Blink: Fast Connectivity Recovery Entirely in the Data Plane

Joint work with

Edgar Costa Molero  ETH Zürich
Maria Apostolaki  ETH Zürich
Stefano Vissicchio  University College London
Alberto Dainotti  CAIDA, UC San Diego
Laurent Vanbever  ETH Zürich

Thomas Holterbach  ETH Zürich

NSDI  26th February 2019

https://blink.ethz.ch
Fire at AT&T facility causes widespread outage in North Texas

Nationwide internet outage affects CenturyLink customers

Time Warner Cable comes back from nationwide Internet outage

Major internet outage hits the U.S. - Affecting customers of Comcast, Verizon, and AT&T
AS level topology in 2015
AS level topology in 2015

Your network
AS level topology in 2015

Your network

Local

www.opte.org
AS level topology in 2015
Upon local failures, connectivity can be quickly restored
Upon local failures, connectivity can be quickly restored

Fast failure detection
using e.g., hardware-generated signals

Fast traffic rerouting
using e.g., Prefix Independent Convergence
or MPLS Fast Reroute
Upon remote failures, the only way to restore connectivity is to wait for the Internet to converge
Upon remote failures, the only way to restore connectivity is to wait for the Internet to converge

… and the Internet converges very slowly*

*Holterbach et al. SWIFT: Predictive Fast Reroute ACM SIGCOMM, 2017
Fire at AT&T facility causes widespread outage in North Texas

Time Warner Cable comes back from nationwide Internet outage

Major internet outage hits the U.S. - Affecting customers of Comcast, Verizon, and AT&T
BGP took *minutes* to converge upon the Time Warner Cable outage in 2014
BGP took *minutes* to converge upon the Time Warner Cable outage in 2014.
Control-plane (e.g., BGP) based techniques typically converge slowly upon remote outages
Control-plane (e.g., BGP) based techniques typically converge slowly upon remote outages

What about using data-plane signals for fast rerouting?
Blink: Fast Connectivity Recovery Entirely in the Data Plane

Thomas Holterbach
ETH Zürich

NSDI
26th February 2019

https://blink.ethz.ch
Outline

1. Why and how to use data-plane signals for fast rerouting

2. **Blink** infers more than *80%* of the failures, often within *1s*

3. **Blink** quickly reroutes traffic to *working* backup paths

4. **Blink** works in practice, on *existing* devices
Outline

1. Why and how to use data-plane signals for fast rerouting

2. Blink infers more than 80% of the failures, often within 1s

3. Blink quickly reroutes traffic to working backup paths

4. Blink works in practice, on existing devices
TCP flows exhibit the same behavior upon failures
TCP flows exhibit the **same** behavior upon failures
TCP flows exhibit the same behavior upon failures

source

S:500

A:1000

destination

failure
TCP flows exhibit the same behavior upon failures

![Diagram showing TCP flows and congestion window](image)
TCP flows exhibit the **same** behavior upon failures

**Retransmission timeout (RTO)**

\[ \text{RTO} = \text{SRTT} + 4 \times \text{RTT\_VAR} \]

- **RTO:** 200ms
TCP flows exhibit the **same** behavior upon failures

**Retransmission timeout (RTO)**

\[ RTO = SRTT + 4 \times RTT\_VAR \]

RTO: 200 ms
TCP flows exhibit the same behavior upon failures

Retransmission timeout (RTO) = SRTT + 4*RTT_VAR

RTO: 200ms

Exponential backoff

$t + 200\text{ms}$

$t + 600\text{ms}$
TCP flows exhibit the same behavior upon failures

Retransmission timeout (RTO) = $S_{RTT} + 4 \cdot RTT_{VAR}$

RTO: 200ms

$t + 200\text{ms}$

exponential backoff

$t + 600\text{ms}$

$t + 1400\text{ms}$

failure

cwnd: 4 pkts (=congestion window)
When multiple flows experience the same failure, the signal is a wave of retransmissions.
When multiple flows experience the same failure the signal is a wave of retransmissions

We simulated a failure affecting 100k flows with NS3

Same RTT distribution than in a real trace*

*CAIDA equinix-chicago direction A, 2015
When multiple flows experience the same failure, the signal is a **wave of retransmissions**

We simulated a failure affecting 100k flows with NS3

Same RTT distribution than in a real trace*

*CAIDA equinix-chicago direction A, 2015
When multiple flows experience the same failure, the signal is a wave of retransmissions.

We simulated a failure affecting 100k flows with NS3.

Same RTT distribution than in a real trace*

*CAIDA equinix-chicago direction A, 2015
When multiple flows experience the same failure, the signal is a **wave of retransmissions**

We simulated a failure affecting 100k flows with NS3.

Same RTT distribution than in a real trace*

*CAIDA equinix-chicago direction A, 2015
When multiple flows experience the same failure, the signal is a wave of retransmissions.

We simulated a failure affecting 100k flows with NS3.

Same RTT distribution than in a real trace.

