Optimal Virtualized In-Network Processing with Applications to Aggregation and Multicast
KuVS Best Master Thesis Award 2015

Matthias Rost
Technische Universität Berlin

NetSys 2015, Cottbus
March 11th, 2015

Advisors / Supervisors
Prof. Anja Feldmann, Ph.D., Technische Universität Berlin
Prof. Dr. Andreas Bley, Universität Kassel
Dr. Stefan Schmid, Technische Universität Berlin
Mindset: Service Deployment

Setting: Network Virtualization, e.g. SDN + NFV
- Routes can be flexibly established on a per flow basis
- Functions can be flexibly placed on specific nodes

Task: Service Deployment
- Given: ‘communication service’ shall be established
- Task: Find an *optimal* virtual topology *and* an embedding of the service on the physical network
Communication Schemes: Multicast (same old! same old?)

Processing $=$ duplication $+$ reroute

sender $\rightarrow$ processing node $\rightarrow$ receiver

Matthias Rost (TU Berlin)

Optimal Virtualized In-Network Processing

NetSys 2015, Cottbus
Processing = duplication + reroute

Figure: Hierarchy of processing nodes
Communication Schemes: Aggregation

processing = merge + reroute

sender
receiver
processing node
sender
sender
Communication Schemes: Aggregation

processing = merge + reroute

Figure: Hierarchy of processing nodes
Introductory Example
Introductory Example

**Aggregation scenario**
grid graph: 14 senders, one receiver

**Virtualized links**
data can be routed arbitrarily

(receiver)  sender
Without in-network processing: Unicast

Solution Method
- minimal cost flow

Solution uses
- 41 edges
- 0 processing nodes

Figure: Unicast solution
With in-network processing at all nodes

Solution Method

- Steiner arborescence

Solution uses

- 16 edges
- 9 processing nodes

Figure: Aggregation tree

receiver

sender

processing node

sender with processing
How to Trade-off?
What we aim for

Solution uses
- 26 edges
- 2 processing nodes

receiver
sender
processing node
Solution Structure

Figure: Virtual Arborescence

Figure: underlying routes
New Model: Constrained Virtual Steiner Arborescence Problem

Definition: CVSAP (Aggregation Variant)

Find a Virtual Arborescence connecting senders to the single receiver, s.t.

1. bandwidth of substrate is not exceeded,
2. inner nodes are capable of processing flow,
3. the processing nodes’ capacities are not exceeded,

minimizing the joint cost for bandwidth allocations and function placement.
Applications
Service Replication
Service Replication
Service Replication

What if backend links are congested?
Service Replication
Service Replication

What if only ‘3’ users can be handled?
Service Replication
## Applications

<table>
<thead>
<tr>
<th>Network</th>
<th>Application</th>
<th>Technology, e.g.</th>
</tr>
</thead>
<tbody>
<tr>
<td>multicast</td>
<td>service replication / cache placement [10, 11]</td>
<td>middleboxes / NFV + SDN</td>
</tr>
<tr>
<td>backbone</td>
<td>optical multicast [6]</td>
<td>ROADM + SDH</td>
</tr>
<tr>
<td>all</td>
<td>application-level multicast [15]</td>
<td>different</td>
</tr>
<tr>
<td>sensor network</td>
<td>value &amp; message aggregation [5, 8]</td>
<td>source routing</td>
</tr>
<tr>
<td>ISP</td>
<td>network analytics: Gigascope [3]</td>
<td>middleboxes / NFV + SDN</td>
</tr>
<tr>
<td>data center</td>
<td>big data / map-reduce: Cam- doop [2]</td>
<td>SDN</td>
</tr>
</tbody>
</table>

**edge capacities** processing node locations processing node capacities

edge costs costs for installing processing functionality
Results
Solution Approaches

Wishful thinking: there exists a

- polynomial time algorithm
- solving CVSAP to optimality
- considering all constraints.
Solution Approaches

Wishful thinking: there exists a
- polynomial time algorithm
- solving CVSAP to optimality
- considering all constraints.

Theorem: Inapproximability of CVSAP
Finding a feasible solution is NP-complete!
Solution Approaches

Wishful thinking: there exists a
• polynomial time algorithm
• solving CVSAP to optimality
• considering all constraints.

