ReNoVatE :
Recovery from Node Failure in Virtual Network Embedding

-Nashid Shahriar, Reaz Ahmed, Aimal Khan, Shihabur R. Chowdhury, Raouf Boutaba, Jeebak Mitra

UNIVERSITY OF WATERLOO
FACULTY OF MATHEMATICS
David R. Cheriton School
of Computer Science

HUAWEI

ReNoVatE Overview

- **Re**covery from a **No**de Failure in **V**irtu**a**l Ne**t**work **E**mbedding
  - Single node failure in the substrate network
  - Recovers a set of virtual networks
  - Treats affected virtual networks fairly
- Goals
  - Maximize the number of recoveries
  - No disruption to the unaffected parts
  - Meet SLA timing requirements

ReNoVatE Overview

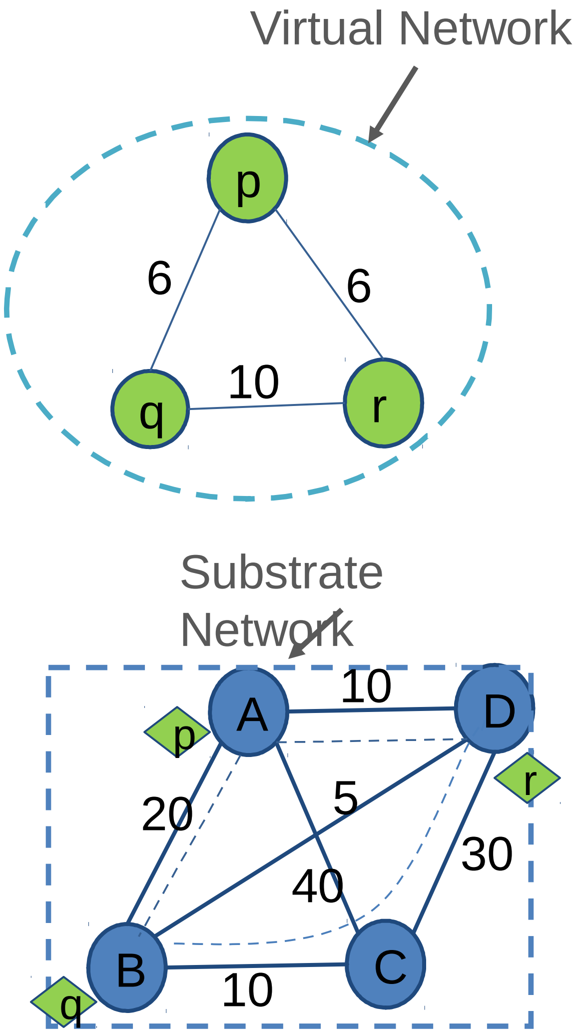
- Opt-ReNoVatE
  - Integer linear program (ILP) formulation
  - Limited to small scale networks
- Fast-ReNoVatE
  - Reformulates as a maximum flow problem
  - Scalable to large scale networks
  - Finds a solution even in a saturated network

Outline

- System model
- Problem statement
- Opt-ReNoVatE
- Fast-ReNoVatE
- Evaluation results
- Conclusion

System Model

- A virtual network is embedded on a substrate network
  - – Node mapping
    - A virtual node is hosted on a substrate node
    - Multiple virtual nodes can coexist
    - Satisfies location constraints
  - – Link mapping
    - A virtual link mapped to a substrate path
    - Substrate link capacities are not exceeded
    - No multi-path embedding



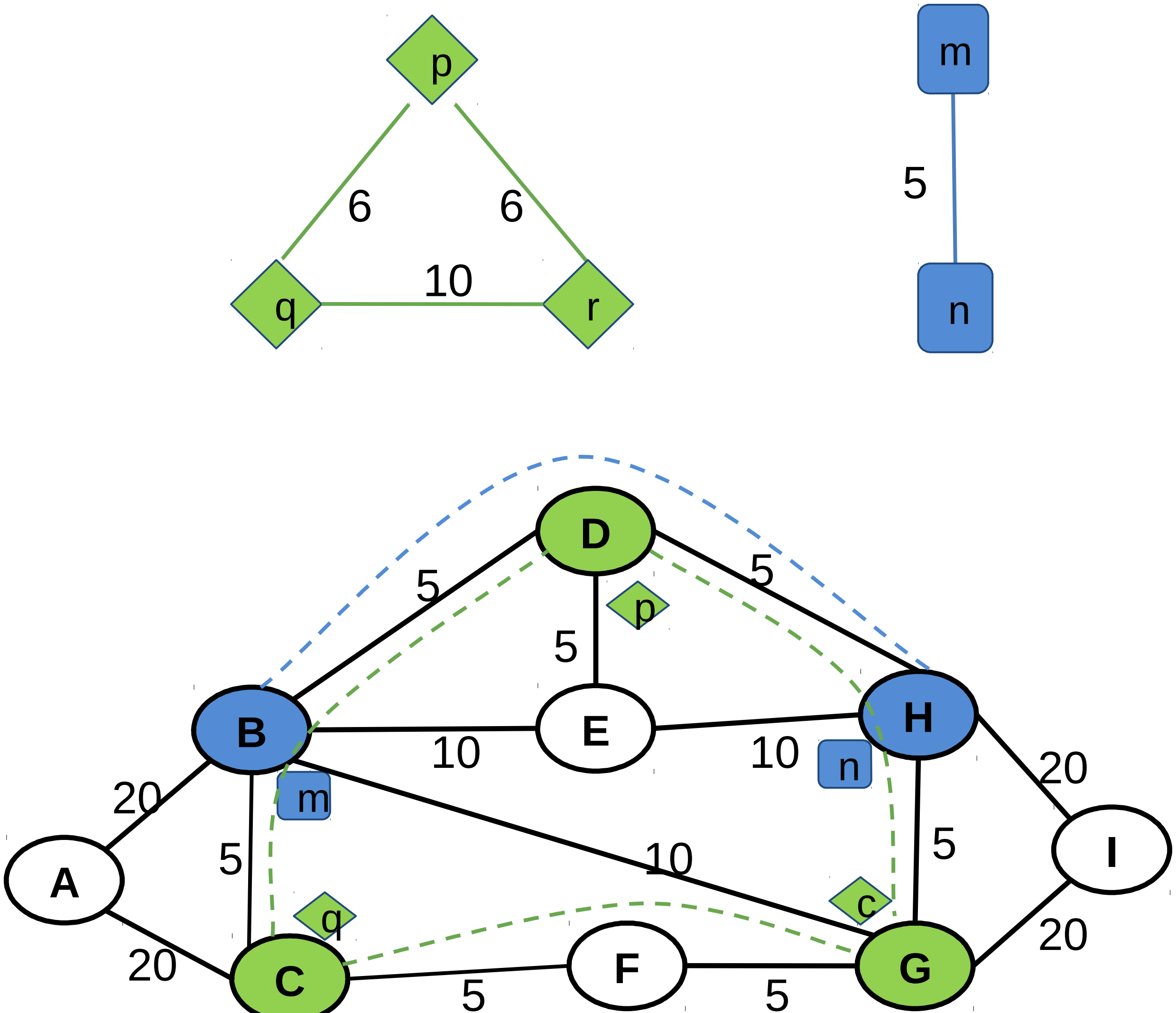
Outline

- System model
- Problem statement
- Opt-ReNoVatE
- Fast-ReNoVatE
- Evaluation results
- Conclusion

Problem Statement

- Given
  - Embedding of a set of virtual networks
  - Single node failure in the substrate network
  - Results in the failure of incident substrate links
- Compute
  - Recovery of the affected virtual networks
  - Migrate failed virtual nodes to other substrate nodes
  - Reroute failed virtual links to alternate substrate paths

