NANOG 74 Hackathon

September 30th, 2018
Vancouver, BC

Sponsored By: ORACLE Cloud Infrastructure

Lab Sponsor: TESUTO network emulation at scale

WIRELESS INFORMATION

- **NANOG-ARIN - This is 802.1X secured using the below user:pass (2.4Ghz and 5Ghz)**
- **NANOG-ARIN-legacy - This is an open, unencrypted network (2.4Ghz and 5Ghz)**

- **The login information for the 802.1x secured network is:**
- **User: nanog**
- **Pass: nanog**

GENERAL LOGISTICS

- Meals & Breaks – in this room
- Washrooms – Georgia Foyer near the escalators
- NANOG Registration 4-6pm (overlaps) – Regency Foyer

AGENDA

- 8:30AM - 9:30AM REGISTRATION & BREAKFAST
- 9:30AM - 10:30AM OPENING/INTRODUCTION, AND TUTORIAL
- 10:30AM BREAK INTO GROUPS
- 12:30PM -1:30PM LUNCH
- 1:30PM RESUME GROUPS
- 3:00PM BREAK – REFRESHMENTS
- 6:00PM HACK DEADLINE, PROTOTYPE DEMOS, VOTING
- 6:50PM CLOSING & RAFFLE GIVEAWAY
- 7:00PM - 8:00PM HACKATHON RECEPTION

THEME:
Segment Routing Automation

TUTORIAL

GROUPS

USING THE SHARED PAD, OR NETWORKING IN PERSON, PLEASE BREAK INTO GROUPS OF 5 OR LESS AND FIND AN IDEA TO HACK ON

COMMUNICATION

SLACK: https://nanog.slack.com/messages/G4Z6B343H/

SHARED PAD - GOOGLE DOC: https://bit.ly/2OlcQdj

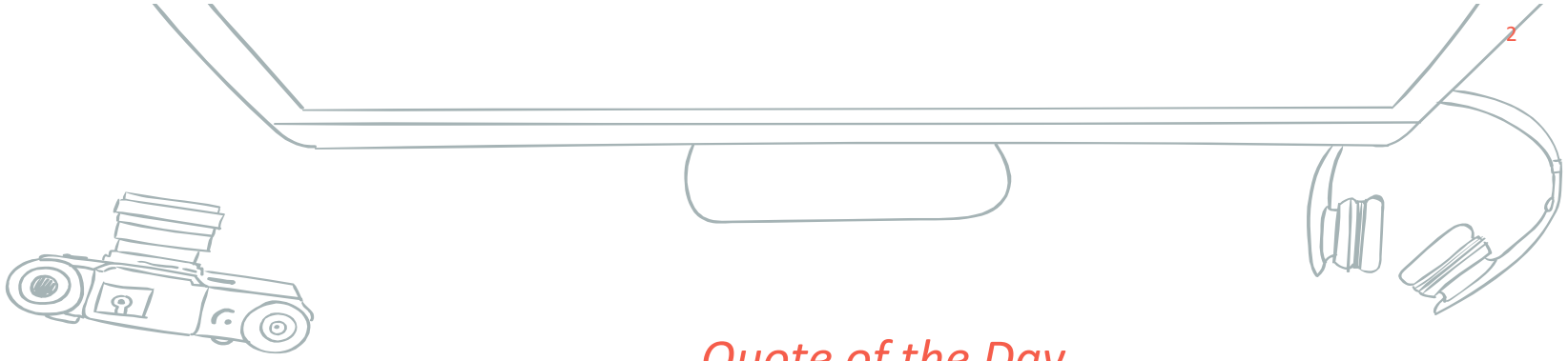
IN PERSON: OLD FASHIONED TALKING

PROTOTYPE FORUM - PRESENT YOUR HACK!

WHAT TO EXPECT:

- 2 - 3M PER GROUP
- MAKE SURE TO SAVE SCREENSHOTS – LABS NOT AVAILABLE TOMORROW
- TUESDAY 3:30PM – WINNER PRESENTATIONS

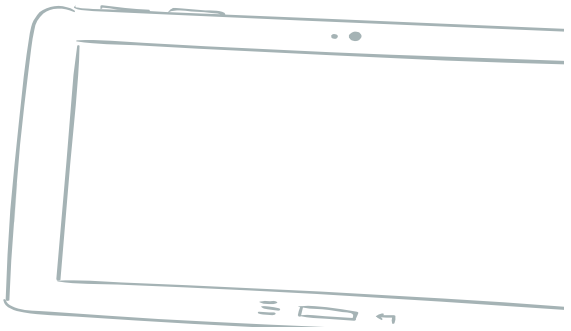
NANOG 74
Hackathon



Quote of the Day

There is nothing so annoying as to have two people talking when you're busy interrupting.

-- Mark Twain



Agenda

- Motivation
- Problem Statement
- 50K Foot view
- Ten minutes of Graph Theory
- Ten minutes of Network Flow Theory
- Two minutes of Segment Routing Overview
- One minute of BGP-LU overview
- BGP-LU as Southbound Protocol

Motivation

- The key idea is to see coding as another tool in your Engineering tool kit.
- It can help us to be a better Network Engineer by allowing us to acquire deeper insights within our domain.

Problem Statement

- As we know that Networks have to meet certain performance requirements and keep the resources well utilized at the same time. This translates into networks solving for various constraints like Cost, Delay, Price, Congestion etc.

- Traffic engineering can help networks to meet those constraints and carry traffic efficiently.

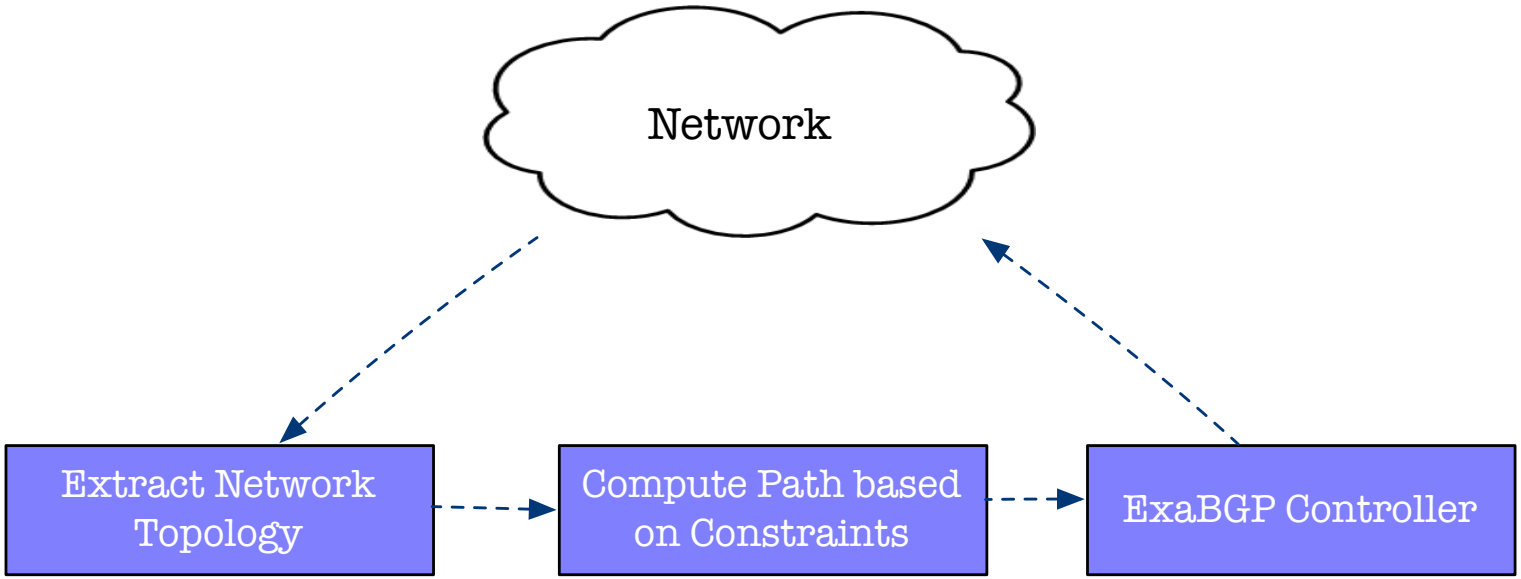
- Our goal is to solve some of the constraints for a given topology.