Theorem: Inapproximability of CVSAP
Finding a feasible solution is NP-complete!

Approximations
• polynomial
• quality guarantee
• weaker models

Exact Algorithms
• non-polynomial
• optimality
• full model

Heuristics
• polynomial
• no solution guarantee
• full model
Thesis' core: comprehensive algorithmic study

**Algorithms**

**Approximations**
- NVSTP
- VSTP
- VSAP

**Exact Algorithms**
- multi-commodity flow
- single-commodity flow → VirtuCast

**LP-based Heuristics**
- FlowDecoRound
- MultipleShots
- GreedyDiving

**Combinatorial Heuristic**
- GreedySelect
Inapproximability
Results

Inapproximability

Reduction from Set Cover: Does a set cover of size $X$ exist?

Theorem:
Finding a feasible solution is already NP-complete.
Approximation Algorithms for Variants
Variants

<table>
<thead>
<tr>
<th>Edge and Node Capacities</th>
<th>Directed</th>
<th>Undirected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CV SAP</td>
<td>NV SAP</td>
<td>VS AP</td>
</tr>
<tr>
<td>CV STP</td>
<td>NV STP</td>
<td>VS STP</td>
</tr>
<tr>
<td>No Capacities</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Approximation via related problems

<table>
<thead>
<tr>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directed</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>both capacities</td>
</tr>
<tr>
<td>node capacities</td>
</tr>
<tr>
<td>no capacities</td>
</tr>
</tbody>
</table>

**Bottom Line**
- Better understanding of the problems core complexity: virtualized links & restricted network function placement
- Obtained lower bounds and approximations
Exact Algorithms for CVSAP
Multi-Commodity Flow (MCF) Integer Program

First approach: MCF IP

- ‘the simple way to do it’
- explicitly represent virtual arborescence
Multi-Commodity Flow (MCF) Integer Program

First approach: MCF IP
- ‘the simple way to do it’
- explicitly represent virtual arborescence

Does not scale well
- number of binary variables: \( \geq \text{#processing nodes} \cdot \text{#edges} \)
Single-Commodity Flow IP

Single-commodity flow formulation

- computes *aggregated* flow on edges independently of the origin
- does not represent virtual arborescence

Figure: Single-commodity
Multi- vs Single-Commodity

Example: 6000 edges and 200 Steiner sites

- Single-commodity: 6000 integer variables
- Multi-commodity: 1,200,000 binary variables

Figure: Single-commodity

Figure: Multi-commodity
VirtuCast Algorithm
VirtuCast Algorithm

Outline of VirtuCast

1. Solve single-commodity flow IP formulation.
2. Decompose IP solution into Virtual Arborescence.

How to decompose?

(a) IP solution  (b) Virtual Arborescence
Complete Formulation

\[
\text{minimize} \quad C_{IP}(x, f) = \sum_{e \in E_G} c_e f_e + \sum_{s \in S} c_s x_s
\]

subject to

\[
f(\delta_{E_{ext}}^+(v)) = f(\delta_{E_{ext}}^-(v)) \quad \forall \ v \in V_G
\]

\[
f(\delta_{E_{ext}}^+(W)) \geq x_s \quad \forall \ W \subseteq V_G, s \in W \cap S \neq \emptyset
\]

\[
f_e \leq u_s x_s \quad \forall \ e = (s, o^-) \in E_{ext}^{S^-}
\]

\[
f_{(r, o^-)} \leq u_r
\]

\[
f_e \leq u_e \quad \forall \ e \in E_G
\]

\[
f_e = 1 \quad \forall \ e \in E_{ext}^{T^+}
\]

\[
f_e = x_s \quad \forall \ e = (o^+, s) \in E_{ext}^{S^+}
\]

\[
x_s \in \{0, 1\} \quad \forall \ s \in S
\]

\[
f_e \in \mathbb{Z}_{\geq 0} \quad \forall \ e \in E_{ext}
\]
**Complete Formulation**

