Supercharged me: Boost Router Convergence with SDN

Thomas Holterbach$^1$

Joint work with Michael Alan Chang$^{1,2}$, Markus Happe$^1$ and Laurent Vanbever$^1$

$^1$ETH Zurich, $^2$Princeton University

University of Strasbourg
ICube laboratory, Network Research Group

June 12, 2015
Plan

1. SDNs
2. Context
3. Today's Convergence time
4. Supercharged Convergence
5. Evaluation
6. Conclusion
Motivations

- Current networks are composed of different types of equipment
  - Routers, switches, firewall, etc
- These equipments are closed
  - Software bundled with hardware
  - Vendor-specific interfaces
- Equipment vendors write the code
- Slow protocol standardization
Motivations

- Routers, switches, etc compute paths the packets will follow thanks to protocols such as OSPF, BGP, STP, etc
  - control plane
- and direct a data packet to an outgoing link accordingly
  - data plane
- Control plane and data plane are coupled in each device
- Decentralized architecture
Motivations

- What if I want to customize my routing and manage my network?
- BGP attributes: local-pref, MED, community, etc
- OSPF: link weights
- Traffic Engineering with MPLS, ACL, etc
Motivations

- What if I want to customize my routing and manage my network?
- BGP attributes: local-pref, MED, community, etc
- OSPF: link weights
- Traffic Engineering with MPLS, ACL, etc

Problems

- Complex and hard to manage
  - vendor specific configurations, hard to debug
- Limitations
  - I can only do things that have been implemented by the vendor
Software Defined Networks

- Decouple the control and the data plane
- Enable programmable networks and support flexible network management

I program *by myself* what I want to do in my **Software Defined** Network
Software Defined Networks

- Decouple the control and the data plane
- Enable programmable networks and support flexible network management

I program *by myself* what I want to do in my **Software Defined Network**
Openflow

- Openflow is a communication protocol
- It gives access to the forwarding plane of switch
- OpenFlow allows remote administration of a switch’s packet forwarding tables
- It can add, modify and remove packet matching rules and actions
SDN Controllers

- Openflow is low-level
- Controllers allow us to express high-level policies
- Controllers relay information between the switch and the applications
The controller populates the switch ahead of time

match(dst_ip = 10.0.0.0/24, fwd : 1)
The controller populates the switch ahead of time

match(dst_ip = 10.0.0.0/24, fwd : 1)
Reactive Flows

- Reactive mode reacts to traffic, consults the OpenFlow controller and creates a rule in the flow table based on the instruction.
- The first packet of a flow is processed by the controller.
- The following stay in the data plane and are forwarded at line rate.
Reactive Flows

- Reactive mode reacts to traffic, consults the OpenFlow controller and creates a rule in the flow table based on the instruction.
- The first packet of a flow is processed by the controller.
- The following stay in the data plane and are forwarded at line rate.
Reactive Flows

- Reactive mode reacts to traffic, consults the OpenFlow controller and creates a rule in the flow table based on the instruction.
- The first packet of a flow is processed by the controller.
- The following stay in the data plane and are forwarded at line rate.

match(dst_ip = 10.0.0.0/24, fwd : 1)
Reactive Flows

- Reactive mode reacts to traffic, consults the OpenFlow controller and creates a rule in the flow table based on the instruction.
- The first packet of a flow is processed by the controller.
- The following stay in the data plane and are forwarded at line rate.

```
match(dst_ip = 10.0.0.0/24, fwd : 1)
```
Reactive Flows

- Reactive mode reacts to traffic, consults the OpenFlow controller and creates a rule in the flow table based on the instruction.
- The first packet of a flow is processed by the controller.
- The following stay in the data plane and are forwarded at line rate.

```
match(dst_ip = 10.0.0.0/24, fwd : 1)
```
Flow entry description

- **Rule**
  - Switch Port
  - MAC src
  - MAC dst
  - Eth type
  - VLAN ID
- **Action**
  - IP Src
  - IP Dst
  - IP Prot
  - TCP sport
  - TCP dport
- **Stats**
  - Packet + byte counters
- **Actions**
  1. Forward packet to port(s)
  2. Encapsulate and forward to controller
  3. Drop packet
  4. Send to normal processing pipeline
Switches hardware and Openflow evolve and provide more and more features. For example:

- multiple flow-tables can be used consecutively (Openflow pipeline)
- "don’t care" state possible on any bit of an IP address

The flow table size in the switches also increases with the time

- As of June 2015, tens of thousands entries maximum
Plan

1. SDNs
2. Context
3. Today’s Convergence time
4. Supercharged Convergence
5. Evaluation
6. Conclusion
Convergence time

When the network changes, routers must adapt their forwarding decisions accordingly.

