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.

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

Failure detection and recovery
The opportunity

Failures are common and diverse, but ...

... the current Internet hides 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.
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.
The opportunity

Failures are common and diverse, but ... the current Internet hides failures.

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

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

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

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.
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.)
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 ...
I incur delay.

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

Also, none of the aforementioned detects latent failures.
<table>
<thead>
<tr>
<th></th>
<th>OS crash</th>
<th>overloaded network</th>
<th>single link failure</th>
<th>multiple link failures</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>connect</code></td>
<td>21s</td>
<td>⊥</td>
<td>3.1s</td>
<td>3.6s</td>
</tr>
<tr>
<td><code>send</code></td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td><code>epoll</code></td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td><code>epoll, error Q</code></td>
<td>⊥</td>
<td>⊥</td>
<td>3.2s</td>
<td>3.6s</td>
</tr>
<tr>
<td><code>sendto</code></td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td><code>sendto, error Q</code></td>
<td>18s</td>
<td>⊥</td>
<td>⊥</td>
<td>3.5s</td>
</tr>
<tr>
<td>ICMP</td>
<td>20s</td>
<td>⊥</td>
<td>⊥</td>
<td>3.5s</td>
</tr>
</tbody>
</table>

⊥ 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 ...

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

application-specific networking

network monitoring

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

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

Could we gain by tracing 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.
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.
Knowledge of failures and their types would be useful in distributed key-value stores.
<table>
<thead>
<tr>
<th>Application</th>
<th>Host Failure</th>
<th>Congestion</th>
<th>Route Failure</th>
<th>Partition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>use alternate server</td>
<td>change primary probabilistically wait</td>
<td>use alternate server</td>
<td>use alternate server return error</td>
</tr>
<tr>
<td>NFS (soft mount)</td>
<td>return error</td>
<td>wait</td>
<td>wait</td>
<td></td>
</tr>
<tr>
<td>Paxos</td>
<td>immediate leader election</td>
<td>invoke election if majority report persistent congestion</td>
<td>wait</td>
<td>invoke leader election if majority report partition</td>
</tr>
<tr>
<td>Primary-backup</td>
<td>immediately failover</td>
<td>use slow failover</td>
<td>wait</td>
<td>use slower failover</td>
</tr>
<tr>
<td>RAMCloud</td>
<td>start recovery</td>
<td>wait</td>
<td>wait</td>
<td>start recovery</td>
</tr>
<tr>
<td>Cassandra</td>
<td>skip replica, report to op.</td>
<td>choose alt. primary replica</td>
<td>choose alt. primary replica</td>
<td>skip replica, report to op.</td>
</tr>
</tbody>
</table>
The opportunity

Failures are common and diverse, but... 
... the current Internet hides 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

Failure detection and recovery
Questions and challenges

What should the interface be?

What should the API be?

Research questions

Costs and deployment

We hypothesize that failure reporting can be cost-effective, if it triggers basic test and rollback policies.

We conjecture that deployment barriers will be lowered by software-defined networks.
What should the interface be?

What failure types should be exposed?

<table>
<thead>
<tr>
<th>A-list</th>
<th>B-list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host failures</td>
<td>Corruption</td>
</tr>
<tr>
<td>Congestion</td>
<td>Route instability</td>
</tr>
<tr>
<td>Route failure</td>
<td>Violation of routing policy</td>
</tr>
<tr>
<td>Partition</td>
<td></td>
</tr>
</tbody>
</table>

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 horizons 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 should the API be?

Research questions

Costs and deployment

We hypothesize that failure reporting can be cost-effective, if it triggers feedback 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 research with E. D. Clark and T. L. (unpublished) on AIP provides a foundation for a new generation of processors. SIGCOMM 1996.

The other side of AIP-enhanced open systems:

- The "Congestion Manager II" (K. Budiu, D. S. Rubenstein, H. S. Bairdi, and S. Seshan), an integrated congestion management architecture for Internet servers, SIGCOMM 1996.

