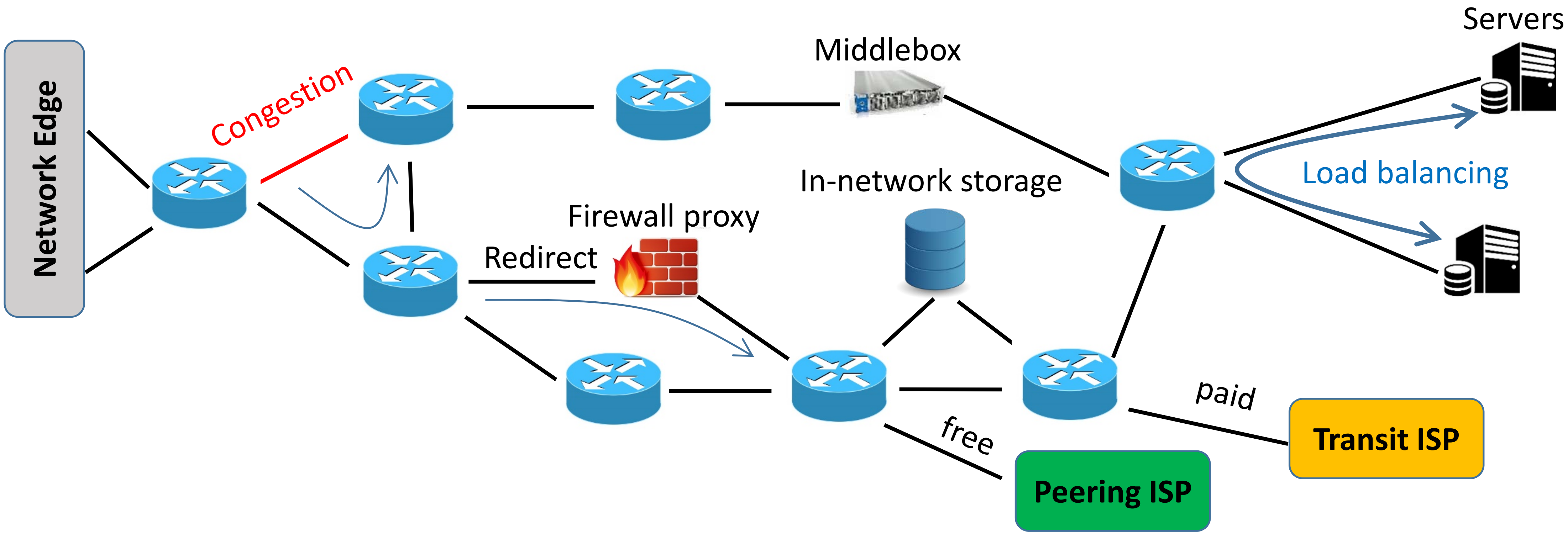
# One Step at a Time: Optimizing SDN Upgrades in ISP Networks

Konstantinos Poularakis
Leandros Tassiulas
Yale University
George Iosifidis
Trinity College Dublin
Georgios Smaragdakis
MIT and TU Berlin

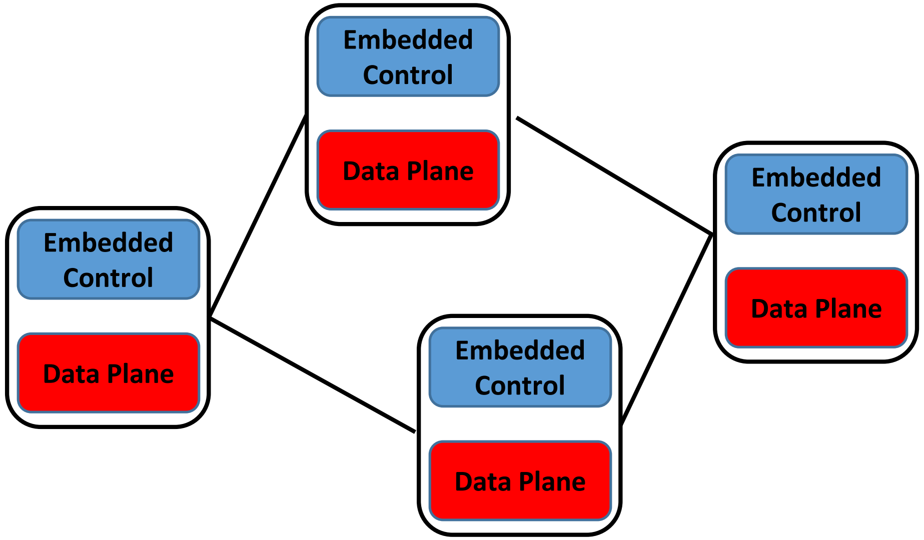
# ISP complex operations today

- ISPs today do a lot more than simply carry data packets.
  - Security policies, traffic engineering, management of different resources (storage, processing), relationships with other networks.

