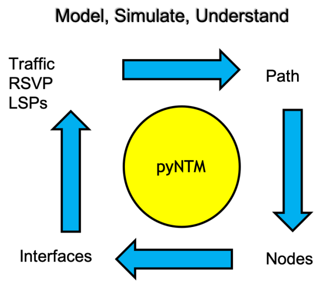
pyNTM Training Module 1 - intro to pyNTM and modeling basics



python3 Network Traffic Modeler (pyNTM)

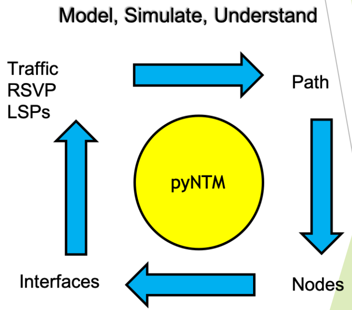
Tim Fiola

Versioning

- Training v1.0
- pyNTM v1.5

Agenda

- Problem statement
- Network modeling provides value
- Simulations
- Network modeling use cases
- We need open source tools in this space
- What is pyNTM?
- pyNTM and *networkx*
- Why is pyNTM helpful?
- pyNTM mechanics
- pyNTM features
- Wide area network modeling options
- Who is pyNTM for?



Problem Statement – Understanding the wide area network during failure states and maintenances and how to grow the network

- In a large, meshy network, it becomes difficult to understand how a given failure will truly impact interface utilization in other parts of the network
  - leads to educated guessing and general *rules of thumb*
- WAN capacity cannot be solved simply throwing money at the problem
  - WAN circuits are expensive
  - WAN circuits are not always available
- Capital in the WAN must be efficiently allocated

Network modeling provides strategic value

- Modeling allows unique, data-based understanding of how network will behave during
  - Failover
  - Changes in the traffic matrix
  - Changes in topology, such as adding RSVP mesh or changing a link metric
- This understanding prevents
  - overbuilding WAN links, which strands capital
  - Underbuilding WAN links, which increases risk

A network model provides value for people in the following roles (to name a few)

- Capacity Planner
  - Plan network to optimize latency, cost, simplest topology, etc
- Network Engineer
  - Test different topologies
- Anyone working a maintenance
  - Simulate the effects of taking down a router (Node) for a maintenance
- Anyone with interest in network performance

A network model has two inputs

- Traffic Matrix
  - Each entry describes a *demand*
  - Each demand has
    - *magnitude*, which describes how much traffic is in that demand
    - A source and destination node
  - The traffic matrix for a network will vary throughout the day, month, season, etc
  - Getting good traffic matrices can be challenging
    - Understanding your network's traffic matrices allows for truly effective engineering and planning
- Topology
  - Layer 3 nodes
  - Circuits (comprised of 2 interfaces) between layer 3 nodes
  - Shared Risk Link Groups (SRLGs)
  - RSVP LSPs

Sample traffic matrix (Mbps)

Source \ Destination	A	B	C
A	-	45	120
B	60	-	
C	75	150	-

This example traffic matrix shows traffic sourced from Node A destined to Node C with a magnitude of 120Mbps

How will this traffic transit the network?

Network modeling provides *simulation* capability

- Applying the traffic matrix to the topology and converging the model produces a *simulation*
  - The *modeling engine* governs behaviors of demands, LSPs, etc in the model
  - pyNTM has a modeling engine
- For a given day, you can produce a simulation for different parts of the day by creating a traffic matrix that reflects source and destination traffic entries for each part of the day
  - What happens during a given failure if it were to occur at different parts of the day?
  - What is the best time to conduct a maintenance on a given router?
  - Where is the best place to augment the network to best handle our holiday traffic matrices?