Convergence process:

1. Detect Change
2. Propagate LSP’s or Adver./Withdraw
3. Recompute RIB
4. Update FIB
Convergence time

Convergence process:

- When the network changes, routers must adapt their forwarding decisions accordingly.
- Bidirectional Forwarding Detection (BFD) protocol: enables very fast failure detection.
When the network changes, routers must adapt their forwarding decisions accordingly.

Bidirectional Forwarding Detection (BFD) protocol: enables very fast failure detection.

We consider only direct neighbor link or node failure.
Convergence time

Convergence process:

1. Detect Change
2. Propagate LSP’s or Adver./Withdraw
3. Recompute RIB
4. Update FIB

- When the network changes, routers must adapt their forwarding decisions accordingly
- Bidirectional Forwarding Detection (BFD) protocol enables very fast failure detection
- We consider only direct neighbor link or node failure
- We focus only on the last two steps
When the network changes, routers must adapt their forwarding decisions accordingly. Bidirectional Forwarding Detection (BFD) protocol enables very fast failure detection. We consider only direct neighbor link or node failure. We focus only on the last two steps. Experiments with real hardware show that these two steps may last several minutes. During this time, traffic is lost ...
When the network changes, routers must adapt their forwarding decisions accordingly.

Bidirectional Forwarding Detection (BFD) protocol: enables very fast failure detection.

We consider only direct neighbor link or node failure.

We focus only on the last two steps.

Experiments with real hardware show that these two steps may last several minutes.

During this time, traffic is lost ...

We introduce a solution where routers always converge within 150ms.
Plan

1. SDNs
2. Context
3. Today’s Convergence time
4. Supercharged Convergence
5. Evaluation
6. Conclusion
Today's (slow) Convergence

AS 100
R1

512k IPv4 prefixes

512k IPv4 prefixes

R2
AS 200
IP: 203.0.113.1
MAC: 01:aa

R3
AS 300
IP: 198.51.100.2
MAC: 02:bb

$
Today’s (slow) Convergence

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>(L2 NH, port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>
Today's (slow) Convergence

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>(L2 NH, port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>
# Today's (slow) Convergence

## R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>(L2 NH, port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>

## Network Diagram

- **R1** (router) FIB
- **AS 100**
- **R2**
  - IP: 203.0.113.1
  - MAC: 01:aa
- **R3**
  - IP: 198.51.100.2
  - MAC: 02:bb
- **AS 300**

- 512k IPv4 prefixes
- 512k IPv4 prefixes
Today’s (slow) Convergence

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>(L2 NH, port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>
Today’s (slow) Convergence

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>(L2 NH, port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>
Today’s (slow) Convergence

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>(L2 NH, port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(01:aa, 0)</td>
</tr>
</tbody>
</table>

R1 → R2: 512k IPv4 prefixes

R3 → 0: 512k IPv4 prefixes

AS 100

AS 200

AS 300
Today's (slow) Convergence

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>(L2 NH, port)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(02:bb, 1)</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(02:bb, 1)</td>
</tr>
</tbody>
</table>
Today’s (slow) Convergence

- FIB entries are updated one-by-one
- The convergence time is linearly proportional to the number of prefixes
Today’s (slow) Convergence

- FIB entries are updated one-by-one
- The convergence time is linearly proportional to the number of prefixes

- As the FIB grows, so does the convergence time
- Updating 500K prefixes may take several minutes ..
BGP Prefix Independent Convergence (PIC)

- Enables sub-second data-plane convergence
- Introduces the idea of using a hierarchical FIB design

C. FilsFils, P. Mohapatra, J. Bettink, P. Dharwadkar, P. De Vriendt, Y. Tsier, V. Van Den Schriek, O. Bonaventure, and P. Francois
BGP Prefix Independent Convergence (PIC)
BGP Prefix Independent Convergence (PIC)

- Enables sub-second data-plane convergence
- Introduces the idea of using a hierarchical FIB design

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(R2)</td>
</tr>
</tbody>
</table>
BGP Prefix Independent Convergence (PIC)

- Enables sub-second data-plane convergence
- Introduces the idea of using a hierarchical FIB design

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(R2)</td>
</tr>
</tbody>
</table>

- Update of the outgoing interface for a BGP nexthop
- Impacts all prefixes tracked by the nexthop
BGP Prefix Independent Convergence (PIC)

- Enables sub-second data-plane convergence
- Introduces the idea of using a hierarchical FIB design