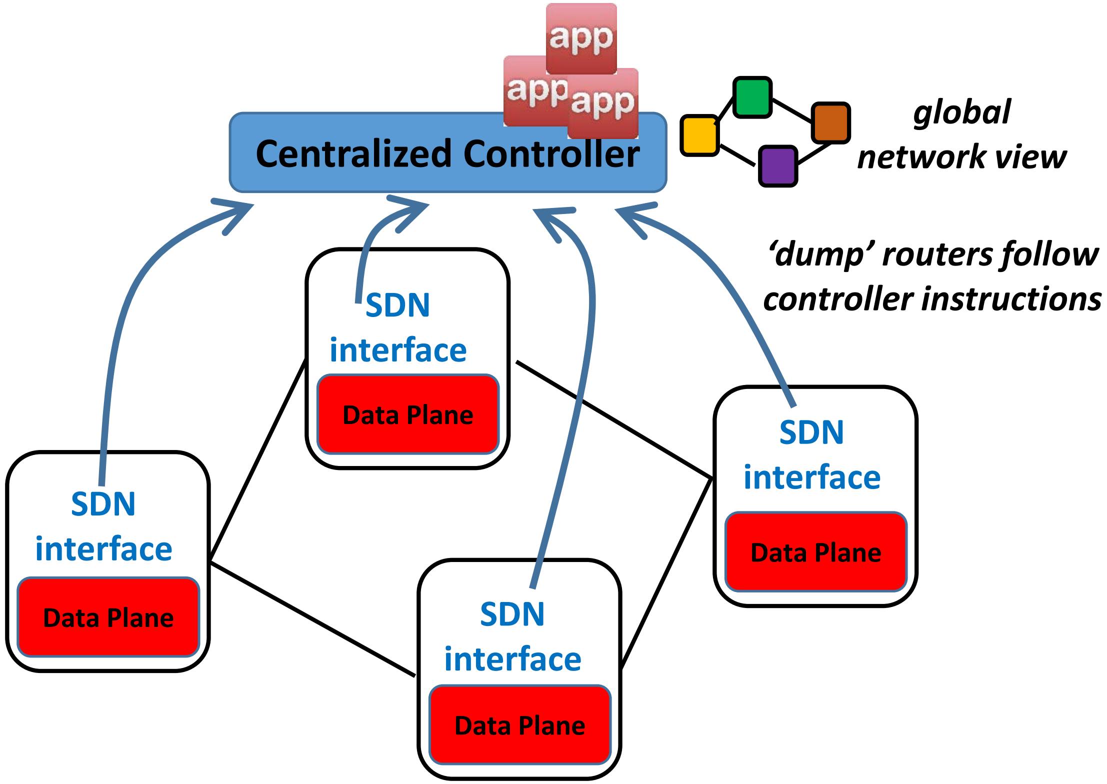


# Shift from legacy to softwarized architectures

- The operations of ISPs for many years were constrained by vendors who *locked* the network functions of their switches/routers.
  - Destination-based IP protocols with limited flexibility.
- **Software Defined Networking (SDN):**
  - ISPs can take back the control of their networks.



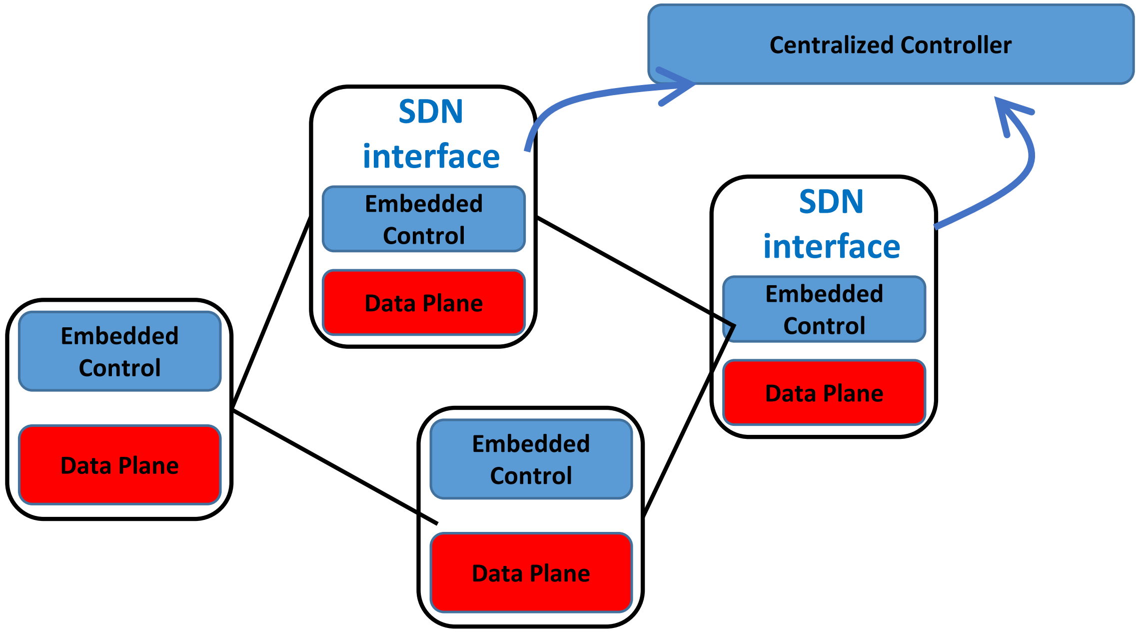

**Traditional Network**




**SDN-enabled Network**

# Hybrid SDN : unavoidable intermediate step

- Difficult to upgrade to SDN the *whole network in one shot*. Simpler to add new SDN functionality in *a part* of the network, while the rest operates the same.
- Even if the ISP decides to upgrade to SDN a specific part of its network (e.g., backbone or edge), it is not trivial how exactly to perform the upgrades:
  - *Operation issues:* need to test the new hardware.
  - *Economic issues*: limited budget for SDN routers.




✓ **Transit period**
Legacy and SDN equipment coexist and interface each other.

✓ ***Hybrid routers***
Support both <u>SDN</u> & <u>local</u> (embedded) control.

# Timing is critical for SDN upgrades

- Like every new technology, SDN equipment *costs reduce* over time as technology matures and the competition among vendors increases.
- ISP's *traffic is increasing* over time.
- Typically, ISPs perform upgrades every 6-12 months by accommodating the lifetime of the existing equipment (3-5 years).
- ***ISP's dilemma:***


- Lack of systematic methodology to answer this dilemma.

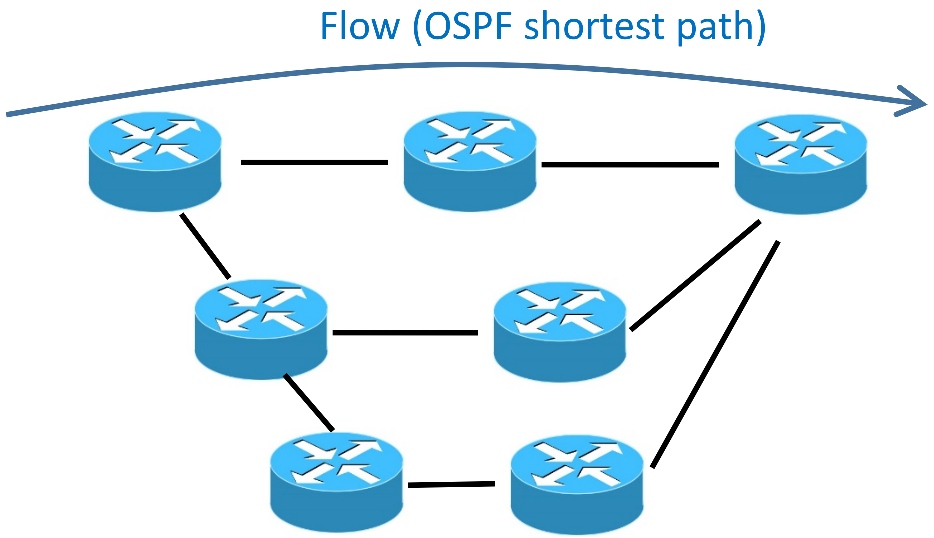
# Key open questions

- It is critical for ISPs to make a *roll-out planning*:

1. **How many** nodes to upgrade to SDN in each **time** period (6-month or year)? Should the ISP upgrade all nodes as early as possible or wait for the prices to fall?

