Network Adaptability under Resource Crunch

Rafael Braz Rebouças Lourenço

Networks Lab - UC Davis

Friday Lab Meeting - April 7th, 2017

UCDAVIS
UNIVERSITY OF CALIFORNIA

Outline

- What is Resource Crunch
- Problem Statement
- Example 1
- Connection Adjacency Graph (CAG)
- Splitting the problem
- Example 2
- Algorithm
- Results

Resource Crunch

- Different from occasional request blocking, consists of a situation in which offered demand cannot possibly be carried by the network
- May be caused by:

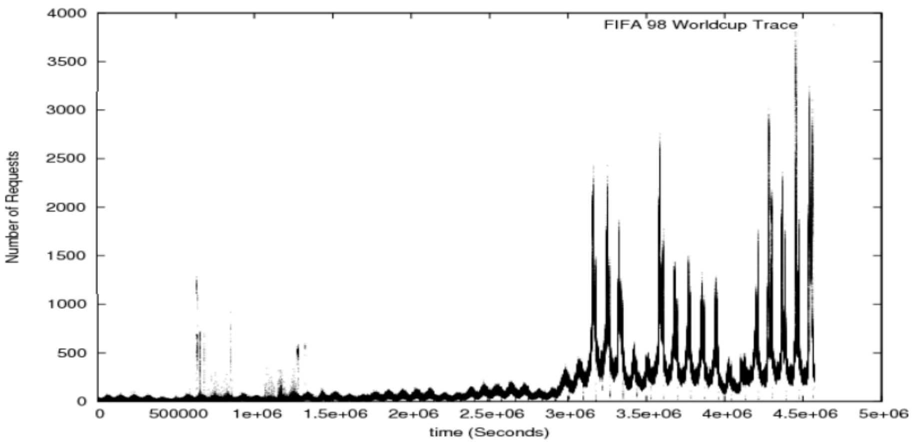
1. Failure arrivals (disasters) ➔ decrease transmission capacity
2. Traffic demand arrivals ➔ increase offered load

➔ How can we deal with Resource Crunch on layer 2.5 (MPLS/SDN flows)?

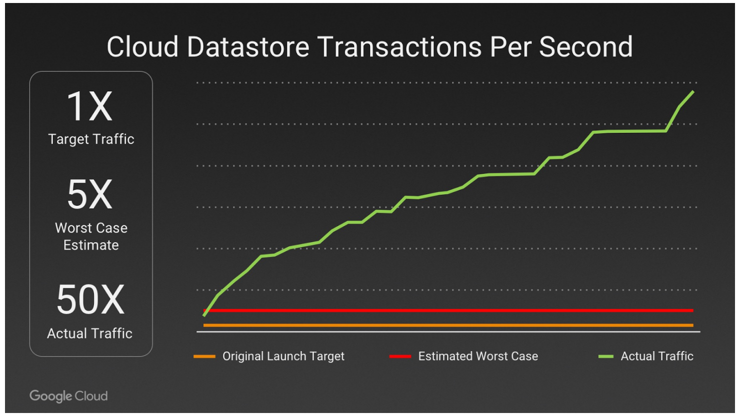
UCDAVIS

FOCUS: Resource Crunch due to Failures or Unexpected Traffic Surges (Flash Crowd)

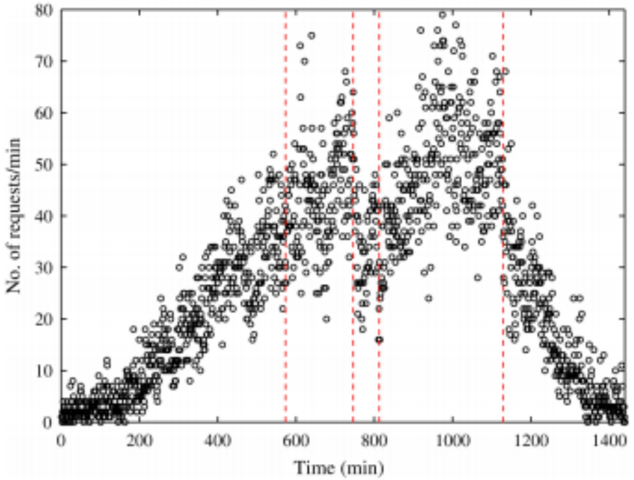
- Unpredicted (or underestimated) spikes in the traffic
- In cloud services environments, is commonly dealt with by spreading computation and/or redirecting traffic



[1] Rapid demand change for the FIFA world cup website



[2] Pokemon Go: Predicted X Observed traffic



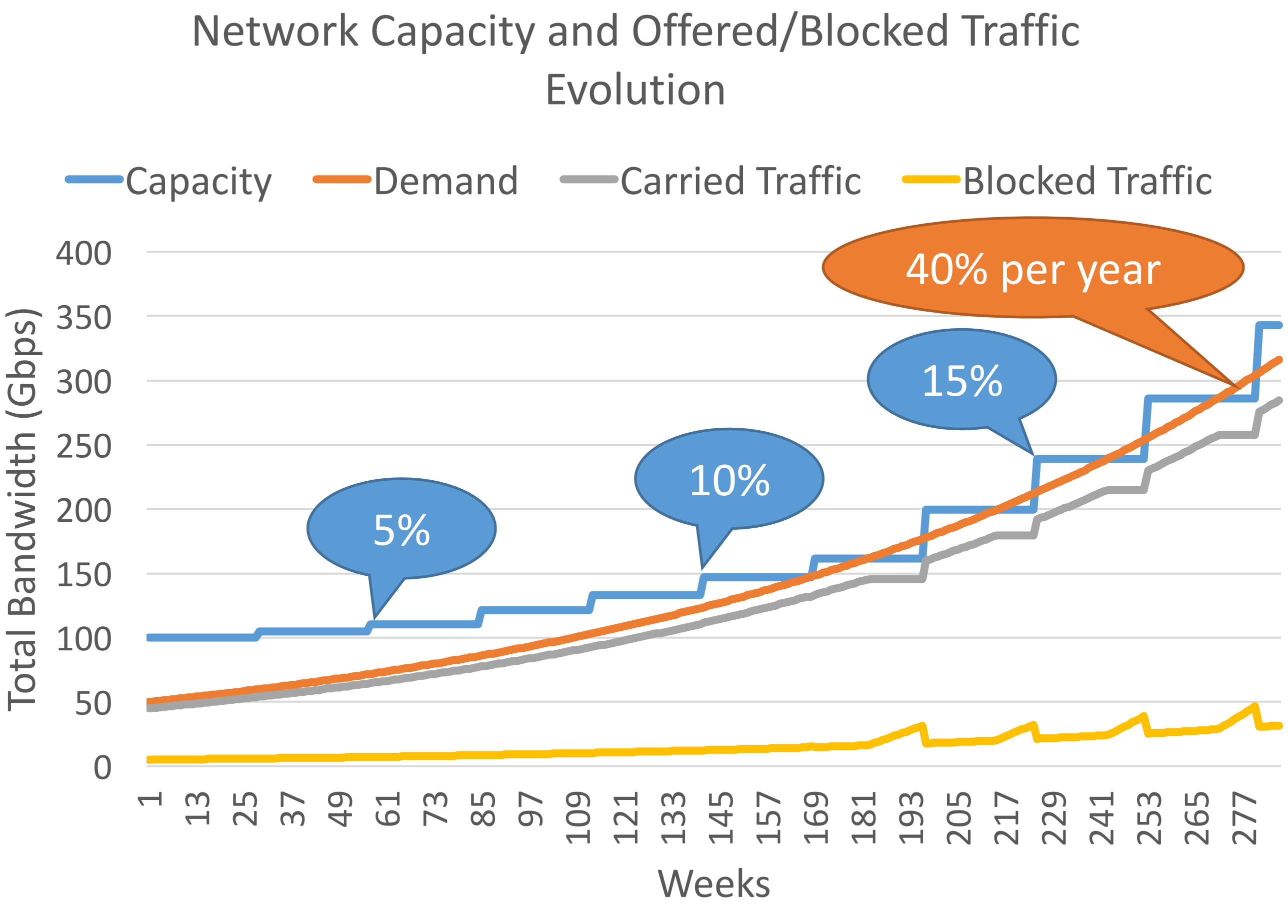
[3] Average Youtube traffic throughout 24h. Abnormalities between red lines

[1] Ali-Eldin, et al. "An adaptive hybrid elasticity controller for cloud infrastructures." *Network Operations and Management Symposium (NOMS), 2012 IEEE*.
[2] https://cloudplatform.googleblog.com/2016/09/bringing-Pokemon-GO-to-life-on-Google-Cloud.html
[3] Tang, et al. "Dynamic request redirection and elastic service scaling in cloud-centric media networks." *IEEE Transactions on Multimedia* 16.5 (2014)

UCDAVIS

But also… Long Term Traffic Growth

- Traffic continually grows, Cisco VNI estimates general internet traffic growth at an average of 22% a year for the next 5 years
- Network engineering activities are cyclically performed to install new network capacity and avoid bottlenecks in the system
- Several networks are already operating well above traditional occupancy levels (intra-datacenter networks, specially)



UCDAVIS

Building Block: Flexible *Service Level Objectives* for different Service Classes

** Different SLOs (such as Degradability, Capacity, or Availability, Latency, and others not shown here) yield in different prices*

Service	Real Time	Degradable	Ratio of all traffic	Requested Gbps	Minimum Gbps	Degradable Capacity	Price per Gbps per Link	Blocking Cost
Control Traffic (SCADA, etc)	Y	N	10%	**2**	**2**	**0**	$5	$10
Big Data Transfer (Backups, etc)	N	Y	20%	**10**	**5**	**up to 5**	$2	$5
Small Data Transfers	N	Y	20%	**5**	**4**	**up to 1**	$3	$5
Video on Demand (Youtube Netflix)	N	Y	30%	**3**	**1**	**up to 2**	$1	$1
HD Real-Time Video (HD TV)	Y	Y	14%	**4**	**2**	**up to 2**	$2	$1
Non-HD Real-Time Video (Regular TV)	Y	Y	6%	**2**	**1**	**up to 1**	$1	$0

UCDAVIS

Network Adaptability Under Resource Crunch

If an incoming demand cannot be placed due to Resource Crunch…

…can it somehow be served? *(by degrading other already allocated demands)*

If so, can we maximize the operator's revenue while serving the demand?

➔ Through which path?

➔ Using what throughput?

➔ At the expense of degrading which other demands?

