Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures

Klaus-T. Foerster (U. Vienna), Manya Ghobadi (Microsoft Research), Stefan Schmid (U. Vienna)

23 July 2018, IEEE/ACM ANCS 2018
Reconfigurable Data Center Networks (DCNs)

- **Helios** (core)
  Farrington et al., SIGCOMM ‘10

- **c-Through** (HyPaC architecture)
  Wang et al., SIGCOMM ‘10

- **ProjecToR** interconnect
  Ghobadi et al., SIGCOMM ‘16

- **Rotornet** (rotor switches)
  Mellette et al., SIGCOMM ‘17

- **Solstice** (architecture & scheduling)
  Liu et al., CoNEXT ‘15

- **REACToR**
  Liu et al., NSDI ‘15

- **FireFly**
  Hamedazimi et al., SIGCOMM ‘14

... and many more ...
Reconfigurable Data Center Networks (DCNs)
Reconfigurable Data Center Networks (DCNs)

• Results and conclusions often not portable
  ◦ Between topologies/technologies
Reconfigurable Data Center Networks (DCNs)

- Results and conclusions often not portable
  - Between topologies/technologies

- Assumption in routing takes away optimality
Reconfigurable Data Center Networks (DCNs)

• Results and conclusions often not portable
  ◦ Between topologies/technologies

• Assumption in routing takes away optimality

• We take a look from a theoretical perspective
Reconfigurable Data Center Networks (DCNs)

• Results and conclusions often not portable
  ◦ Between topologies/technologies

• Assumption in routing takes away optimality

• We take a look from a theoretical perspective
  ◦ With average path length as an objective
Reconfigurable Data Center Networks (DCNs)

• Results and conclusions often not portable
  ◦ Between topologies/technologies

• Assumption in routing takes away optimality

• We take a look from a theoretical perspective
  ◦ With average path length as an objective
  ◦ For one switch (with/without this assumption)
  ◦ Also briefly for multiple switches
The Static Case
The Static Case

Communication frequency: $A \rightarrow E: 10$, $A \rightarrow G: 5$
The Static Case

Communication frequency: $A \rightarrow E: 10$, $A \rightarrow G: 5$
The Static Case

Communication frequency: $A\rightarrow E: 10$, $A\rightarrow G: 5$

Weighted average path length: $4 \times 10 + 6 \times 5 = 70$
Adding Reconfigurability

Communication frequency: \(A \rightarrow E: 10, \ A \rightarrow G: 5\)

Weighted average path length: \(4 \times 10 + 6 \times 5 = 70\)
Adding Reconfigurability

Communication frequency: \( A \rightarrow E: 10, A \rightarrow G: 5 \)

Weighted average path length: \( 4 \times 10 + 6 \times 5 = 70 \)
Adding Reconfigurability

Communication frequency: $A \rightarrow E: 10$, $A \rightarrow G: 5$

Weighted average path length: $4 \times 10 + 6 \times 5 = 70$
Adding Reconfigurability

Communication frequency: $A \rightarrow E: 10$, $A \rightarrow G: 5$

Weighted average path length: $4 \times 10 + 6 \times 5 = 70$
Adding Reconfigurability

Weighted average path length: $1 \times 10 + 6 \times 5 = 40$

Communication frequency: $A \rightarrow E: 10$, $A \rightarrow G: 5$

Weighted average path length: $4 \times 10 + 6 \times 5 = 70$

static
Adding Reconfigurability

Weighted average path length: $1 \times 10 + 6 \times 5 = 40$

Communication frequency: $A \rightarrow E: 10$, $A \rightarrow G: 5$

Weighted average path length: $4 \times 10 + 6 \times 5 = 70$

reconfig

static
Adding Reconfigurability

Weighted average path length: \(1 \times 10 + 6 \times 5 = 40\)

\[\text{optimum} \quad 1 \times 10 + (1+2) \times 5 = 25\]

Communication frequency: \(A \rightarrow E: 10, \ A \rightarrow G: 5\)

