Augmenting Anycast Network Flows

Sebastian Brandt, Klaus-Tycho Förster, Roger Wattenhofer
January 06, 2016 @ ICDCN 2016 - Singapore
size of each flow: 1
capacity of links: 1
Motivation

Size of each flow: 1
Capacity of links: 1
Motivation

size of each flow: 1
capacity of links: 1
Network Updates

• The Internet: Designed for selfish participants
  – Often inefficient (low utilization of links), but robust

• But what happens if the WAN is controlled by a single entity?
  – Examples: Microsoft & Amazon & Google ...
  – They spend hundreds of millions of dollars per year
Network Updates

Think: Google, Amazon, Microsoft
Network Updates

• The Internet: Designed for selfish participants
  – Often inefficient (low utilization of links), but robust

• But what happens if the WAN is controlled by a single entity?
  – Examples: Microsoft & Amazon & Google ...
  – They spend hundreds of millions of dollars per year

• Possible solution: **Software Defined Networking (SDN)**

• General Idea: Separate data & control plane in a network
• Centralized controller updates networks rules for optimization
  – Controller (*control plane*) updates the switches/routers (*data plane*)
Motivation

size of each flow: 1
capacity of links: 1
Motivation

size of each flow: 1
capacity of links: 1
Motivation

size of each flow: 1
capacity of links: 1
Motivation

- size of each flow: 1
- capacity of links: 1

network updates
Motivation

size of each flow: 1
capacity of links: 1
Motivation

network updates

size of each flow: 1
capacity of links: 1
Motivation

network updates

size of each flow: 1
capacity of links: 1
Structure of the Talk

• Motivation & Software Defined Networking
• Related Work & Splittable Flows
• Our Approach
• Extension beyond Anycast Flows
Network Updates of Flows without Congestion

- **State of the art:** (Partial) moves of flows using linear programming (LPs), e.g.,
  - *SWAN* [Hong et al., SIGCOMM 2013], *zUPDATE* [Liu et al., SIGCOMM 2013]
  - *Dionysus* [Jin et al., SIGCOMM 2014]

- **Open problems:**
  - When are network updates without congestion *possible*?
  - How can we do them *fast*?

- **This paper:** Addresses the case of one (logical) destination for splittable flows
Swapping of Flows

size of each flow: 2
capacity of links: 3
Just Switch? Congestion!

size of each flow: 2

capacity of links: 3
Migrate only parts of the flow

size of each flow: 2
capacity of links: 3
Can even do both flows at once

size of each flow: 2
capacity of links: 3
Done in two steps

size of each flow: 2
capacity of links: 3
But not always possible!

size of each flow: 2
capacity of links: 2
How about other paths?

Number of steps can be unbounded

size of each flow: 2
capacity of links: 2
How about other paths?

Number of steps can be **unbounded**

Binary search with LPs **ineffective**

size of each flow: 2
capacity of links: 2
Structure of the Talk

• Motivation & Software Defined Networking
• Related Work & Splittable Flows
• Our Approach
• Extension beyond Anycast Flows
Our Approach

• Compute our own new rules
  – Based off the new demands

• Deviate from linear programming binary search
  – Go combinatorial with augmenting flows
    • “Push back” flows for migration
Augmenting Flows

size of each flow: 1
capacity of links: 1
Consider Residual Network

*size of each flow: 1
*capacity of links: 1
Find a Way in the Residual Network

size of each flow: 1
capacity of links: 1
Push back the old Flow

size of each flow: 1
capacity of links: 1
Insert the new Flow

size of each flow: 1
capacity of links: 1
Migrated without Congestion

size of each flow: 1
capacity of links: 1
Similar to “Addition”

size of each flow: 1
capacity of links: 1
Also works as “Subtraction”

size of each flow: 1
capacity of links: 1
High-level Mechanism Idea

For all commodities (iteratively):
  – Increase demand and calculate new flow with LP
    • offline calculation
  – Apply augmenting flow from the difference
    • linear # re-routing in the network
High-level Mechanism Idea