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(R2)</td>
</tr>
</tbody>
</table>

- Update of the outgoing interface for a BGP nexthop
- Impacts all prefixes tracked by the nexthop
BGP Prefix Independent Convergence (PIC)

- Enables sub-second data-plane convergence
- Introduces the idea of using a hierarchical FIB design

R1 (router) FIB

<table>
<thead>
<tr>
<th>pos.</th>
<th>prefix</th>
<th>NH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0.0.0/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>2</td>
<td>1.0.0.1/24</td>
<td>(R2)</td>
</tr>
<tr>
<td>3</td>
<td>1.0.0.2/24</td>
<td>(R2)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>256k</td>
<td>100.0.0.1/24</td>
<td>(R2)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>512k</td>
<td>200.0.0.1/24</td>
<td>(R2)</td>
</tr>
</tbody>
</table>

- Update of the outgoing interface for a BGP nexthop
- Impacts all prefixes tracked by the nexthop

Problems

- Not available on all routers
- Requires expensive line-cards update
Plan

1. SDNs
2. Context
3. Today’s Convergence time
4. Supercharged Convergence
5. Evaluation
6. Conclusion
Supercharging Router R1: Idea

- Connect a SDN switch to the router and install a controller (Floodlight, POX, etc)
Supercharging Router R1: Idea

When R2 becomes down ...
Supercharging Router R1: Idea

- When R2 becomes down ...
- ...push a rule that directs traffic towards R3 rather than R2
Challenges

- What if there are two possible backup routers (R3 and R4)
- R3 advertises 256K prefixes and R4 advertises the other prefixes
Challenges

- What if there are two possible backup routers (R3 and R4)
- R3 advertises 256K prefixes and R4 advertises the other prefixes
Challenges

- What if there are two possible backup routers (R3 and R4)
- R3 advertises 256K prefixes and R4 advertises the other prefixes

```
match(256k, fwd : 1 )
match(256k', fwd : 2 )
```
Challenges

- How to match a subset of the traffic?

  \[
  \text{match}(256k, \text{fwd} : 1) \\
  \text{match}(256k', \text{fwd} : 2)
  \]

- One rule per prefix advertised by R2 would produce too many rules
We use a BGP daemon to catch advertisements/withdraws.
Solution

- Thanks to the BGP deamon, we are able to compute *backup-groups*

**Backup-group definition**

- A backup-group is the couple (primary NH, backup NH)
- Many IP prefixes share the same primary and backup Next-Hop

- We match each IP prefix with a backup-group
Thanks to the BGP deamon, we are able to compute *backup-groups*

**Backup-group definition**

- A backup-group is the couple (primary NH, backup NH)
- Many IP prefixes share the same primary and backup Next-Hop

We match each IP prefix with a backup-group
Solution

- We assign a Virtual Next-Hop IP address and MAC Address on each backup-group.
- We program a module on the controller that answers ARP requests for VNH.

Backup-groups:
- R2, R3: 256k prefixes (10.0.0.1, 99:01)
- R2, R4: 256k prefixes (10.0.0.2, 99:02)
- R3, R2: (10.0.0.3, 99:03)
- R3, R4: (10.0.0.4, 99:04)
- R4, R2: (10.0.0.5, 99:05)
- R4, R3: (10.0.0.6, 99:06)
Solution

- We assign a Virtual Next-Hop IP address and MAC Address on each backup-group.
- We program a module on the controller that answers ARP requests for VNH.

We can use the router to *tag* the traffic according to the backup group it belongs to.

and use the switch to *redirect* the tagged traffic to the master or backup NH, depending on its status.
Example

- R2 advertises 512k prefixes
- R3 advertises half of these 512k prefixes
- R4 advertises the other half
The BGP daemon modifies the next-hop of each advertisement according to the backup group the advertised prefix belongs to.
Example

- When the router wants to send a packet, it first sends an ARP request.
Example

- The ARP Handler in the controller replies the Virtual NH Mac Address
We direct the traffic according to its tag
Both VNH MAC 99:01 and VNH MAC 99:02 to R2

Flow-table:
match (eth_dst : 99:01) actions (fwd : 0, eth_dst : 01:aa)
match (eth_dst : 99:02) actions (fwd : 0, eth_dst : 01:aa)
Example

When R2 becomes down ...

Flow-table:
match (eth_dst : 99:01) actions (fwd : 0, eth_dst : 01:aa)
mismatch (eth_dst : 99:02) actions (fwd : 0, eth_dst : 01:aa)
We push new rules that direct the traffic tagged with the VNH MAC 99:01 and VNH 99:02 to their backup routers.