Weighted average path length: \(4 \times 10 + 6 \times 5 = 70\)

\[\text{static} \]
Adding Reconfigurability

Weighted average path length: \(1 \times 10 + 6 \times 5 = 40\)  
\(1 \times 10 + (1+2) \times 5 = 25\)

Communication frequency: \(A \rightarrow E: 10, A \rightarrow G: 5\)

Weighted average path length: \(4 \times 10 + 6 \times 5 = 70\)
Beyond a Single Switch

• Especially important at scale: **multiple** reconfigurable switches

*Rotornet*

Mellette *et al.*, SIGCOMM ‘17

*A Tale of Two Topologies*

Xia *et al.*, SIGCOMM ‘17
One Switch: Segregated Routing Policies
One Switch: Segregated Routing Policies

• Model: Either just 1 reconfig or just static
One Switch: Segregated Routing Policies

- Model: Either just 1 reconfig or just static

Communication frequency: \(A \rightarrow E: 10, A \rightarrow G: 5\)
One Switch: Segregated Routing Policies

- Model: Either just 1 reconfig or just static

Communication frequency: \( A \rightarrow E: 10 \), \( A \rightarrow G: 5 \)
One Switch: Segregated Routing Policies

● Model: Either just 1 reconfig or just static

Why this solution?

**Benefit of A→E: 10:**
• Static-Reconfig: 40-10=30

**Benefit of A→G: 5:**
• Static-Reconfig: 30-5=25

Communication frequency: A→E: 10, A→G: 5
One Switch: Segregated Routing Policies

• Model: Either just 1 reconfig or just static
One Switch: Segregated Routing Policies

• Model: Either just 1 reconfig or just static

• Optimal solution in polynomial time:
One Switch: Segregated Routing Policies

• Model: Either just 1 reconfig or just static

• Optimal solution in polynomial time:
  1. Compute & assign benefit to every matching edge
One Switch: Segregated Routing Policies

• Model: Either **just 1 reconfig** or **just static**

• Optimal solution in polynomial time:
  1. Compute & assign benefit to every matching edge
  2. Compute optimal weighted matching
One Switch: Segregated Routing Policies

• Model: Either just 1 reconfig or just static

• Optimal solution in polynomial time:
  1. Compute & assign benefit to every matching edge
  2. Compute optimal weighted matching
     – E.g., weighted Edmond’s Blossom algorithm
One Switch: Segregated Routing Policies

• Model: Either just 1 reconfig or just static

• Optimal solution in polynomial time:
  1. Compute & assign benefit to every matching edge
  2. Compute optimal weighted matching
     – E.g., weighted Edmond’s Blossom algorithm

• Downside: Only optimal under (artificially!?) segregated routing policy!
One Switch: Segregated Routing Policies

- Model: Either just 1 reconfig or just static

- Optimal solution in polynomial time:
  1. Compute & assign benefit to every matching edge
  2. Compute optimal weighted matching
     - E.g., weighted Edmond’s Blossom algorithm

- **Downside**: Only optimal under (artificially!?) segregated routing policy!
  - *Not optimal under arbitrary routing policies*
One Switch: Non-Segregated Routing
One Switch: Non-Segregated Routing

Can improve routing quality
One Switch: Non-Segregated Routing

Can improve routing quality

NP-hard to optimally compute
One Switch: Non-Segregated Routing

Can improve routing quality

NP-hard to optimally compute

Already for simple settings

*sparse communication patterns, unit weights etc.*
One Switch: Non-Segregated Routing

Can improve routing quality

NP-hard to optimally compute

Already for simple settings
(sparse communication patterns, unit weights etc.)

Approximation algorithms & restricted topologies
One Switch: Non-Segregated Routing

- Can improve routing quality
- NP-hard to optimally compute
- Already for simple settings (sparse communication patterns, unit weights etc.)
- Approximation algorithms & restricted topologies

Future Work
One Switch: Non-Segregated Routing

Can improve routing quality

NP-hard to optimally compute

Already for simple settings
(sparse communication patterns, unit weights etc.)