Problem Statement

- Given:
  - Network topology
  - ***Potentially flexible SLOs*** for different Service Classes

Simplified Problem Statement:

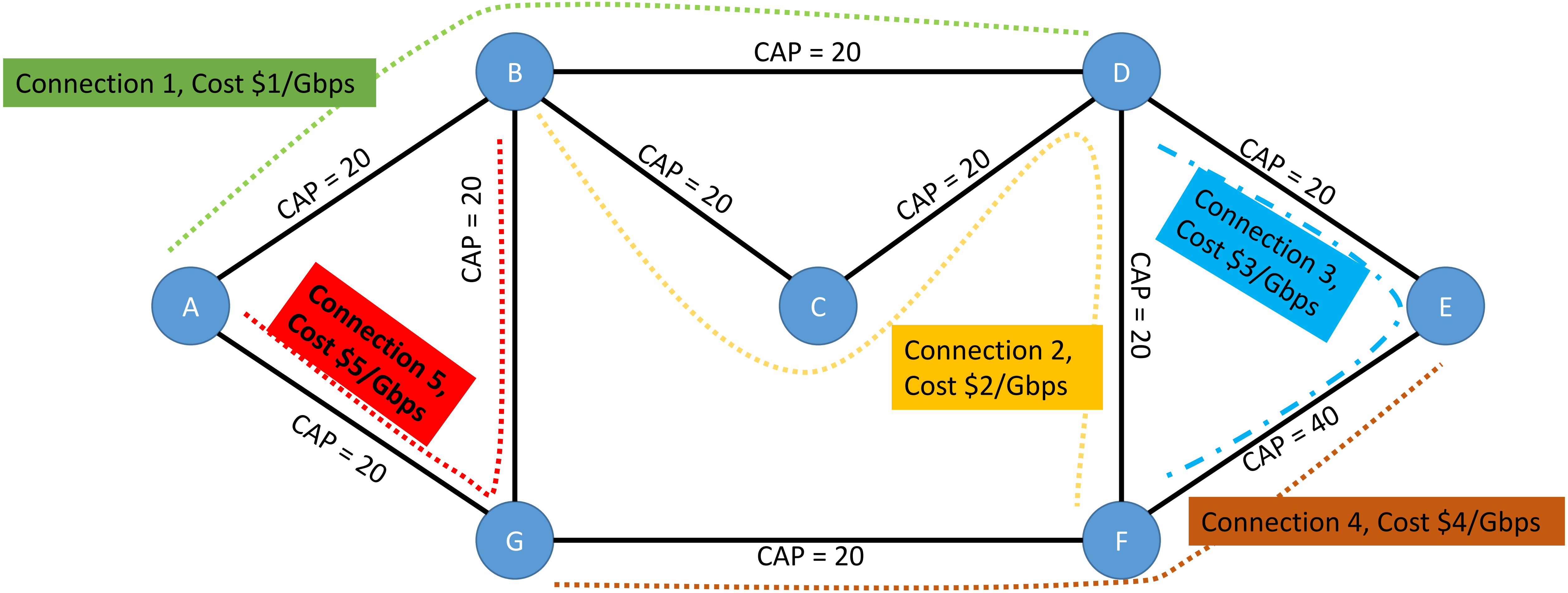
If an incoming demand cannot be normally served due to Resource Crunch: which other connections should we degrade in order to serve this demand (or should we not serve it at all)?

throughput, and by degrading which other demands

- Goal:
  - Maximize the overall revenue of the network operator
- Constraints:
  - Link rates, SLOs, network topology

UCDAVIS

Illustrative Example - 1

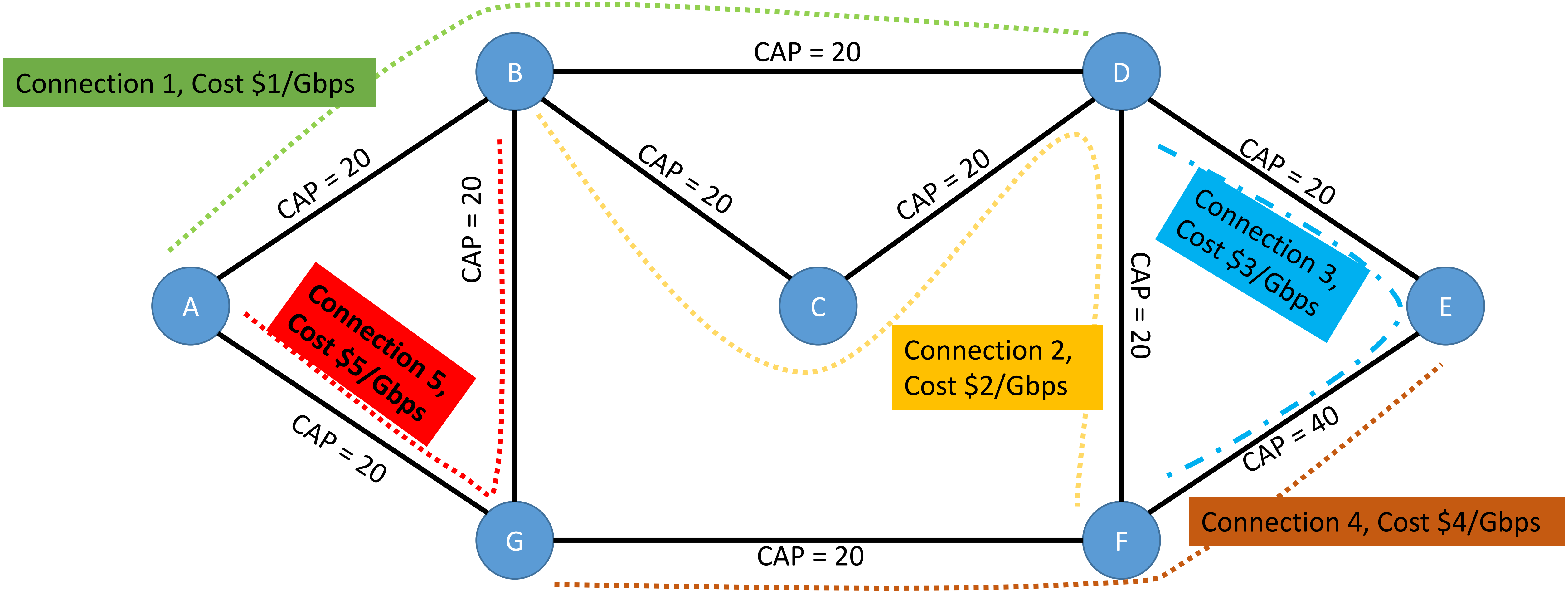


All connections have the same minimum and maximum required throughputs:
Min(C1)=Min(C2)=Min(C3)=Min(C4)=Min(C5) = 10Gbps
Max(C1)=Max(C2)=Max(C3)=Max(C4)=Max(C5) = 20Gbps

Thus, all link are being fully utilized.

UCDAVIS

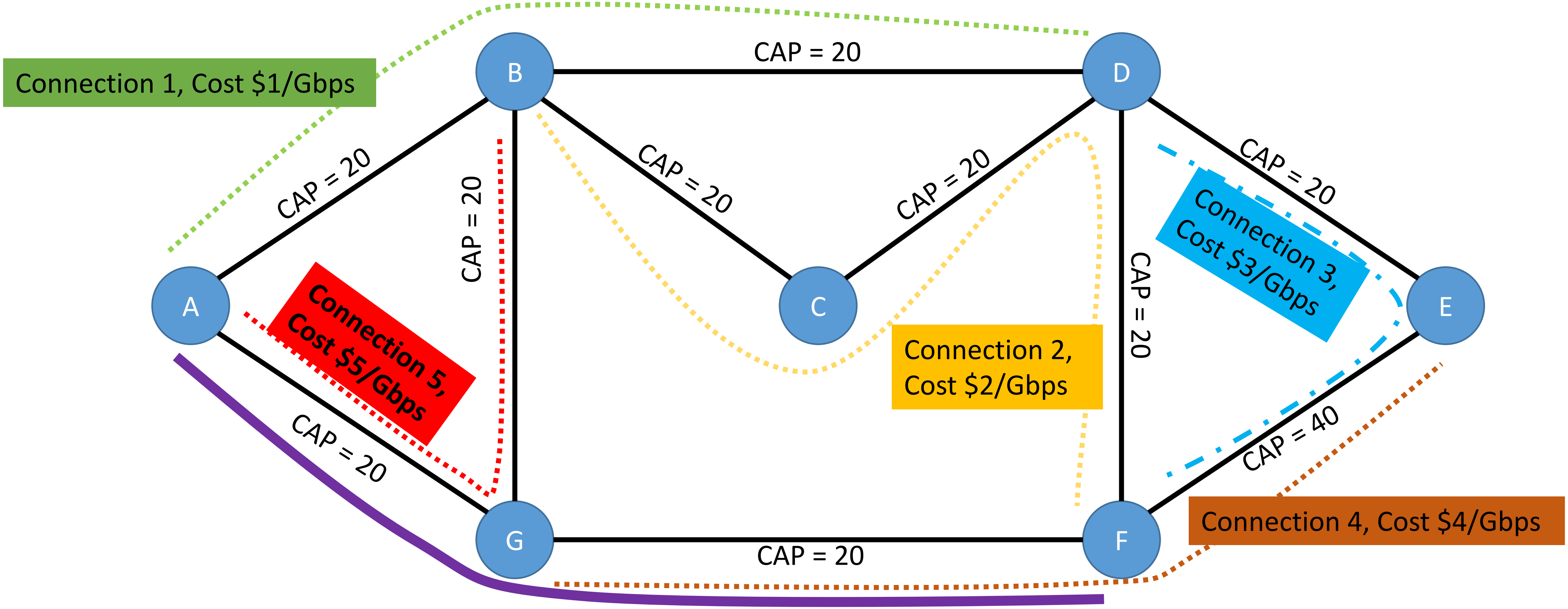
Illustrative Example - 1



A new request of 10Gbps arrives from A to F and cannot be normally served due to Resource Crunch. This new request offers to pay $4/Gbps.

UCDAVIS

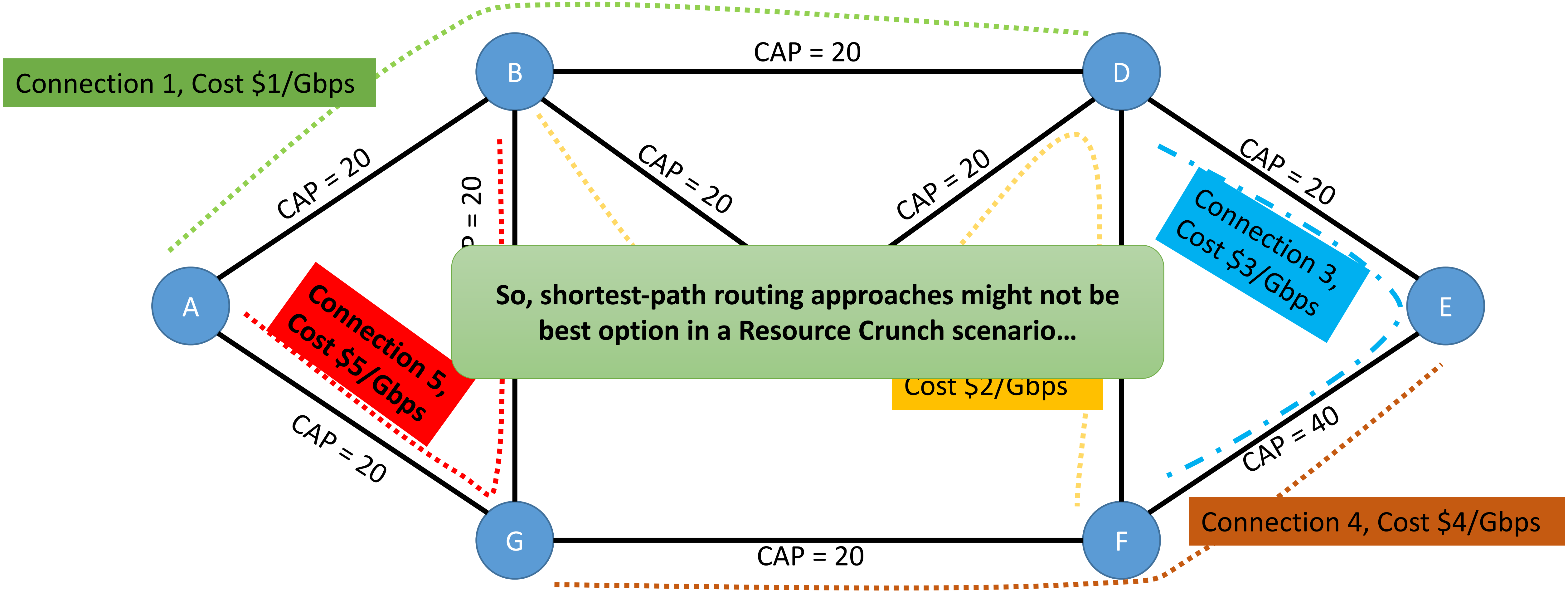
One Idea: Shortest path routing…



Degrade Connections 5 and 4, and place the new request on the shortest path.
The degradation would **decrease** the revenue in $(5x10+4x10) and the new request would **increase** it in $40.
Total revenue decreased by $50.