ReNoVatE - Example



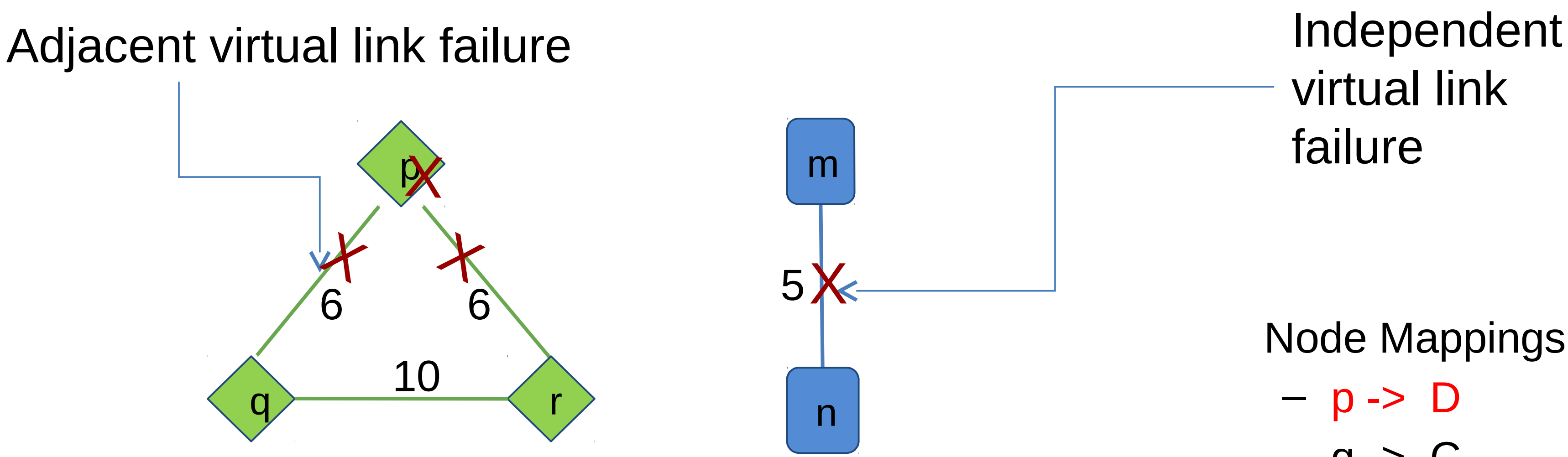
Node Mappings

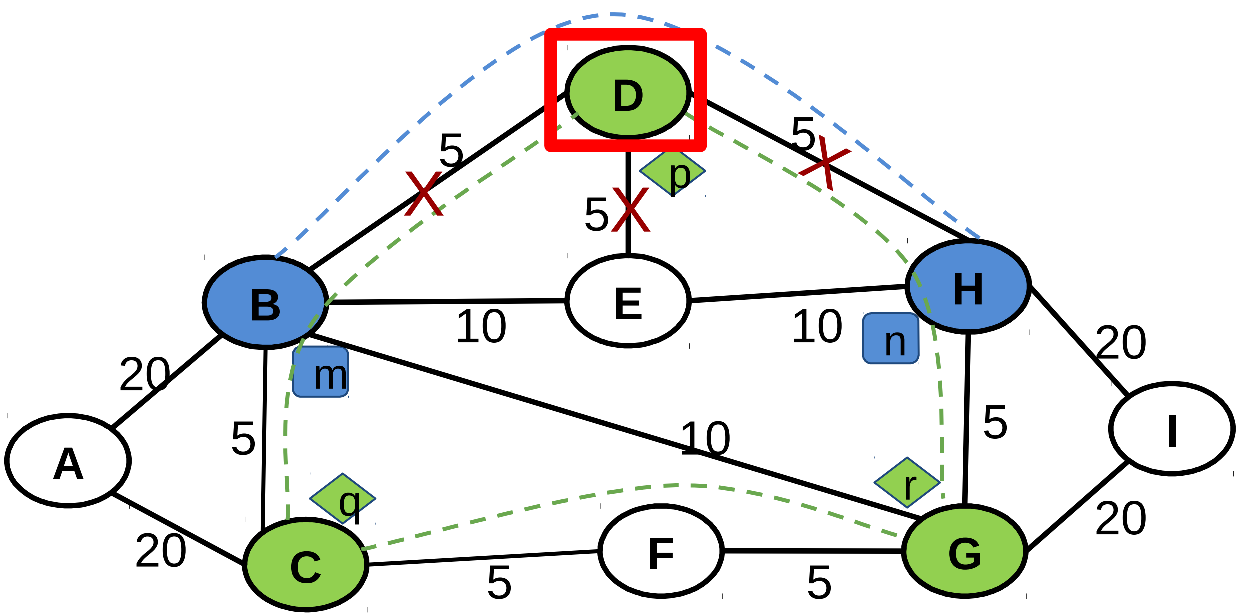
- p -> D
- q -> C
- r -> G
- m -> B
- n-> H

Link Mappings

- (p, q) -> D-B-C
- (p, r) -> D-H-G
- (q, r) -> C-F-G
- (m, n) -> B-D-H

ReNoVatE - Single Node Failure





Node Mappings

- p -> D
- q -> C
- r -> G
- m -> B
- n -> H

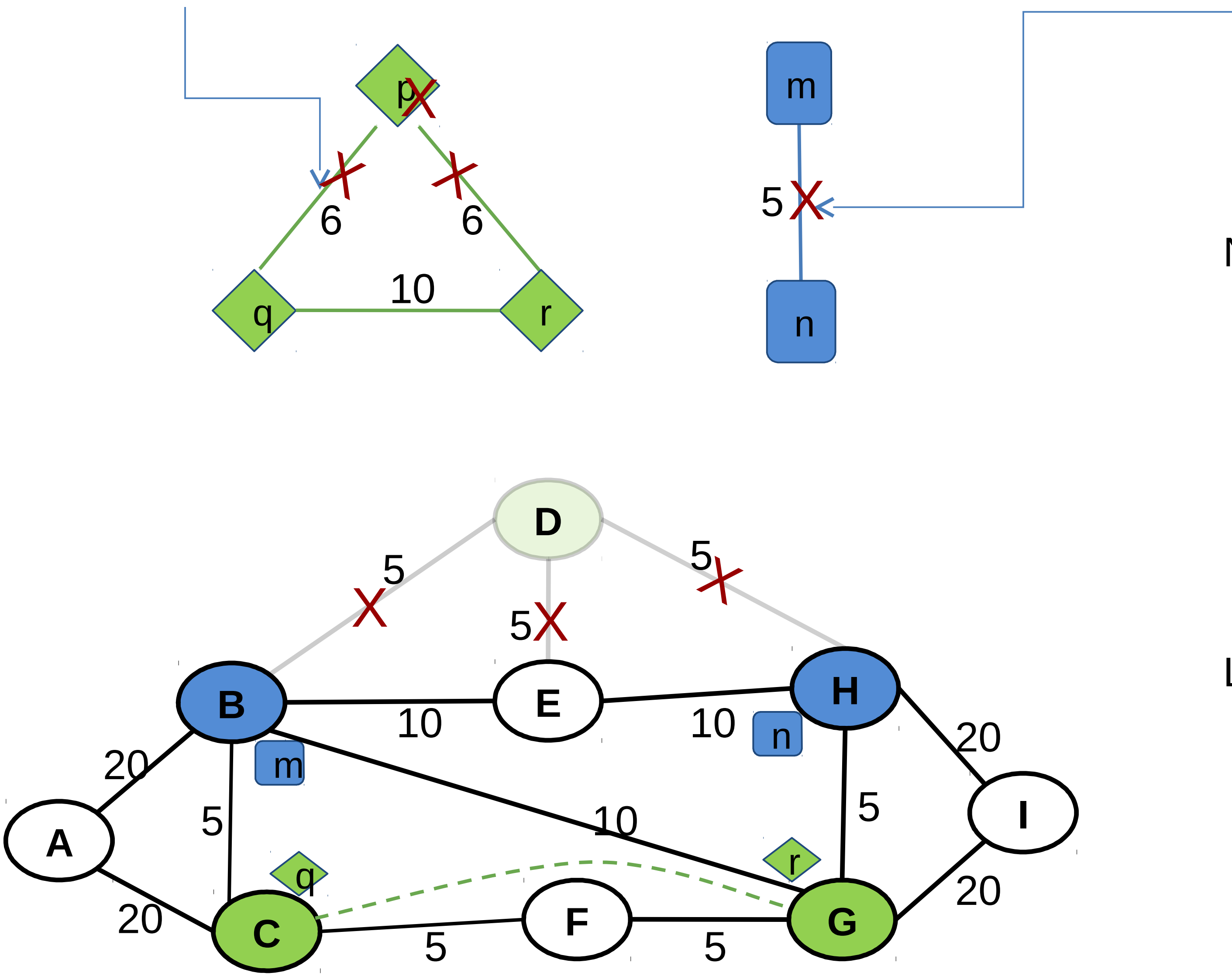
Link Mappings

- (p, q) -> D-B-C
- (p, r) -> D-H-G
- (q, r) -> C-F-G
- (m, n) -> B-D-H

ReNoVatE - Single Node Failure

Adjacent virtual link failure

Independent virtual link failure



Node Mappings

- p -> D
- q -> C
- r -> G
- m -> B
- n-> H

Link Mappings

- (p, q) -> D-B-C
- (p, r) -> D-H-G
- (q, r) -> C-F-G
- (m, n) -> B-D-H

Outline

- System model
- Problem statement
- Opt-ReNoVatE
- Fast-ReNoVatE
- Evaluation results
- Conclusion

Opt-ReNoVatE

- Which virtual links (or networks) are recovered?
  - Some virtual links (or networks) may not be recovered due to resource inadequacy
- Primary maximization objective
  - Number of recovered virtual links
  - May lead to partial recovery of virtual networks
- Secondary minimization objective
  - Cost of bandwidth consumption for recovery
  - Breaks tie among solutions having same primary objective

Opt-ReNoVatE - Constraints

- Link Mapping Constraints
  - Un-splittable path constraints
  - Substrate link capacities are not violated
- Node Mapping Constraints
  - Adheres to provided location constraints
  - Virtual link mapping implies adjacent node mapping
- Intactness of unaffected parts
  - Unaffected mappings are not changed
  - Excludes failed substrate node and links

Outline

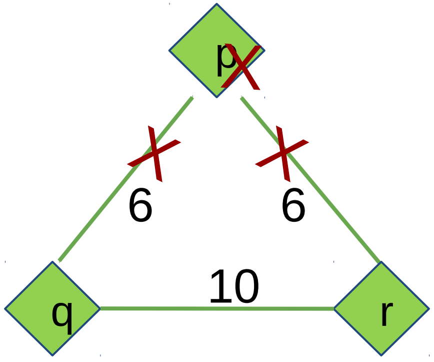
- System model
- Problem statement
- Opt-ReNoVatE
- Fast-ReNoVatE
- Evaluation results
- Conclusion