Problem Statement

- Some common Traffic Engineering(TE) goals:
  - Minimum Cost Routing
  - Flow Aware Min Cost Routing
  - Disjoint Path Routing

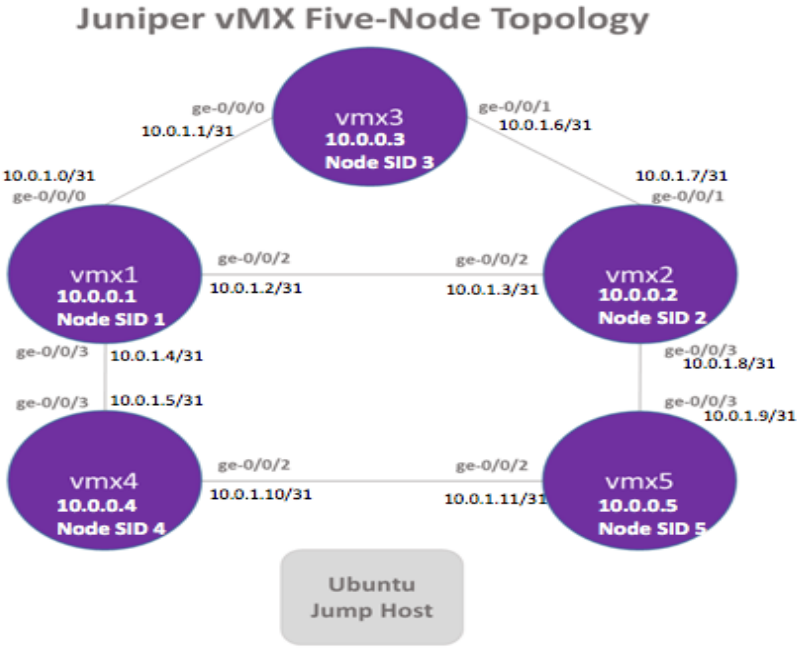
- Our goals are:
  - Create a model of the given network topology.
  - Solve some of the TE goals.
  - Program the network accordingly.

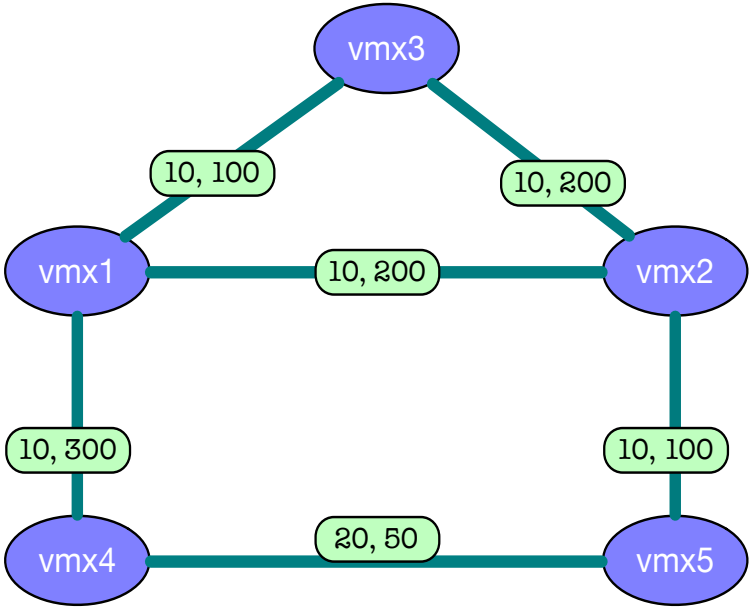
50K foot view of the Problem



Problem Statement

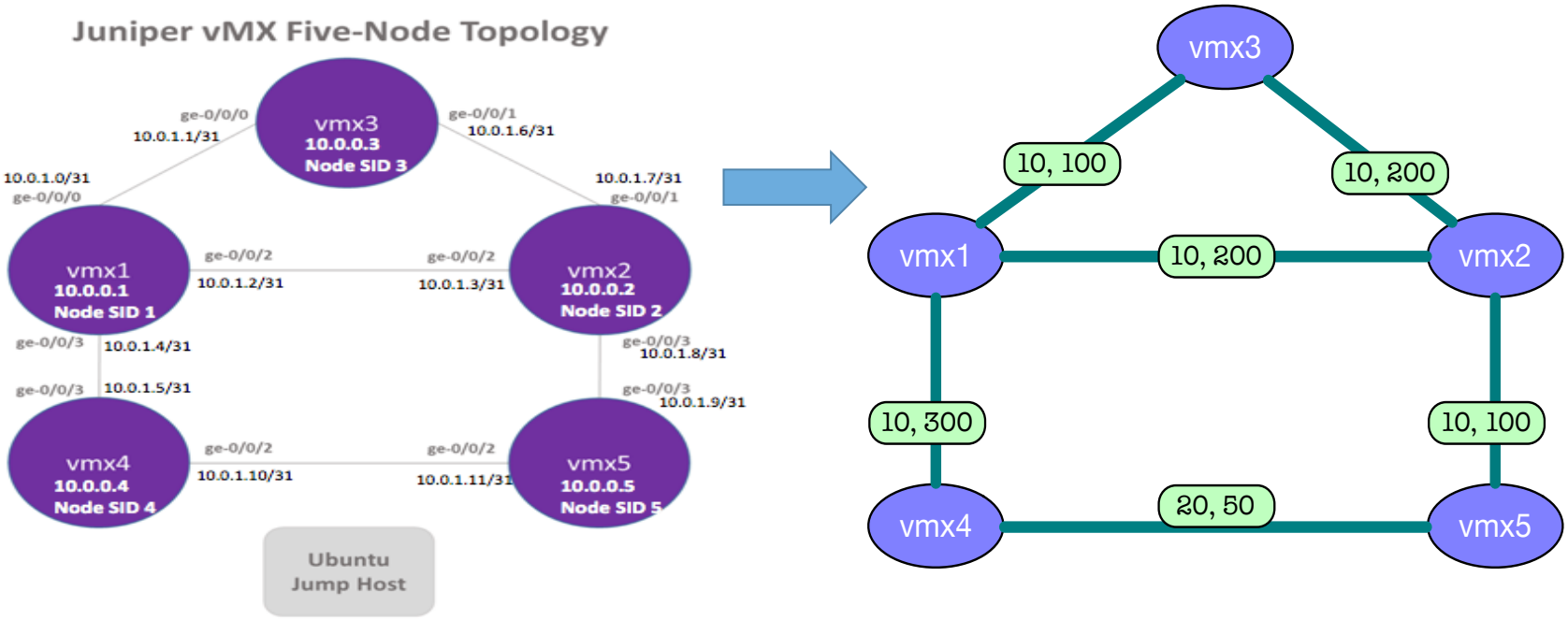
- Below is the Physical Topology
- Link Cost and Capacity is labelled for each link (Cost, Capacity).
- Capacity units are in mbps





Problem – Step 1(Extract Topology)

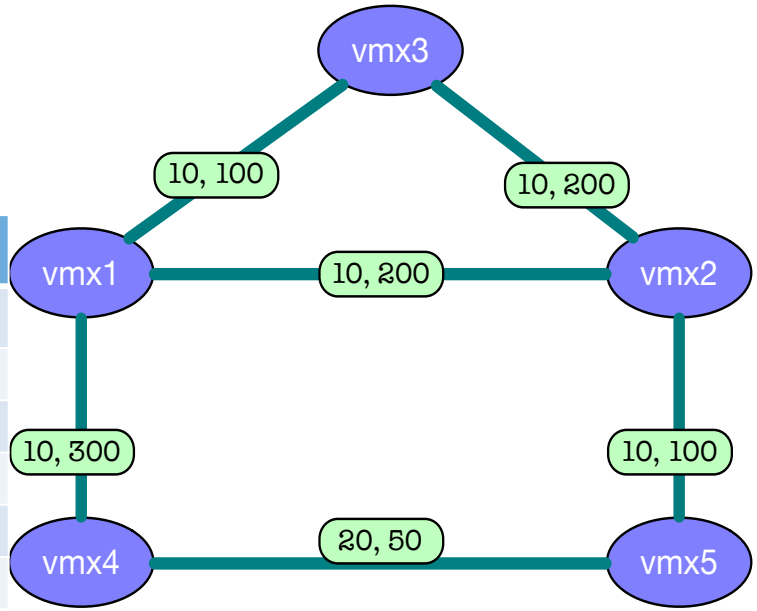
Extract Topology info and create a Network model



Problem – Step2(Compute Constrained Path)