For all commodities (iteratively):
  - Increase demand and calculate new flow with LP
    • *offline* calculation
  - Apply augmenting flow from the difference
    • linear # re-routing *in the network*

<table>
<thead>
<tr>
<th>OLD</th>
<th>CURRENT</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>
High-level Mechanism Idea

For all commodities (iteratively):

- Increase demand and calculate new flow with LP
  - offline calculation
- Apply augmenting flow from the difference
  - linear # re-routing in the network

<table>
<thead>
<tr>
<th>OLD</th>
<th>CURRENT</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>
High-level Mechanism Idea

For all commodities (iteratively):

- Increase demand and calculate new flow with LP
  - *offline* calculation
- Apply augmenting flow from the difference
  - linear # re-routing *in the network*

<table>
<thead>
<tr>
<th>OLD</th>
<th>CURRENT</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>
High-level Mechanism Idea

For all commodities (iteratively):
  – Increase demand and calculate new flow with LP
    • offline calculation
  – Apply augmenting flow from the difference
    • linear # re-routing in the network

<table>
<thead>
<tr>
<th>OLD</th>
<th>CURRENT</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>
High-level Mechanism Idea

For all commodities (iteratively):

– Increase demand and calculate new flow with LP
  • offline calculation

– Apply augmenting flow from the difference
  • linear # re-routing in the network

<table>
<thead>
<tr>
<th>OLD</th>
<th>CURRENT</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>
High-level Mechanism Idea

For all commodities (iteratively):

- Increase demand and calculate new flow with LP
  - offline calculation
- Apply augmenting flow from the difference
  - linear # re-routing in the network

<table>
<thead>
<tr>
<th>OLD</th>
<th>CURRENT</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
Number of “Push Back” - Links decreases

size of flows: 1, 2, 1, 1
capacity of links: 1 (or marked)
Number of “Push Back”- Links decreases

size of flows: 1, 2, 1, 1
capacity of links: 1 (or marked)
Number of “Push Back”- Links decreases

size of flows: 1, 2, 1, 1
capacity of links: 1 (or marked)
Number of “Push Back”- Links decreases

size of flows: 1, 2, 1, 1

capacity of links: 1 (or marked)
Number of “Push Back”- Links decreases

size of flows: 1, 2, 1, 1

capacity of links: 1 (or marked)
Number of “Push Back” - Links decreases

size of flows: 1+3=4, 2, 1, 1
capacity of links: 1 (or marked)
Structure of the Talk

• Motivation & Software Defined Networking
• Related Work & Splittable Flows
• Our Approach
• Extension beyond Anycast Flows
Flow Augmentation for many Destinations

- Flows end up at the wrong destination!
- So let’s stick with augmenting flows that don’t mix destinations
Extension beyond one logical destination?

size of each flow: 1
capacity of each links: 1
Augmenting flows that don’t mix up the destinations?

size of each flow: 1
capacity of each links: 1
Augmenting flows that don’t mix up the destinations?

- $s_1$ connected to $t_1$
- $s_2$ connected to $t_2$
- $s_3$ connected to $t_3$

- Size of each flow: 1
- Capacity of each link: 1
But impossible to migrate!

size of each flow: 1
capacity of each links: 1
But impossible to migrate!

“It is unlikely that similar techniques can be developed for constructing multicommodity flows”

[Hu, 1963]

size of each flow: 1
capacity of each links: 1
Summary

• We studied how to migrate flows with one logical destination in SDNs without congestion
  – We can decide fast if demands can be met
  – We can migrate with linear # re-routings in the network using augmenting flows

• Open question:
  – How to extend beyond one logical destination?
Thank you

Sebastian Brandt, **Klaus-Tycho Förster**, Roger Wattenhofer
January 06, 2016 @ ICDCN 2016 - Singapore