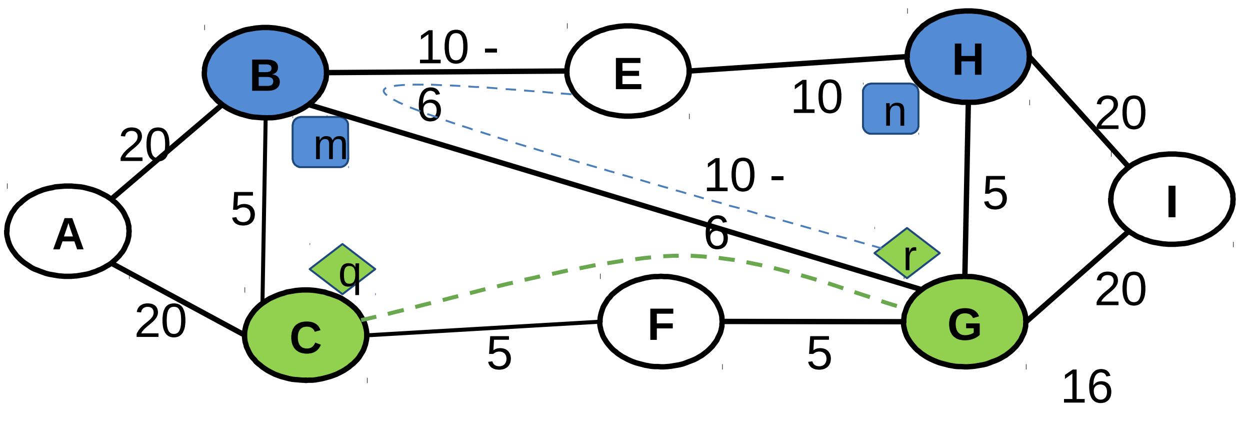
Fast-ReNoVatE - Node Recovery

- Virtual networks are recovered in increasing order of lost bandwidth
  - Increases probability of recovery
  - Re-embeds the failed virtual node based on its location constraint
  - Iterate over all candidate substrate nodes in the location constraint set
  - Select the substrate node yielding the maximum number of recovered paths
  - Tie-break through lower cost of bandwidth for link mapping

Finding Maxpaths - A Naive Approach

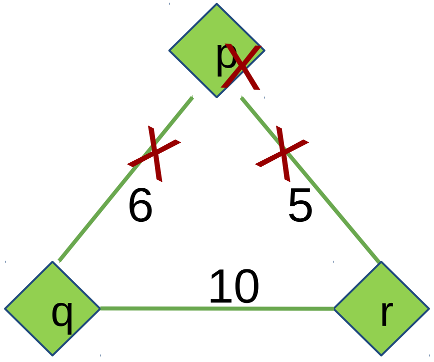
- Sequentially find shortest path for each failed virtual link
  - Suffer from bottleneck links
- Let, *E* is candidate node for *p*
- Shortest path for virtual link *pr*
  - {EB, BG}
- Bottleneck substrate link, BG
  - No other virtual links can be recovered!

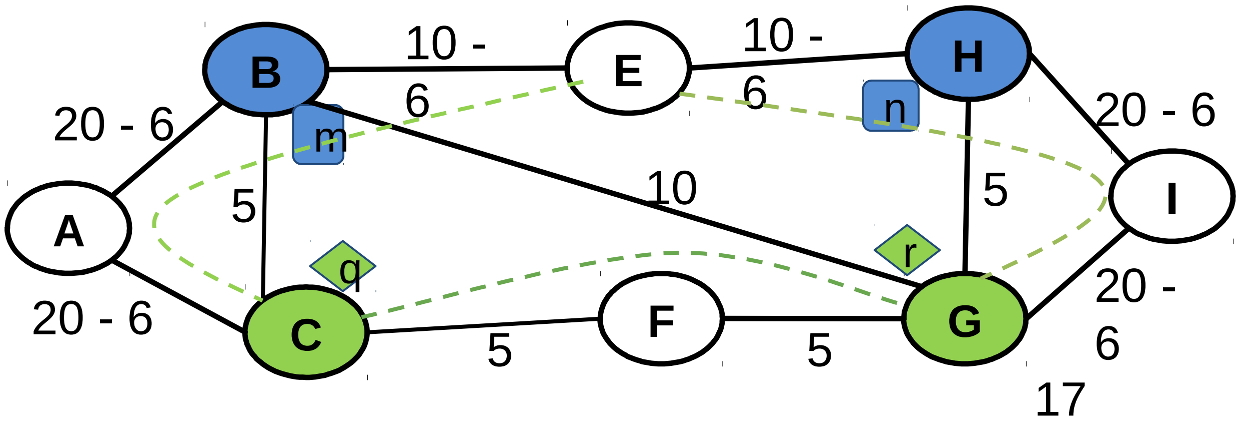




Finding Maxpaths - A Better Approach

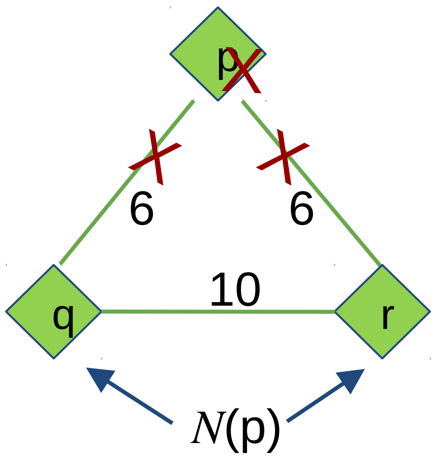
- Compute maximum flow from a source to a sink
  - Avoid bottleneck links
- Send unit flow from the source to the sink
  - Paths carrying the maximum flow yield maximum number of paths
- May result in longer paths