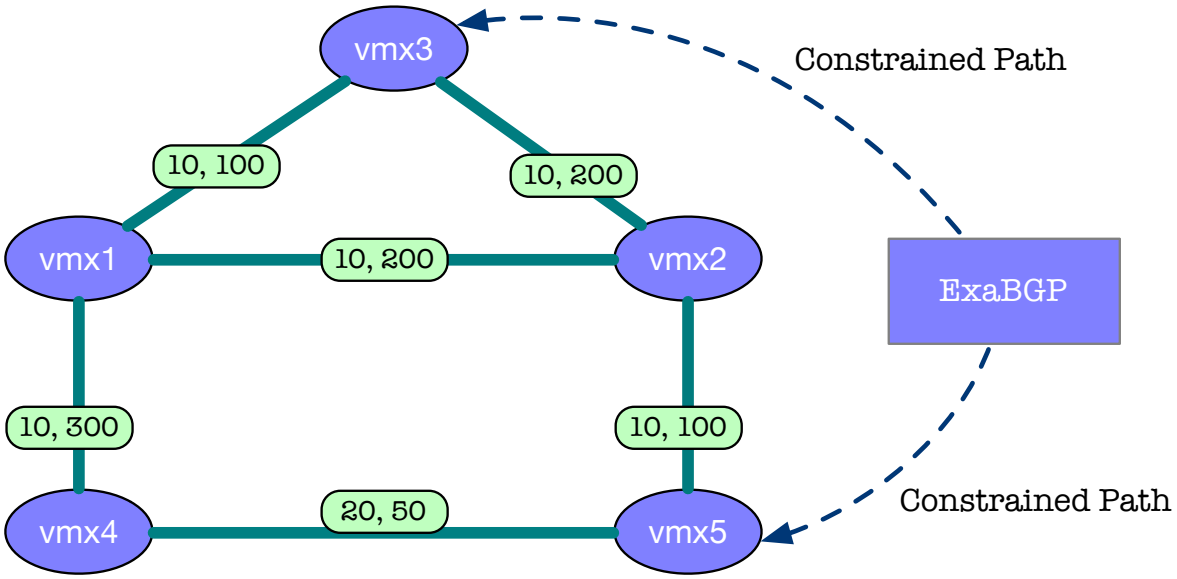
- Find Path's between the Source and destination for each problem given in the below table.

No.	Source	Destination	Path	Traffic Demand
1	vmx3	vmx4	Shortest	0
2	vmx3	vmx4	Shortest	200 Mbps
3	vmx3	vmx4	Shortest	200 Mbps
	vmx5	vmx4	Shortest	60 Mbps
4	vmx3	vmx4	Disjoint	0
	vmx3	vmx4	Disjoint	0



Problem – Step3(Program Paths on the Head ends)

Program the respective Head end's with the paths meeting the constraints for each problem.



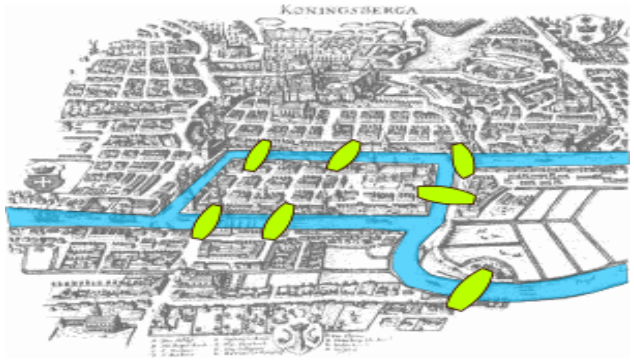
Ten Minutes of Graph Theory

- In order to create a Network Model we have to understand Graphs.
- Graph theory is a branch of Mathematics which can be used to model Communication Networks.
- Just like we can not approach physics without using calculus, we can not approach networks without notion of graphs.

Ten Minutes of Graph Theory

- Graph Theory began in 1736 with Euler who was asked to find a nice path across the seven Koenigsberg bridges.

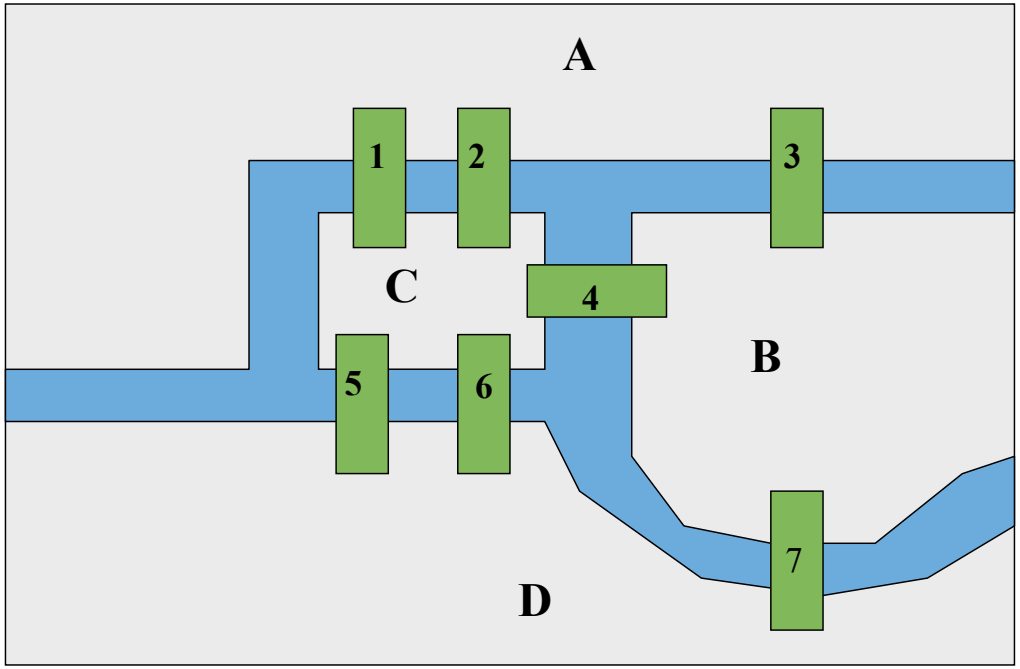
- Is it possible to take a walk, end up where you started from, with crossing each bridge exactly once.



Ref: https://commons.wikimedia.org/w/index.php?curid=112920

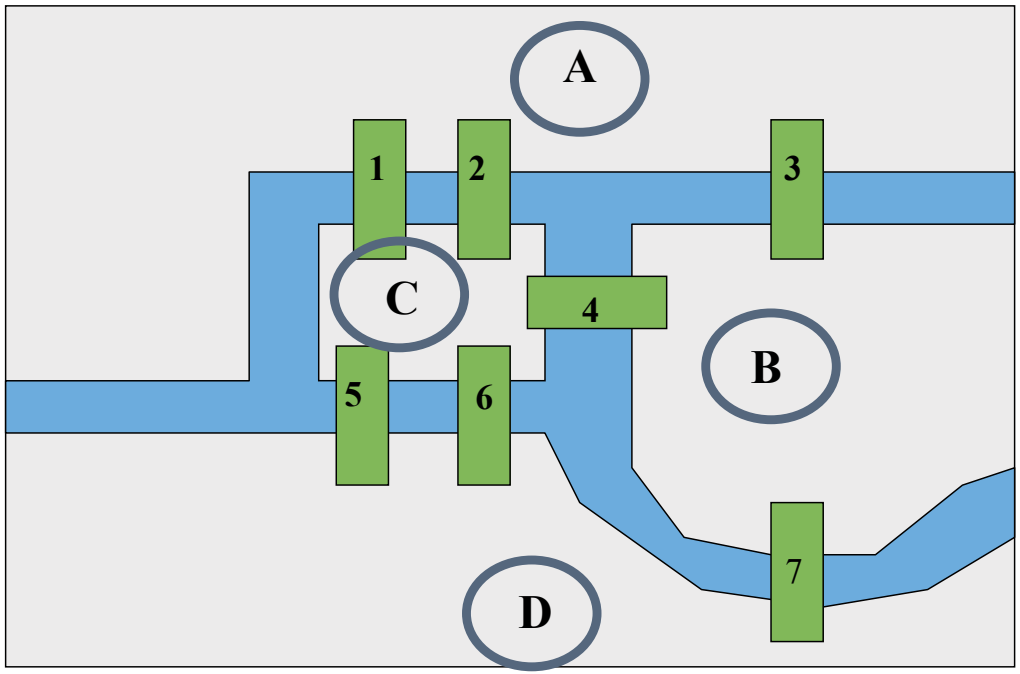
The Bridges of Koenigsberg

Is it possible to start in A, cross over each bridge exactly once, and end up back in A?