2. After deciding the number of nodes to be upgraded, **which** specific nodes to select and **when**?

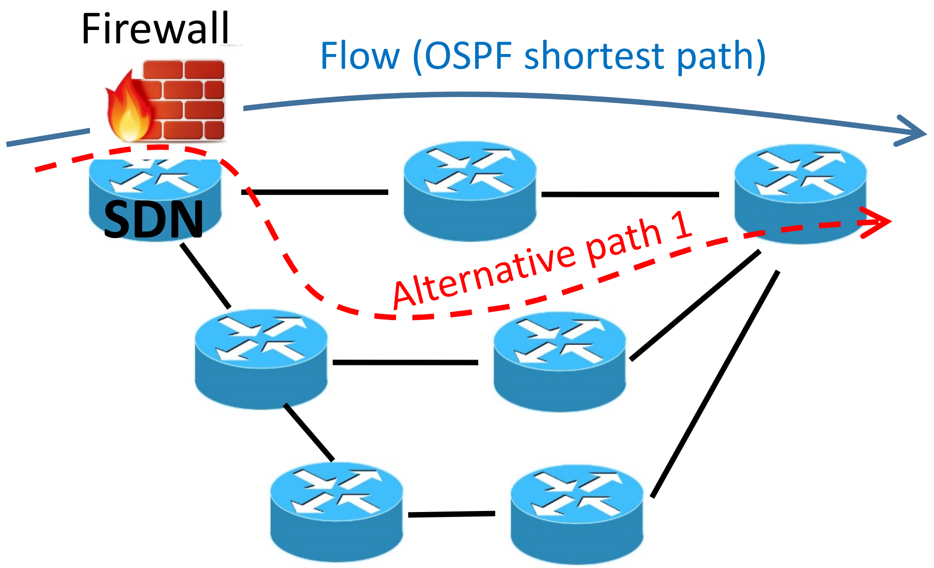
# Understanding the benefits of hybrid SDN




***An example network***

# Understanding the benefits of hybrid SDN

- If a flow that crosses *at least one* SDN-enabled node we can:
  - Implement access policies (firewall) and other middlebox-supported network services.
  - Dynamically reroute the flow towards alternative routing paths by overwriting the OSPF protocol.




***An example network***

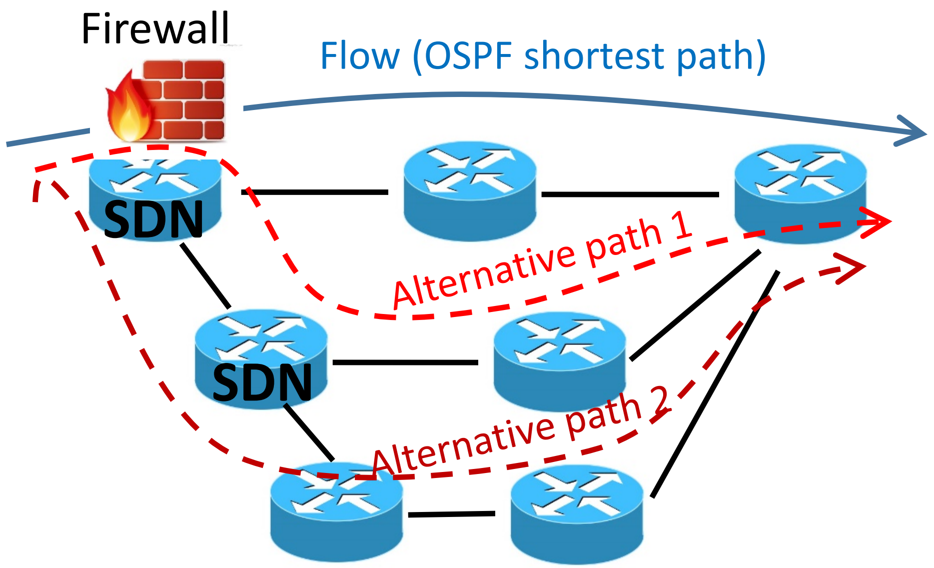
***Performance objectives:***

*Obj1: 'Programmable traffic':*

traffic that crosses at least one SDN node.

# Understanding the benefits of hybrid SDN

- If a flow that crosses *at least one* SDN-enabled node we can:
  - Implement access policies and other middlebox-supported network services.
  - Dynamically reroute the flow towards alternative routing paths by overwriting the OSPF protocol.
- If the flow crosses *more than one* SDN-enabled node there are even more dynamic routing options.




***An example network***

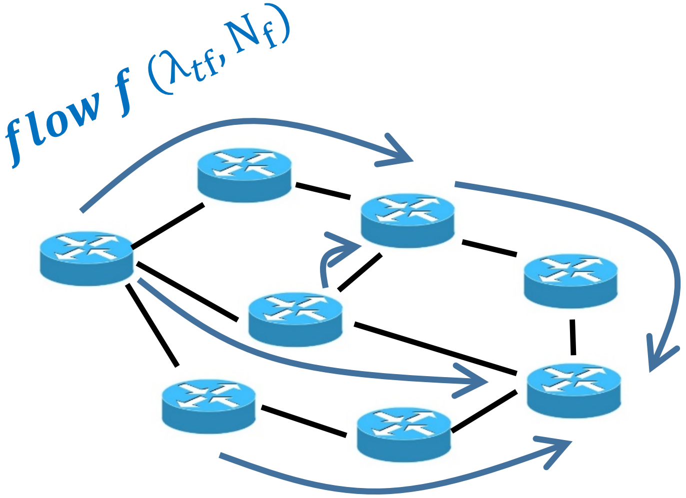
***Performance objectives:***

*Obj1: 'Programmable traffic':*
traffic that crosses at least one SDN node.

