This file contains

1. Remote\_scripts folder - it contains scripts that can be executed on Fabric nodes.   
   It has 1 files -   
   init\_mtp.sh - this file modifies the tmux configuration after it gets installed in EIBPDeps.
2. graphs folder - this folder contains graphml files.

# manager is the object name for class FabOrchestrator. We use it when we need to upload, download files in the code.

1. Local\_books - it contains multiple files to run, compile and execute the C code.  
     
   It contains one folder Slice\_setup - which includes   
     
   EIBPCheck- to check whether the interfaces are UP or not in a slice.  
     
   EIBPDeps - this code will upload configuration script (init\_mtp.sh)-to install utilites. It will install dependencies like - chronyc, c compiler, tshark. It will also install gdb. Sets up IP address.Turns off IP routing. It creates a base for EIBP domain nodes.  
     
   SliceBuilderGraphml\_meas - it adds measurement nodes and submit slices. Create ssh tunnel link and grafana dashboard link. We currently use this.  
     
   SliceBuilderGraphml\_MF - old setup for measurement framework.  
     
   SlicebuilderGraphmleibp1.ipnyb - has the scripts to disable network manager on the interfaces used by the EIBP code.  
     
   Topology\_info - we are storing following into a txt file -   
   "Nodename", "ifindex", "ifname", "operstate", "address", "altnames", "addr\_info": { "local":, "prefixlen"}  
     
   Slice\_setup folder ends here.
2. FabUtlis - Collection of scripts to aid in the testing of FABRIC experiments.  
   [Fabutlis.py ReadME](https://docs.google.com/document/d/15T9kC77AroC_ZPLOesEImY1XBj2QTeQzm8ybJBZoH2Y/edit?usp=sharing)

b.SliceBuilderGraphmleibp - it creates a slice, adding nodes to the slice. Parse the GraphML file.   
- Parses a GraphML file to extract node and edge information.

- Creates a network slice, adding nodes with specified properties and connecting them to prefixes based on the parsed data.  
-Submits the configured network slice.   
Add Basic IPv4 Addressing:   
-This code assigns IPv4 addresses to network interfaces in a network slice, considering whether the network is an edge network and whether it's associated with client nodes.   
-It iterates through networks and interfaces, configuring addresses based on specified rules.

1. EIBPDeps - this code will upload configuration script (init\_mtp.sh)-to install utilites. It will install dependencies like - chronyc, c compiler, tshark. It will also install gdb. Sets up IP address.Turns off IP routing. It creates a base for EIBP domain nodes.  
     
   *Initialization and Configuration Script Upload:*

Initialises the FabOrchestrator for a specified network slice and uploads an initialization script (init\_mtp.sh) for configuration.   
  
*Dependency Installation and Script Execution*:  
Install essential packages and run the initialization script on multiple nodes in parallel, including nodes with prefixes.   
"sudo dnf install -q -y tmux wireshark":  
Install - tmux: Terminal multiplexer that allows you to create and manage multiple terminal sessions within a single terminal window.   
- wireshark: Network protocol analyzer that allows you to capture and analyse network traffic.   
"sudo dnf groupinstall -q -y "Development Tools": This command install various tools and utilities, such as:  
GNU GCC C compiler (gcc)   
GNU Make (make)   
Debuggers (e.g., gdb)   
  
*Adding Default Routes to Compute Nodes*:   
Sets up default routes for IP-based forwarding on compute nodes (prefix), allowing them to route traffic outside their local network.   
  
*Turning off IP-based Forwarding on EIBPNodes:*  
Disables traditional IP-based forwarding on EIBP nodes (prefixes) using sysctl commands.

1. EIBPStart- It uploads code, compiles code.  
   It uses uploadDirectoryParallel from the Fabutlis.py.
2. EIBPTest- Access the slice, Deleting the logs.  
   -takes a specified network interface down, and then it kills a tmux session named MNLR in parallel across multiple nodes.  
   -creates a log directory if it doesn't exist, downloads log files in parallel from multiple nodes with modified names, and then brings a network interface backup
3. Config\_small.txt - contains sudo commands

Start\_MNLR.py - will upload MNLR\_code from the local machine to the remote node, compile the MNLR code, and then run the code by providing node specific parameters.

If it is tier 1 node, it is given the tier value after the parameter -T

T is tier level, L is label assigned. For L, only for the first node we manually assign labels and for the rest of the nodes it gets automatically assigned. -N is depicting whether it is connected to any access node(IP node). -N 1 means it's not connected to any access node or IP node. -N 0 means it’s connected to IP nodes and we give IP address and the eth port to which it is connected to.

To find an IP address we are currently using Add Basic IPv4 addressing from the Slicebuilder.py (Peter’s code).

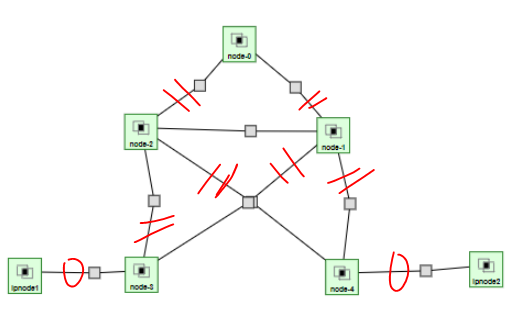


Fig: 1

// - core network

O - edge network

* Node name convention: Slice is created with pre-defined core and edge node name prefixes. Example - core= “node” , edge = “ipnode”
* Network name convention - networks are named core or edge based on the nodes on the network, a core and edge node make an edge network. This results in topology as Fig 1.
* Take each network, determine if its core or edge, and give each node an IP address.

Core = 192.168. X .1 and 192.168. X . 2  
Edge = 192.168. X . 254 for core node  
 192.168. X . 1 for edge node

* For edge node IP routing, add a ***default route*** pointing to 192.168. X . 254.
* For all core nodes, turn off IP finding.