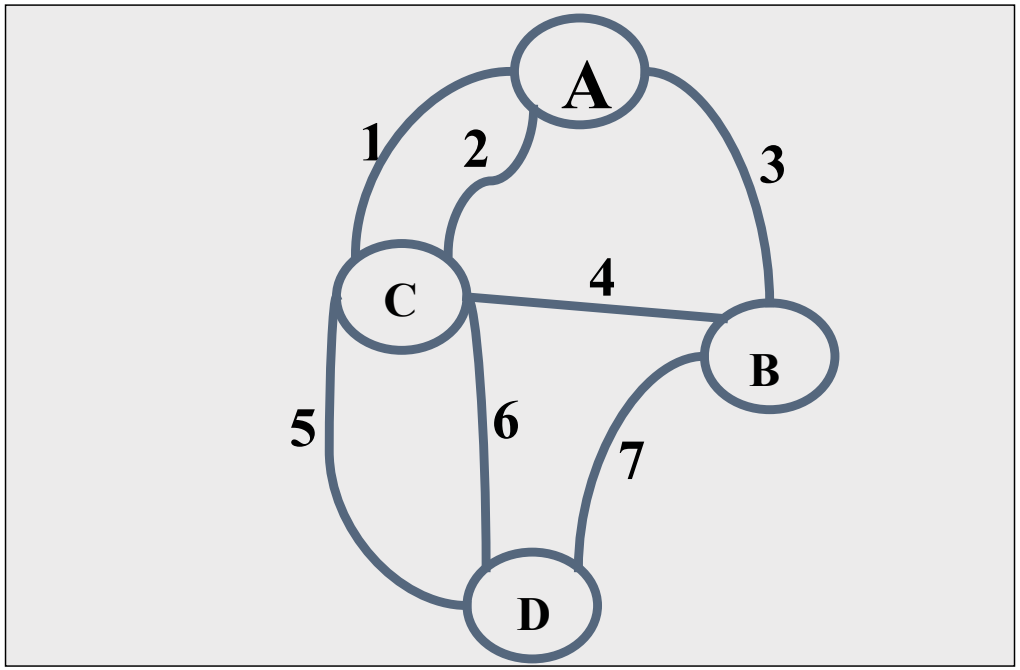
The Bridges of Koenigsberg

Think of Land masses as “nodes” or “vertices” and Bridges as “arcs” or “edges”



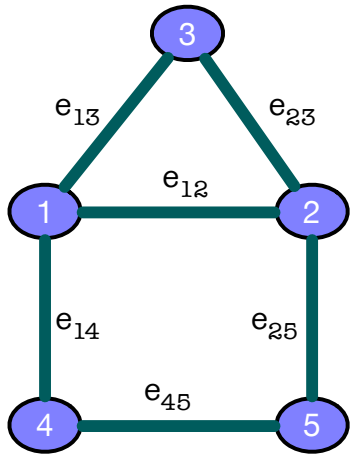
The Bridges of Koenigsberg

Is it possible to walk across all the seven bridges so that each bridge is crossed exactly once on the same walk?



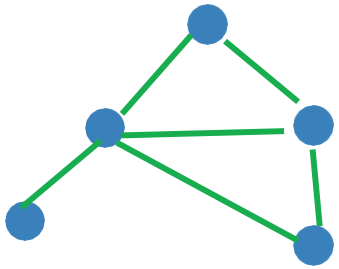
What are Graphs ?

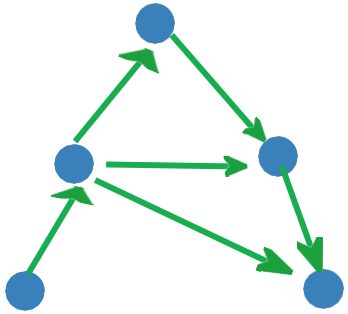
- Graph is a way of representing the relationship among a collection of objects.
- Graph consists of Nodes and Edges
- A graph G = (N, E) is a pair of Nodes (or vertices) N and a set of Edges (or Arcs) E , assumed finite i.e. |N | = n and |E | = m.
- Two nodes are neighbors if they are connected by an edge.
- Degree of a node is the number of edges ending at that node.



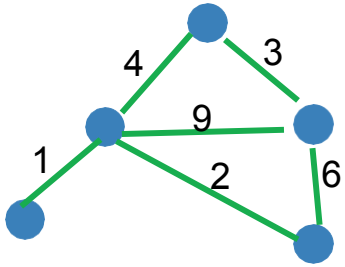
What are Graphs ?

- Graphs can be Undirected or Directed (Digraph)



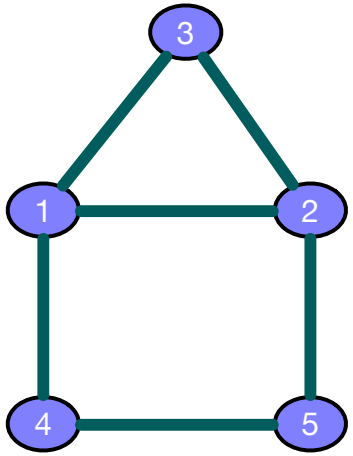


- Edges can have Weights like IGP Cost

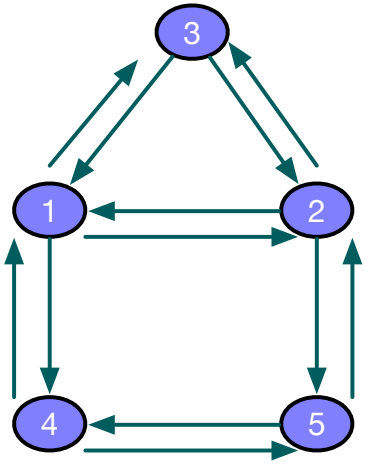


Undirected to Directed Graph

An Undirected Graph can be converted into a Directed graph by duplicating edges.



An Undirected Graph



A Directed Graph

For a directed graph, the in-degree and out-degree of a node refer to numbers of edges incoming to or outgoing from the node.

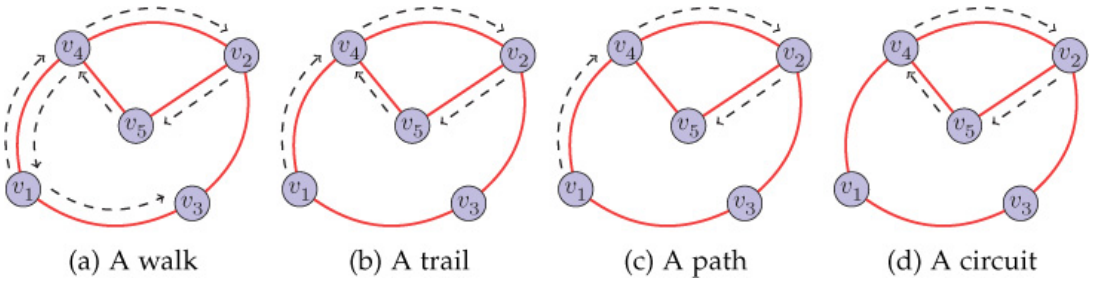
Some more Graph Definitions

a) **Walk**: A walk in graph G is an alternating sequence of Nodes and Edges that form a route across the graph. A Walk starts with a Node and also terminates at a Node. The number of edges incorporated by a walk is known as a length of walk.

b) **Trail**: A walk with no repeated Edges.

c) **Path**: A walk with no repeated Edges and Nodes.

d) **Circuit or Loop**: If a Path is closed i.e. terminates back to the Node.



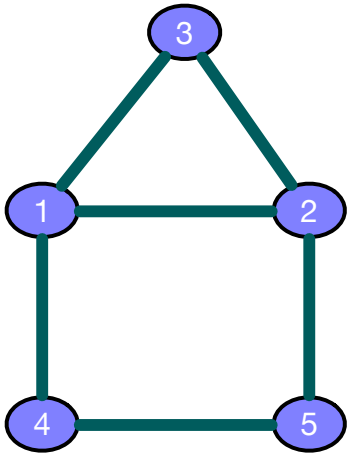
(a) A walk (b) A trail (c) A path (d) A circuit

Ref: Complex Networks: A Networking and Signal Processing Perspective

Graph Representation

Few ways to represent graph

- Adjacency Matrix (an N x N Matrix)
- Incidence Matrix (an N x M Matrix)
- Adjacency List



A=

	1	2	3	4	5	Degree
1	0	1	1	1	0	3
2	1	0	1	0	1	3
3	1	1	0	0	0	2
4	1	0	0	0	0	2
5	0	1	0	1	0	2

An Adjacency Matrix

A^2=

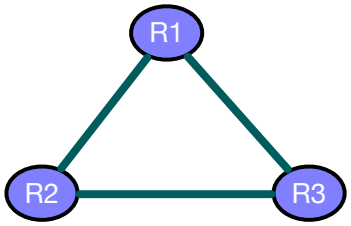
	1	2	3	4	5
1	3	1	1	0	2
2	1	2	0	1	1
3	1	0	2	1	0
4	0	2	1	2	0
5	2	0	0	2	1

2 Length Distinct Walks

Creating Graphs

Most languages have Graph libraries like:

- Python → NetworkX, iGraph
- GoLang → Goraph



Sample Graph

```
>>> import networkx
```
Import library

```
>>> g = networkx.Graph()
```
Create new undirected graph

```
>>> g.add_node("R1")
>>> g.add_node("R2")
>>> g.add_node("R3")
```
Add new nodes with unique IDs.

```
>>> g.add_edge("R1", "R3")
>>> g.add_edge("R1", "R2")
>>> g.add_edge("R2", "R3")
```
Add new edges referencing associated node IDs.