UCDAVIS

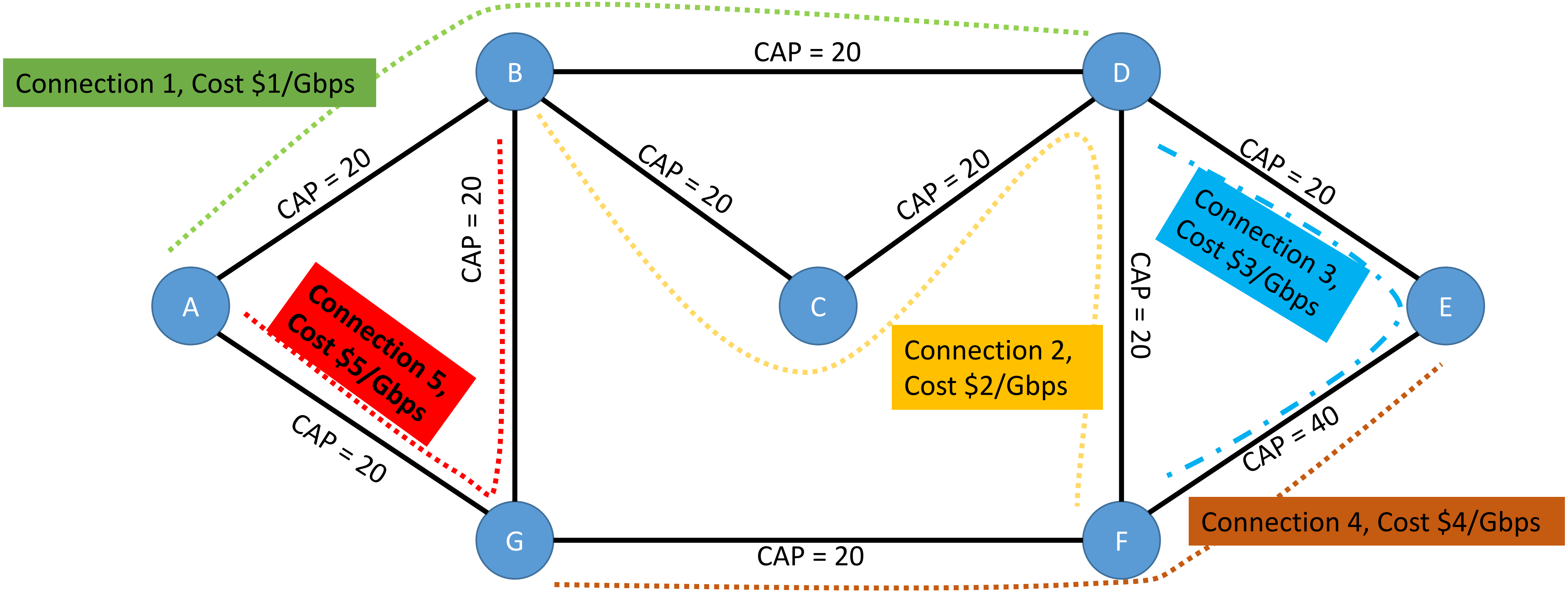
One Idea: Shortest path routing on prices…



So, shortest-path routing approaches might not be best option in a Resource Crunch scenario…

Does not work ➔ Once you pay the first cost, the following edges of the connection should be "free"…

However…



Notice how every time a connection is degraded it frees-up capacity throughout its entire path…
Wouldn't there be a more efficient way to utilize the capacity that was liberated?

Another Idea: Connection Adjacency Graph (CAG)

Connection5
[A, G, B]

Connection1
[A, B, D, E]

First, create super-vertices representing each connection. Annotate in each super-vertex the set of physical nodes that connection touches.

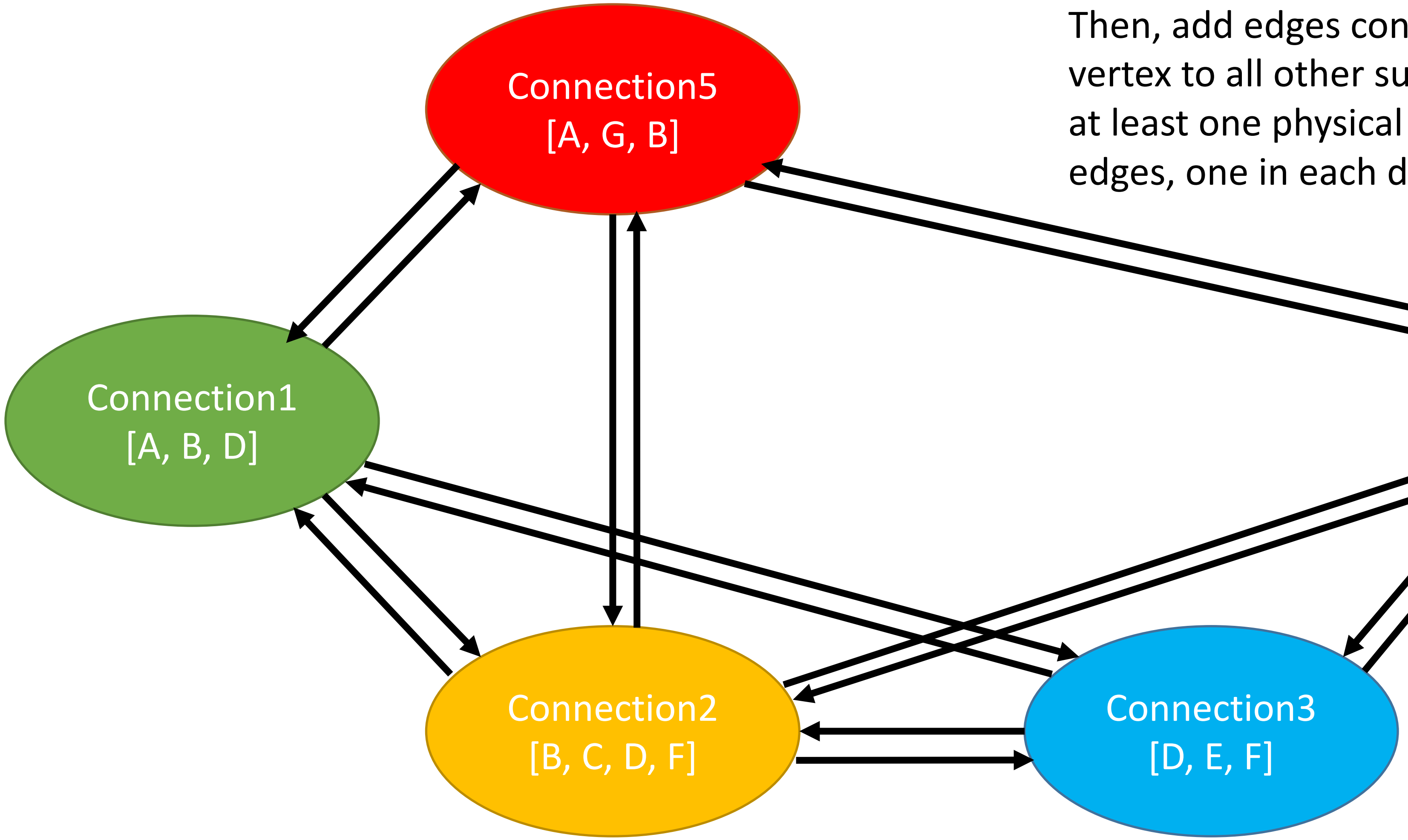
Connection4
[G, F, E]

Connection2
[B, C, D, F]

Connection3
[D, E, F]

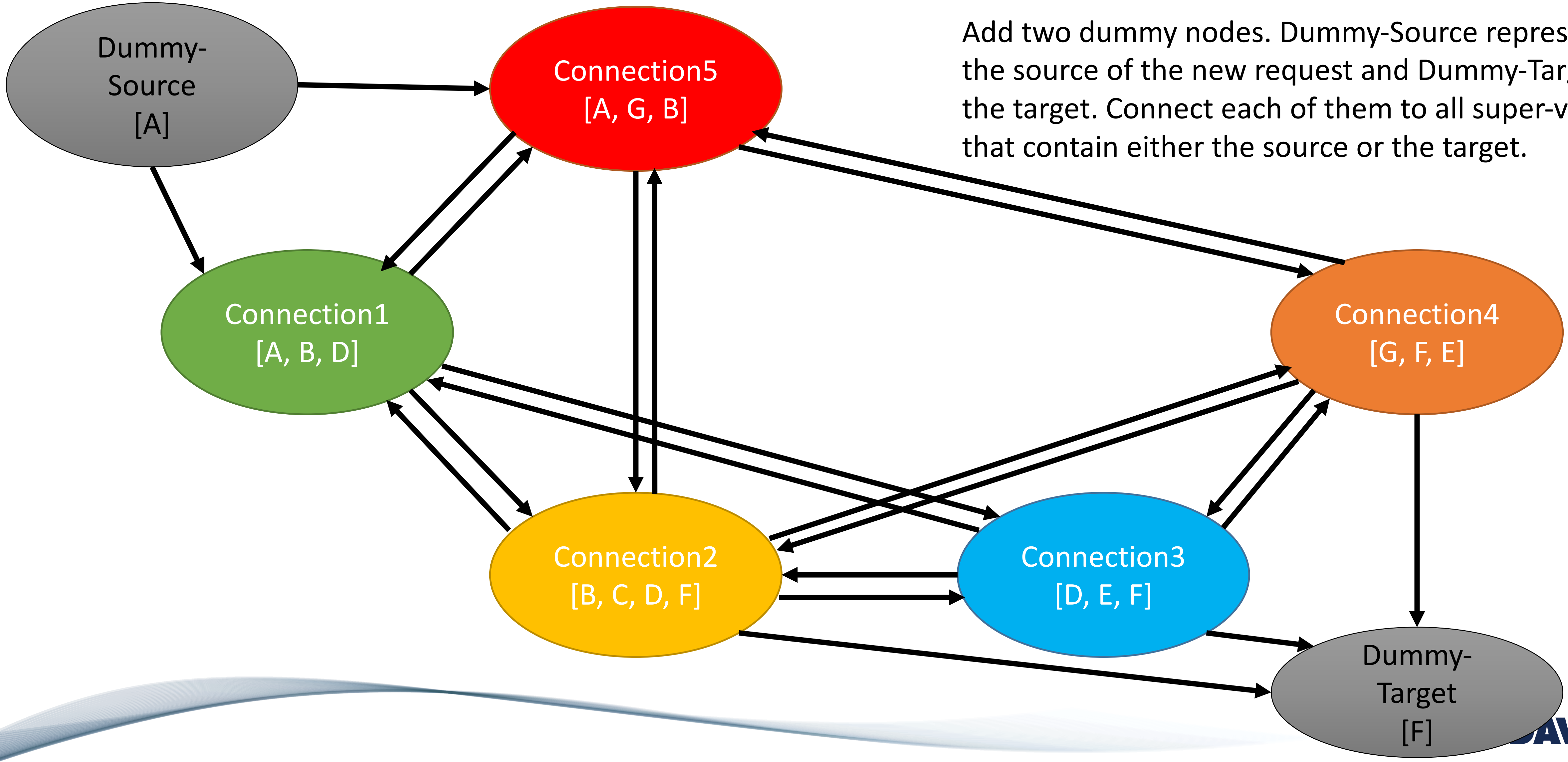
Another Idea: Connection Adjacency Graph (CAG)

Then, add edges connecting each super-vertex to all other super-vertices that contain at least one physical node in common. Add 2 edges, one in each direction.



Another Idea: Connection Adjacency Graph (CAG)

Add two dummy nodes. Dummy-Source representing the source of the new request and Dummy-Target, for the target. Connect each of them to all super-vertices that contain either the source or the target.