Approximation algorithms & restricted topologies

Future Work

Already some work in different settings, e.g.:
- network forms a dynamic tree [Schmid et al., ToN ’16]
- constant degree and sparse demands [Avin et al., DISC ’17]
- degree depends on node popularity [Avin et al., Inf. Pr. Let. ‘18]

(These works assume all links are reconfigurable)
Multiple Reconfigurable Switches
Multiple Reconfigurable Switches

• Makes the setting more scalable 😊
Multiple Reconfigurable Switches

• Makes the setting more scalable 😊

• But of course, still NP-hard 😞

  (already for one switch)
Multiple Reconfigurable Switches

• Makes the setting more scalable 😊

• But of course, still NP-hard 😞
  (already for one switch)

• Let’s make things simpler
Multiple Switches: More than One Flow
Multiple Switches: More than One Flow

• Can we optimize max. path length?
Multiple Switches: More than One Flow

• Can we optimize max. path length?
  ◦ For 2 flows?
Multiple Switches: More than One Flow

• Can we optimize max. path length?
  ◦ For 2 flows?
    – NP-hard again 😞
Multiple Switches: One Flow
Multiple Switches: One Flow

Communication frequency: $A \rightarrow G: 1$
Multiple Switches: One Flow

Communication frequency: A→G: 1
Multiple Switches: One Flow

- Consider weights

Communication frequency: A→G: 1
Multiple Switches: One Flow

• Consider weights

Communication frequency: A→G: 1
Multiple Switches: One Flow

- Consider **weights**

Communication frequency: A→G: 1
Multiple Switches: One Flow

- Consider weights

Communication frequency: A→G: 1

How to formalize?
Multiple Switches: One Flow

• Challenge:
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm

• Idea: Use flow algorithms
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm

• Idea: Use flow algorithms
  ◦ Min-cost integral flow is polynomial
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm

• Idea: Use flow algorithms
  ◦ Min-cost integral flow is polynomial
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm

• Idea: Use flow algorithms
  ◦ Min-cost integral flow is polynomial
  
  ![Diagram of network with label A and capacity = 1]

*some small strings attached*
Multiple Switches: One Flow

- Challenge:
  - Proper matchings
  - Polynomial algorithm

- Idea: Use flow algorithms
  - Min-cost integral flow is polynomial

*some small strings attached*
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm

• Idea: Use flow algorithms
  ◦ Min-cost integral flow is polynomial

Unidirectionality
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm

• Idea: Use flow algorithms
  ◦ Min-cost integral flow is polynomial

Unidirectionality

capacity =1

*some small strings attached
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm

• Idea: Use flow algorithms
  ◦ Min-cost integral flow is polynomial

Unidirectionality
Multiple Switches: One Flow

- Challenge:
  - Proper matchings
  - Polynomial algorithm

- Idea: Use flow algorithms
  - Min-cost integral flow is polynomial

Unidirectionality

capacity =1

*some small strings attached
Multiple Switches: One Flow

- Challenge:
  - Proper matchings
  - Polynomial algorithm

- Idea: Use flow algorithms
  - Min-cost integral flow is polynomial

Unidirectionality

- Same conceptual idea

capacity = 1

*some small strings attached
Multiple Switches: One Flow

• Challenge:
  ◦ Proper matchings
  ◦ Polynomial algorithm

• Idea: Use flow algorithms
  ◦ Min-cost integral flow is polynomial

Unidirectionality

• Same conceptual idea
  ◦ capacity =1

A

capacity =1

A

*some small strings attached

A_{out}

capacity =1

A_{in}
Summary and Outlook

- **one** reconfigurable switch
  - not seg.: NP-hard. Improves solutions.

- **multiple** reconfigurable switches
  - multiple flows: NP-hard
  - just one flow: Easy.

- Next steps
  - approximation algorithms
  - special topologies
Thank you! 😊