*Obj2: 'TE flexibity':*
# alternative paths for dynamic rerouting enabled by SDN nodes.

# ISP model

- **A general ISP network:**
  - *N* set of nodes can be upgraded to SDN.
  - *T* time periods (e.g., years) in which upgrades take place.
  - *F* traffic flows with increasing rates over time:
    $\lambda_{tf}$ where $\lambda_{1f} \leq \lambda_{2f} \leq \cdots \leq \lambda_{Tf}, \forall flow\ f$
    $N_f \subseteq N$ *nodes along the OSPF path of flow f*




  - $b_{tn}$ (\$) cost for upgrading node n at time period t
  - *B (\$) total budget for upgrades:* $\sum_{period\ t} \sum_{node\ n} b_{tn} x_{tn} \leq B$

    *Binary variable: 1 if node n is upgraded at time t*

  - *Each node can be upgraded in at most one period:* $\sum_{period\ t} x_{tn} \leq 1, \forall n$

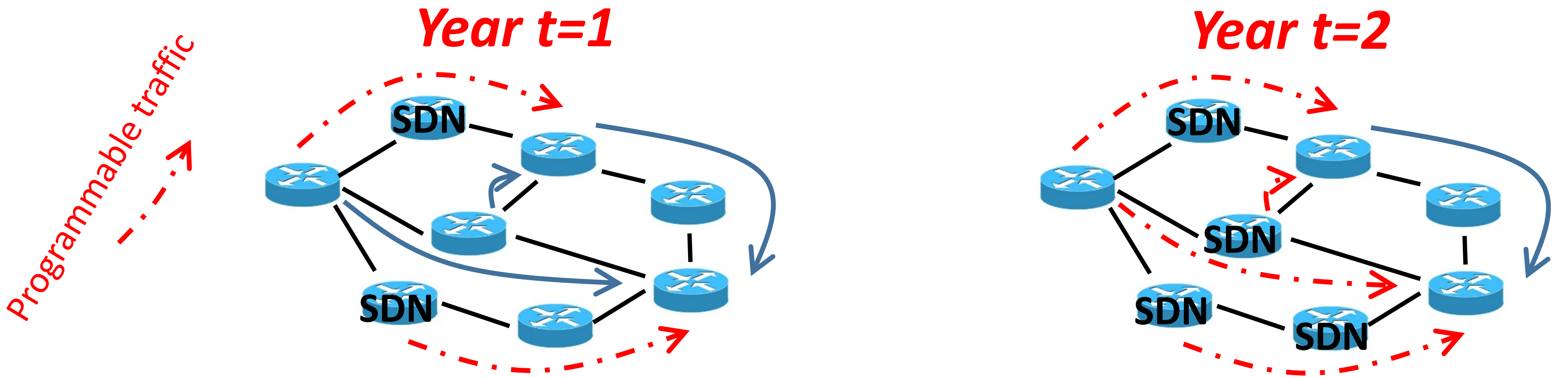
# Formulating programmable traffic maximization (Obj1)

- *Maximize programmable traffic* within a total time window.
  - $\sum_{period\ t}\sum_{flow\ f}\boldsymbol{\lambda}_{tf}\,\mathbf{1}_{\{\sum_{node\ n\,\in N_f}\sum_{period\ t'\le t} x_{t'n}\ge 1\}}$

*Indicator function $1_{\{c\}}$=1 if condition c is true; otherwise 0.*

*Programmable if at least one node on the OSPF path of flow f has been upgraded to SDN by period t.*

- **Example:**



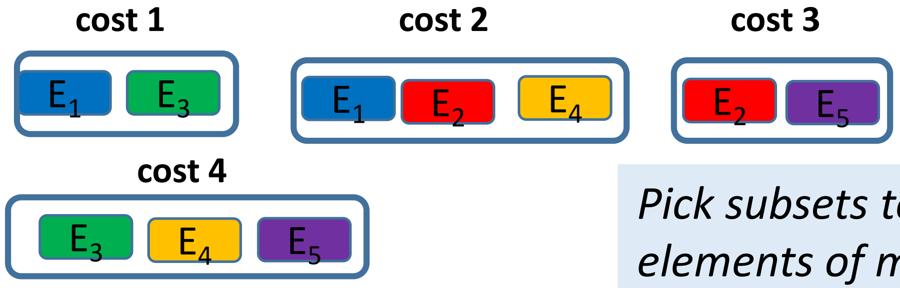
# Analyzing the complexity for Obj1

- Simple case: T=1 period

***Lemma 1:*** The SDN upgrading problem for T=1 and the programmable traffic maximization objective is equivalent to the ***Budgeted Maximum Coverage problem***.

A set of **elements**:


A set of ***subsets***:




*Pick subsets to cover elements of max weight given a cost budget.*

- NP-Hard problem.