Associate with each edge a cost (aka, weight). For any edge, except those incoming to the Dummy-Target (whose cost are all 0), the cost of this edge is the cost of degrading in one throughput-unit the connection the super-vertex that edge points to represents.

Connection 1, Cost $1/Gbps

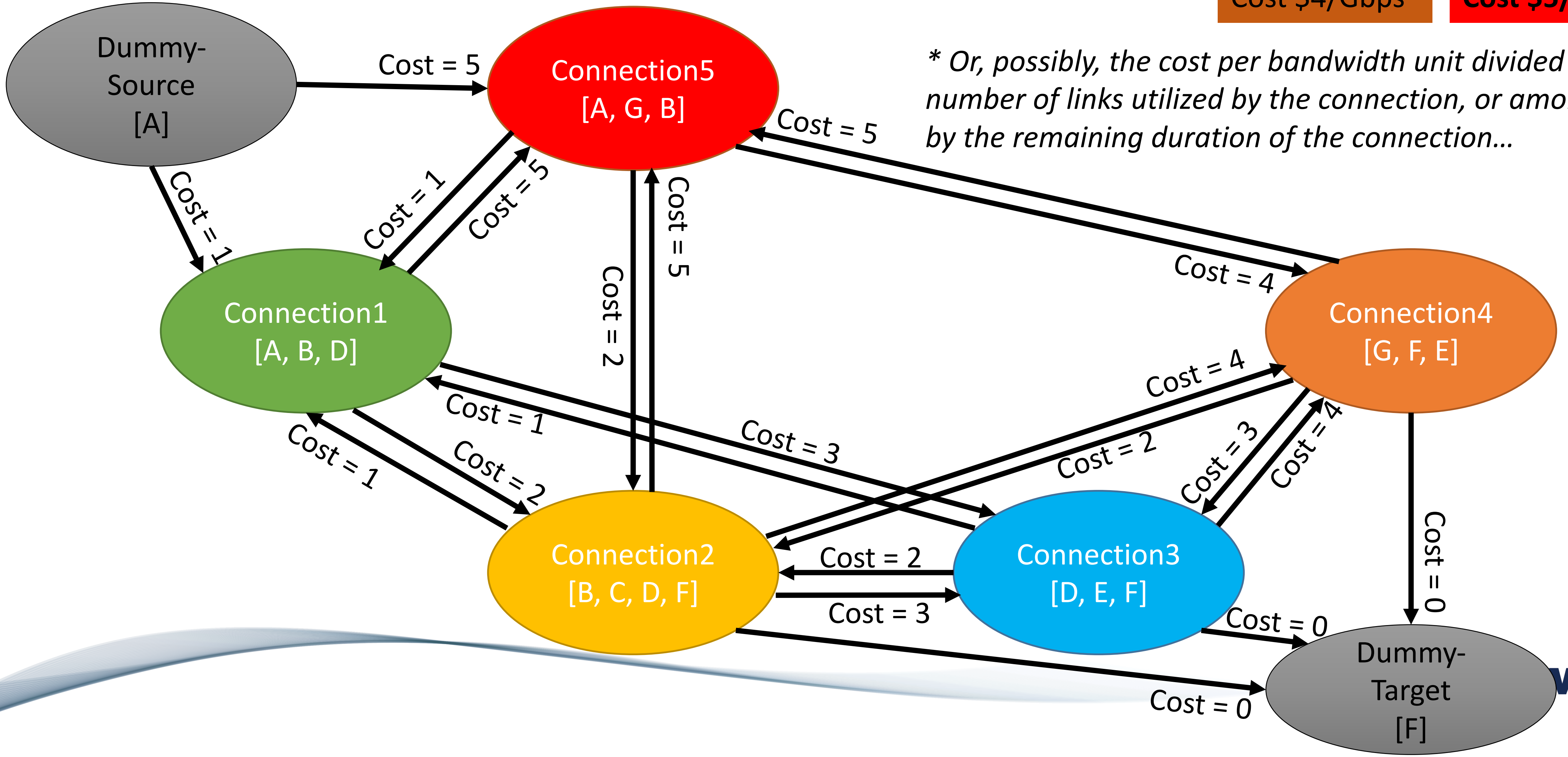
Connection 2, Cost $2/Gbps

Connection 3, Cost $3/Gbps

Connection 4, Cost $4/Gbps

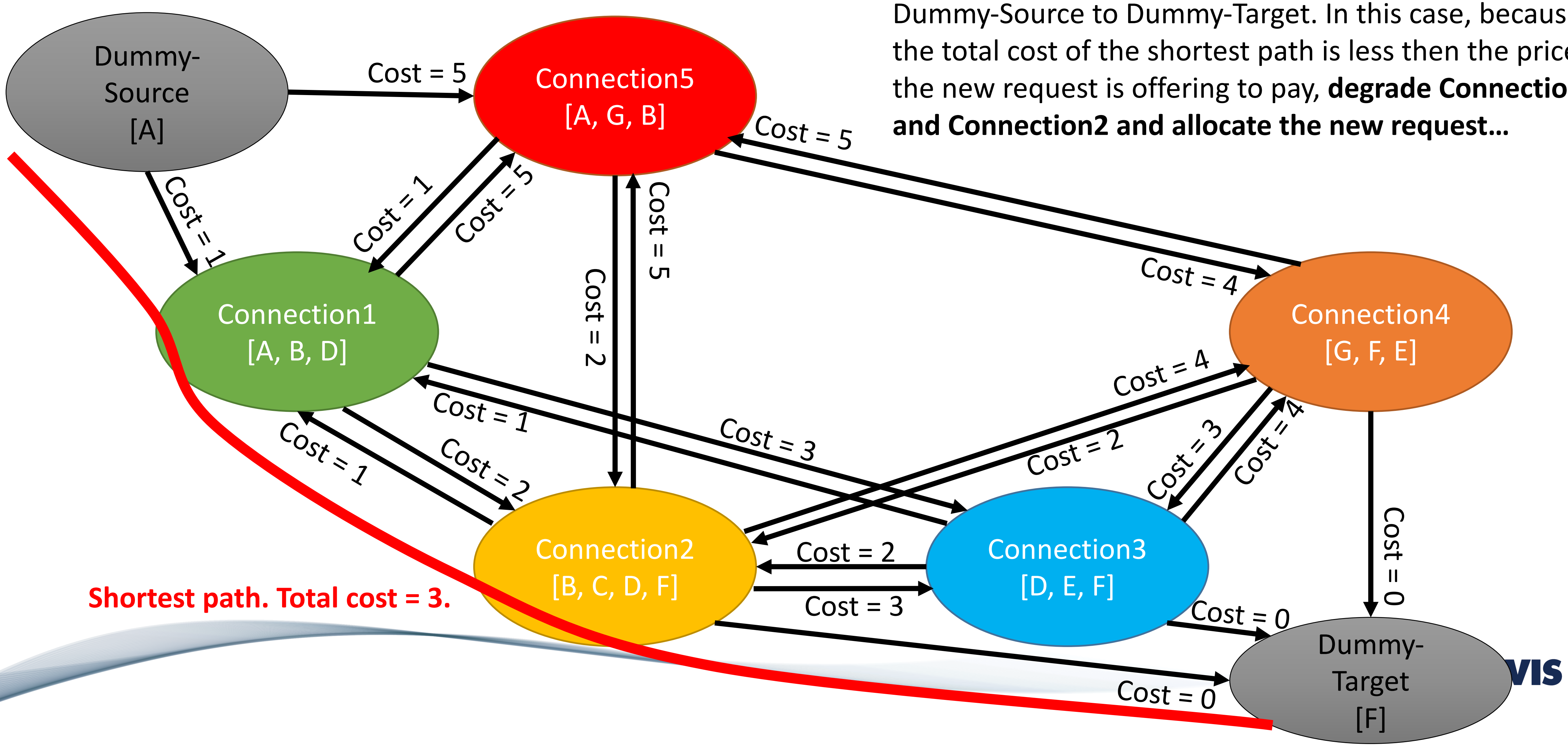
Connection 5, Cost $5/Gbps

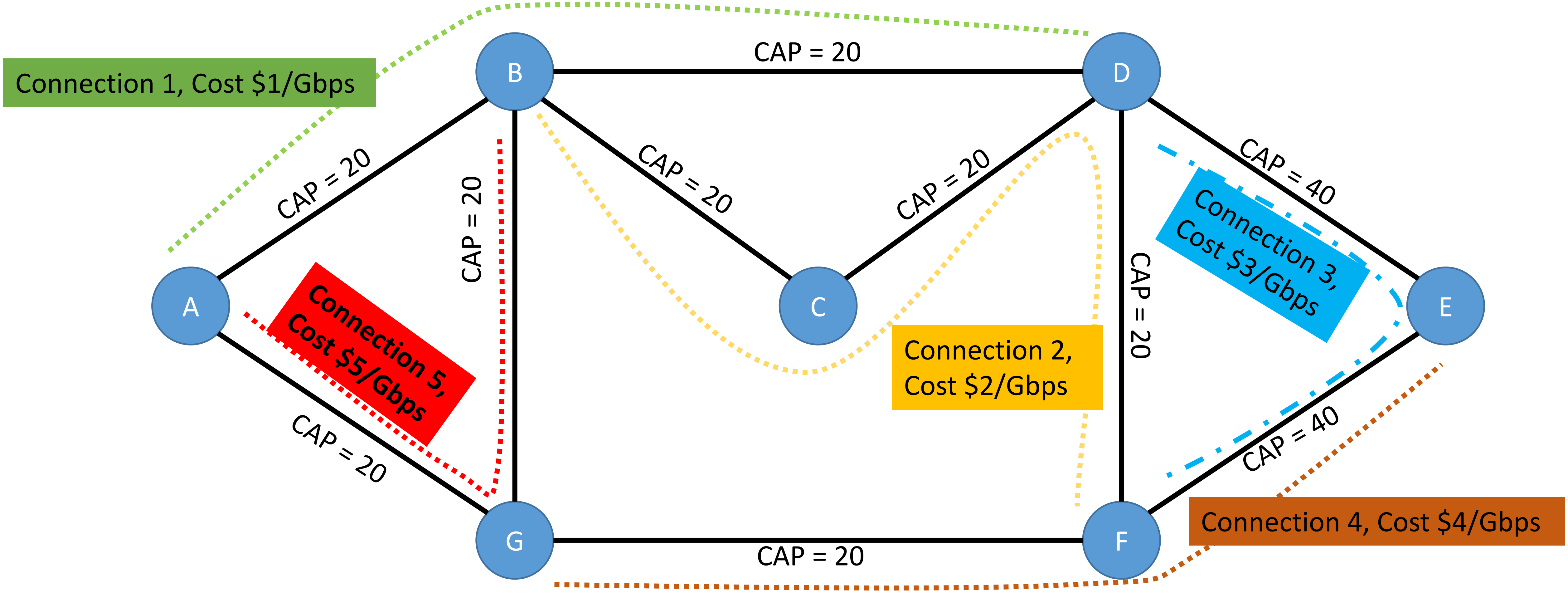
** Or, possibly, the cost per bandwidth unit divided by the number of links utilized by the connection, or amortized by the remaining duration of the connection…*