Wide area network modeling vs legacy techniques

Non-modeling techniques

- Rules of thumb, such as augmenting a circuit when it reaches 50% utilization, don’t guarantee that a given circuit will be able to handle the events *you are interested* in
- Planning an auto-bandwidth RSVP network adds additional complexity
  - The demands a link handles can change throughout the day/week/season

Modeling

- Simulations allow for
  - Understanding how your network traffic will behave during a failure event
  - Understanding how your network will behave with additional traffic
  - Better understanding of how RSVP LSP meshes will behave
- Understanding your network allows you to
  - Efficiently allocate capacity/capital where it’s really needed
  - Plan for and understand events you care about
- At a minimum, allows you to make a more educated estimation

Some example use cases for network model

Understanding current network topology

How many ECMP paths does a given demand take across the network?

Understanding failover by modeling failures

- Link(s)
- Node(s)
- Shared risk link group(s) (SRLG)

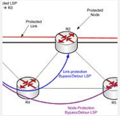
Understanding where it makes sense to augment a network

- Deploy capital where it’s most needed
- Don’t strand capital

More example use cases for network model

Understanding how changes in the network affect traffic flow

- More/less traffic
- Adding capacity to existing link(s)
- New link(s)
- Metric changes



RSVP Implementations

- Adding RSVP overlay to IGP network
- Adding/removing parallel LSPs
- Failover

The need

- Open-source tools that allow programmatic network modeling and simulation
- Specifically, there are two needed components
  - Open source modeling engines
  - Open source tools to create reasonable traffic matrices
- Nice to have: open source GUI for visualizaton

So, what is pyNTM?

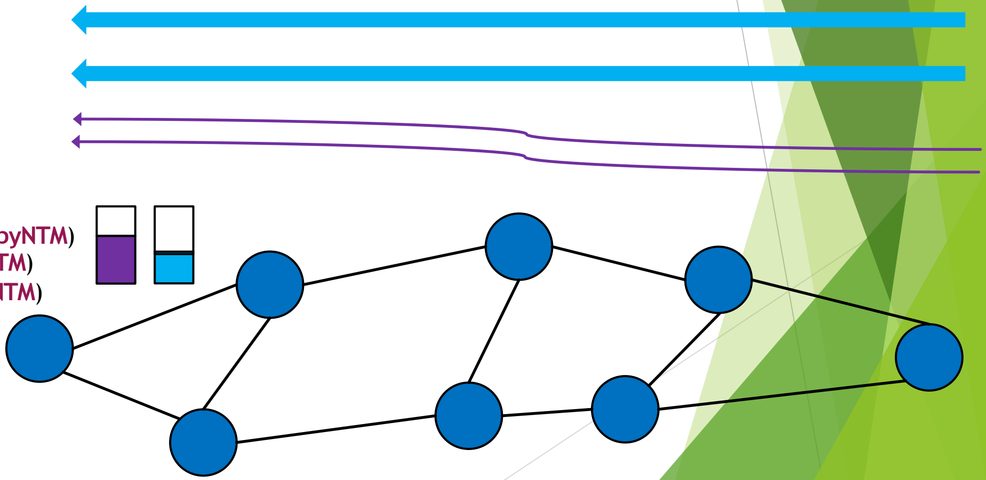
- pyNTM is the python3 Network Traffic Modeler
- pyNTM is an **open source** WAN *modeling engine*
- Applies a traffic matrix to a network topology to route traffic as the network would
  - Uses *networkx* module to get the topology path info
  - *networkx* is a GREAT tool to get path info in a topology . . .
  - . . . but there's more to modeling than just path info

Networkx and pyNTM roles in pyNTM simulations

Demand routes (networkx)

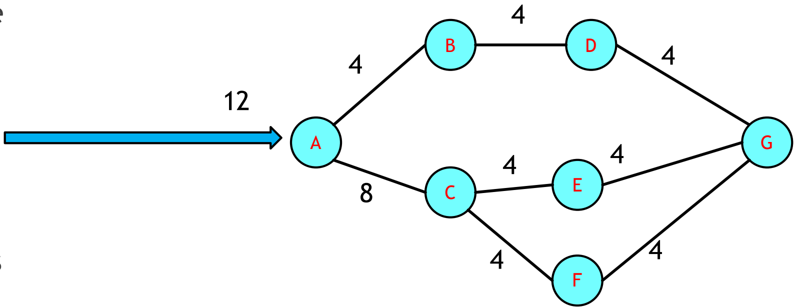
RSVP routes (networkx)