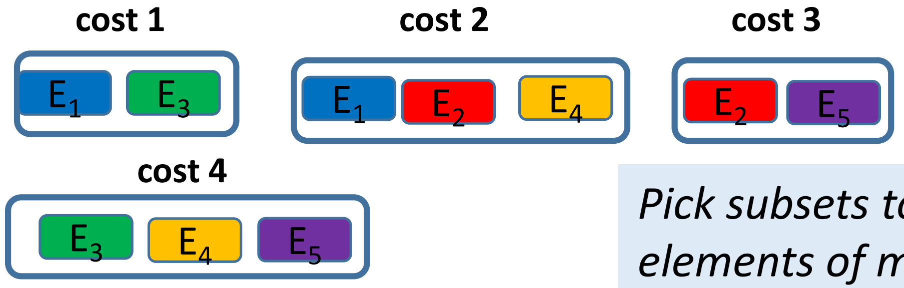
# Analyzing the complexity for Obj1

- Simple case: T=1 period

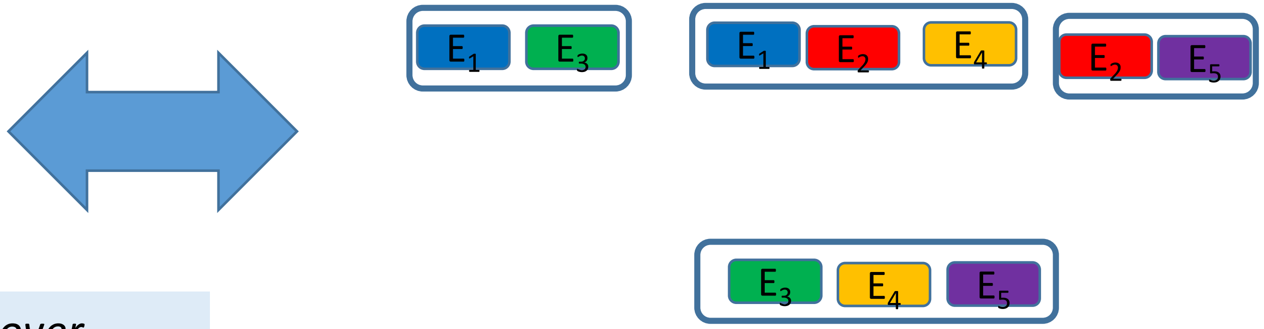
***Lemma 1:*** The SDN upgrading problem for T=1 and the programmable traffic maximization objective is equivalent to the ***Budgeted Maximum Coverage problem***.

A set of **elements**:


A set of ***subsets***:




✓ ***Create a node per subset***




*Pick subsets to cover elements of max weight given a cost budget.*

- NP-Hard problem.

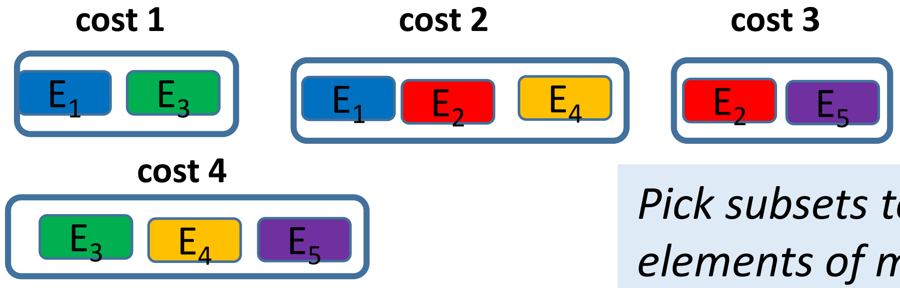
# Analyzing the complexity for Obj1

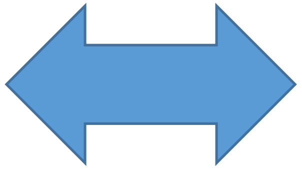
- Simple case: T=1 period

***Lemma 1:*** The SDN upgrading problem for T=1 and the programmable traffic maximization objective is equivalent to the ***Budgeted Maximum Coverage problem***.

A set of **elements**:

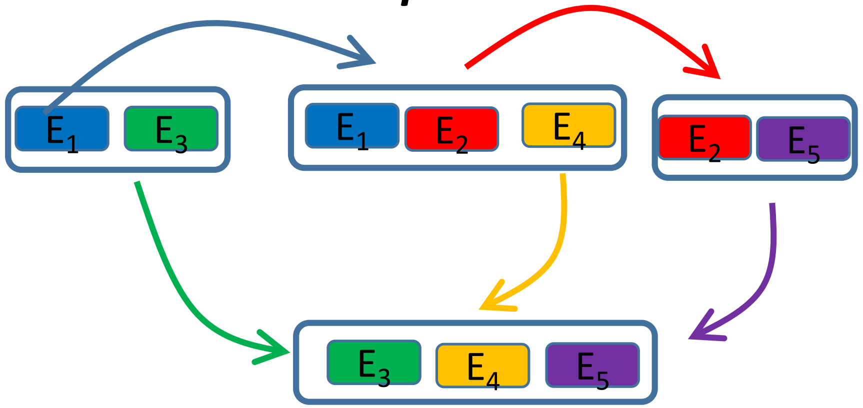

A set of ***subsets***:



*Pick subsets to cover elements of max weight given a cost budget.*

✓ ***Create a node per subset***




✓ ***Create a flow per element***

- NP-Hard problem.

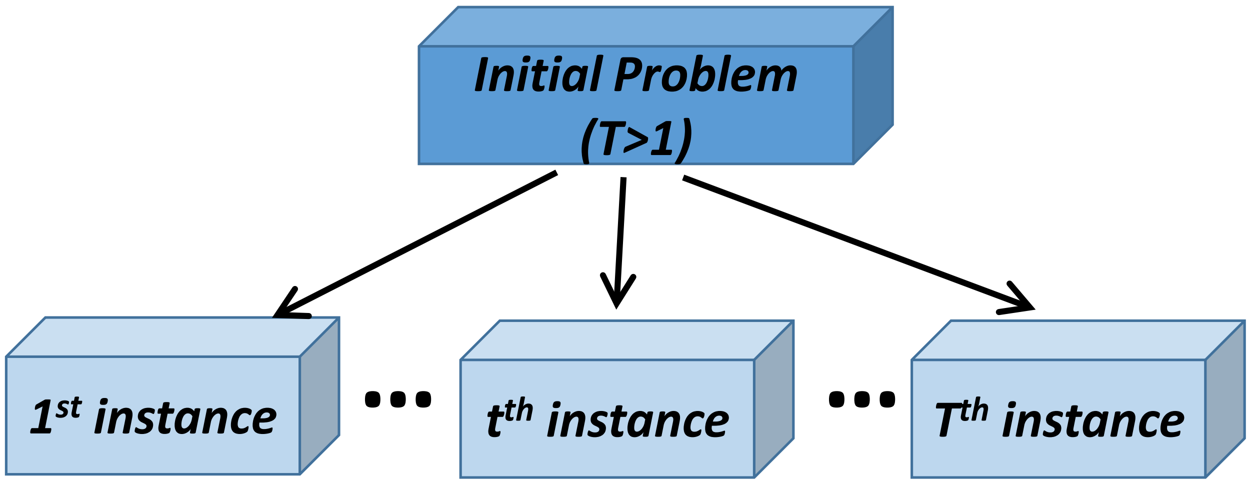
# Approximation Algorithms for Obj1 for T=1

- *Modified-greedy algorithm* achieves *(1-1/e)*-approx. ratio (best possible).