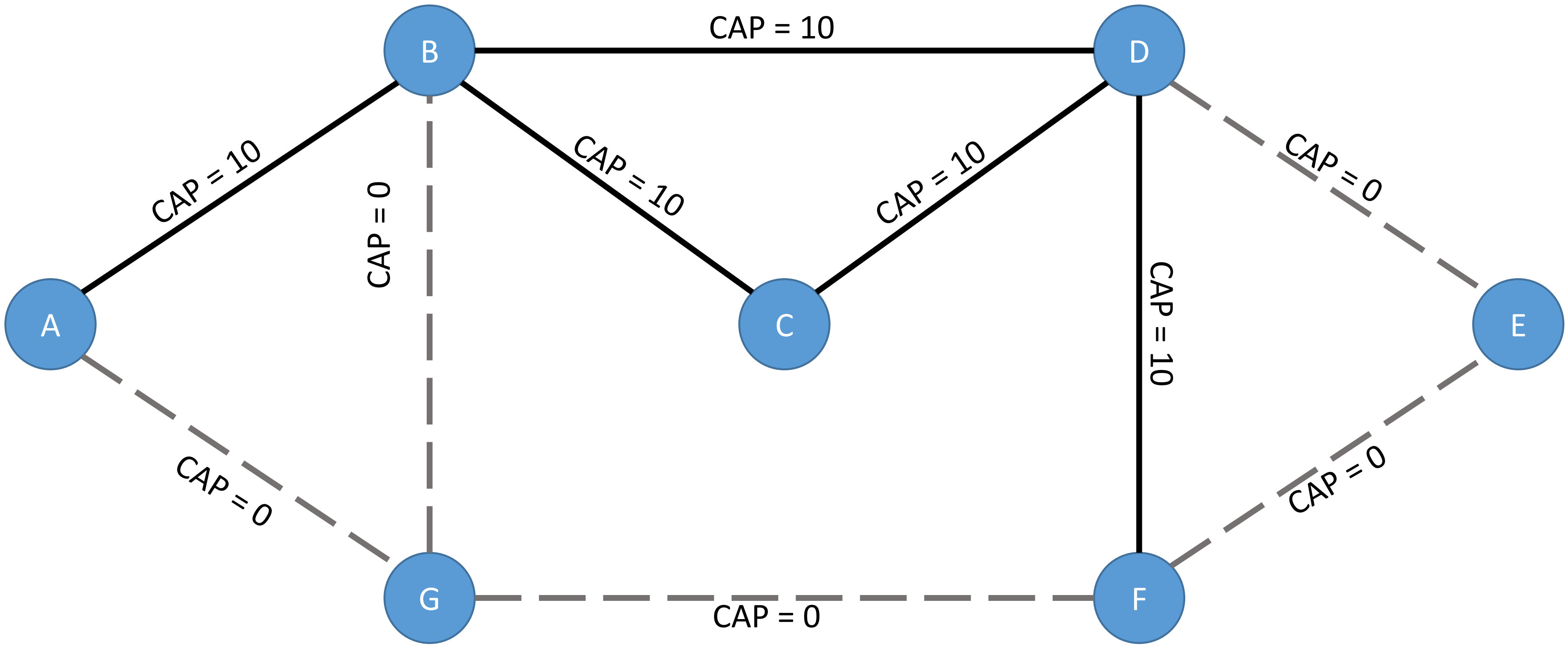
Another Idea: Connection Adjacency Graph (CAG)

Finally, calculate the shortest (cheapest) path from Dummy-Source to Dummy-Target. In this case, because the total cost of the shortest path is less then the price the new request is offering to pay, **degrade Connection1 and Connection2 and allocate the new request…**



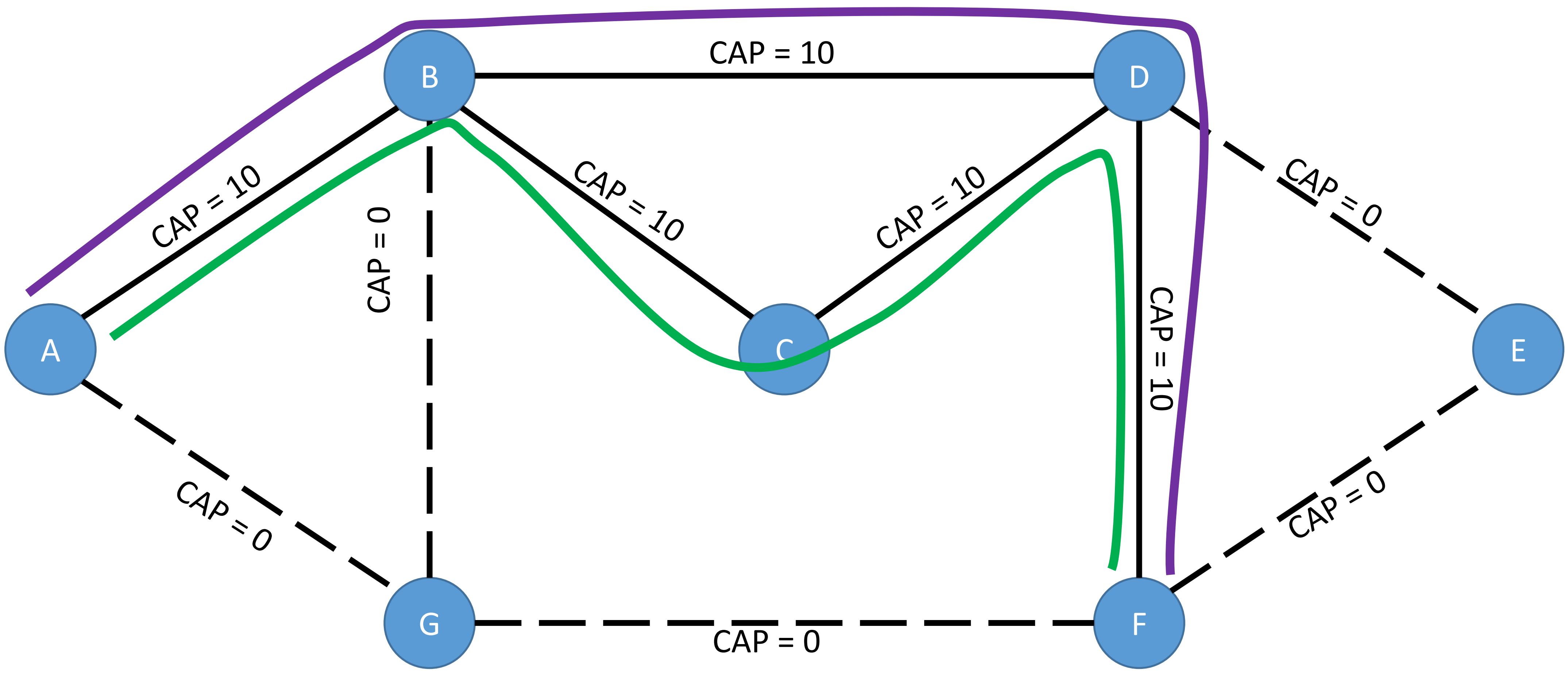


Degrade Connection1 and Connection 2 to their minimum (i.e., 10Gbps)….

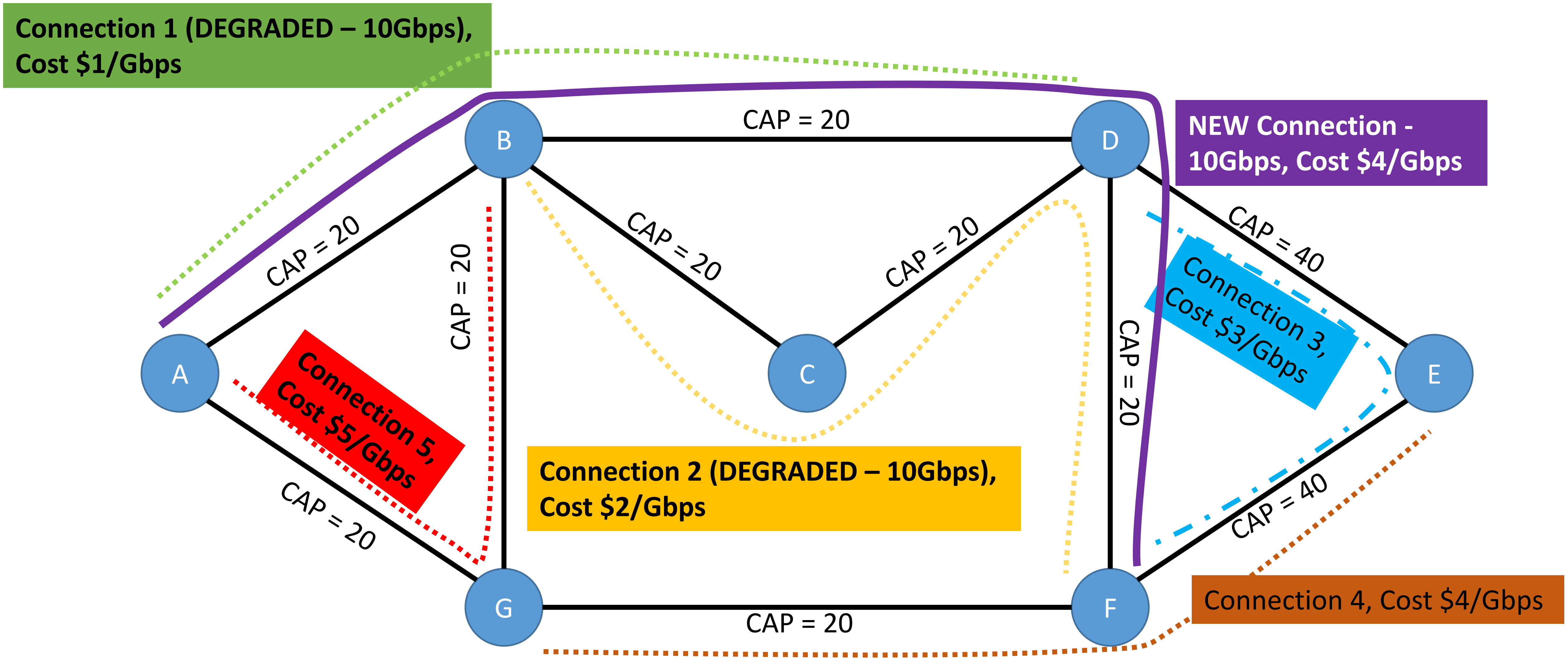


The residual graph after capacity is freed-up due to the degradation of Connections 1 and 2.

UCDAVIS



Now, two paths (of 10Gbps capacity) are available. Choose the shortest one.



Total revenue ➔ - (10 x 1) - (10 x 2) + (10 x 4)

➔ **Request not blocked and revenue increased in $10**

Proof of Correctness

- CAG contains all possible combinations of connection degradations that one can perform in **any** feasible path through the network
- CAG precisely describes the cost of degrading a connection (since edges weights are the costs of degrading their target super-vertex connection)
- There is a one-to-one mapping from any path in the physical network to a path in the CAG. There is a one-to-many relationship between one path in the CAG to many paths in the physical network

➢ **Thus, a shortest (cheapest) S-T path in the CAG necessarily maps to a cheapest set of connection degradations that create room for a path to be routed from source to destination**

However…

- Note that in the example, the purple request only asked for 10 Gbps
- Note how **every** initially allocated connection had a **degradable capacity of 10 Gbps**

Simplified Problem Statement:
If an incoming demand cannot be normally served due to Resource Crunch: which other connections should we degrade in order to serve this demand (or should we not serve it at all)?