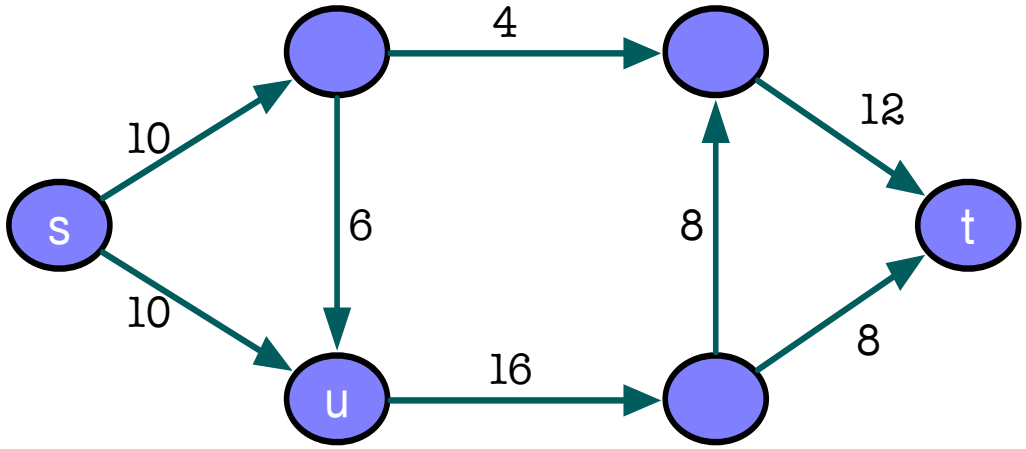
```
>>> print g.number_of_nodes()
3
>>> print g.number_of_edges()
3
>>> print g.nodes()
['R1','R2','R3']
```
Print details of our newly- created graph.

Extracting Topology Info: Parsing a TED

```
tesuto@mx3> show ted database extensive
TED database: 5 ISIS nodes 5 INET nodes
NodeID: mx1.00(10.0.0.1)                                    << Node ID
  Protocol: IS-IS(2)
   10.0.0.1                                                 << Router Loopback
    To: mx3.00(10.0.0.3), Local: 10.0.1.0, Remote: 10.0.1.1
      IGP metric: 10                                        << IGP Cost
      Reservable BW: 100Mbps                                << Bandwidth available for Reservation
    To: mx4.00(10.0.0.4), Local: 10.0.1.4, Remote: 10.0.1.5
      IGP metric: 10
      Reservable BW: 300Mbps
    To: mx2.00(10.0.0.2), Local: 10.0.1.2, Remote: 10.0.1.3 <<Local and Remote IP on the link to MX2
      IGP metric: 10
      Reservable BW: 200Mbps
   Prefixes:
     10.0.0.1/32                                            << Router Loopback. SR Details coming later.
      Metric: 0, Flags: 0x00
      Prefix-SID:
        SID: 1, Flags: 0x40, Algo: 0                        << SID Index
   SPRING-Capabilities:
     SRGB block [Start: 800000, Range: 4096, Flags: 0xc0] << SRGB Details
```

Ten Minutes of Network Flow Theory

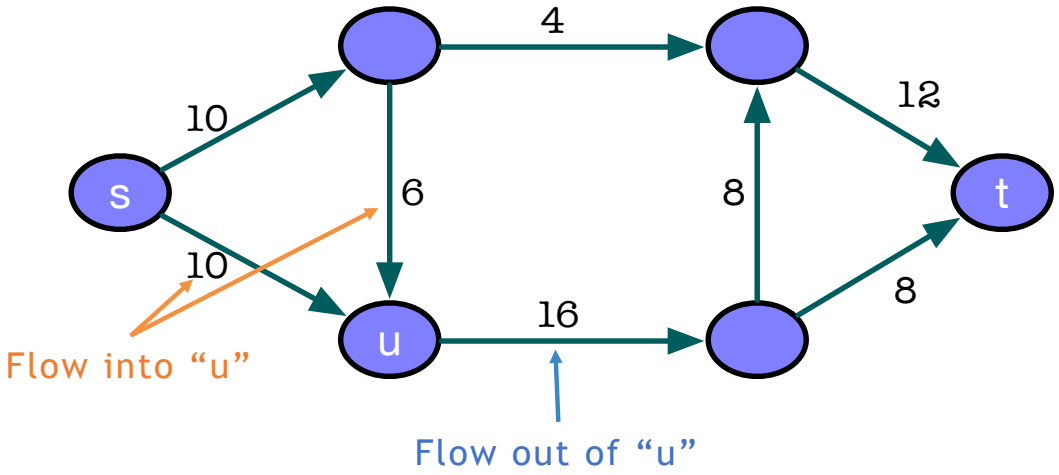
- A flow network consists of a Directed Graph with a source “s” and a sink “t”. Each edge has a nonnegative capacity “c”.
- Flow needs to meets the capacity constraint i.e. <= “c”



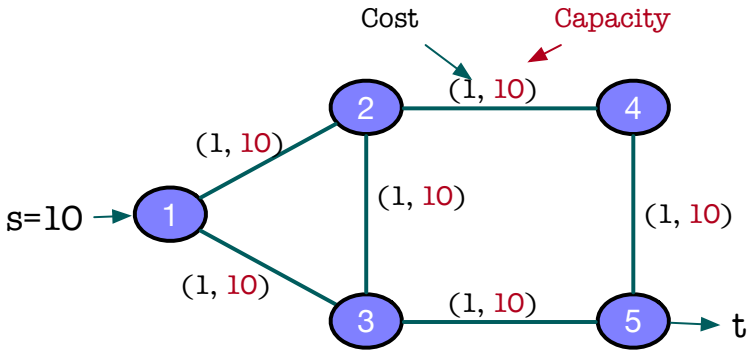
A Directed Graph with flows

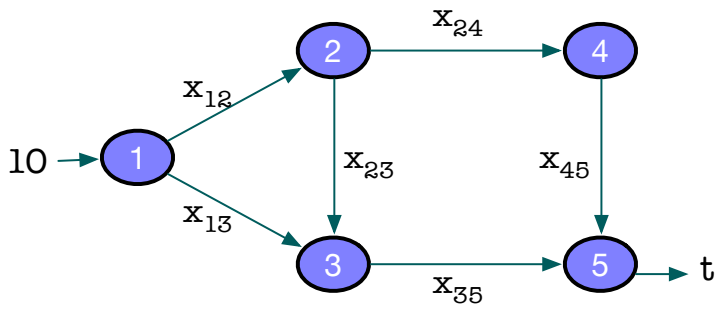
Law of Flow Conservation

- Flow conservation law is also known as Kirchoff's current law.
- Flow into "u" = 10+6 = 16 = Flow out of "u"



Shortest Path Problem as Network Flow Problem





$$\min\ x_{12}+ x_{13}+ x_{24}+ x_{23}+ x_{35}+ x_{45}$$

$$x_{12}+ x_{13} = 10$$
$$x_{12} - x_{24} - x_{23} = 0$$
$$x_{24} - x_{45} = 0$$
$$x_{13} + x_{23} - x_{35} = 0$$

$$0 \le x_{ij} \le 10 \text{ for all } x_{ij}$$

Solving that

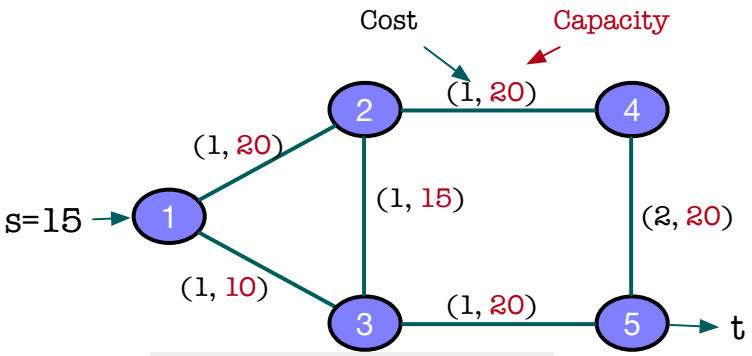
$$x_{12} = x_{23} = x_{24} = x_{45} = 0$$
$$x_{13} = x_{35} = 10$$

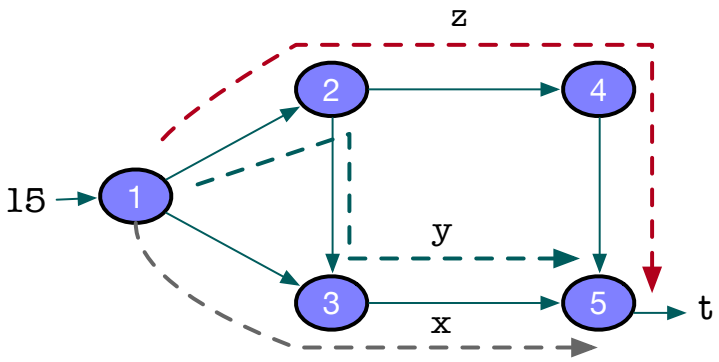
Dijkstra's Shortest Path algorithm

Simplicity is a great virtue but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better. - *Edsger Dijkstra*

- Dijkstra shortest path is used for finding shortest path for weighted graphs.
- It's like BFS for weighted graphs.
- If all costs are equal, Dijkstra = BFS
  - Explores nodes in increasing order of cost from source.
- Because of it's greedy  nature it does not work with Graphs with Negative Edges.
- **Bellman Ford** is another Shortest Path algorithm which works with Negative edges.