1. For each ***triplet*** of nodes:
   a. Upgrade the three nodes in the triplet.
   b. Repeat:
      - Upgrade the node with the highest ratio of traffic that becomes programmable over upgrading cost.
      - If the above upgrade will exceed the budget, skip this node.
   c. Until all nodes are examined.
2. Pick the solution with the highest programmable traffic.
3. Compare the solution found with any other solution of cardinality one or two, and replace it if this improves programmable traffic.

# Extend Modified-greedy for many periods (T>1)

- Define a set of *single time period problems* where all the budget B is spent within that single period (no budget is spent in the rest periods).




- ✓ ***Solve the T problems independently (e.g., by running Modified-greedy T times).***
- ✓ ***Pick the solution with the best performance.***
- ✓ ***Simple but log(T) loss in approx. factor. Can we do better?***

***Lemma 2:*** We can extend any α-approx. for the T=1 case of the SDN upgrading problem to obtain an **O( α / log(T) )**-approx. for the general case (T>1).

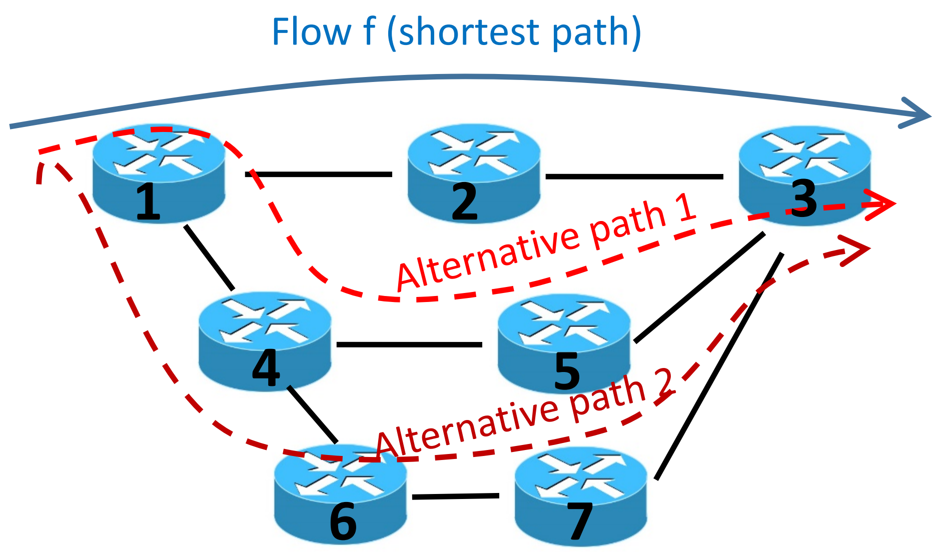
# Do better than log(T)-approx.

- We show the *submodular* property of the programmable traffic function:
  - Diminishing return rule: the benefits of upgrading a node decrease as the set of already upgraded nodes expands.
- Valuable result as several approx. algorithms are known for this class of problems.
- Cast as the *maximization of a submodular function* subject to:
  - *a knapsack constraint* (budget), and
  - *a matroid constraint* (each node can be upgraded at most in one period).
- ***Pipage rounding algorithm*** ⟶ ( 1 - 1/e - ε )-approx., ∀ε>0
- ***Local search algorithm & enumeration method*** (more practical)

⟶ ( 1/(2+ε) )-approx., ∀ε>0

# TE flexibility maximization (Obj2)

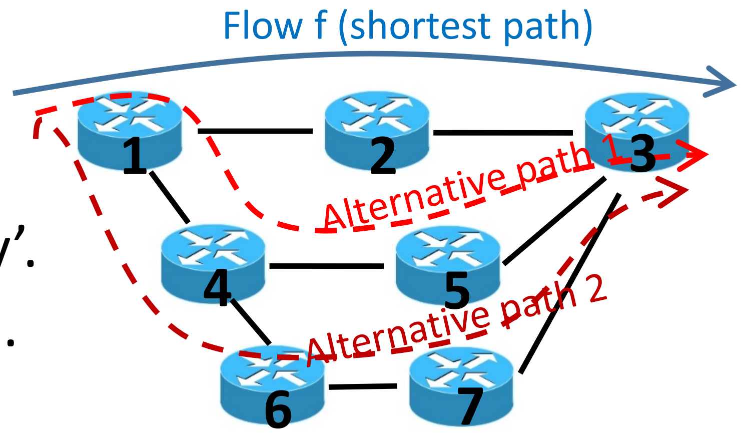
- TE flexibility = number of dynamically selectable routing paths enabled by SDN nodes.
- A group of *key nodes* need to be upgraded to SDN to be able to select an alternative routing path.




- ✓ ***Node 1 is the key node for alternative path 1.***
- ✓ ***Both nodes 1 and 4 are key nodes for alternative path 2.***
- ✓ ***TE flexibility can be expressed as a function of the key nodes and the SDN upgrading policy.***

# TE flexibility maximization algorithm

- This is a very different problem:
  - *Not a submodular function*. The diminishing return rule does not hold.
  - We cannot apply the same methods used for Obj1.




- We can use the concept of *supermodular degree (D)*.
  - Captures the `deviation of a function from submodularity'.
  - This depends on the instance, e.g., in the next figure D=1.

- Super-greedy ( 1/( 2(D+1)+1 ) )-approx. for uniform upgrading costs.
  - Iteratively, pick a pair (node, time) and a **subset** of pairs that increase the marginal gain of the former. Greedily, augment them to the current solution.

# State of the art on hybrid SDN

- Hybrid SDN *models and tradeoffs* **[Vissicchio2014].**
- Design of *routing policies* in hybrid SDN networks **[Agarwal2013].**
- Hybrid SDN *upgrading strategies* that *neglect the timing issue*, do not provide approximation guarantees and have different objectives:
  - **[Levin2014]**: PANOPTICON heuristics minimize path stretch when all traffic is programmable.
  - **[Hong2016]**: heuristics minimize maximum link utilization.
  - **[Kar2016]**: heuristics maximize two similar coverage metrics.
  - **[Caria2015]**: divide & conquer to partition the network into OSPF subnetworks.
  - **[Wang2016]**: maximizes a network connectivity metric to prevent flooding attacks.
  - **[Xu2017]**: incrementally adds SDN nodes instead of replacing the legacy with SDN ones.
  - **[Caria2013]**: schedules SDN upgrades over time yet, it *does not provide tight bounds*, nor it analyzes the *impact of equipment cost reduction*.