\[
\begin{align*}
\text{minimize} & \quad C_{IP}(x, f) = \sum_{e \in E_G} c_e f_e + \sum_{s \in S} c_s x_s \\
\text{subject to} & \quad f(\delta^+_{E_{\text{ext}}}(v)) = f(\delta^-_{E_{\text{ext}}}(v)) \quad \forall \ v \in V_G \\
& \quad f(\delta^+_{E_{\text{ext}}}(W)) \geq x_s \quad \forall \ W \subseteq V_G, s \in W \cap S \neq \emptyset \\
& \quad f_e \leq u_s x_s \quad \forall \ e = (s, o^-) \in E^s_{\text{ext}} \\
& \quad f_e \leq u_o \quad \forall \ e \in E_G \\
& \quad f_e = 1 \quad \forall \ e \in E^{T+}_{\text{ext}} \\
& \quad f_e = x_s \quad \forall \ e = (o^+, s) \in E^s_{\text{ext}} \\
& \quad x_s \in \{0, 1\} \quad \forall \ s \in S \\
& \quad f_e \in \mathbb{Z}_{\geq 0} \quad \forall \ e \in E_{\text{ext}}
\end{align*}
\]
## Connectivity Inequalities

### STP Excursion [7]

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i) $x(\delta(W)) \geq 1$, for all $W \subset V$, $W \cap T \neq \emptyset$, $(V \setminus W) \cap T \neq \emptyset$,</td>
<td></td>
</tr>
<tr>
<td>(ii) $0 \leq x_e \leq 1$, for all $e \in E$,</td>
<td></td>
</tr>
<tr>
<td>(iii) $x$ integer,</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\min \quad & c^T x \\
\text{(i)} \quad & x(\delta(W)) \geq 1, \quad \text{for all } W \subset V, W \cap T \neq \emptyset, \\
\text{(ii)} \quad & 0 \leq x_e \leq 1, \quad \text{for all } e \in E, \\
\text{(iii)} \quad & x \text{ integer}, 
\end{align*}
\]
Connectivity Inequalities

STP Excursion [7]

\[
\begin{align*}
\text{(uSP)} & \quad \min c^T x \\
(i) & \quad x(\delta(W)) \geq 1, \quad \text{for all } W \subseteq V, W \cap T \neq \emptyset, \\
(ii) & \quad 0 \leq x_e \leq 1, \quad \text{for all } e \in E, \\
(iii) & \quad x \text{ integer,}
\end{align*}
\]

\[\forall W \subseteq V_G, s \in W \cap S \neq \emptyset. \quad f(\delta_{E_{\text{ext}}}^+(W)) \geq x_s\]

‘From each processing node there exists a path towards the sender.’

Exponentially many constraints, but …

can be separated in polynomial time.
Decomposing flow is non-trivial (5 pages proof)!

Flow solution . . .
- contains cycles and
- represents *arbitrary* hierarchies.

Result
- Decomposition is *always* feasible
- Constructive proof: polynomial time algorithm
Combinatorial Heuristic: GreedySelect
Combinatorial Heuristics

On Chickens and Eggs
- How and when to place processing functionality?
- How and when to reserve bandwidth for routes?
- How to react to infeasibilities?

Our Approach
- Place processing functionality and reserve bandwidth jointly.
- Try to avoid infeasibilities by proactive routing decisions.
GreedySelect Heuristic

Greedily either...

- connect a single node to the connected component of the receiver or
- connect multiple nodes to an inactive processing node

minimizing the averaged discounted cost per connected node.

Selecting processing node + terminals + paths: $\mathcal{O}(|V| \cdot |E| + |V|^2 \log |V|)$

compute $P_{\bar{s}} \triangleq \{ \bar{s} \in \bar{S}, T' \subseteq \bar{T}, P_{T'} = \{ P_{t,\bar{s}} | t \in T' \} \}$,

such that $P_{t,\bar{s}}$ connects $t$ to $\bar{s}$,

$u^{\bar{s}}(e) - |P_{T'}[e]| \geq 0$ for all $e \in E_G$,

$2 \leq |T'| \leq u_{r,s}(\bar{s})$

minimizing $c_{\bar{s},T'} \triangleq \left( \sum_{t \in T'} \left( c_E(P_{t,\bar{s}}) - c_E(P_{t,R}) \right) + c_E(P_{\bar{s},R}) + c_S(\bar{s}) \right) / |T'|$
VirtuCast Based Heuristics
Overview VirtuCast Based Heuristics