*CAIDA equinix-chicago direction A, 2015
When multiple flows experience the same failure the signal is a wave of retransmissions

We simulated a failure affecting 100k flows with NS3

Same RTT distribution than in a real trace*

*CAIDA equinix-chicago direction A, 2015
When multiple flows experience the same failure, the signal is a wave of retransmissions.

We simulated a failure affecting 100k flows with NS3.

Same RTT distribution than in a real trace*.

4. Blink works in practice, on existing devices

1. Why and how to use data-plane signals for fast rerouting

2. Blink infers more than 80% of the failures, often within 1s

3. Blink quickly reroutes traffic to working backup paths

4. Blink works in practice, on existing devices
To detect failures, *Blink* looks at TCP retransmissions
To detect failures, Blink looks at TCP retransmissions. **Problem:** TCP retransmissions can be unrelated to a failure (*i.e.*, noise).
To detect failures, *Blink* looks at TCP retransmissions

**Problem**: TCP retransmissions can be unrelated to a failure (*i.e.*, noise)
To detect failures, *Blink* looks at TCP retransmissions

**Problem**: TCP retransmissions can be unrelated to a failure (*i.e.*, noise)
To detect failures, **Blink** looks at TCP retransmissions

**Problem**: TCP retransmissions can be unrelated to a failure (*i.e.*, noise)
To detect failures, *Blink* looks at TCP retransmissions.

**Problem**: TCP retransmissions can be unrelated to a failure (*i.e.*, noise).
Solution #1: *Blink* looks at consecutive packets with the same sequence number
Solution #1: **Blink** looks at consecutive packets with the same sequence number.

**Retransmission timeout (RTO)**

\[ RTO = SRTT + 4 \times RTT\_VAR \]

- **RTO: 200ms**
- **t + 200ms exponential backoff**
- **t + 600ms**
- **t + 1400ms**

---

![Diagram showing the process of retransmission timeout and exponential backoff](image-url)
Solution #2: *Blink* monitors the number of flows experiencing retransmissions over time using a sliding window.
Solution #2: *Blink* monitors the number of flows experiencing retransmissions over time using a sliding window.
Solution #2: Blink monitors the number of flows experiencing retransmissions over time using a sliding window.
Solution #2: *Blink* monitors the number of flows experiencing retransmissions over time using a sliding window.

![Graph showing number of retransmissions and flows experiencing retransmissions over time.](image)

- Number of retransmissions
- Number of flows experiencing retransmissions
- Congestions
- One "bogus" flow
- Failure
Solution #2: **Blink** monitors the number of flows experiencing retransmissions over time using a sliding window.
Solution #2: *Blink* monitors the number of flows experiencing retransmissions over time using a sliding window.
Solution #2: *Blink* monitors the number of flows experiencing retransmissions over time using a sliding window.
Solution #2: Blink monitors the number of flows experiencing retransmissions over time using a sliding window.
Solution #2: *Blink* monitors the number of flows experiencing retransmissions over time using a sliding window.
Blink is intended to run in programmable switches
Blink is intended to run in programmable switches

**Problem**: those switches have very limited resources
Solution #1: *Blink* focuses on the popular prefixes, *i.e.*, the ones that attract data traffic
Solution #1: *Blink* focuses on the popular prefixes, i.e., the ones that attract data traffic.

As Internet traffic follows a Zipf-like distribution* (1k pref. account for >50%), *Blink* covers the vast majority of the Internet traffic.

*Sarra et al. Leveraging Zipf’s Law for Traffic offloading ACM CCR, 2012*
Solution #2: *Blink* monitors a sample of the flows for each monitored prefix.
Solution #2: *Blink* monitors a sample of the flows for each monitored prefix.
To monitor active flows, *Blink* evicts a flow from the sample if it does not send a packet for a given time (default 2s)
To monitor active flows, *Blink* evicts a flow from the sample if it does not send a packet for a given time (default 2s)

and *selects* a new one in a *
first-seen, first-selected* manner
Blink infers a failure for a prefix when the majority of the monitored flows experience retransmissions
Blink infers a failure for a prefix when the majority of the monitored flows experience retransmissions.
Blink infers a failure for a prefix when the majority of the monitored flows experience retransmissions.
We evaluated *Blink* failure inference using 15 real traces, 13 from CAIDA, 2 from MAWI, covering a total of 15.8 hours.
We evaluated *Blink* failure inference using 15 real traces, 13 from CAIDA, 2 from MAWI, covering a total of 15.8 hours.

We are interested in:

**Accuracy**: True Positive Rate vs False Positive Rate

**Speed**: How long does Blink take to infer failures
As we do not have ground truth, we generated synthetic traces following the traffic characteristics extracted from the real traces.
As we do not have ground truth, we generated **synthetic traces** following the traffic characteristics extracted from the real traces

**Step #1** - We extracted the RTT, Packet rate, Flow duration from the real traces

**Step #2** - We used NS3 to replay these flows and simulate a failure

**Step #3** - We ran a Python-based version of *Blink* on the resulting traces


*Blink* failure inference accuracy is above 80% for 13 real traces out of 15

---

**True Positive Rate**

![Graph showing True Positive Rate vs Real traces ID]
*Blink* failure inference accuracy is above 80% for 13 real traces out of 15.
**Blink** avoids incorrectly inferring failures when packet loss is below 4%

<table>
<thead>
<tr>
<th>packet loss %</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive Rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Blink** avoids incorrectly inferring failures when packet loss is below 4%.

