failures and their types would be useful in distributed key-value stores.



application	host failure	congestion	route failure	partition
DNS	use alternate server	change primary probabilistically	use alternate server	use alternate server
NFS (soft mount)	return error	wait	wait	return error
Paxos	immediate leader election	invoke election if majority report persistent congestion	wait	invoke leader election if majority report partition
Primary-backup	immediately failover	use slow failover	wait	use slower failover
RAMcloud	start recovery	wait	wait	start recovery
Cassandra	skip replica, report to op.	choose alt. primary replica	choose alt. primary replica	skip replica, report to op.


application
transport
network
failure reporting
target
target

The opportunity

Failures are common and diverse, but ...

... the current Internet hides failures.

Today's interfaces to failures

What could we gain by exposing failures?

Questions and challenges

What should the interface be?

Research questions

Costs and deployment

Prior work

Application-specific networking

Network monitoring

We organize this area according to two axes:
• Who is the intended recipient?
• What information is gathered?

Failure detection and recovery

Questions and challenges

What should the interface be?

What failure types should be exposed?

A-list	B-list
Host failures	Corruption
Congestion	Route instability
Route failure	Violation of routing policy
Partition	

What should the API be?

The interface should be independent of specific failures, to allow for pluggable implementations.

Should it be callback-based (probably) or query-based?

Should the interface report only the type of failure or even more fine-grained information?

Research questions

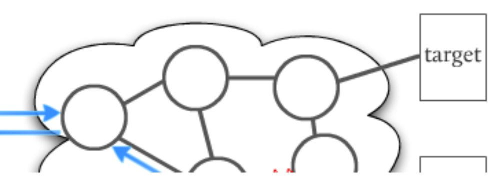
Defining

Detecting and reporting

Costs and deployment

We hypothesize that failure reporting can be cost-effective, if it piggy-backs on existing protocols.

We conjecture that deployment barriers will be lowered by software-defined networks.

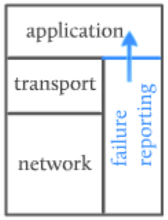


What should the interface be?

What failure types should be exposed?

A-list	B-list
Host failures	Corruption
Congestion	Route instability
Route failure	Violation of routing policy
Partition	

What should the API be?



The interface should be independent of specific failures, to allow for pluggable implementations.

Should it be callback-based (probably) or query-based?

Should the interface report only the type of failure or even more fine-grained information?

Research questions

Defining

How should we actually define these failures?

For instance, when should we say that the path between A and B is experiencing congestion?

Detecting and reporting

Mechanistically, how should we detect these failures?

- Can we do it in such a way that different detectors for different failures can plug in to a coherent architecture?
- Can the mechanisms scale as the number of monitored elements and monitoring hosts increases?

Can a network report failures without revealing sensitive information?

Defining

How should we actually define these failures?

For instance, when should we say that the path between A and B is experiencing congestion?

Detecting and reporting

Mechanistically, how should we detect these failures?

- Can we do it in such a way that different detectors for different failures can plug in to a coherent architecture?

- Can the mechanisms scale as the number of monitored elements and monitoring hosts increases?

Can a network report failures without revealing sensitive information?

Costs and deployment

We hypothesize that failure reporting can be cost-effective, if it piggy-backs on existing protocols.

We conjecture that deployment barriers will be lowered by software-defined networks.

Questions and challenges

What should the interface be?

What failure types should be exposed?

A-list	B-list
Host failures	Corruption
Congestion	Route instability
Route failure	Violation of routing policy
Partition	

What should the API be?

The interface should be independent of specific failures, to allow for pluggable implementations.

Should it be callback-based (probably) or query-based?

Should the interface report only the type of failure or even more fine-grained information?

Research questions

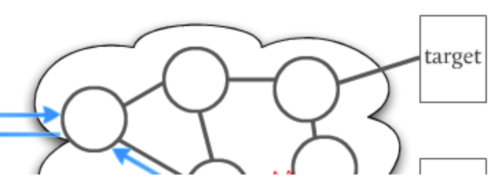
Defining

Detecting and reporting

Costs and deployment

We hypothesize that failure reporting can be cost-effective, if it piggy-backs on existing protocols.