Shortest Path with Flow Constraint – Non Splittable Flows





$$\min \quad 2x + 3y + 4z$$

$$x + y + z = 1$$
$$15z \le 20$$
$$15y \le 15$$
$$15x \le 10$$

Solving that

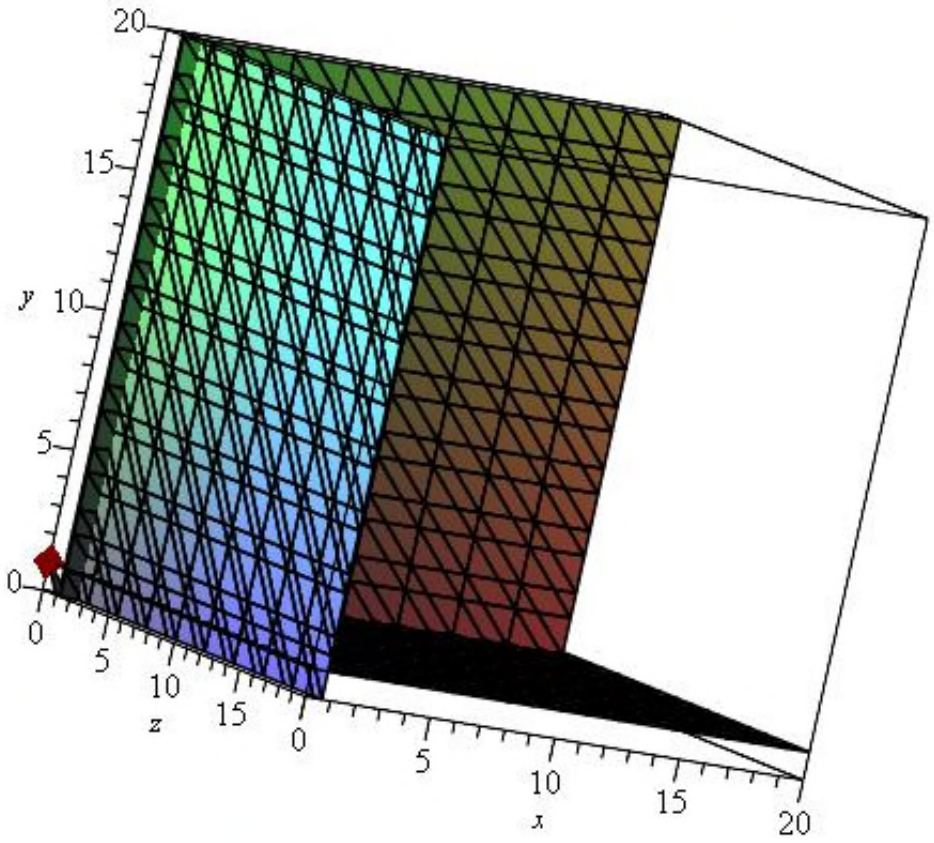
LPSolve(obj, cnsts, assume = binary)

Min Cost = 3

X = 0, y = 1, z = 0

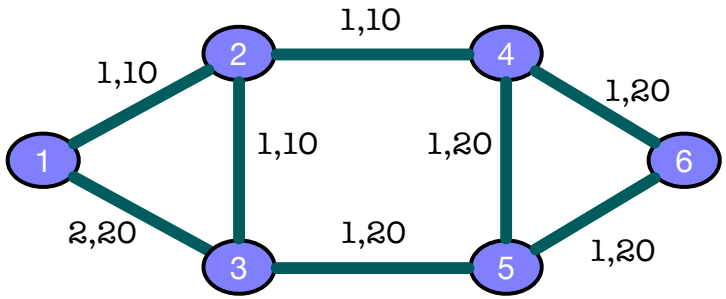
[3, [x = 0, y = 1, z = 0]]

Shortest Path with Flow Constraint – Visualization

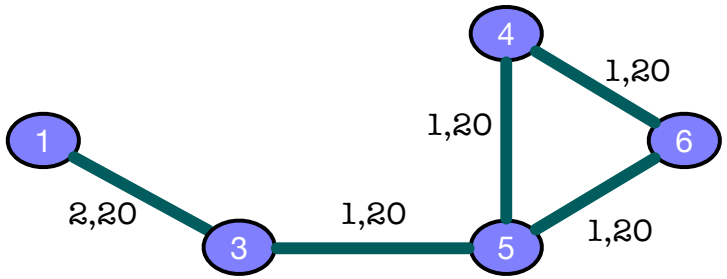


Handling Capacity Constraint with Dijkstra

- If we want to only find the shortest path along the edges which satisfies a certain capacity constraint then first prune the edges which does not satisfy the capacity constraint and then run Dijkstra SPF.
- Example: Find Shortest path between Node 1 and 6 with capacity of 20.



Graph



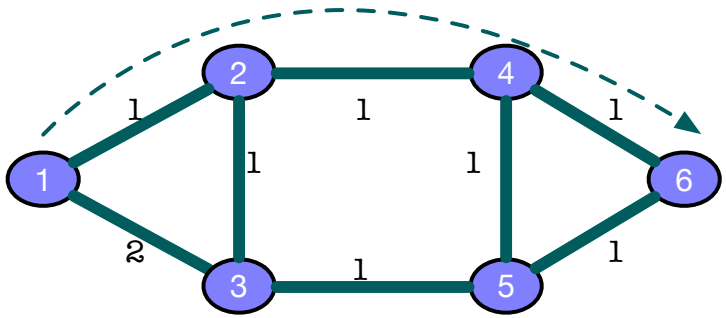
Residual Graph

Disjoint Path Problems

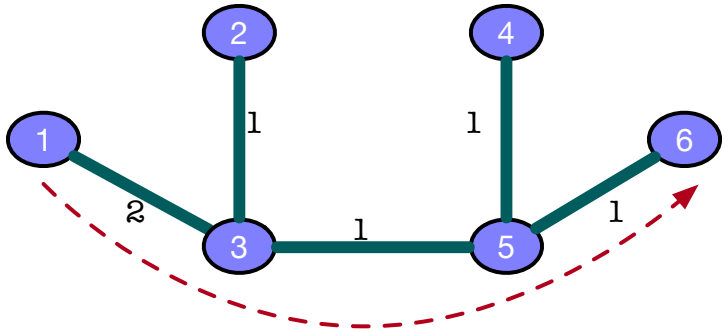
- Disjoint Path Problem could be seen as an extension of Shortest Path Problem.
- Rather than just computing the Shortest Path, Compute Several Paths that do not share any common links (or Nodes).
- Node Disjoint and Link Disjoint Path Problems. Node Disjoint is more restrictive as that means paths are also Link Disjoint.
- Maximally Disjoint Paths is where the disjoint paths can partially overlap rather than being fully Disjoint.

Handling Disjoint Path constraints with Dijkstra

- We can find the shortest path first and then remove those intermediate Edges/Nodes from the Graph and run SPF again
  - Edges for Edge Disjoint Path
  - Nodes and Edges for Node Disjoint Path
- Example: Find an Edge Disjoint path between Node 1 and 6



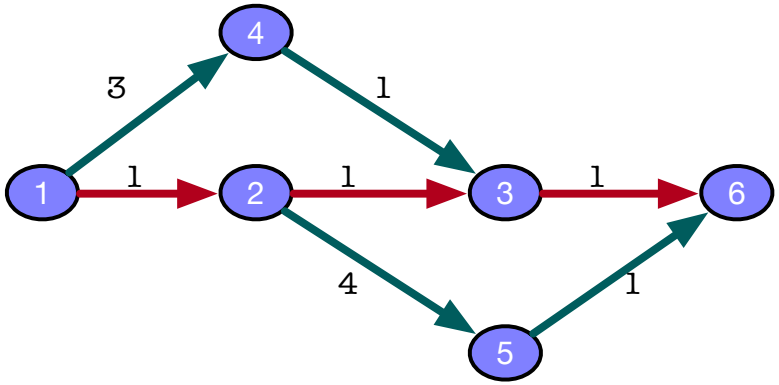
First Path



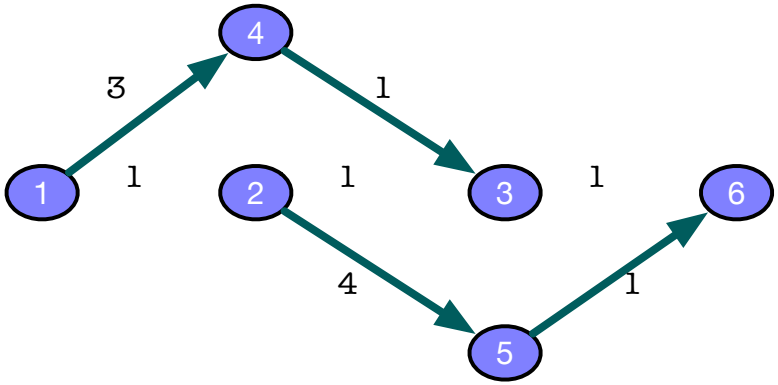
Second Edge Disjoint Path

Disjoint Paths between 1 and 6 with Dijkstra?