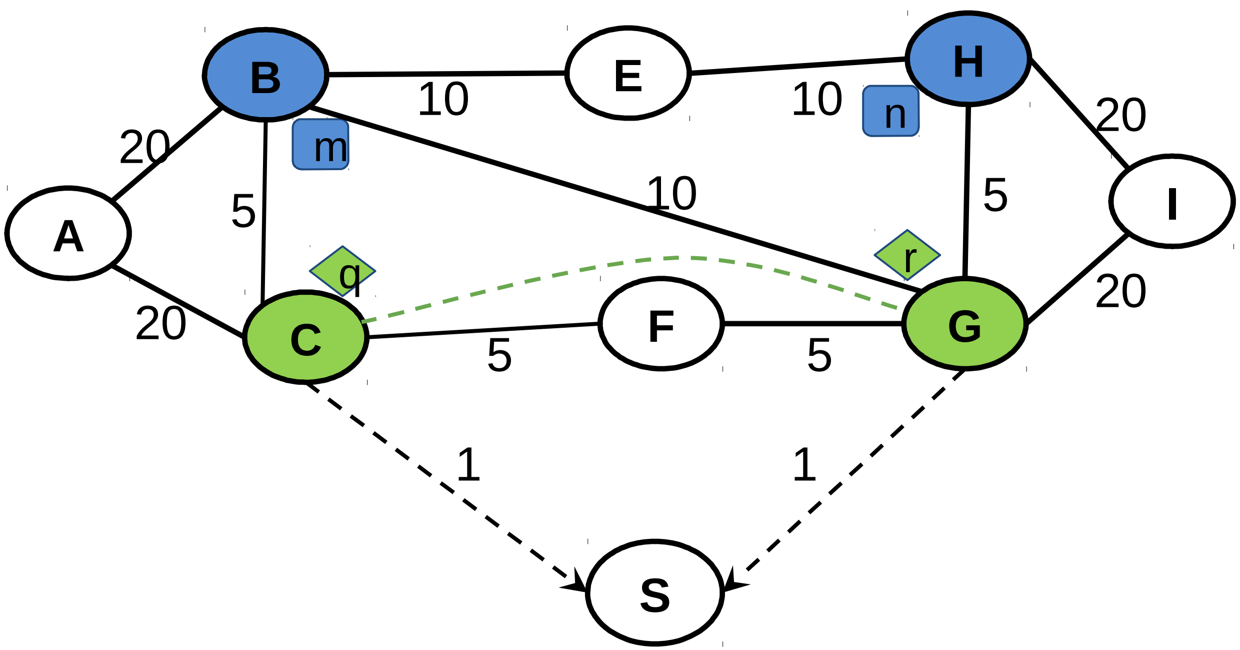




Maxflow Realization - Step 1

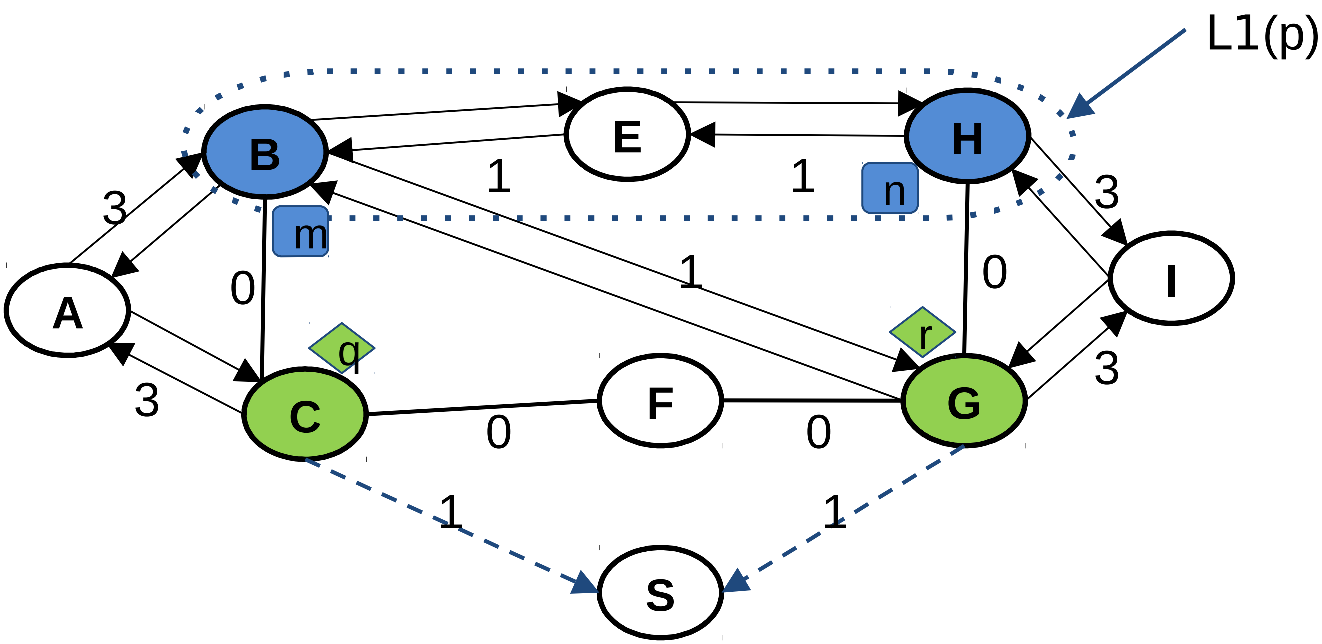
- Augment the SN with a pseudo sink node, S
- Add pseudo links from substrate nodes that host other ends of the failed virtual links to S





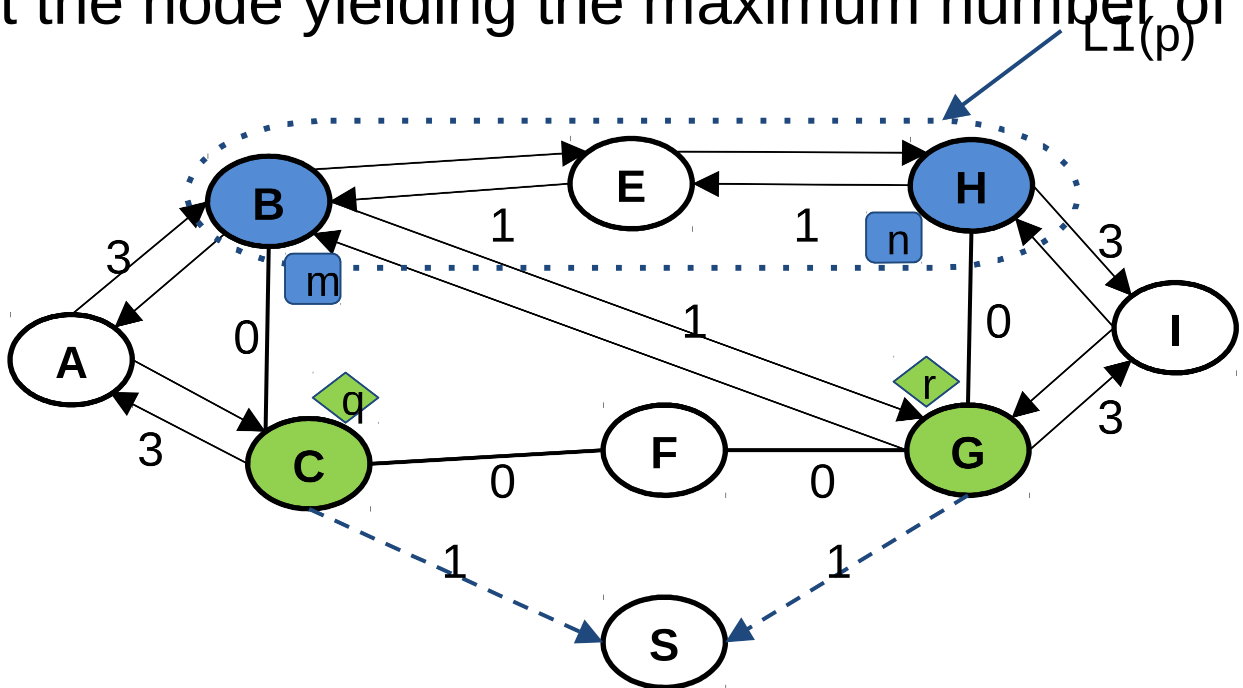
Maxflow Realization - Step 2

- Replace each substrate link with two unidirectional links
- Discretize each link's capacity using an estimation
  - 1/maximum demand of all the failed virtual links in the virtual network
- Other functions such as minimum and average demand could result in oversubscription of bandwidth



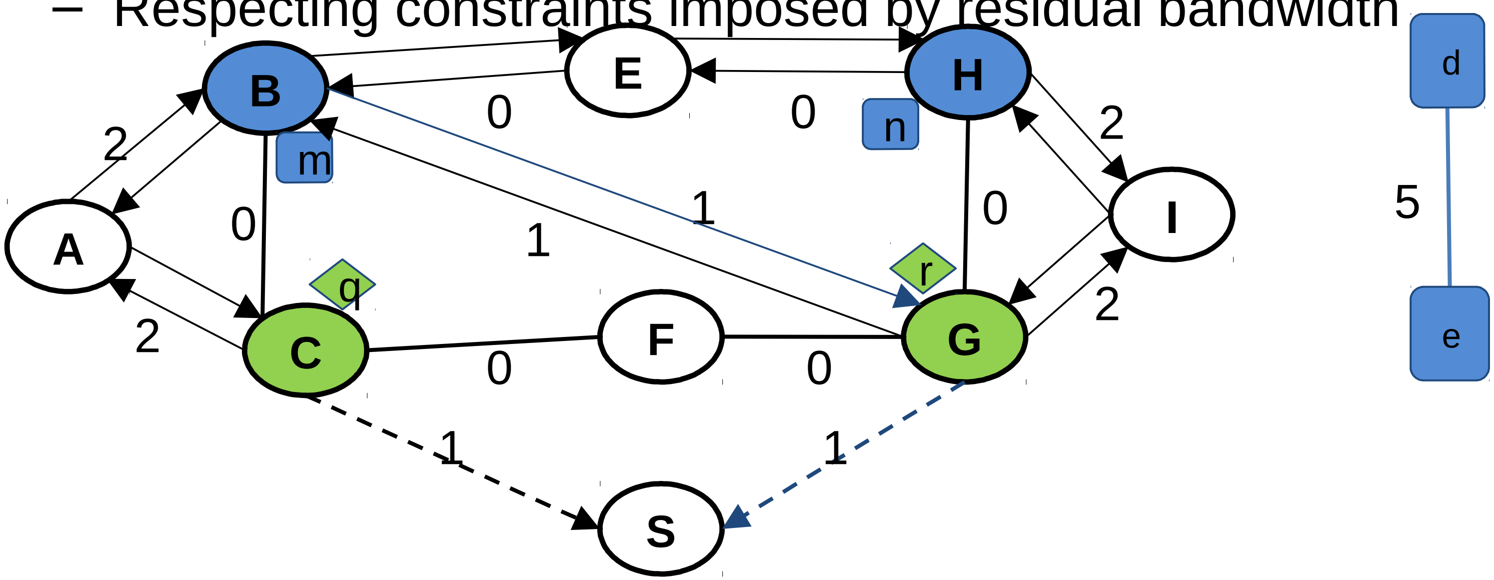
Fast-ReNoVatE - Adjacent Links

- Use Edmond-Karp algorithm to compute augmenting paths from each candidate node in L1(p) to S
- If a new path cancels the flow of a link assigned by a earlier path, re-arrange both paths to exclude the link
- Select the node yielding the maximum number of paths



Fast-ReNoVatE - Independent Links

- Previous approach doesn't apply as it may lead to invalid paths
- Re-embed virtual links in increasing order of bandwidth demand
  - Find alternate substrate path using a minimum cost path approach
  - Use modified version of Dijkstra's shortest path algorithm
  - Respecting constraints imposed by residual bandwidth