# Compare with two state-of-the-art heuristics

- *DEG [Hong2016]:* upgrades the nodes with the highest degrees (number of incoming and outgoing adjacent links) in the topology graph. All the upgrades take place at the first time period (t=1).

- *VOL [Hong2016], [Levin2014]:* upgrades the nodes with the highest traffic volume that traverses them. All the upgrades take place at the first time period (t=1).

# Evaluation setup

- *Abilene dataset*[1]*:* backbone network in North America.
  - 12 nodes and 30 directed links.
  - 12x12=144 flows.
  - traffic matrices.
  - OSPF weights per link.

- Practical time windows of T= 1,2,3,4,5 years are examined.

- Our evaluation code is *publicly available online*[2].

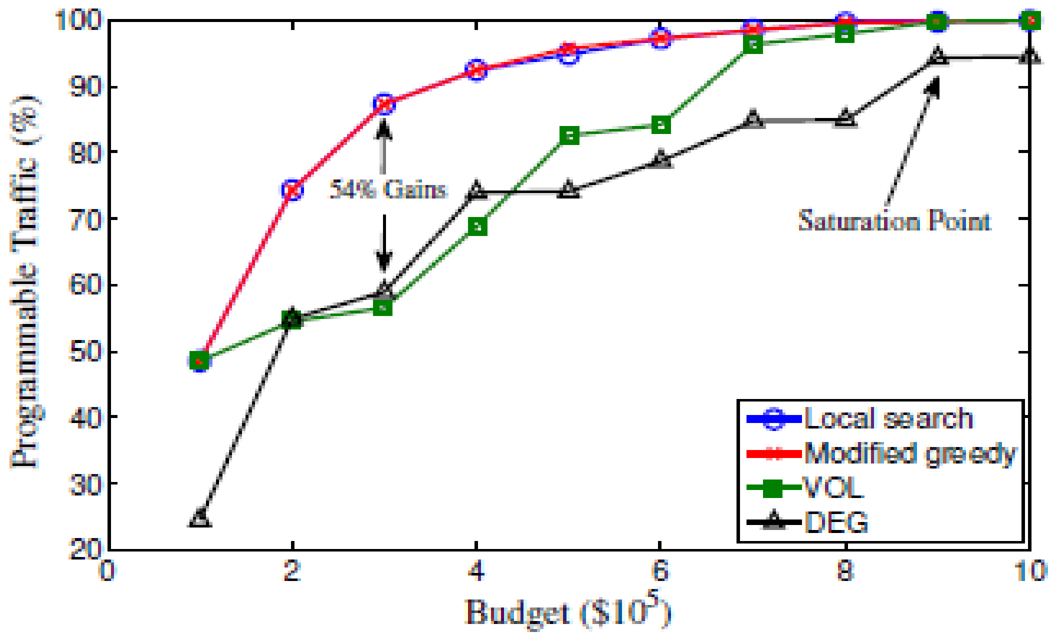
[1] http://www.cs.utexas.edu/yzhang/research/AbileneTM

[2] goo.gl/EXoZZZ

# Evaluation results for T=1

- Fair comparison with VOL and DEG.
  - *Benefits up to 54%*.
  - Saturation point as budget increases.

**_Setup:_** \$100K cost per router ($b_{tn}$). +22% traffic per year ($\lambda_{tf}$).

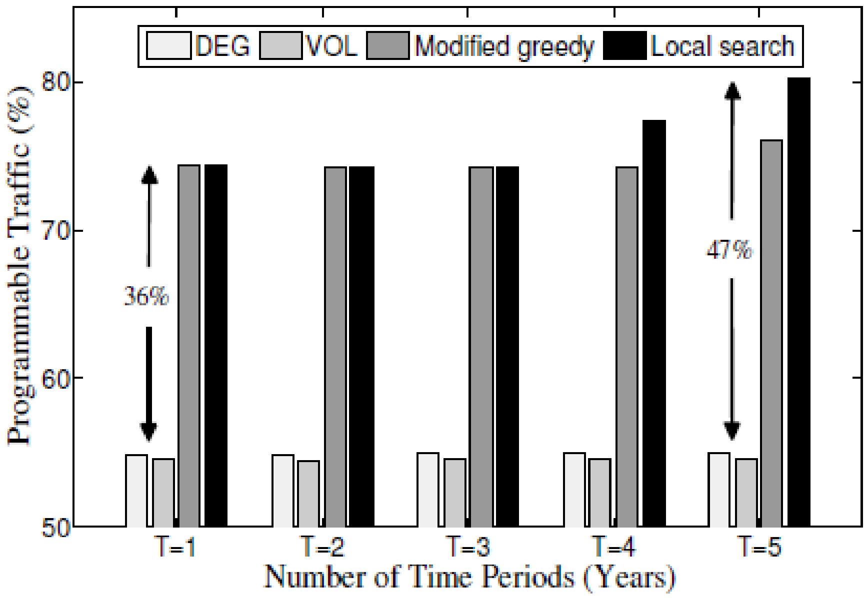



- The performance depends on the network structure:
  - 12% gains are also reported for a larger network with >100 nodes (Deltacom,US).

# Evaluation results for T>1

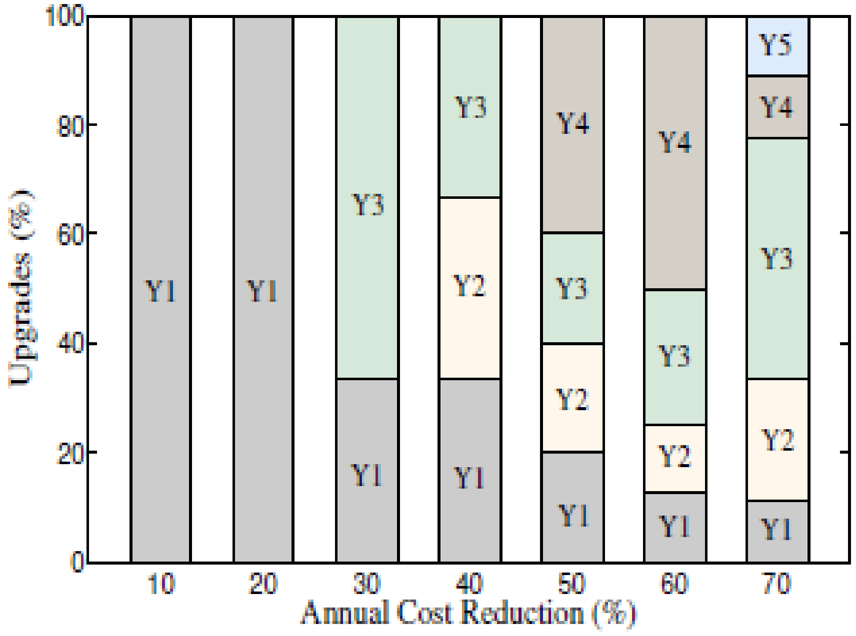
- Impact of number of periods T.
  - Benefits over state-of-the-art methods increase with T.
  - Local search performs better than Modified-greedy for sufficiently large T (T>3).