- It's not always possible to apply Dijkstra for Disjoint Path routing.
- In the below graph(Pathological case), Shortest Path between 1 and 6 is colored in RED.
- Removing the edges along the Shortest Path leaves the graph disconnected.
- Can we do it better ? Exercise for user to explore other algorithms.



Trap Topology



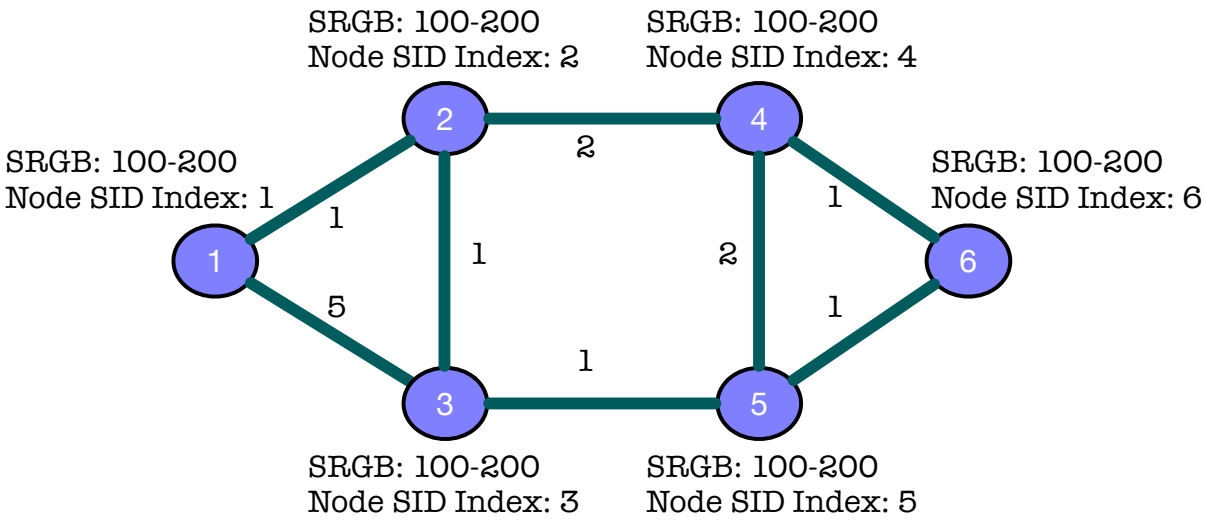
Residual Graph

Two minutes of Segment Routing - Terminology

- Segment: An instruction a node executes on an incoming packet.
- Global Segment: Instruction is supported by all SR- Capable Routers. Example: Node SID, Anycast SID.
- Local Segment: Only Supported by the router originating it. Example: Adjacency SID
- SRGB: MPLS label block reserved for global segments.
- A Segment is represented as a MPLS label (In the case where dataplane is MPLS)
- An ordered list of Segments is encoded as a stack of labels

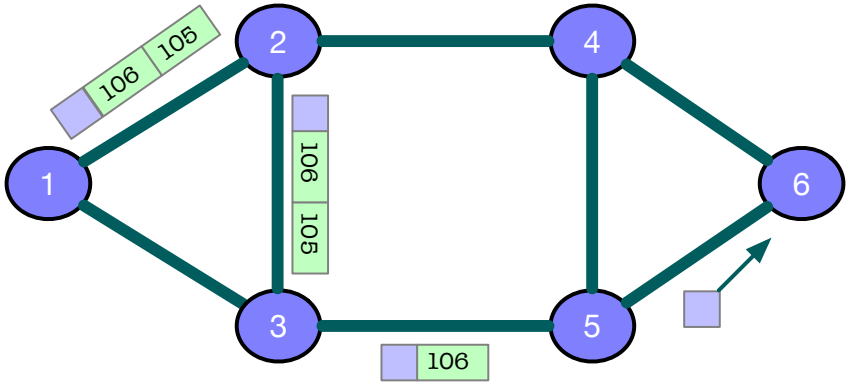
SR Example

Assume below is a Sample SR topology and we want to find send traffic from Node 1 to Node 6 via Node 5.



SR Example

Assume below is a Sample SR topology and we want to find send traffic from Node 1 to Node 6 via Node 5.

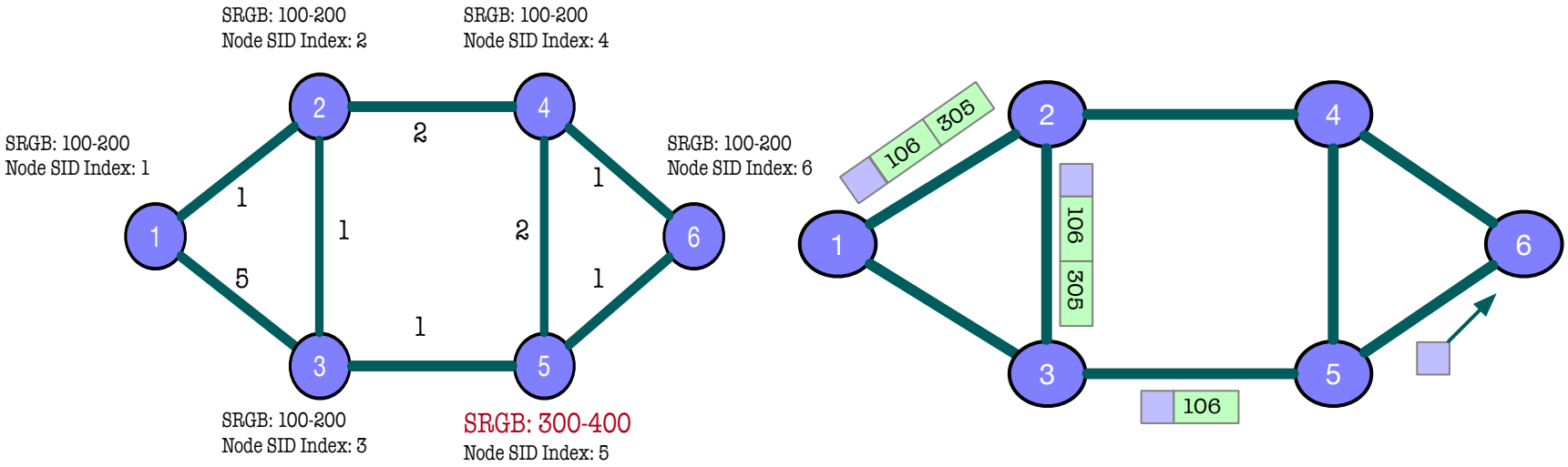


What if Node 5 has a different SRGB

SRGB of Node 5 = 300-400

SID Index of Node 5 = 5

Node SID = 300+5 = 305



One minute of BGP-LU Overview

- BGP-LU allows us to advertise unicast routes with an MPLS label binding, a prefix and label(s).
- Provides capability to send and receive BGP updates with stack of labels
- Traditionally it has been used for inter-AS VPN services like Inter-AS Option C or Carrier Supporting Carrier.
- RFC-3107 was the original RFC which was updated by RFC-8277.

BGP-LU as Southbound Protocol

- Once we have computed a Path, we will need a southbound protocol to communicate with the headend to program the paths.
- One of the option is BGP-LU and SR supported is available on latest ExaBGP stack.