Outline

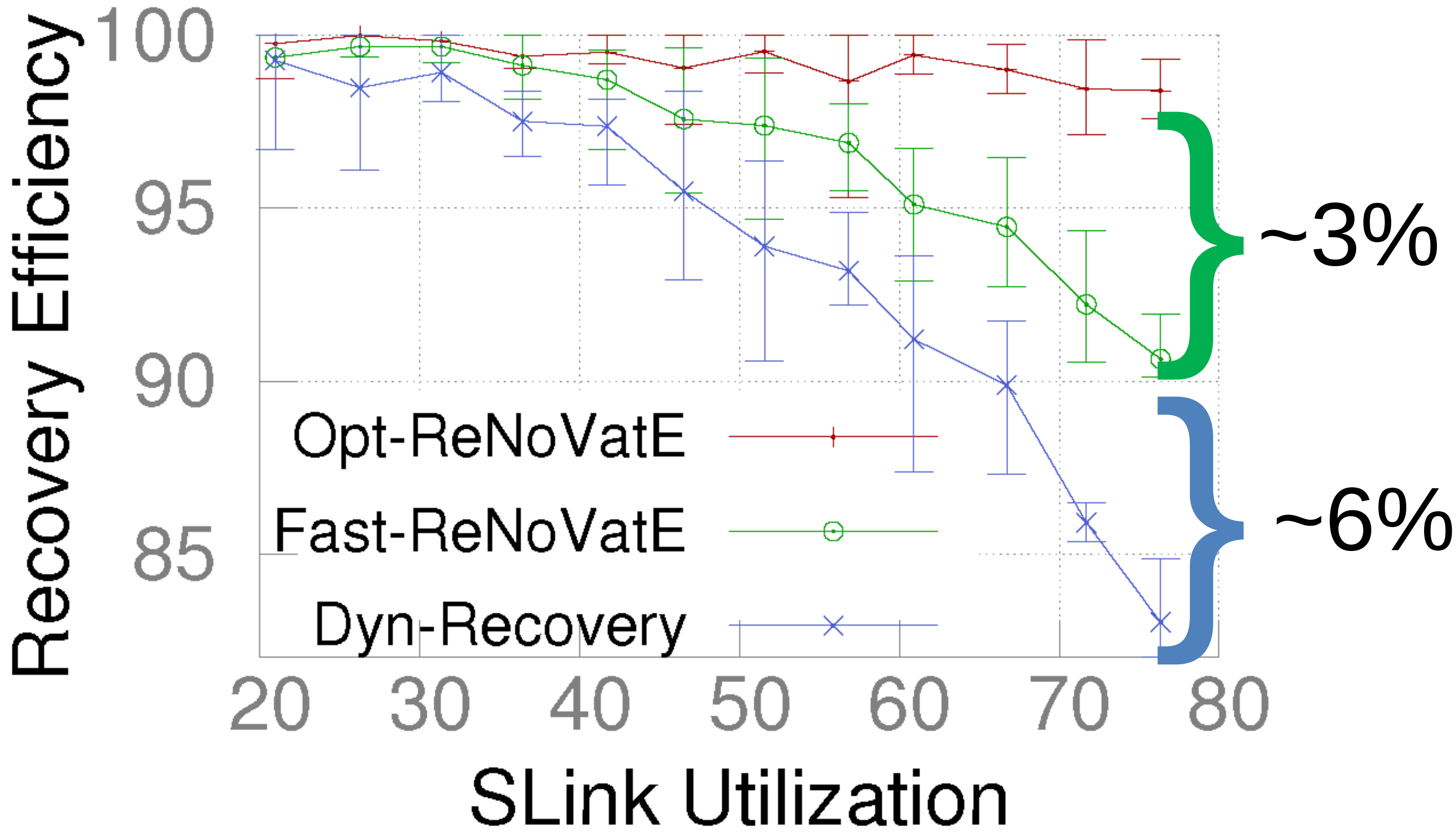
- System model
- Problem statement
- Opt-ReNoVatE
- Fast-ReNoVatE
- Evaluation results
- Conclusion

Evaluation Results - Settings

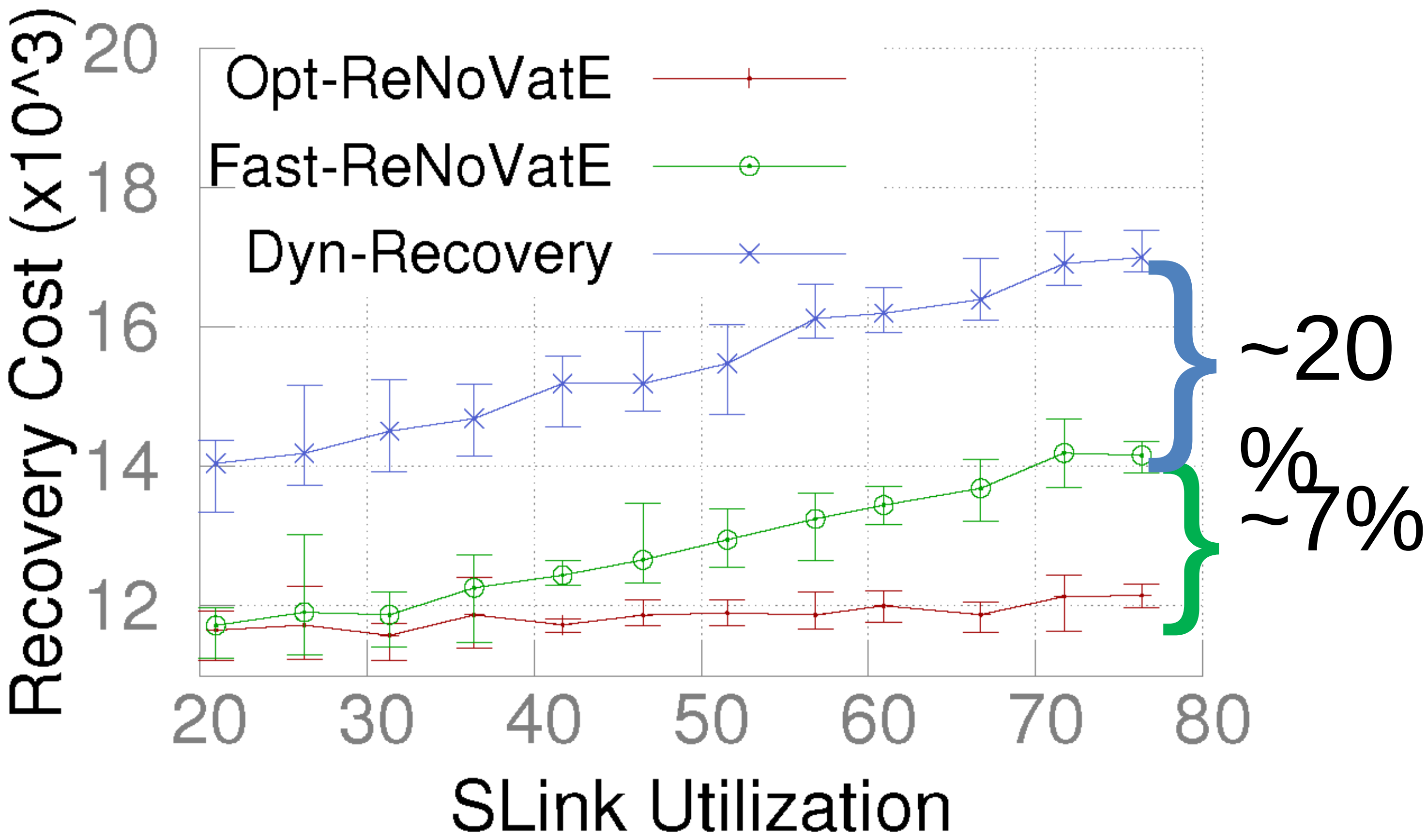
- Compared approaches
  - Opt-ReNoVatE : ILP implementation using IBM's ILOG CPLEX
  - Fast-ReNoVatE : C++ implementation of the heuristic algorithm
  - Dyn-Recovery : C++ implementation of the state-of-the-art1
  - Doesn't allow partial recovery of a virtual network
- Simulation parameters
  - Small scale : 50 substrate nodes and up to 30 VNs embedded on SN
  - Large scale : 1000 substrate nodes and up to 500 VNs embedded
  - Bandwidth demand is ~10-15% of substrate link capacity

1. B. LU et. al., "Dynamic Recovery for Survivable Virtual Network Embedding," The Journal of China Universities of Posts and Telecommunications, vol. 21, pp. 77–84, Jun 2014.