P

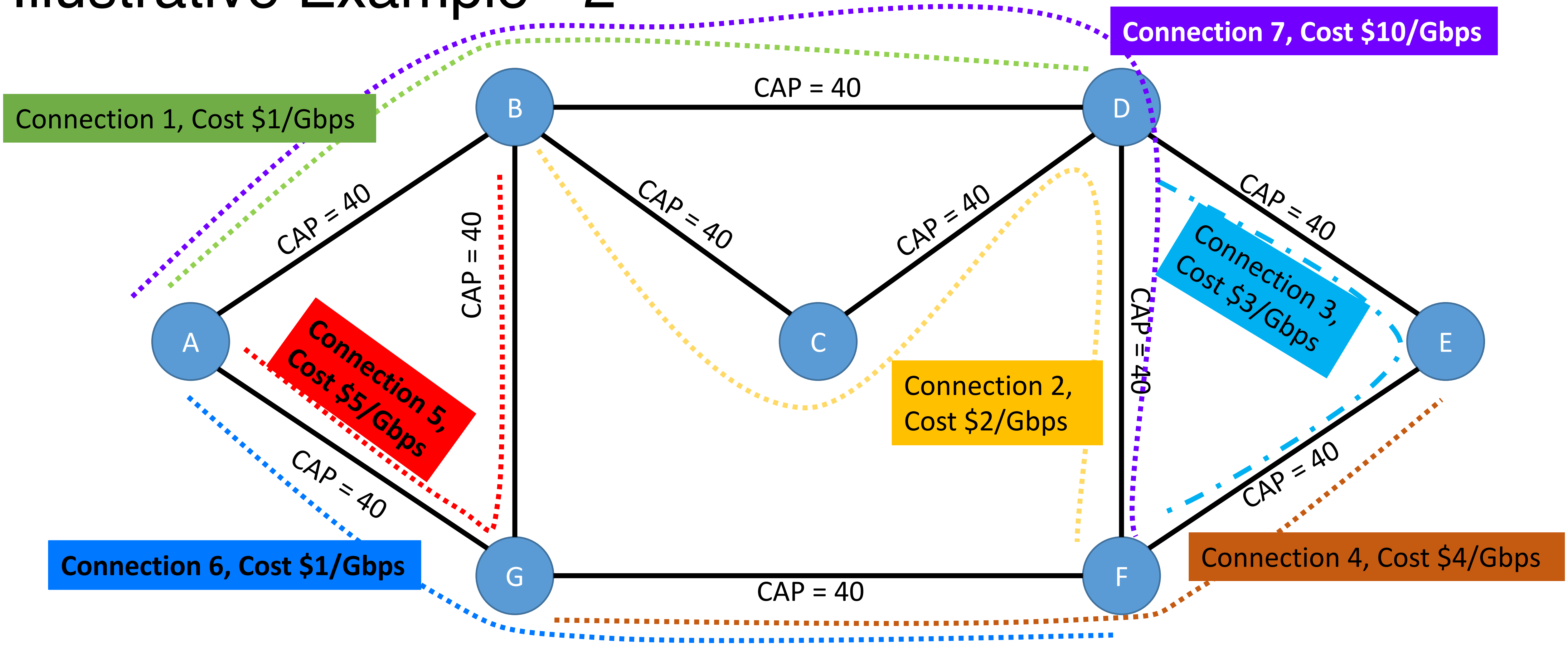
➔ (…) given that the incoming demand fits in each and every degradable capacity of each already allocated connection?

+

Not in P

➔ (…) given that the incoming demand does not necessarily fit in each and every degradable capacity of each already allocated connection?

Illustrative Example - 2



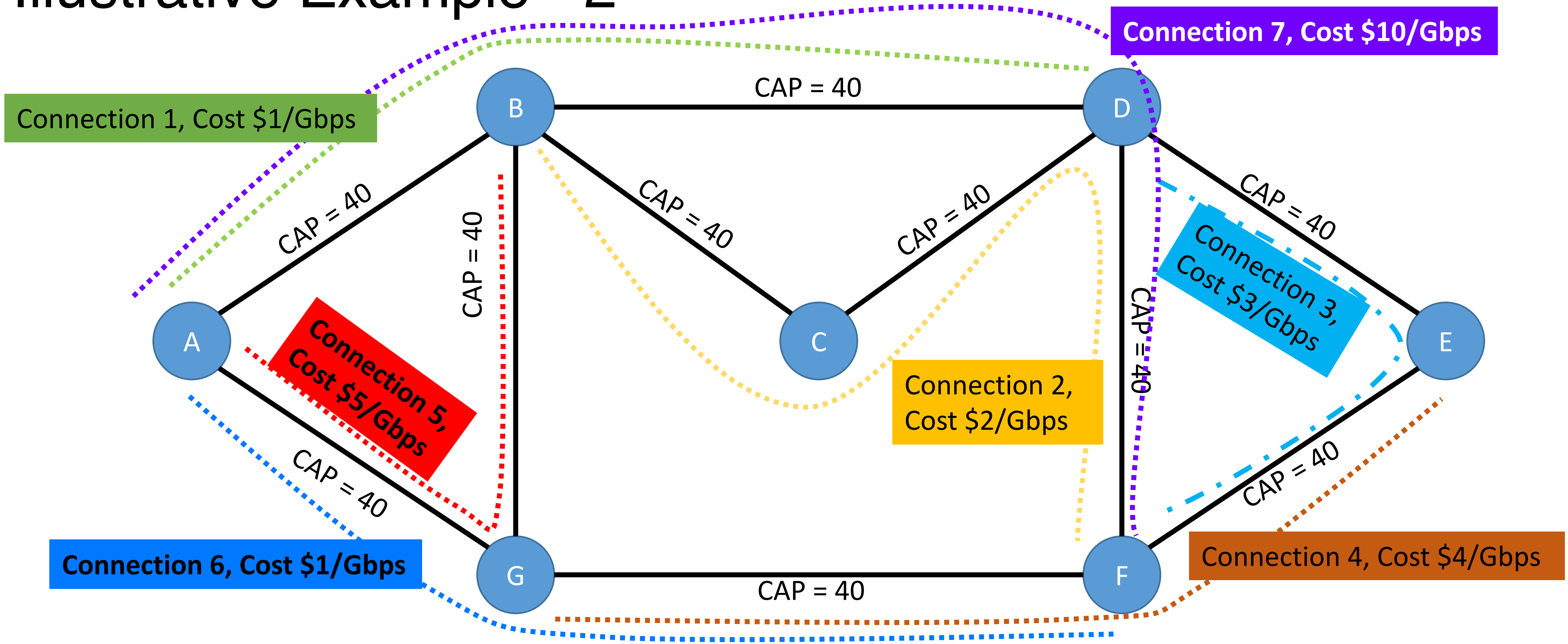
All Links now have 40Gbps capacities.

All connections have the same minimum and maximum required throughputs:

Min(C1)=Min(C2)=Min(C3)=Min(C4)=Min(C5)=Min(C6) = 10Gbps

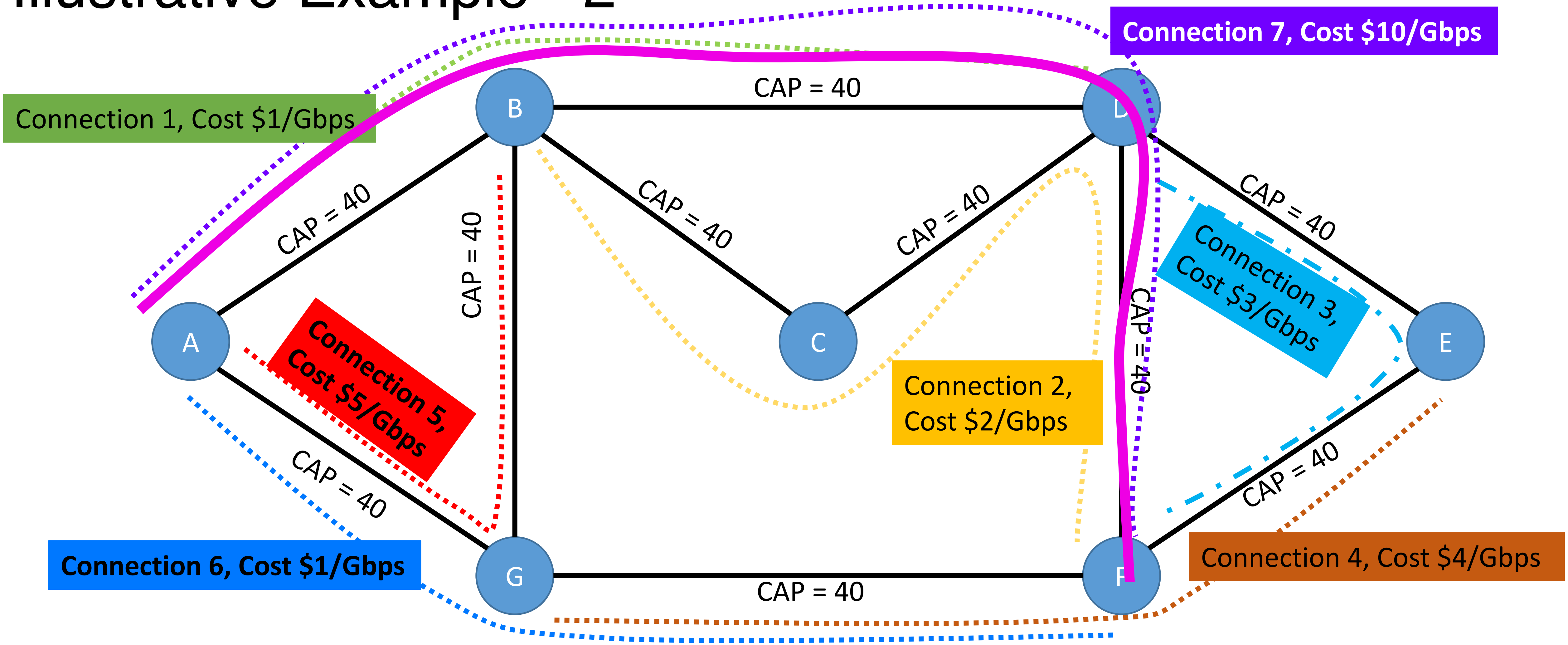
Max(C1)=Max(C2)=Max(C3)=Max(C4)=Max(C5)=Max(C6) = 20Gbps

Illustrative Example - 2



A new request of 20Gbps arrives from A to F and cannot be normally served due to Resource Crunch. This new request offers to pay $11/Gbps.

Illustrative Example - 2



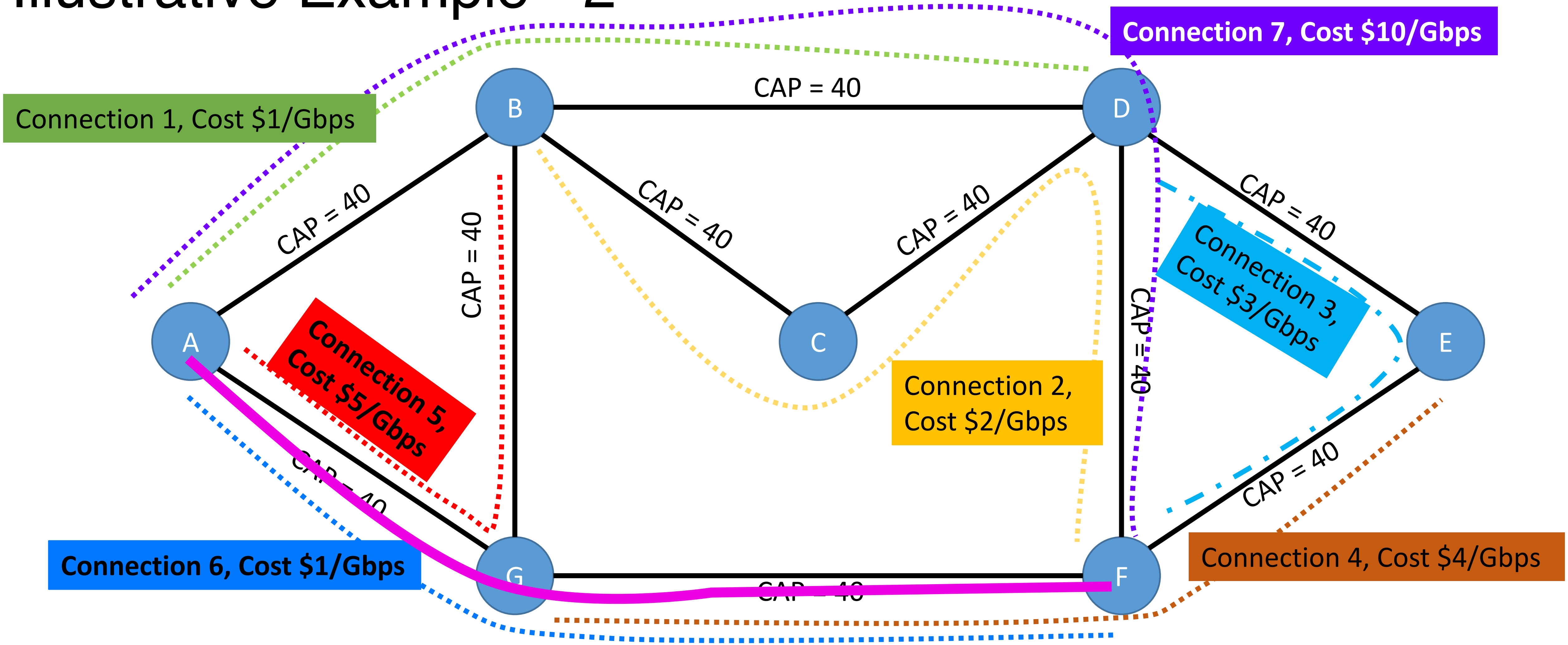
One option: Degrade Connections 1, 2 and 7.