We conjecture that deployment barriers will be lowered by software-defined networks.



Prior work

Application-specific networking

In this strand of work, functions normally hidden from applications are exposed to them.

This area began with D. D. Clark's and D. L. Tennenhouse's 1990 paper on ALF ("Architectural considerations for a new generation of protocols," SIGCOMM 1990).

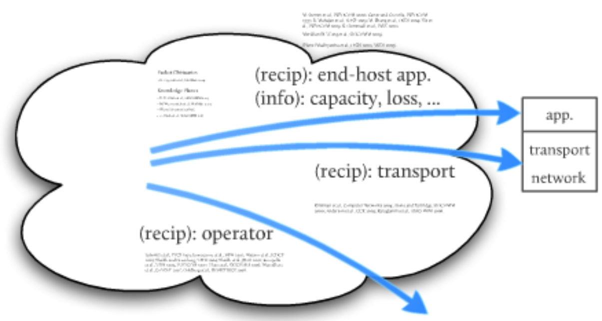
The ethos of ALF influenced other systems:

- The exokernel (D. A. Wallach, D. R. Engler, and M. F. Kaashoek, "ASHs: application-specific handlers for high-performance messaging," TON 1997).
- Plexus (M. E. Fiuczynski and B. N. Bershad, "An extensible protocol architecture for application-specific networking," USENIX ATC 1996).
- The Congestion Manager (H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for Internet hosts," SIGCOMM 1999).

Network monitoring

We organize this area according to two axes:

- Who is the intended recipient?
- What information is gathered?



Failure detection and recovery

Failure recovery is at the heart of the network's design, but researchers have proposed techniques for making the network even more robust:

- Failure-carrying packets (K. Lakshminarayanan et al., "Achieving convergence-free routing using failure-carrying packets," SIGCOMM 2007).
- SafeGuard (A. Li, X. Yang, and D. Wetherall, "SafeGuard: Safe forwarding during route changes," CoNEXT 2009).
- Data-driven connectivity (J. Liu, S. Shenker, M. Schapira, and B. Yang, "Data-driven network connectivity," HotNets 2011).

Failure detection was formalized by T. D. Chandra and S. Toueg ("Unreliable failure detectors for reliable distributed systems," JACM 1996).

- Many failure detectors use end-to-end timeouts to infer end-host crashes (Bertier et al. DSN 2002; Chen et al. 2002; Hayashibara et al., SRDS 2004; So and Sirer, EUROSYS 2007).
- Recently, Leners et al. proposed a failure detector based on layer-specific probes in the target ("Detecting failures in distributed systems with the FALCON spy network", SOSP 2011).

Application-specific networking

In this strand of work, functions normally hidden from applications are exposed to them.

This area began with D. D. Clark's and D. L. Tennenhouse's 1990 paper on ALF ("Architectural considerations for a new generation of protocols," SIGCOMM 1990).