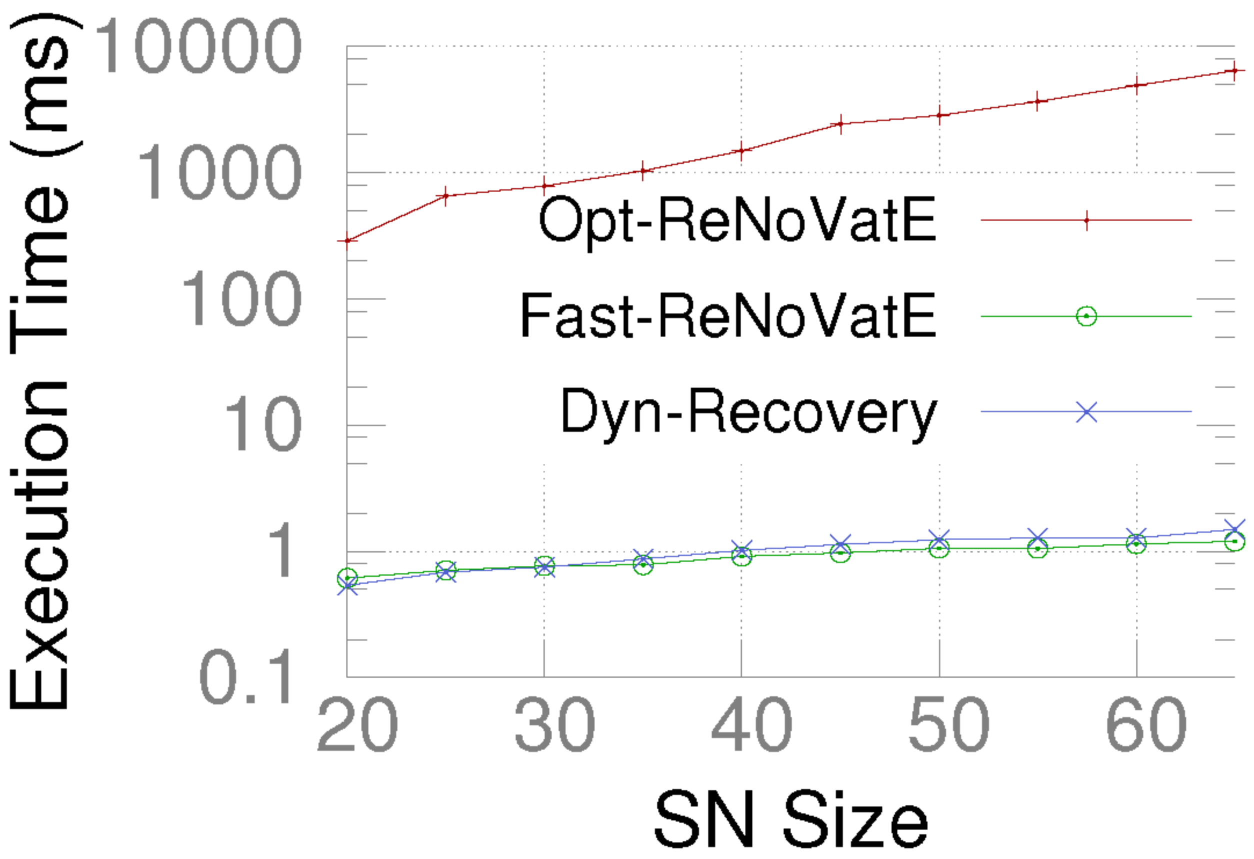
Evaluation Results - Small Scale



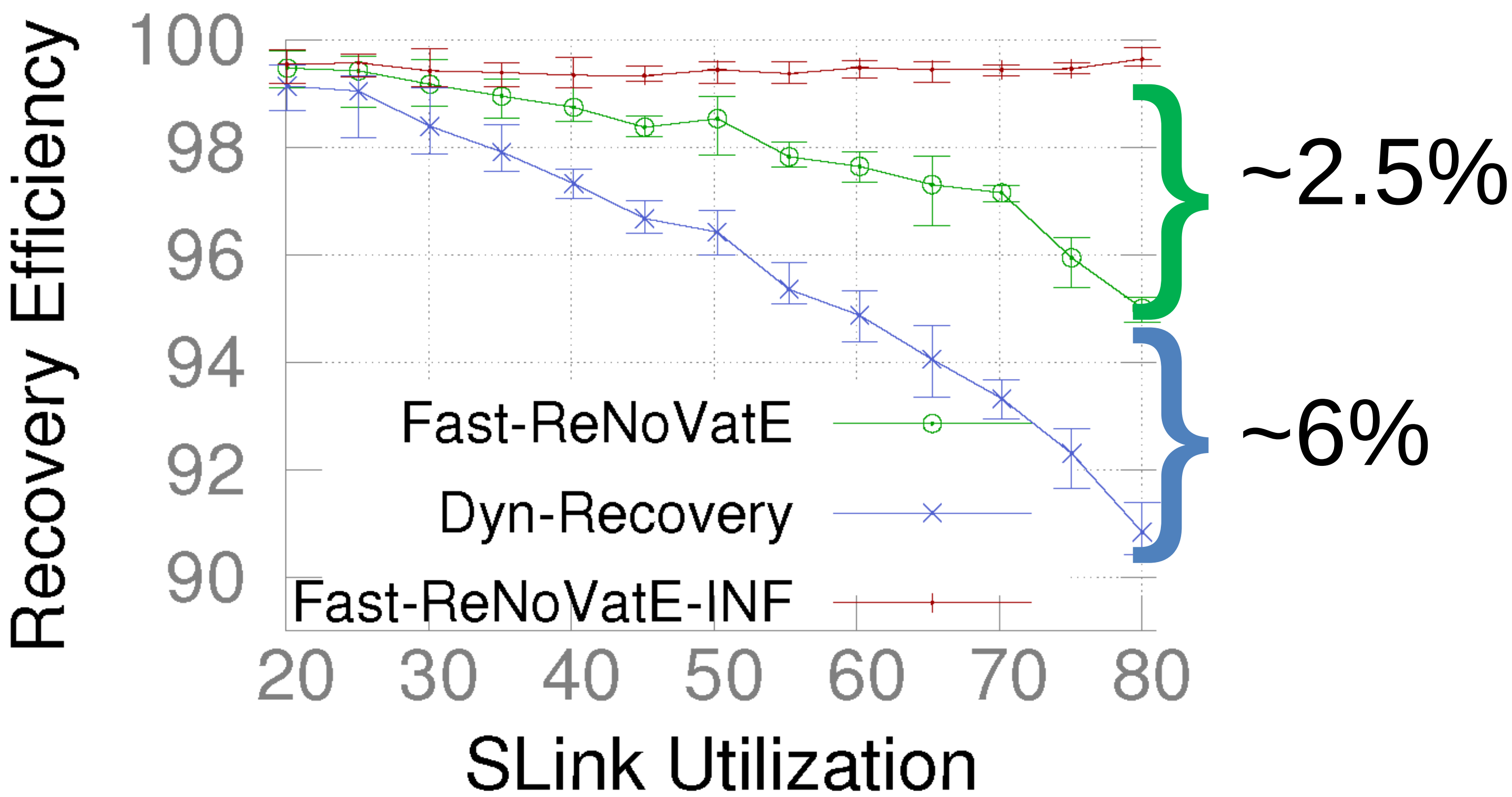
Evaluation Results - Small Scale



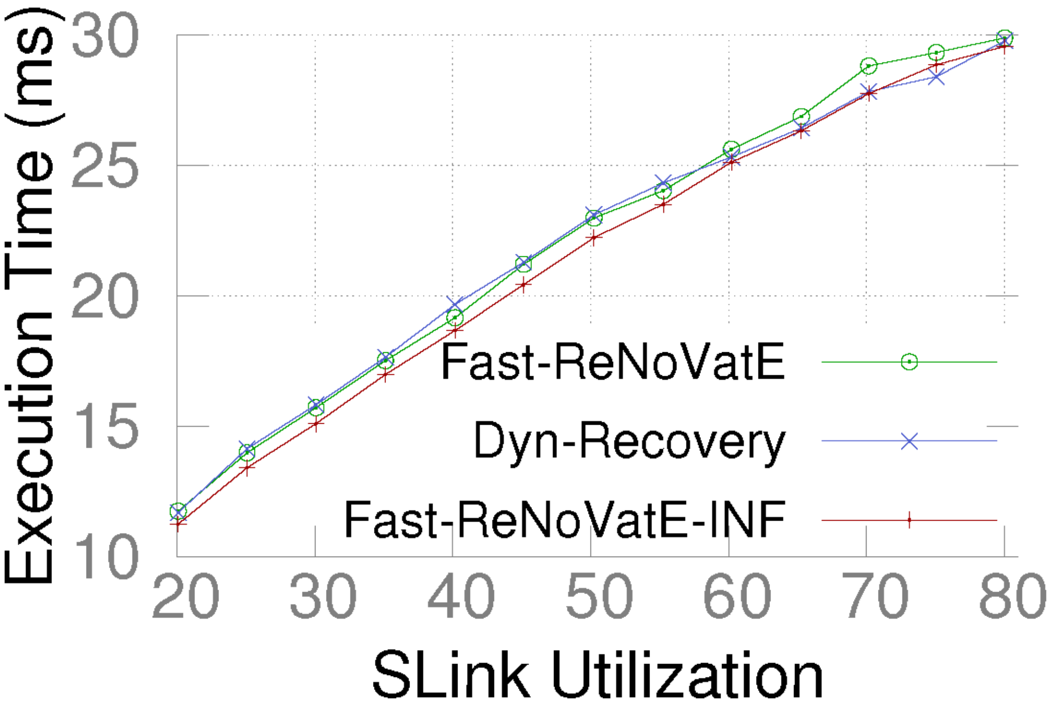
Evaluation Results - Small Scale



Evaluation Results - Large Scale



Evaluation Results - Large Scale



Outline

- System model
- Problem statement
- Opt-ReNoVatE
- Fast-ReNoVatE
- Evaluation results
- Conclusion

Conclusion

- Recovery from a substrate node failure
  - Re-embeds failed virtual nodes and virtual links
  - Maximizes number of recoveries
  - Minimizes cost of re-embedding as secondary goal
- An optimal approach based on ILP formulation for small scale networks
- A fast heuristic approach more scalable than ILP and outperforming a state-of-the-art solution

Future Work

- Evaluate using real testbed experiments
- Prioritize the affected VNs based on the following and adhere to that priority
  - SLA requirements priorities
  - Impacts of failure
  - Profits

Thank you

ReNoVatE Overview

- Validation through extensive simulations
  - Fast-ReNoVatE performs close to Opt-ReNoVatE
  - Outperform a state-of-the-art approach
- Treats affected virtual networks fairly
- Can be extended to consider
  - Service level agreement requirements priorities
  - Profit of individual virtual network
  - Impact of failures