The degradation would **decrease** the revenue in $ (1x10+2x10+10x10) = $130 and the new request would **increase** it in $110.

Total revenue decreased by $20.

UCDAVIS

Illustrative Example - 2



Another option: Degrade Connections 4, 5, and 6.

The degradation would **decrease** the revenue in $ (5x10+6x10+4x10) = $100 and the new request would **increase** it in $110.

Total revenue increased in $10.

UCDAVIS

Demands that don't fit in every single Degradable Capacity

- Splittable demand ➔ find CAG cheapest path, allocate (same procedure as before), repeat…
- Non-splittable demand ➔ since the CAG contains all possible degradations, the solution to this problem can be found within one of the possible combinations of CAG paths

  - Finding all simple paths in a graph: NP hard
  - All combinations: Exponential

Dealing with Intractability

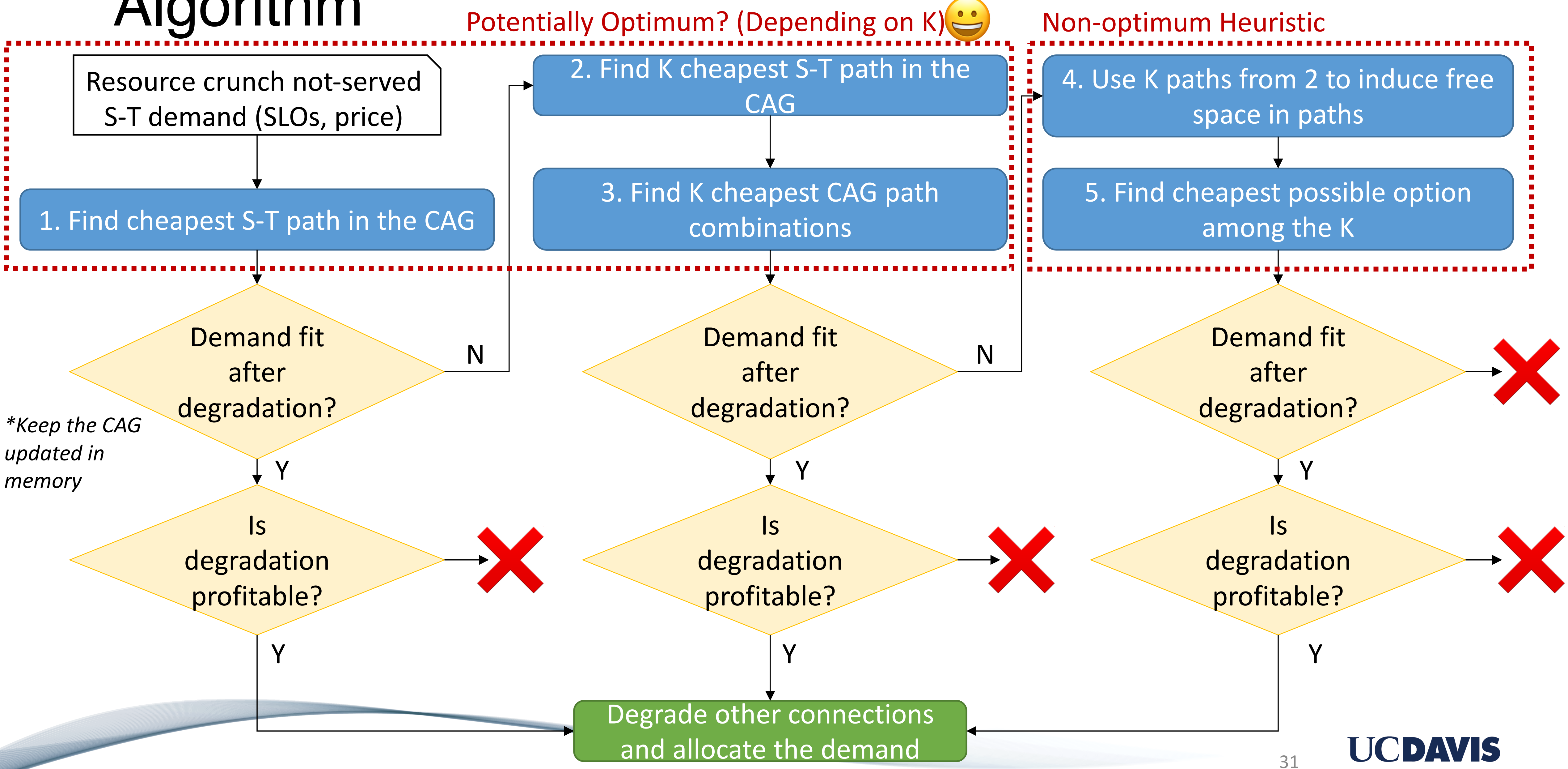
1. Instead of calculating all CAG paths ➔ only calculate the ”K-cheapest”

2. Instead of calculating all combinations ➔ (with the help of a bipartite Degradation Oriented Graph…) only analyze the ”K- shortest” combinations of paths that share physical links

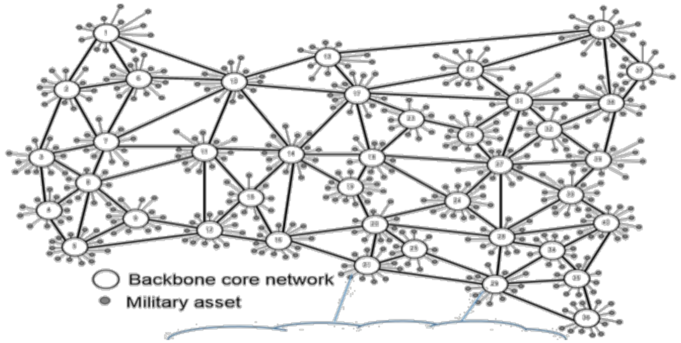
3. If no solution is found ➔ use the CAG paths of #1 to guide the search of a cheap (non-optimum) degradation

UCDAVIS

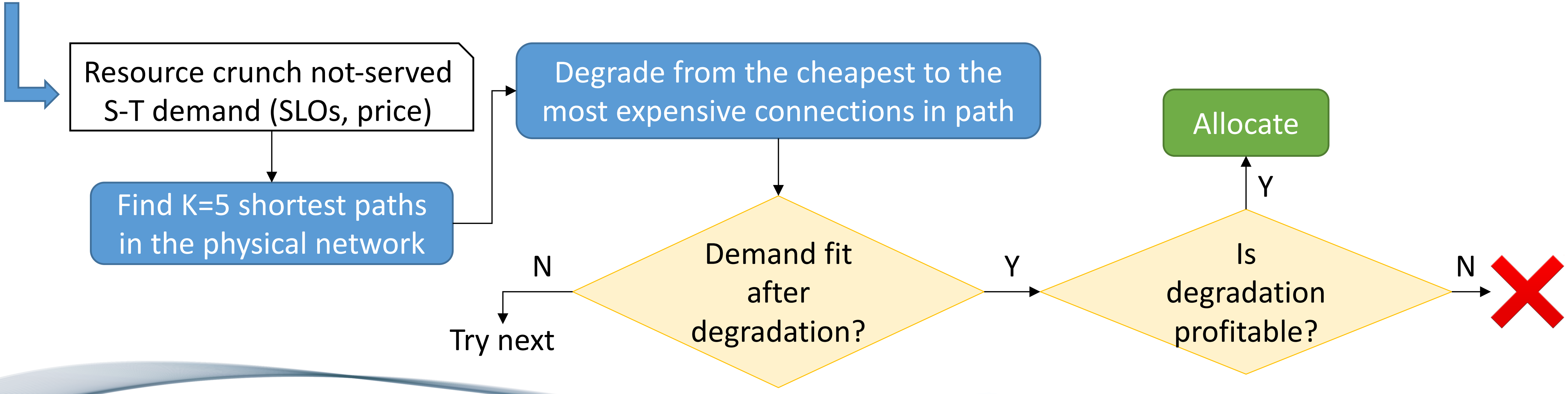
Algorithm


Potentially Optimum? (Depending on K)
Non-optimum Heuristic
Resource crunch not-served S-T demand (SLOs, price)
1. Find cheapest S-T path in the CAG
2. Find K cheapest S-T path in the CAG
3. Find K cheapest CAG path combinations
4. Use K paths from 2 to induce free space in paths
5. Find cheapest possible option among the K
Demand fit after degradation?
N
Y
Is degradation profitable?
Y
Demand fit after degradation?
N
Y
Is degradation profitable?
Y
Demand fit after degradation?
Y
Is degradation profitable?
Y
*Keep the CAG updated in memory
Degrade other connections and allocate the demand