Flow-table:
- match (eth_dst : 99:01) actions (fwd : 0, eth_dst : 01:aa)
- match (eth_dst : 99:02) actions (fwd : 0, eth_dst : 01:aa)
- match (eth_dst : 99:01) actions (fwd : 1, eth_dst : 02:bb)
- match (eth_dst : 99:02) actions (fwd : 2, eth_dst : 03:cc)
The FIB becomes hierarchical (such as PIC) but spanning two devices. The convergence is therefore drastically reduced.
Plan

1. SDNs
2. Context
3. Today’s Convergence time
4. Supercharged Convergence
5. Evaluation
6. Conclusion
Evaluation

- We evaluate the convergence time of a recent router prior and after supercharging it

**Hardware used**

- 3 routers Cisco Nexus 7K
- 1 switch HP E3800 Openflow enabled
- 2 FPGAs for the traffic generation

- We always use the BFD protocol
- We evaluate the convergence with different number of advertised prefixes
  - 1K, 5K, 10K, 50K, 100K, 200K, 300K, 400K, 500K
- We repeated the experiment 3 times per number of advertised prefixes
Traffic generation

- Custom-built hardware-based traffic generator using FPGAs
- One FPGA source and one FPGA sink
- 100 different IP addresses used
  - The convergence time differs between the IP addresses
  - We include the first and the last IP address in the lexical order
- The FPGA source sends one packet per flow every $70\mu s$
- We measure the convergence time by monitoring the maximum inter-packet delay seen by each flow between the two FPGAs
- The convergence time is then measured with a precision of $70\mu s$
Non-supercharged Router
Non-supercharged Router

Convergence time (s) - log scale

Number of prefixes

1K 5K 10K 50K 100K 200K 300K 400K 500K

Convergence time (s) - log scale

0.9 1.6 3.4 13.8 29.2 56.9 86.4 113.1 140.9
non-supercharged
Non-supercharged Router

- The convergence time is roughly linear in the number of prefixes in the FIB
- The first entry is updated almost immediately
- The last entry updated must wait for all the preceding FIB entries to be updated

The non-supercharged router takes close than 2.5min to converge when loaded with 512K prefixes
Supercharged Router

- Cisco Nexus 7K
- R1
- HP E3800 Openflow enabled
- ExaBGP
  - Floodlight
- R2
- R3
- 512k IPv4 prefixes
- Convergence time
Setup

- Supercharged controller
- Cisco Nexus 7K
- FPGAs
- Supercharged Router
- HP E3800
  Openflow enabled
Supercharged Router

![Box plot comparison of convergence times between supercharged and non-supercharged routers for varying numbers of prefixes.](image)

- **Supercharged Router**
- **Convergence time** (s) - log scale
- **Number of prefixes**: 1K, 5K, 10K, 50K, 100K, 200K, 300K, 400K, 500K
- **Convergence time**
  - **1K**: 0.9 ms
  - **5K**: 1.6 ms
  - **10K**: 3.4 ms
  - **50K**: 13.8 ms
  - **100K**: 29.2 ms
  - **200K**: 56.9 ms
  - **300K**: 86.4 ms
  - **400K**: 113.1 ms
  - **500K**: 140.9 ms

**Conclusion**
- Supercharged routers achieve significantly lower convergence times compared to non-supercharged routers.
The supercharged router *always converges within 150ms*, for all prefixes and irrespective of the number of prefixes.

- This constitutes a 900x improvement factor.
- The worst-case convergence time of a supercharged router is still more than two times faster than the best-case convergence time of its standalone counterpart (375ms).
The supercharged router always converges within 150 ms
  - regardless of the total number of prefixes to update

This constitutes a 900x improvement factor

Supercharging a router only requires
  - a low-cost OpenFlow device
  - and a lightweight reconfiguration of the supercharged router

Current SDN switches are orders of magnitude cheaper than fully-equipped routers

Benefits observed with only one SDN switch

Provides new incentives for network operators to kickstart SDN deployment
Further work

Reminder
We consider only direct neighbor link or node failure

- Can we supercharge the convergence time when a link or node *not* directly connected to the supercharged router becomes down?
Further work
Can we supercharge this convergence time?

Further work
Can we supercharge this convergence time?

Further work
Further work

- We plan to supercharge the FIB size of router with the help of a SDN switch
  - The FIB is splitted between the router and the SDN switch
  - The router takes care of the most used prefixes, the SDN switch takes care of the other prefixes
- We plan to supercharge the load-balancing
Thank you!