Challenges of ReNoVatE

- Recovery of adjacent virtual links of a VN
  - *NP-Hard Single-source unsplittable flow problem*
- Recovery of independent virtual links
  - *NP-Hard Multi-commodity unsplittable flow problem*
- When a batch of VNs to recover
  - Exponential number of sequences of VNs
- Resource contention due to the failure
  - Create bottleneck nodes and links

State-of-the-art

Proactive approaches

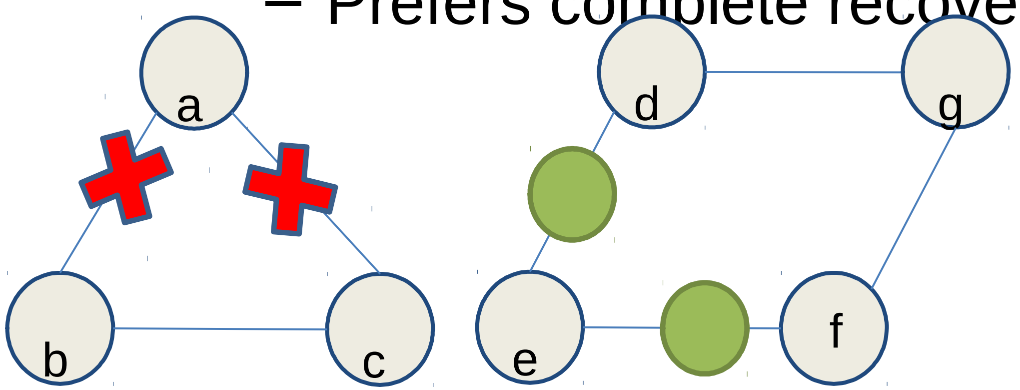
- Guaranteed recovery for certain failure scenarios e.g., single node failure
- May require a very high level of resource redundancy
  - Expensive and not scalable to large VN topologies

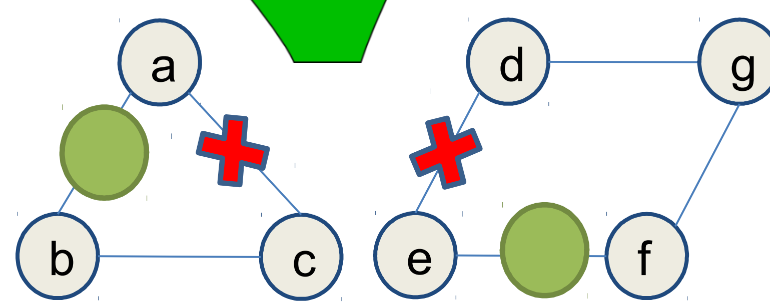
Reactive approaches

- Some approaches try to re-embed the failed links on minimum cost paths
  - ***Bottleneck*** links may cause some failed links not recoverable
- In the event of resource insufficiency, they re-embed the whole/part of the VN
  - VN goes offline for unstipulated time causing service disruption
- None of the approaches deal with a batch of VN failures

Opt-ReNoVatE: Primary Objective

- Primary maximization objective
  - Number of recovered virtual links
  - May lead to partial recovery of VNs
  - Assuming that all virtual links may not be recovered due to resource inadequacy in SN
  - Number of recovered virtual nodes
  - Prefers complete recovery of VNs



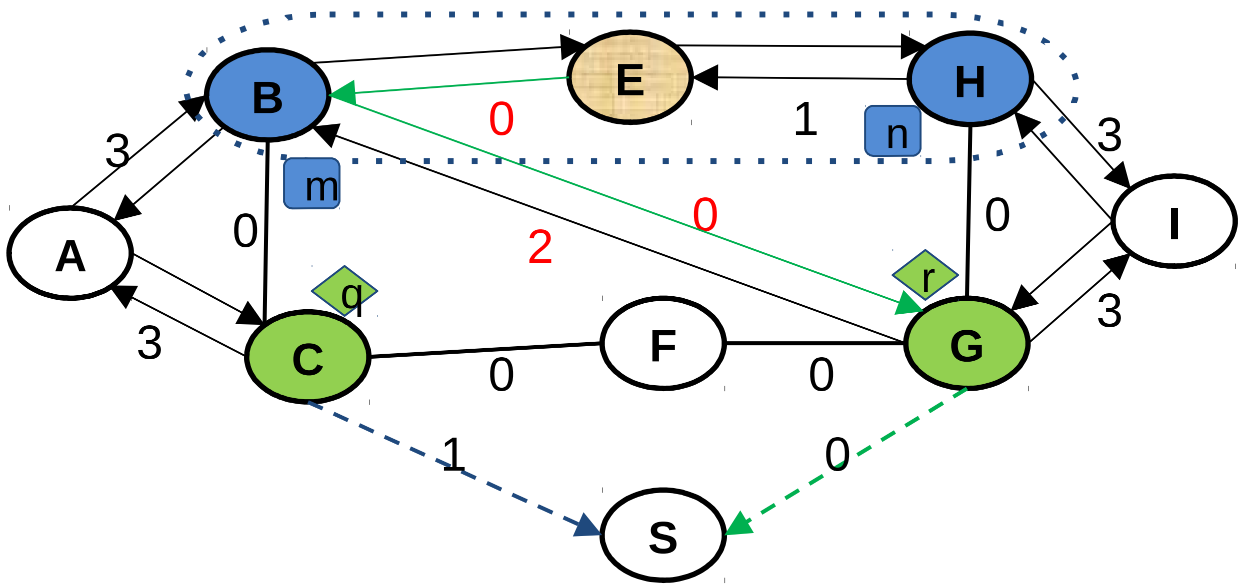


Opt-ReNoVatE: Secondary Objective

- Secondary minimization objective
  - Physical network cost
  - Cost of bandwidth consumption
  - Agitation in the network
  - Embedding failed virtual links in completely new paths require new flow rules to be installed

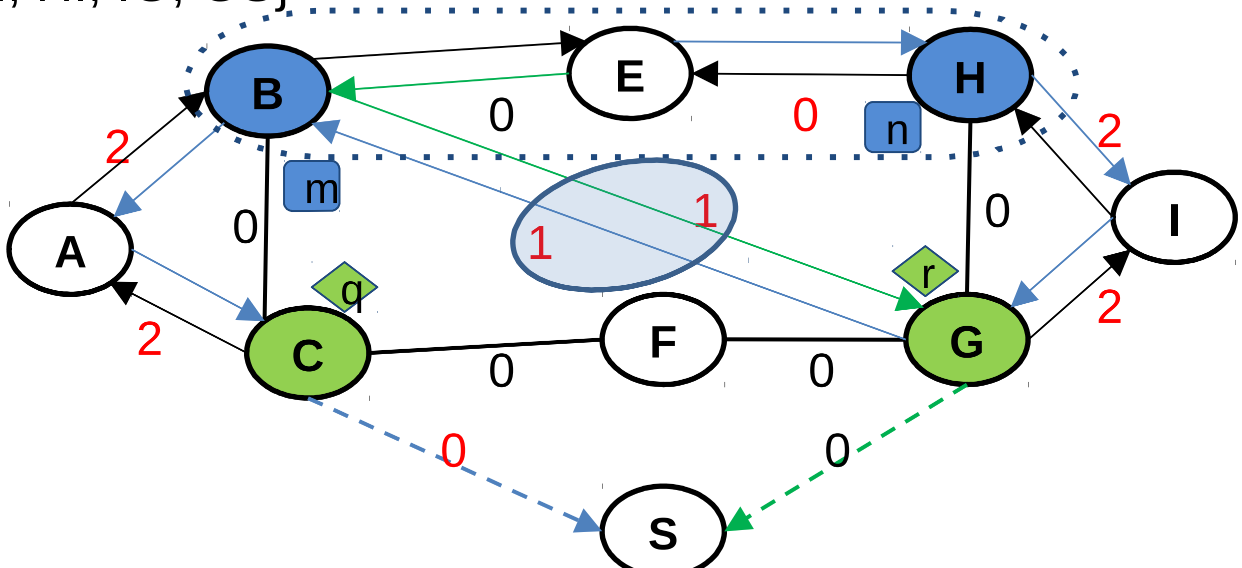
Fast-ReNoVatE: In Action

- Let *E* be the candidate node for p
- First augmenting path
  - {EB, BG, GS}
- Update residual capacities along the augmenting path