Network monitoring

We organize this area according to two axes:

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

Failure detection and recovery

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


Failure detection was formalized by T. D. Chun and C. L. T. (Ethereal) failure detectors for reliable distributed systems. 1996.

- Many failure detectors use end-to-end timers to infer end-host reachability (Bertossi et al., DSN 1992, Chun et al. 2000, Fischman et al., SME'98, and A. M. 2000).
- Recently, Luecke et al. proposed a failure detector based on event-specific priori to the target ("Detecting failures in distributed systems with the FailCone protocol"), MOP 2000.
Application-specific networking

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


The ethos of ALF influenced other systems:


Network monitoring

We organize this area according to two axes:

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

(recip): end-host app.
(info): capacity, loss, ...

(recip): transport

(recip): operator

Packet Obitsuaries
- K. Sengupta et al., HotNets 2011

Knowledge Planes
- D.S. Chutjian et al., SIGCOMM 2009
- M. Wpentowski et al., HotNets 2011
- Reflection-based obits
- A. Shach et al., SIGCOMM 2009

Kilgour et al., Computer Networks 2004; Soma and Patridge, SIGCOMM 2000; Anderson et al., CCR 2001; Kargi et al., SIGCOMM 2008.

LaBarre et al., ITCSS 1999; Ianin et al., INFOCOM 2001; Watson et al., ITCSS 2000; Shach et al., ITCSS 2000; Shach et al., ITCSS 2001; Shach et al., ITCSS 2002; Shach et al., INFOCOM 2003; Zhou et al., SIGCOMM 2008; Dhindokh et al., CONEXT 2000; Godfrey et al., SIGMETRICS 2001.
(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.
(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?

(recip): end-host app.
(info): capacity, loss, ...

(recip): transport

(recip): operator

Packet Obitsuaries
- K. Regan et al., HotNets 2014

Knowledge Planes
- I. Stahl et al., SIGCOMM 2013
- M. Aboba et al., HotNets 2013
- RFC 7901 draft
- A. Black et al., SIGCOMM 2013


Merritman (K. Wong et al., SIGCOMM 2004).

iPlane (M. Fischetti et al., OSHI 2006; NSDI 2009).

Krilman et al., Computer Networks 2004; Soma and Patridge, SIGCOMM 2000; Anderson et al., CCR 2003; Karigiannis et al., SIGCOMM 2004.

K. Black et al., PICS 1997; A. Lawrence et al., IMW 1999; Watson et al., UdACS 2002; Stahl et al., GNS 2006; Stahl et al., JAC 2005; Korpela et al., NSDI 2005; INFOCOM 2007; Zhao et al., SIGCOMM 2000; Bhattacharya et al., ComNET 2000; Gohberg et al., SIGMETRICS 2000.
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 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 research begins with R. D. Clark's and D. L. Dhem's work on CIP (Communication Infrastructure for an Open Platform) and subsequent research for a new generation of protocols. (SIGCOMM 1996).

The ethos of CIP influenced other projects:

- Planet (M. L. Hornak, L. S. Frans, and W. N. Hurst), an extensible protocol architecture for application-transparent networking. (USENIX 1997).

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.

- Reliable transport (B. Liskov, M. Malis, et al.), including congestion avoidance protocols (TCP, SIGCOMM 1983).
- SafeGuard (S. S. Sengupta, D. Wetherall), SafeGuard: Safe forwarding during service changes. (HotOS 1996).


- Many failure detectors use an event to infer end-host crash (Berger et al., PDS 2003; Chandra et al., 2003; Huang/Huang et al., SIGCOMM 2004; and Sesh). (IEEE/OS 2007).
- Recently, Levent et al. proposed a failure detector based on an event-specific process in the target (“Detecting failures in distributed systems with the FAILOFF network”. SIGOPS 2001).
The opportunity

Failures are common and diverse, but... the current Internet hides 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 propose tools and techniques to reveal failures.

Failure detection and recovery

Network monitoring

We propose tools and techniques to reveal failures.