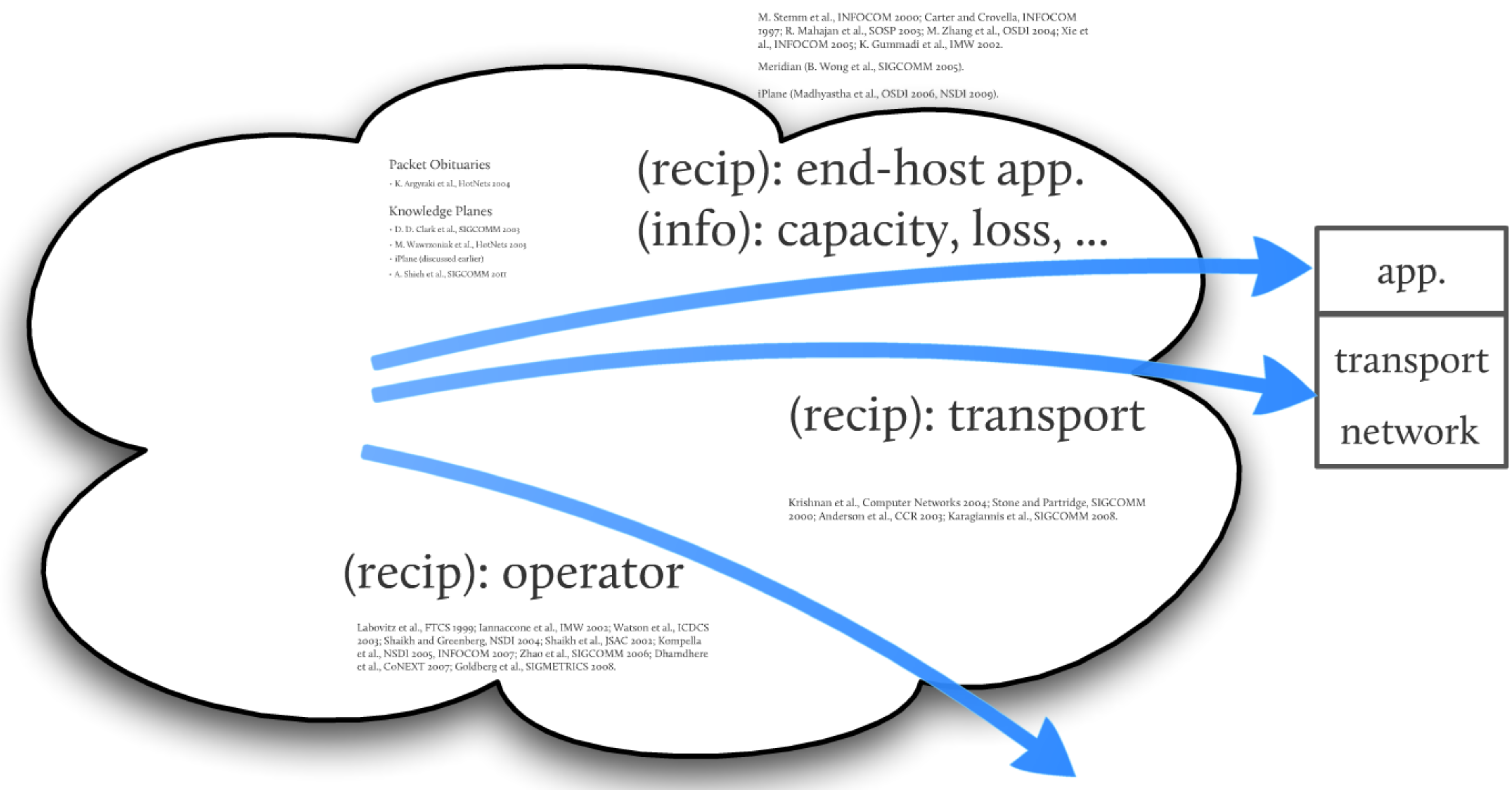
The ethos of ALF influenced other systems:

- The exokernel (D. A. Wallach, D. R. Engler, and M. F. Kaashoek, "ASHs: application-specific handlers for high-performance messaging," TON 1997).
- Plexus (M. E. Fiuczynski and B. N. Bershad, "An extensible protocol architecture for application-specific networking," USENIX ATC 1996).
- The Congestion Manager (H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for Internet hosts," SIGCOMM 1999).

Network monitoring

We organize this area according to two axes:

- Who is the intended recipient?
- What information is gathered?



(recip): operator

Labovitz et al., FTCS 1999; Iannaccone et al., IMW 2002; Watson et al., ICDCS 2003; Shaikh and Greenberg, NSDI 2004; Shaikh et al., JSAC 2002; Kompella et al., NSDI 2005, INFOCOM 2007; Zhao et al., SIGCOMM 2006; Dhamdhere et al., CoNEXT 2007; Goldberg et al., SIGMETRICS 2008.

(recip): transport

Krishnan et al., Computer Networks 2004; Stone and Partridge, SIGCOMM 2000; Anderson et al., CCR 2003; Karagiannis et al., SIGCOMM 2008.

M. Stemm et al., INFOCOM 2000; Carter and Crovella, INFOCOM 1997; R. Mahajan et al., SOSP 2003; M. Zhang et al., OSDI 2004; Xie et al., INFOCOM 2005; K. Gummadi et al., IMW 2002.

Meridian (B. Wong et al., SIGCOMM 2005).

iPlane (Madhyastha et al., OSDI 2006, NSDI 2009).

(recip): end-host app.