**FlowDecoRound**
- based on (simple) flow decomposition and rounding

**MultipleShot**
- treats processing (node) variables as probabilities
- iteratively tries to construct a solution using a MST variant
- recomputes LP and iterates, if unsuccessful

**Greedy Diving**
- activates single *best* processing (node) iteratively, recomputes LP
- afterwards fixing of flow variables in a similar fashion
- complex fallback mechanisms
## Overview VirtuCast Based Heuristics

### FlowDecoRound
- Based on (simple) flow decomposition and rounding

### MultipleShot
- Treats processing (node) variables as probabilities
- Iteratively tries to construct a solution using a MST variant
- Recomputes LP and iterates, if unsuccessful

### Greedy Diving
- Activates single best processing (node) iteratively, recomputes LP
- Afterwards fixing of flow variables in a similar fashion
- Complex fallback mechanisms
## Overview VirtuCast Based Heuristics

**FlowDecoRound**
- based on (simple) flow decomposition and rounding

**MultipleShot**
- treats processing (node) variables as probabilities
- iteratively tries to construct a solution using a MST variant
- recomputes LP and iterates, if unsuccessful

**Greedy Diving**
- activates single *best* processing (node) iteratively, recomputes LP
- afterwards fixing of flow variables in a similar fashion
- complex fallback mechanisms
### Overview VirtuCast Based Heuristics

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FlowDecoRound</strong></td>
<td>based on (simple) flow decomposition and rounding</td>
</tr>
<tr>
<td><strong>MultipleShot</strong></td>
<td>treats processing (node) variables as probabilities, iteratively tries to construct a solution using a MST variant, recomputes LP and iterates, if unsuccessful</td>
</tr>
<tr>
<td><strong>Greedy Diving</strong></td>
<td>activates single <em>best</em> processing (node) iteratively, recomputes LP, afterwards fixing of flow variables in a similar fashion, complex fallback mechanisms</td>
</tr>
</tbody>
</table>
Computational Evaluation
Topologies

3D torus

Fat tree

An ISP topology generated by IGen with 2400 nodes.
Computational Setup & Instances

Setup

- 225 instances of five different graph sizes, costs, ...
- 1 hour runtime limit for computations
- All algorithms implemented in C/C++ using SCIP [1]

<table>
<thead>
<tr>
<th></th>
<th>Nodes</th>
<th>Edges</th>
<th>Processing Locations</th>
<th>Senders</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fat tree</strong></td>
<td>1584</td>
<td>14680</td>
<td>720</td>
<td>864</td>
</tr>
<tr>
<td><strong>3D torus</strong></td>
<td>1728</td>
<td>10368</td>
<td>432</td>
<td>864</td>
</tr>
<tr>
<td><strong>IGen</strong></td>
<td>4000</td>
<td>16924</td>
<td>401</td>
<td>800</td>
</tr>
</tbody>
</table>

**Table:** Largest graph sizes
VirtuCast + LP-based Heuristics
VirtuCast + LP-based Heuristics

<table>
<thead>
<tr>
<th>Fat Tree</th>
<th>IGen</th>
<th>Torus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Obj. Gap [%]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Dual Bound Improv. [%]</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graph Sizes
MCF-IP
MCF-IP: Performance

### Computational Evaluation

#### Results

**Fat Tree**

**IGen**

**Torus**

<table>
<thead>
<tr>
<th>Root Relaxations</th>
<th>Fat Tree</th>
<th>IGen</th>
<th>Torus</th>
</tr>
</thead>
<tbody>
<tr>
<td>I, II, III, IV</td>
<td><img src="chart1" alt="" /></td>
<td>![chart1]</td>
<td>![chart1]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dual Bound Gap [%]</th>
<th>Fat Tree</th>
<th>IGen</th>
<th>Torus</th>
</tr>
</thead>
<tbody>
<tr>
<td>I, II, III, IV</td>
<td>![chart2]</td>
<td>![chart2]</td>
<td>![chart2]</td>
</tr>
</tbody>
</table>