Simulation

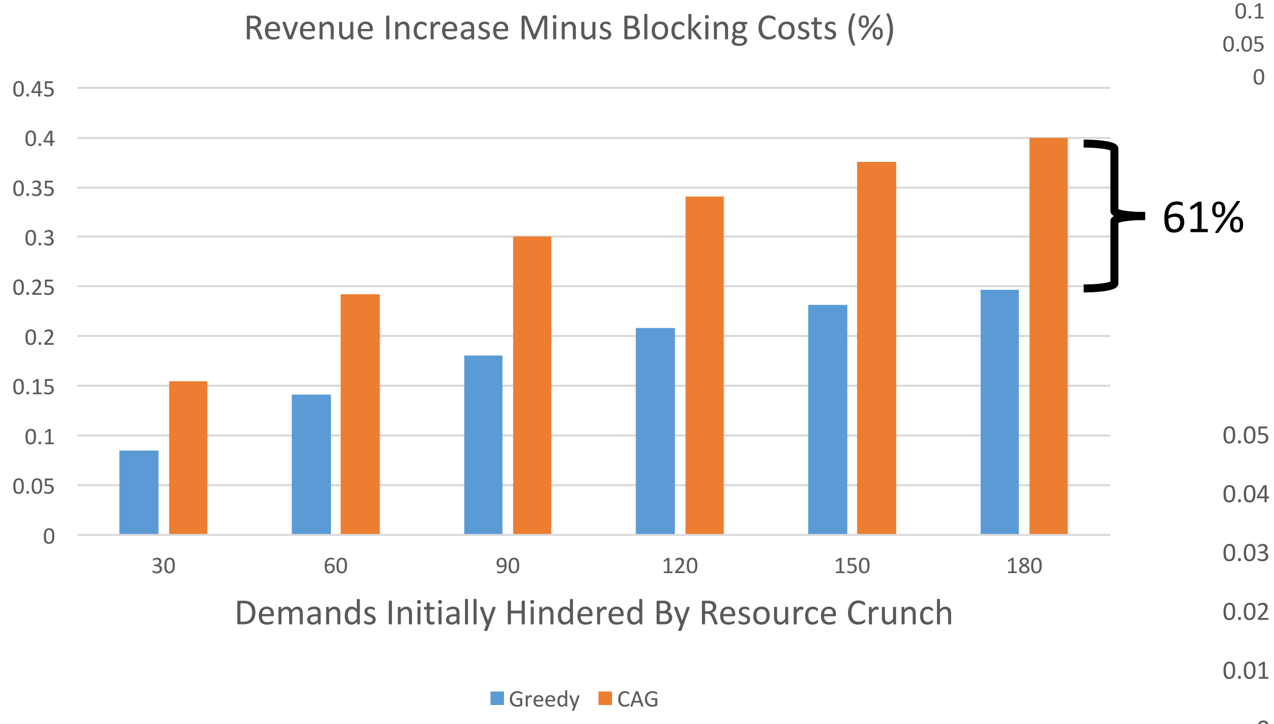


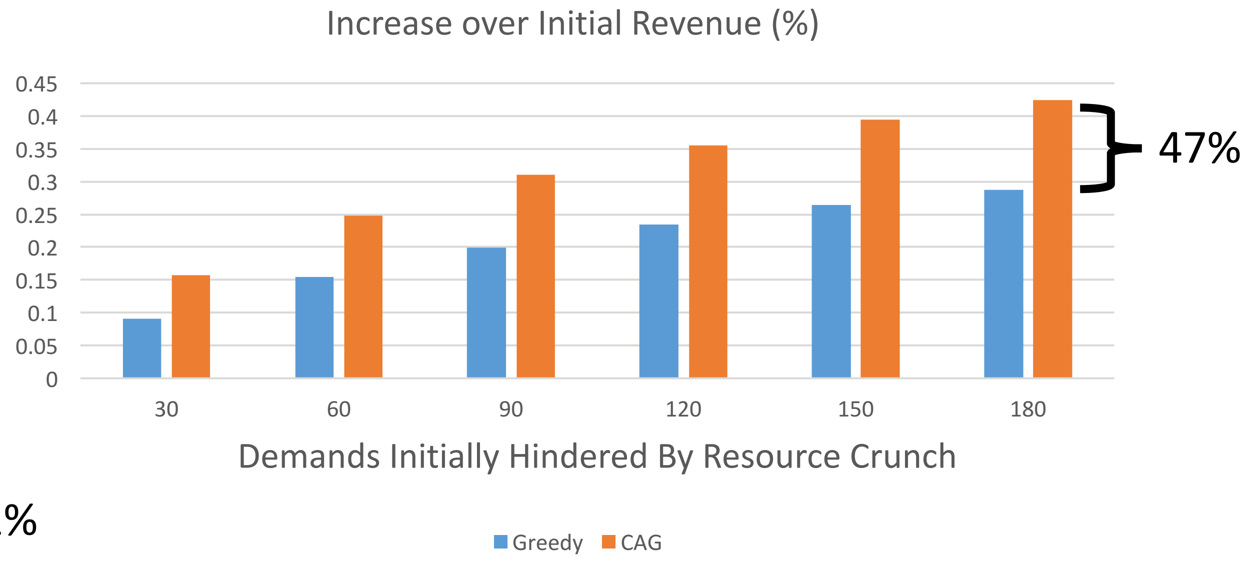
- Use traffic mixture from the previous table (slide 6)
- Statically occupy network up to average link utilization = 60%
- Generate X random demands that would otherwise be blocked
- Compare with a greedy approach [based on 1 and Journal submission]

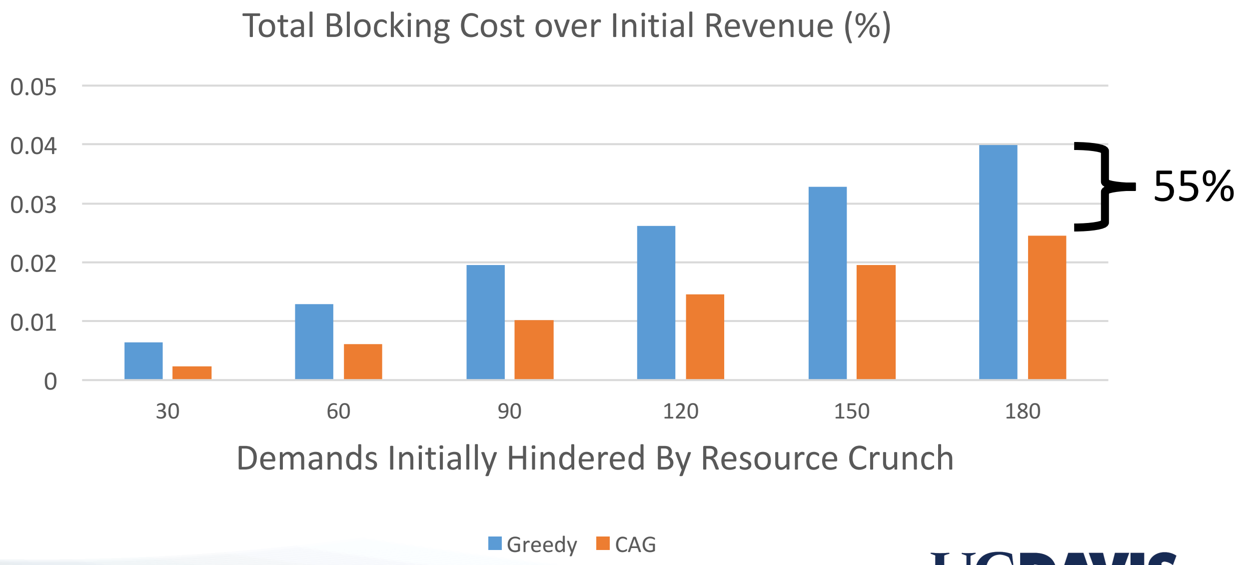


[1] Roy, Abhishek, M. Farhan Habib, and Biswanath Mukherjee. "Network adaptability under resource crunch." *Advanced Networks and Telecommuncations Systems (ANTS), 2014 IEEE International Conference on*. IEEE, 2014.

Results

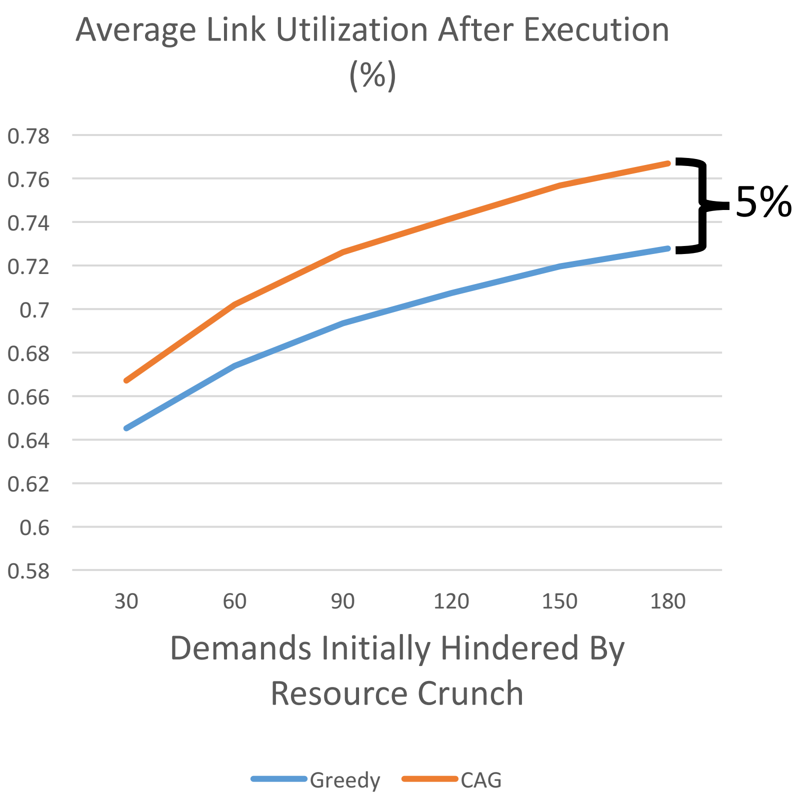


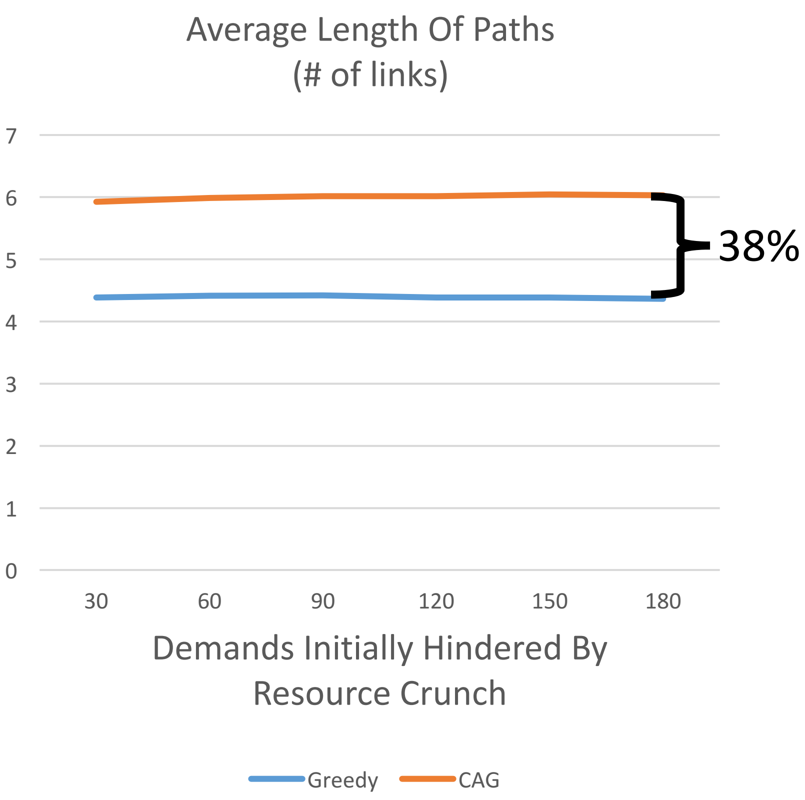


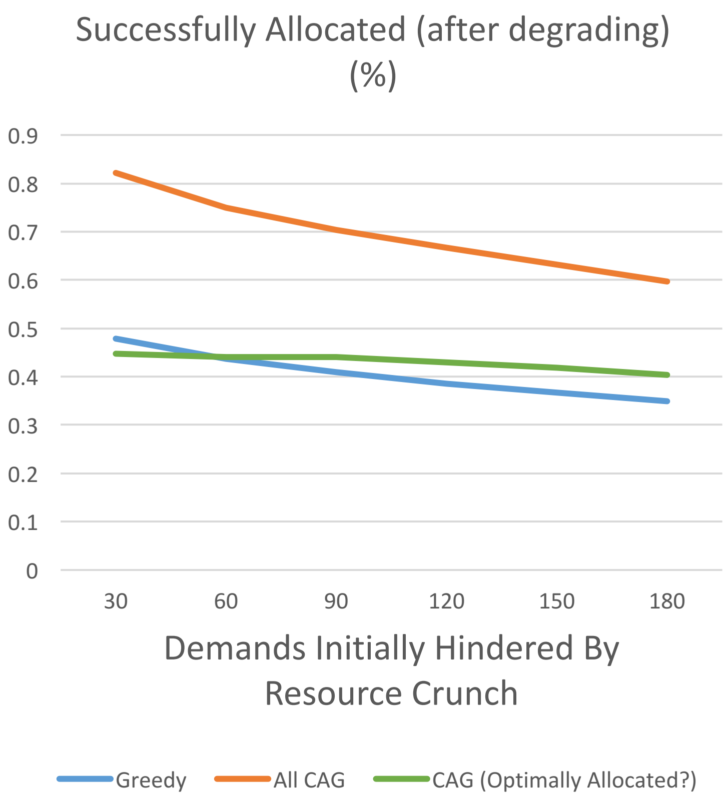


UCDAVIS

Results – Cont.







UCDAVIS