(info): capacity, loss, ...

Packet Obituaries

- K. Argyraki et al., HotNets 2004

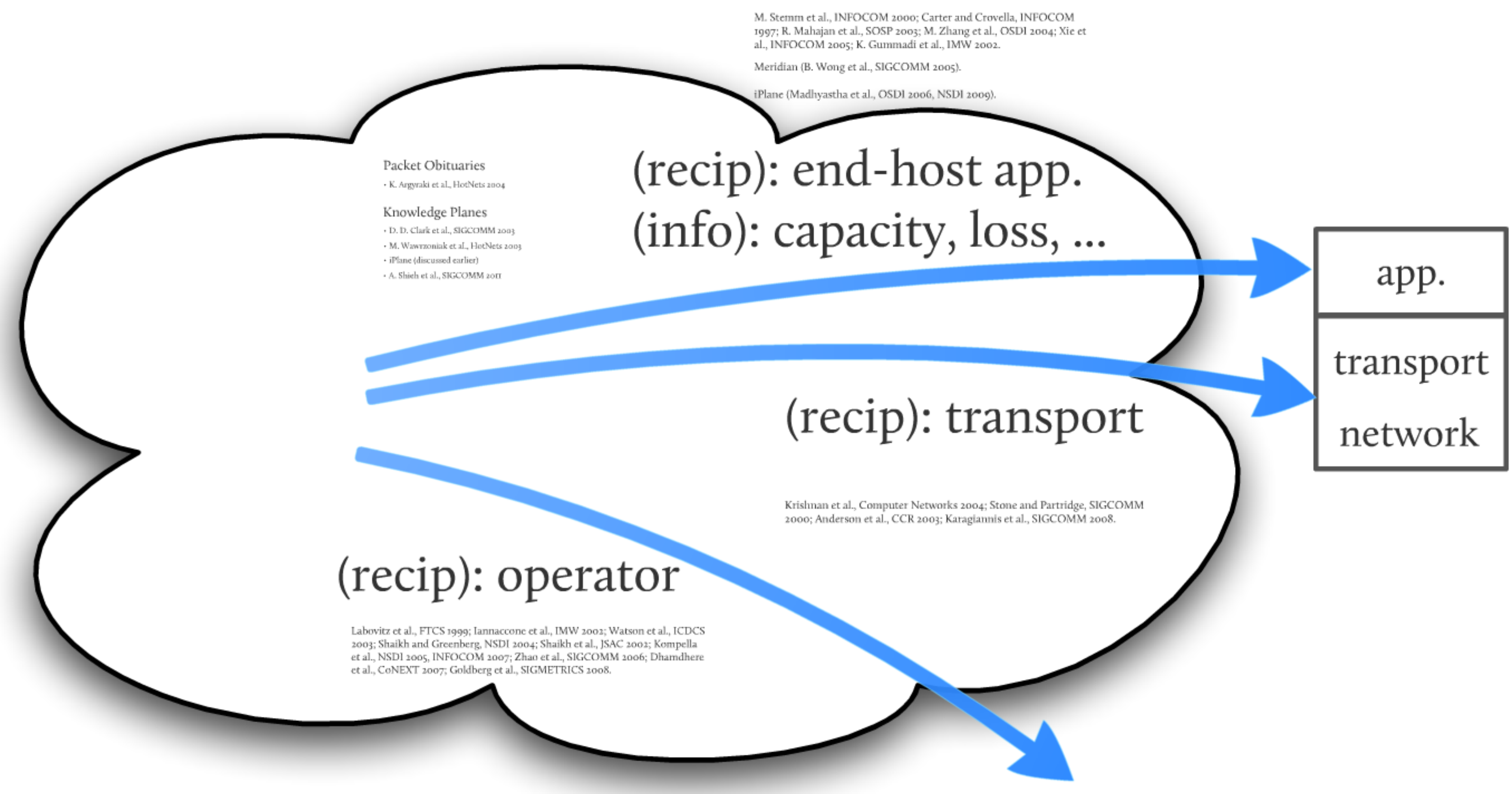
Knowledge Planes

- D. D. Clark et al., SIGCOMM 2003
- M. Wawrzoniak et al., HotNets 2003
- iPlane (discussed earlier)
- A. Shieh et al., SIGCOMM 2011

Network monitoring

We organize this area according to two axes:

- Who is the intended recipient?
- What information is gathered?