Interface utilization (pyNTM)
Interface reserved bandwidth (pyNTM)
Demand path's interfaces (pyNTM)
RSVP LSP path's interfaces (pyNTM)



Networkx and pyNTM with ECMP traffic

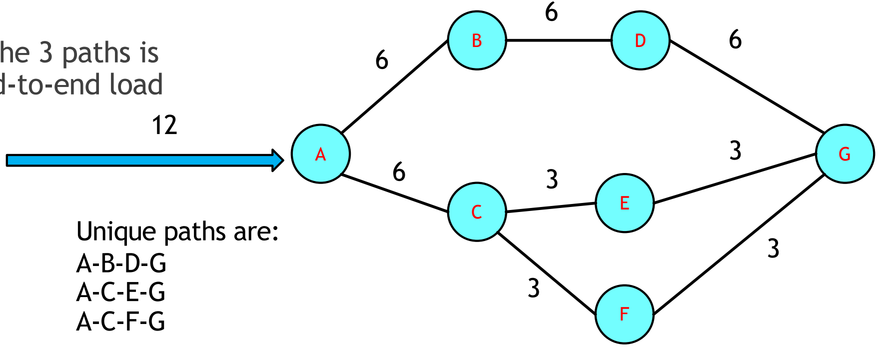
- Networkx is great to find all the unique paths through a topology
- However, you can't just model a demand with ECMP by splitting traffic evenly across all the unique paths
  - That would be ***end-to-end*** load balancing
- In the example to the right, a demand with a magnitude of 12 enters the topology
- Spreading the traffic evenly across the 3 unique paths results in the traffic spread shown



Unique paths are:
A-B-D-G
A-C-E-G
A-C-F-G

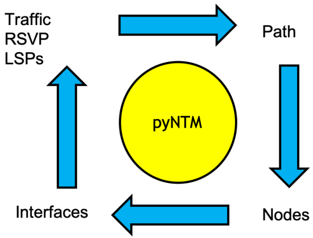
Networkx and pyNTM with ECMP traffic

- pyNTM models ***hop-by-hop*** ECMP across the 3 unique paths
  - This is how OSPF and ISIS load balance
- In the example shown, a demand with a magnitude of 12 enters the topology
- Hop-by-hop load balancing results in the traffic spread shown
- This traffic spread across the 3 paths is very different than the end-to-end load balancing traffic spread



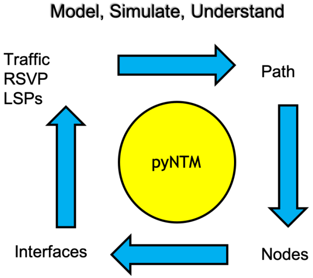
Why is pyNTM helpful?

Model, Simulate, Understand



- PyNTM leverages path information from *networkx* in a *network state-specific context*, allowing for modeling of *network-specific* state:
  - Modeling utilization on interfaces
  - Modeling traffic demands consuming interface bandwidth
  - Modeling RSVP LSPs consuming reservable interface bandwidth
  - Associating a traffic demand with the specific interfaces along the path(s)
  - Determining the available path(s) that have a given amount of reservable bandwidth
- pyNTM APIs allow for programmatic network modeling capability

Why is pyNTM helpful? (continued)



- pyNTM allows users to easily modify the network topology and determine alternate network state based on that change
  - Failing layer3 Nodes, Circuits, SRLGs, etc
  - Adding new Nodes, Interfaces, traffic Demands, etc to the topology
  - Adding new/additional auto-bandwidth LSPs to the network
- pyNTM is specifically designed to easily relate the following items:
  - Traffic Demands
  - RSVP LSPs
  - Path info
  - Interfaces
  - Nodes

pyNTM Mechanics

- pyNTM *Model* object describes the network topology
- The Model object contains the following objects:
  - Interfaces/Circuits
  - Nodes
  - Demands (traffic)
  - RSVP LSPs
  - Shared Risk Link Groups
- The Model object applies the traffic matrix to the topology, allowing the traffic to flow as it would in a real network
  - This produces a simulation
- Valuable data can be mined from simulation results

pyNTM features (as of v1.5)