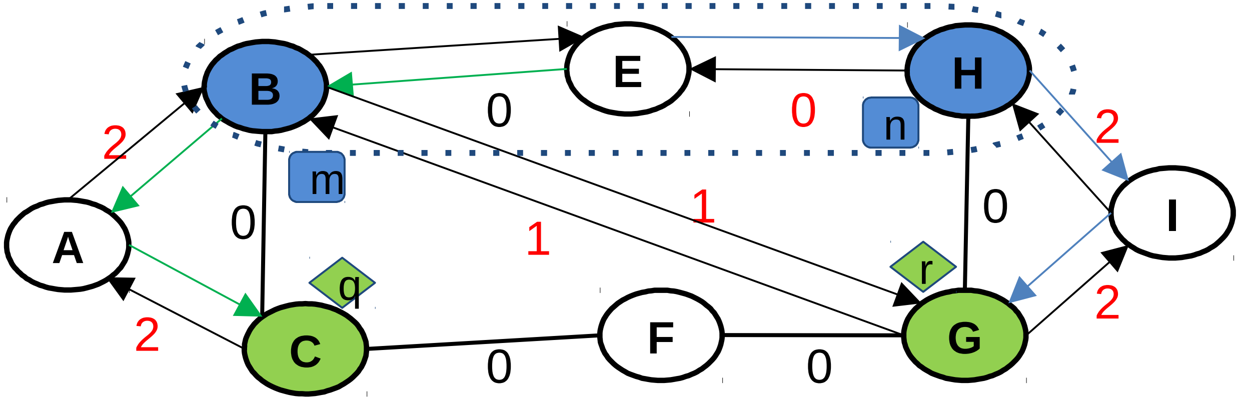
Fast-ReNoVatE: In Action

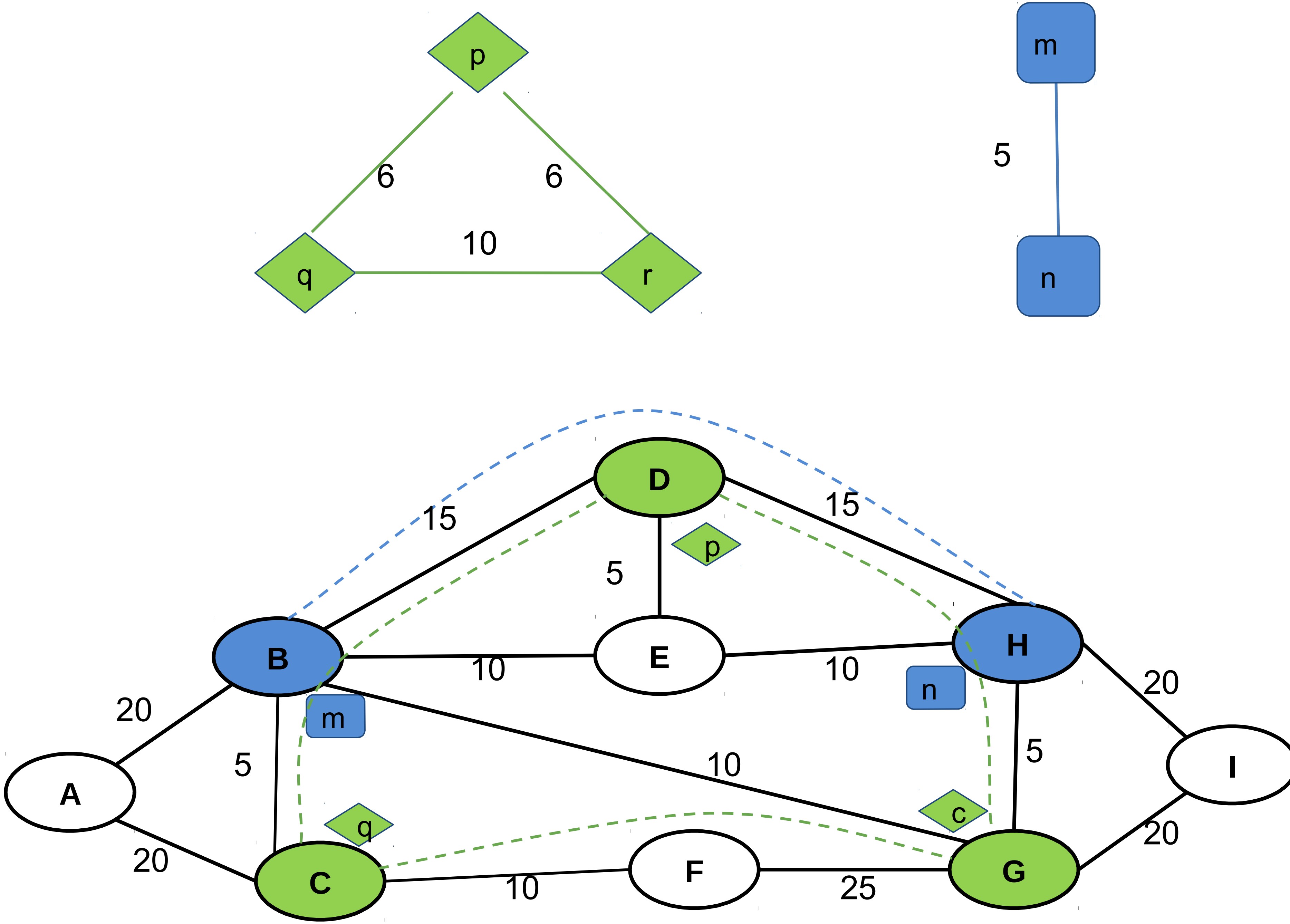
- Second augmenting path
  - {EH, HI, IG, GB, BA, AC, CS}
- It cancels previous flow between B and G in the previous path
- Re-arrange the paths
  - {EB, BA, AC, CS}
  - {EH, HI, IG, GS}

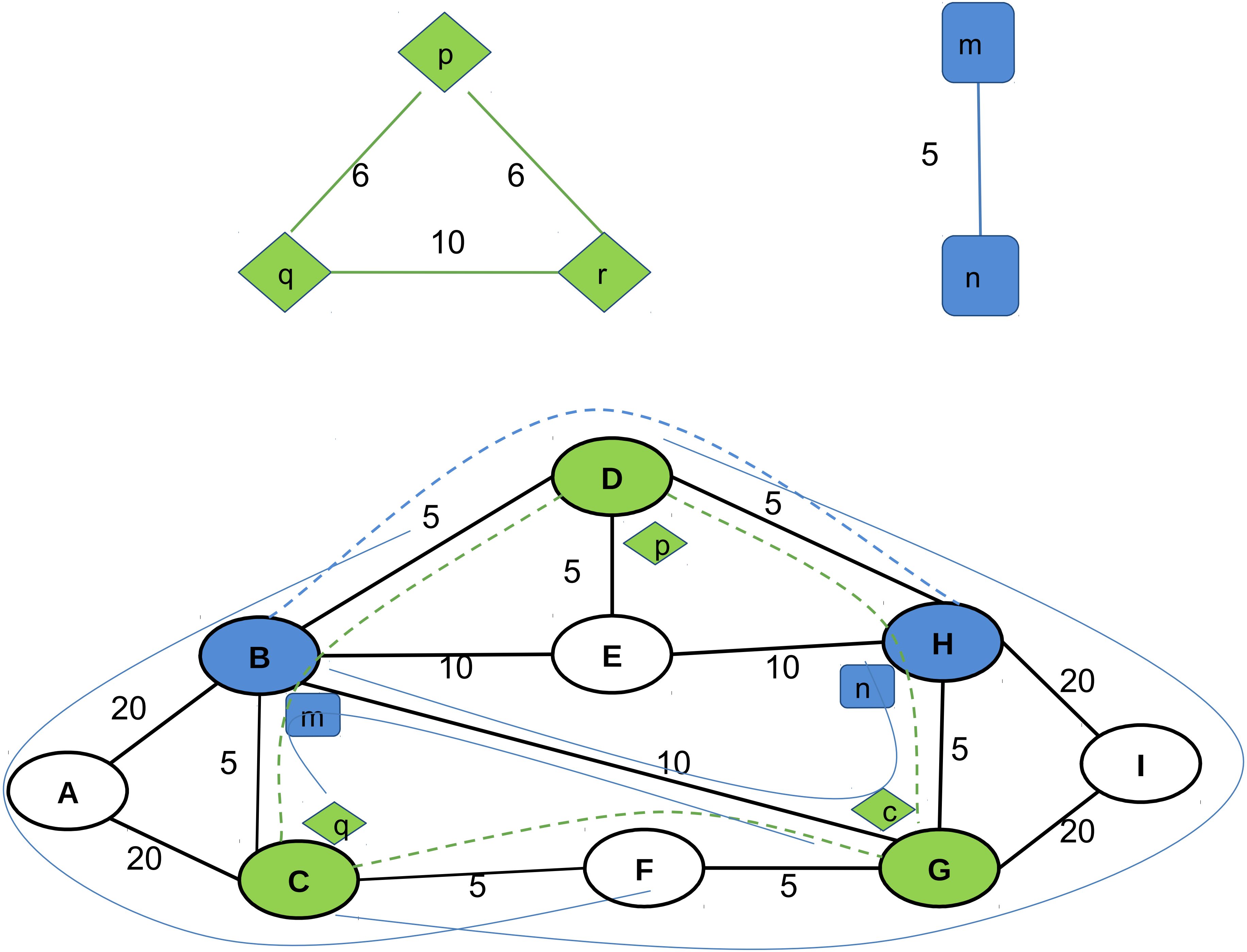


Fast-ReNoVatE: In Action

- Repeat the same steps for other candidates, B and H
- Select the node yielding the maximum number of paths to recover adjacent virtual links
  - Use cost of the path in case of a tie
- If E is selected, computed paths after removing the links to S
  - {EB, BA, AC}
  - {EH, HI, IG}




p
6
6
q
10
r
m
5
n
D
15
15
p
5
B
10
E
10
H
n
20
m
20
5
10
5
20
A
I
q
c
20
C
10
F
25
G


p
6
6
q
10
r
m
5
n
D
5
5
p
5
B
10
E
10
H
20
n
m
20
5
10
5
A
I
q
c
20
20
C
5
F
5
G