Matthias Rost  (TU Berlin)  Optimal Virtualized In-Network Processing  NetSys 2015, Cottbus 45
GreedySelect
GreedySelect: Efficacy

<table>
<thead>
<tr>
<th>Solutions found</th>
<th>Fat Tree</th>
<th>IGen</th>
<th>Torus</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>14</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>13</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>III</td>
<td>13</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>IV</td>
<td>13</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>V</td>
<td>13</td>
<td>13</td>
<td>1</td>
</tr>
</tbody>
</table>
GreedySelect: Performance

Computation Evaluation

Results

Matthias Rost (TU Berlin)  Optimal Virtualized In-Network Processing  NetSys 2015, Cottbus
LP-based Heuristics
### Computational Evaluation

**Results**

**LP-based Heuristics: Performance on graph size V**

<table>
<thead>
<tr>
<th>Fat Tree</th>
<th>IGen</th>
<th>Torus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obj. Gap [%]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GD</td>
<td>GSD</td>
<td>FGSD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Runtime [s] | | |
| GD | GSD | FGSD | MSS | MS | FDR |
| | | | | | |
Conclusion
Most Important Results

VirtuCast Formulation

- *Single-commodity flow* IP formulation; considers only aggregate flow values
- Based on *novel* flow decomposition scheme
- *Enables* derivation of highly-efficient linear heuristics

**IP formulation + linear heuristics**

Highly efficient solver for CVSAP
Conclusion

Summary

Publications
Matthias Rost, Stefan Schmid: OPODIS 2013 & arXiv [14, 13]

Applications → Concise definition of CVSAP

Inapproximability

Approximations
• NVSTP
• VSTP
• VSAP

Exact Algorithms
• multi-commodity flow
• single-commodity flow → VirtuCast

Heuristics
• FlowDecoRound
• MultipleShots
• GreedyDiving
• GreedySelect

Extensive explorative Computational Evaluation
Thanks
References I


References II

Aggregation tree construction in sensor networks.

Ethernet aggregation and core network models for efficient and reliable iptv services.

Solving steiner tree problems in graphs to optimality.

Modelling data-centric routing in wireless sensor networks.

Hierarchies to Solve Constrained Connected Spanning Problems.
References III


Streaming cache placement.

Optimal Virtualized In-Network Processing with Applications to Aggregation and Multicast, 2014.


Virtucast: Multicast and aggregation with in-network processing.
A proposal for a scalable internet multicast architecture.
In Washington University, 2001.
### Related Work

**Molnar: Constrained Spanning Tree Problems [9]**
- Shows that optimal solution is a ‘spanning hierarchy’ and not a DAG.

**Oliveira et. al: Flow Streaming Cache Placement Problem [11]**
- Consider a weaker variant of multicasting CVSAP without bandwidth
- Give weak approximation algorithm

**Shi: Scalability in Overlay Multicasting [15]**
- Provided heuristic and showed improvement in scalability.
Future Work

Model Extensions
- prize-collecting variants
- concurrent multicast / aggregation sessions

Application Modeling
- Stratosphere II: Big Data
- UNIFY Project: flow analytics

IP formulation
- try to derive further cuts / facets
Future Work: UNIFY / Network Analytics

EU FP7 IP UNIFY [4]

- Considers *service chaining* in the wide-area network, connecting e.g. customers at home to (possibly multiple) datacenter
- Business perspective: SLAs must be guaranteed strictly, otherwise fines
  - KPIs need to be monitored constantly
  - Different measurements need to be collected the whole time

Information Distribution

- Use multicast variant of CVSAP to distribute measurements.
- Placing processing nodes everywhere should be avoided due to the synchronization overhead (latencies).
Future Work: UNIFY / Network Analytics

EU FP7 IP UNIFY [4]

- Considers *service chaining* in the wide-area network, connecting e.g. customers at home to (possibly multiple) datacenter
- Business perspective: SLAs must be guaranteed strictly, otherwise fines
  - KPIs need to be monitored constantly
  - Different measurements need to be collected the whole time

Information Aggregation

- Use aggregation variant of CVSAP to compute (subfunctions) of the KPIs on-the-fly
- Processing nodes may offer multicast functionality (see above) as well.