Failure detection and recovery

Failure recovery is at the heart of the network's design, but researchers have proposed techniques for making the network even more robust:

- Failure-carrying packets (K. Lakshminarayanan et al., "Achieving convergence-free routing using failure-carrying packets," SIGCOMM 2007).
- SafeGuard (A. Li, X. Yang, and D. Wetherall, "SafeGuard: Safe forwarding during route changes," CoNEXT 2009).
- Data-driven connectivity (J. Liu, S. Shenker, M. Schapira, and B. Yang, "Data-driven network connectivity," HotNets 2011).

Failure detection was formalized by T. D. Chandra and S. Toueg ("Unreliable failure detectors for reliable distributed systems," JACM 1996).

- Many failure detectors use end-to-end timeouts to infer end-host crashes (Bertier et al. DSN 2002; Chen et al. 2002; Hayashibara et al., SRDS 2004; So and Sirer, EUROSYS 2007).
- Recently, Leners et al. proposed a failure detector based on layer-specific probes in the target ("Detecting failures in distributed systems with the FALCON spy network", SOSP 2011).

Prior work

Application-specific networking

In this strand of work, functions normally hidden from applications are exposed to them.

This area began with D. D. Clark's and D. L. Tennenhouse's 1990 paper on ALF ("Architectural considerations for a new generation of protocols," SIGCOMM 1990).

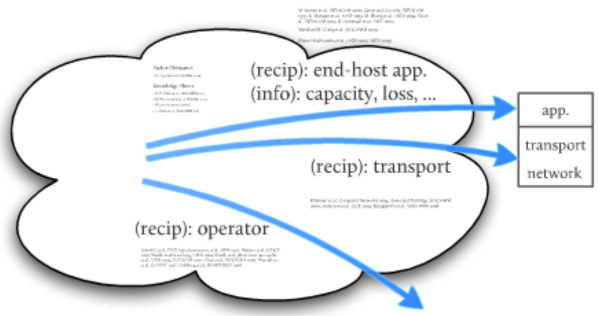
The ethos of ALF influenced other systems:

- The exokernel (D. A. Wallach, D. R. Engler, and M. F. Kaashoek, "ASHs: application-specific handlers for high-performance messaging," TON 1997).
- Plexus (M. E. Fiuczynski and B. N. Bershad, "An extensible protocol architecture for application-specific networking," USENIX ATC 1996).
- The Congestion Manager (H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for Internet hosts," SIGCOMM 1999).

Network monitoring

We organize this area according to two axes:

- Who is the intended recipient?
- What information is gathered?



Failure detection and recovery

Failure recovery is at the heart of the network's design, but researchers have proposed techniques for making the network even more robust:

- Failure-carrying packets (K. Lakshminarayanan et al., "Achieving convergence-free routing using failure-carrying packets," SIGCOMM 2007).
- SafeGuard (A. Li, X. Yang, and D. Wetherall, "SafeGuard: Safe forwarding during route changes," CoNEXT 2009).
- Data-driven connectivity (J. Liu, S. Shenker, M. Schapira, and B. Yang, "Data-driven network connectivity," HotNets 2011).

Failure detection was formalized by T. D. Chandra and S. Toueg ("Unreliable failure detectors for reliable distributed systems," JACM 1996).

- Many failure detectors use end-to-end timeouts to infer end-host crashes (Bertier et al. DSN 2002; Chen et al. 2002; Hayashibara et al., SRDS 2004; So and Sirer, EUROSYS 2007).
- Recently, Leners et al. proposed a failure detector based on layer-specific probes in the target ("Detecting failures in distributed systems with the FALCON spy network", SOSP 2011).

The opportunity

Failures are common and diverse, but ...

... the current Internet hides failures.

Today's interfaces to failures ...

What could we gain by exposing failures?

Questions and challenges

What should the interface be?

Research questions

Costs and deployment

Prior work

Application-specific networking

Network monitoring

We organize this area according to two axes:
- Who is the intended recipient?
- What information is gathered?

Failure detection and recovery