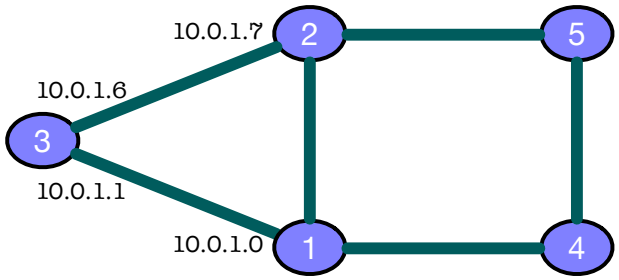
```
neighbor Router3 {
  family {
    ipv4 nlri-mpls;
  }
  announce {
    ipv4 {
      nlri-mpls 10.0.0.4 next-hop 10.0.1.7 label [ 800001 800004 ];
    }
  }
}
```

Destination

Headend outgoing next-hop

SR Label Stack

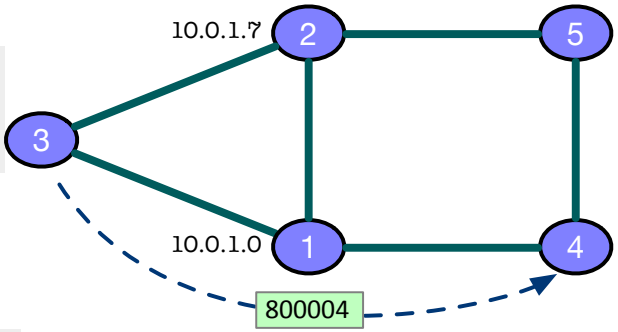
Sample Exabgp config



BGP-LU as Southbound Protocol

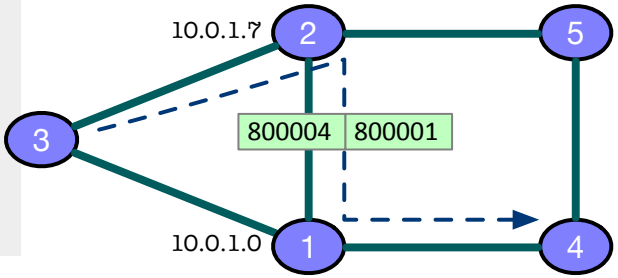
Default Path to 10.0.0.4 without BGP-LU Route

```
tesuto@mx3> show route 10.0.0.4
10.0.0.4/32        *[IS-IS/18] 11:53:58, metric 20
                    > to 10.0.1.0 via ge-0/0/0.0
```

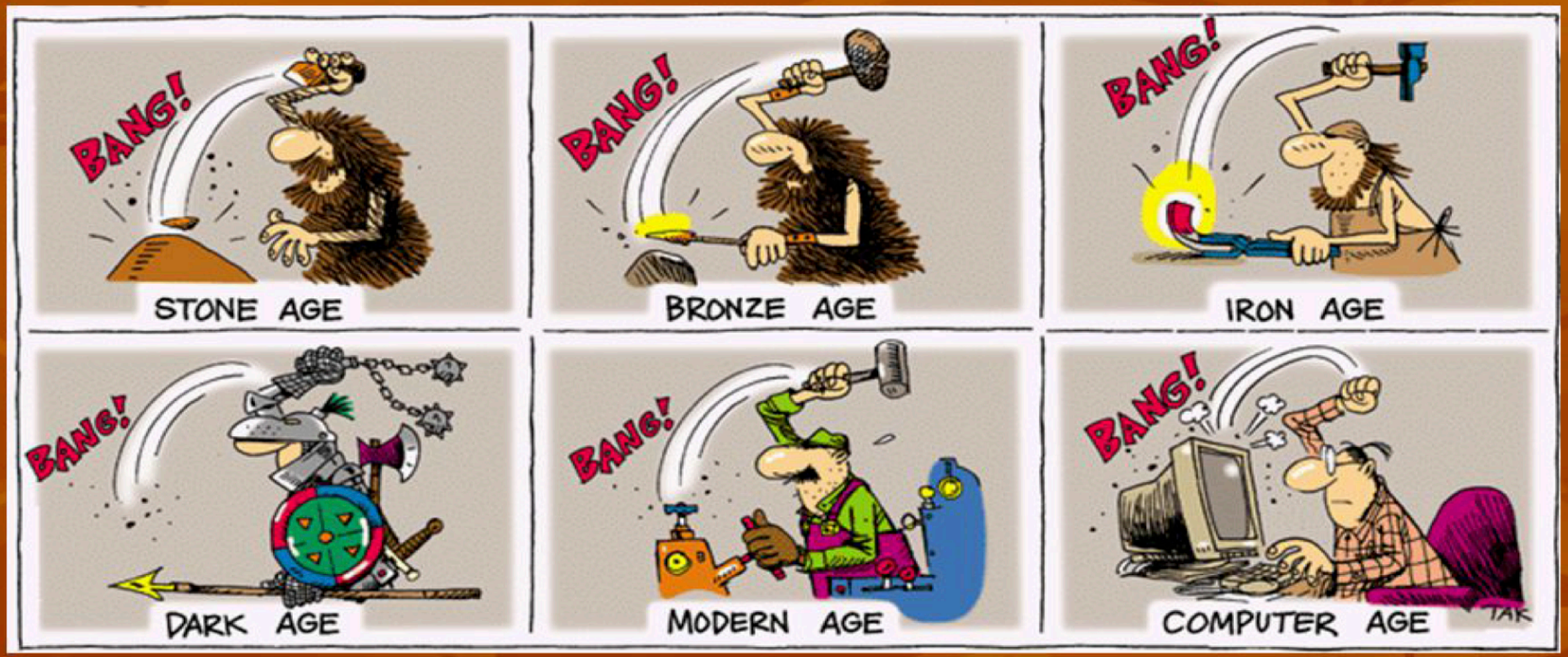


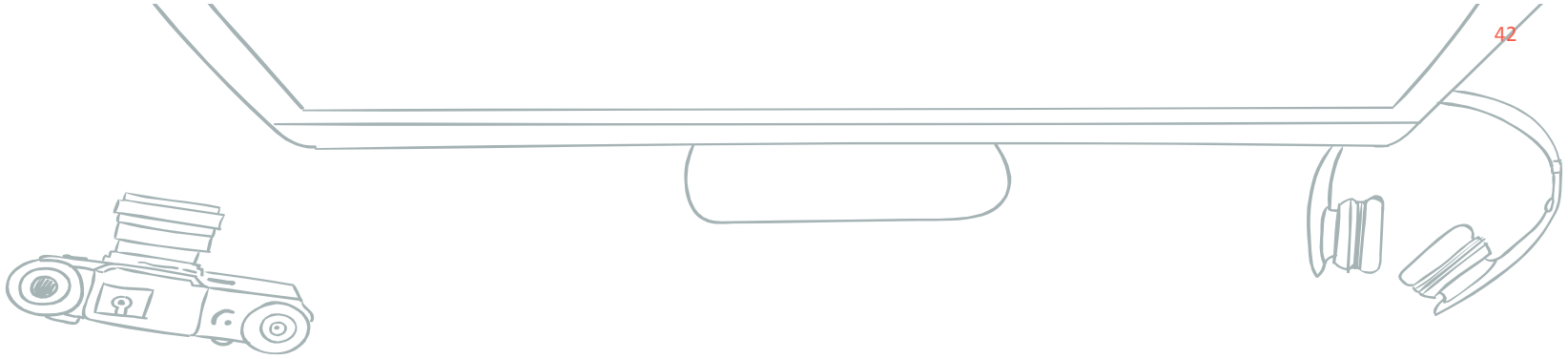
After BGP-LU path pushed from ExaBGP

```
tesuto@mx3> show route 10.0.0.4
10.0.0.4/32     *[BGP/13] 00:00:07, localpref 100, from 67.160.98.142
                  AS path: I, validation-state: unverified
                 > to 10.0.1.7 via ge-0/0/1.0, Push 800004, Push 800001(top)
tesuto@mx3> traceroute 10.0.0.4
 1  10.0.1.7 (10.0.1.7)  65.576 ms  13.990 ms  16.017 ms
     MPLS Label=800001 CoS=0 TTL=1 S=0
     MPLS Label=800004 CoS=0 TTL=1 S=1
 2  10.0.1.2 (10.0.1.2)  23.934 ms  22.245 ms  24.019 ms
     MPLS Label=800004 CoS=0 TTL=1 S=1
 3  10.0.0.4 (10.0.0.4)  31.872 ms  22.277 ms  23.831 ms
```

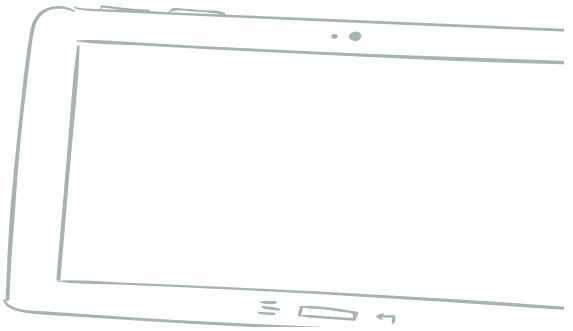


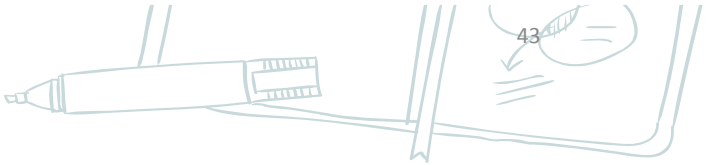
Nothing can stop automation





Questions





Special Thanks to our Lab Partner Tesuto

TESUTO


ORACLE®