- IGP (OSPF/ISIS)
- RSVP LSPs
  - Supports full mesh
    - IGP shortcuts is on the roadmap
  - Auto-bandwidth
  - Fixed/manually-assigned setup bandwidth
- Shared Risk Link Groups (SRLGs)
- Feature requests are accepted on GitHub!
- Currently only supports modeling a single link between layer 3 nodes
  - Modeling multiple/parallel links between nodes may incur a large performance cost
  - We have top people looking into that problem . . . *TOP* . . . *PEOPLE*

Options for modeling/simulation

Feature	Commercial Options	pyNTM
Cost	$tens-hundreds of thousands	$0
APIs for programmatic modeling	**Y**	**Y**
Includes capability to create traffic matrix	Y	N
Sophisticated GUI for visualization	Y	N
Open Source	N	Y
Dependent on vendor	Y	N

- Commercial option examples
  - Cariden MATE/Cisco Wan Automation Engine ← I used to work at Cariden
  - WANDL/Juniper Northstar ← Juniper is my current employer

So who is pyNTM for?

- If your org/company can generate a reasonable traffic matrix
  - Access to data scientists
  - PMACCT and NetFlow
  - Forecasted traffic demands
- If your org has basic python coding skills
- If your org does not want to rely on and/or manage external modeling vendors

pyNTM provides the open source modeling and simulation engine and can help you today

Next steps . . . .

- Continue with the follow-on pyNTM training modules that explain
  - How to get pyNTM from github or pypi
  - How to set up pyNTM in a virtual environment
  - How to use the example scripts included in the github repository
  - How to use pyNTM to solve for some basic, but very powerful, use cases

FIN

Backup slides

The value proposition for network modeling

- In a large, meshy network it becomes very difficult to understand what traffic flow will look like for a given failure event
  - This difficulty causes operators to augment and/or turn up additional, expensive WAN circuits in response, using general rules of thumb which don't guarantee that a given circuit will be able to handle failure events that you are interested in
    - augmenting a circuit when it reaches 50% utilization
  - This difficulty causes network planners to spend hours/days/months understanding how their network should be planned to deal with any given event (new traffic, failures, etc.)
- WAN capacity cannot be solved by throwing money at the problem
  - WAN circuits are expensive
  - WAN circuits are not always available
- Simulations allow for
  - Understanding how your network traffic will behave during a failure event
  - Understanding how your network will behave with additional traffic

The value proposition for network modeling (continued)

- Understanding your network allows you to
  - Smartly augment your network
    - Don’t augment where it’s not needed (stranding capital) for the failure scenarios you care about
    - Ensure you augment where it will be needed for the failure scenarios you care about
  - Understand how new traffic will affect the existing network in steady state and failover

Modeling options

Commercial options

- Examples (cost$$)
  - Cariden MATE/Cisco Wan Automation Engine
  - WANDL/Juniper Northstar
- Commercial options typically provide
  - GUI for humans
  - Capability to automatically create a traffic matrix for your network via gathering network gear stats
  - API set for programmatic modeling

pyNTM

- Cost - $0
- API set for programmatic modeling
- With basic python skills, you can run network simulations
- Does not create a traffic matrix

Objects you'll find in a network model

- Interface
  - Represents an interface on one end of a Circuit
- Circuit
  - Two-way link between 2 Nodes
  - Comprised of an Interface on each side
  - Each component Interface sends traffic in a single direction (toward remote Node)
- Node
  - Layer 3 device
- Demand
  - Represents traffic load from a source Node to a destination Node
  - Has a magnitude that represents the amount of traffic
- RSVP LSP
- Shared Risk Link Group (SRLG)
  - A grouping of Interfaces and/or Nodes that share a similar risk
  - When the SRLG fails, all component objects fail