***Setup:*** $200K total budget (B)
-40% cost per year ($b_{tn}$)



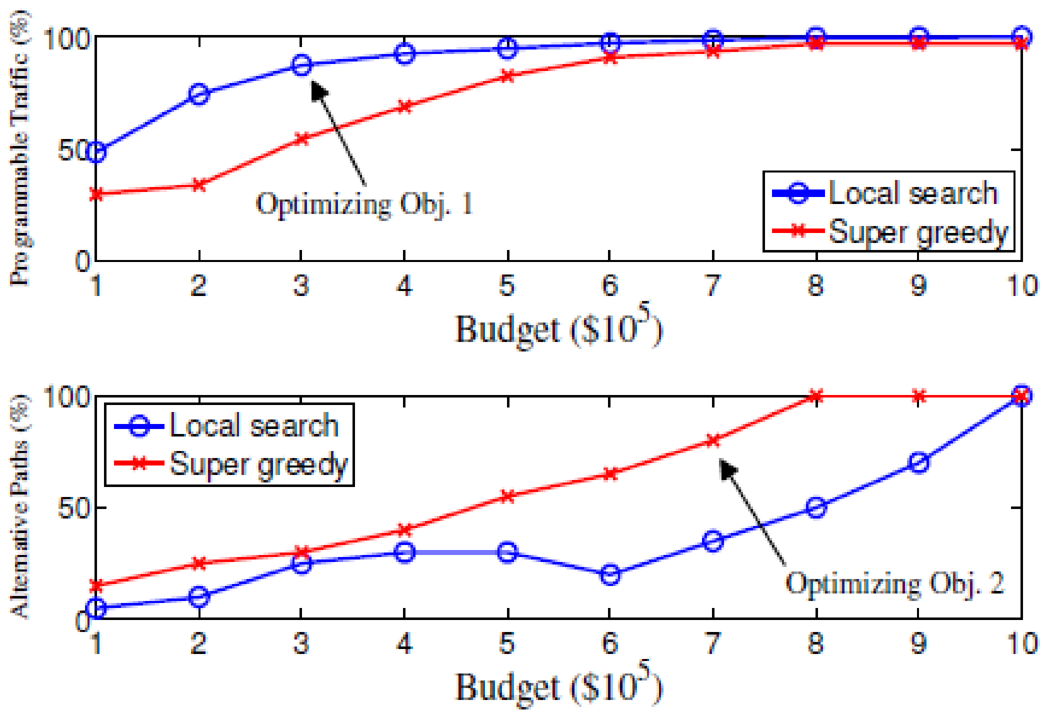
# Evaluation results: upgrades over time

- Upgrades are *spread across many years* by Local-search algorithm.
    - This is more pronounced when the *rate of reduction in the equipment cost* is high.
    - When this rate is $\leq 20\%$, then all the upgrades should be performed in the first year. Otherwise spread them across years.



# Evaluation results: different objectives

- *Interplay between the two objectives:*
  - Programmable traffic vs TE flexibility.
  - Maximizing one has a positive impact on the other metric.
  - Up to *a factor of 2 performance loss*.



# Conclusion

- We studied the problem of roll-out SDN deployment. In particular, we decided *which* part of the network to upgrade and *when*.
- We also proposed and studied approximation algorithms for two different objectives.
- By applying our algorithms in real topologies and traffic measurements, we showed that they can yield better performance than state-of-the-art methods, *especially when the upgrades span multiple years*.
- We are currently evaluating additional objectives such as resilience to malfunction of SDN equipment, network failures and malicious attacks.

# References


1. S. Vissicchio, L. Vanbever, and O. Bonaventure, "**Opportunities and research challenges of hybrid software defined networks**", *ACM SIGCOMM Comput. Commun. Rev.*, 2014.
2. S. Agarwal, M. Kodialam, and T. Lakshman, "**Traffic engineering in software defined networks**", in *Proc. IEEE INFOCOM*, 2013.
3. D. Levin *et al.*, "**Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks**", in *Proc. USENIX ATC*, 2014.
4. D.K. Hong, et at., "**Incremental Deployment of SDN in Hybrid Enterprise and ISP Networks"**, In Proc. ACM SOSR 2016.
5. B. Kar, et al., "**The Budgeted Maximum Coverage Problem in Partially Deployed Software Defined Networks**", IEEE TNSM 2016.
6. M. Caria, T. Das, and A. Jukan, "**Divide and conquer: Partitioning OSPF networks with SDN**", in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, 2015.
7. L. Wang, et al., "**Towards Mitigating Link Flooding Attack Via Incremental SDN Deployment"**, In Proc. IEEE ISCC 2016.
8. H. Xu, et al., "**Incremental Deployment and Throughput Maximization Routing for a Hybrid SDN"**, IEEE/ACM Trans. Networking, 2017.
9. M. Caria, A. Jukan, M. Hoffmann, "**A Performance Study of Network Migration to SDN-enabled Traffic Engineering**", in Proc. IEEE Globecom, 2013.

- BACKUP SLIDES

# Local-search algorithm

- Local-search algorithm:
  - Starts with a random feasible solution.
  - Iteratively, adds at most 2/ε elements and deletes at most 4/ε elements from the solution, if this local move improves the objective.
  - Until there is not such local move.

- 'Classic' local search algorithm works for problems with *two matroid constraints*. Here, we have *one matroid and one knapsack*.

- Use an enumeration technique to *convert the knapsack to a set of matroids*.