<table>
<thead>
<tr>
<th>packet loss %</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive Rate</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
<td>0.67</td>
<td>...</td>
<td>1.3</td>
<td>2.7</td>
</tr>
</tbody>
</table>
**Blink** infers a failure within 1s for the majority of the cases.
**Blink** infers a failure within 1s for the majority of the cases.
Outline

1. Why and how to use data-plane signals for fast rerouting

2. Blink infers more than 80% of the failures, often within 1s

3. Blink quickly reroutes traffic to working backup paths

4. Blink works in practice, on existing devices
Upon detection of a failure, **Blink** immediately activates backup paths pre-populated by the control-plane
Problem: since the rerouting is done entirely in the data-plane, *Blink* cannot prevent forwarding issues
**Problem:** since the rerouting is done entirely in the data-plane, *Blink* cannot prevent forwarding issues
Problem: since the rerouting is done entirely in the data-plane, *Blink* cannot prevent forwarding issues.
**Problem:** since the rerouting is done entirely in the data-plane, *Blink* cannot prevent forwarding issues.
**Problem:** since the rerouting is done entirely in the data-plane, *Blink* cannot prevent forwarding issues.

![Diagram showing network topology and problem statement](image)
Solution: As for failures, *Blink* uses data-plane signals to pick a working backup path.
Solution: As for failures, **Blink** uses data-plane signals to pick a **working** backup path.

The probing period lasts up to 1s.
Solution: As for failures, Blink uses data-plane signals to pick a working backup path.
As for failures, *Blink* compares the sequence number of consecutive packets to detect blackholes or loops*

*See the paper for an evaluation of the rerouting*
Outline

1. Why and how to use data-plane signals for fast rerouting

2. Blink infers more than 80% of the failures, often within 1s

3. Blink quickly reroutes traffic to working backup paths

4. Blink works in practice, on existing devices
We ran $\textit{Blink}$ on the 15 real traces (15.8 hours)
We ran *Blink* on the 15 real traces (15.8 hours) and it detected 6 outages, each affecting *at least* 42% of *all* the flows.
On current programmable switches, Blink supports up to 10k prefixes
On current programmable switches, Blink supports up to 10k prefixes.
On current programmable switches, Blink supports up to 10k prefixes.
On current programmable switches, **Blink** supports up to 10k prefixes.
*Blink* works on a real *Barefoot* Tofino switch
Blink works on a real Barefoot Tofino switch
Blink works on a real Barefoot Tofino switch
**Blink** works on a real Barefoot Tofino switch

![Graph showing number of packets every 100ms and RTTs in [10ms; 300ms]]
Blink works on a real Barefoot Tofino switch
Blink works on a real Barefoot Tofino switch
**Blink**: Fast Connectivity Recovery Entirely in the Data Plane

Infers failures from data-plane signals with more than 80% accuracy, and often within 1s

Fast reroutes traffic at line rate to working backup paths

Works on real traffic traces and on existing devices

https://blink.ethz.ch
**Blink**: Fast Connectivity Recovery Entirely in the Data Plane

Thomas Holterbach  
ETH Zürich

**NSDI**  
26th February 2019

Joint work with

- **Edgar Costa Molero**  
  ETH Zürich
- **Maria Apostolaki**  
  ETH Zürich
- **Stefano Vissicchio**  
  University College London
- **Alberto Dainotti**  
  CAIDA, UC San Diego
- **Laurent Vanbever**  
  ETH Zürich
When multiple flows experience the same failure the signal is a wave of retransmissions

We simulated a failure affecting 100k flows with NS3

Same RTT distribution than in a real trace*

*CAIDA equinix-chicago direction A, 2015
When multiple flows experience the same failure, the signal is a wave of retransmissions.
**Blink** failure inference accuracy is close to a best case scenario, and is above 80% for 13 real traces out of 15.
**Blink** infers a failure within 1s for the majority of the cases

"best case", i.e., no sampling but threshold still 32
**Blink** avoids incorrectly inferring failures when packet loss is below 4%.

<table>
<thead>
<tr>
<th>packet loss %</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Blink</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.67</td>
<td>0.67</td>
<td>...</td>
<td>1.3</td>
<td>2.7</td>
</tr>
<tr>
<td>no sampling but threshold still 32</td>
<td>59</td>
<td>85</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>...</td>
<td>97</td>
<td>98</td>
</tr>
</tbody>
</table>
**Blink** quickly infers and avoids forwarding loops

![Graph](image)

Number of packets every 100ms