**SliceBuilderGraphmleibp.ipynb**

a. 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***  
- In this system, 192.168.0.0/16 is the address space for all interfaces on the FABRIC slice. - - Each network is a /24 subnet of this network. Edge networks have client/compute devices with lower address (ex: .1) and networking nodes with higher addresses.  
- This Python code iterates through networks retrieved from slice.get\_networks(). It configures IPv4 addresses for interfaces within each network based on whether it's an "edge" network or not. For edge networks on networking nodes (not clients), it assigns the highest available address; otherwise, it assigns the lowest available address from the respective network's host addresses, incrementing the third octet for each subsequent network.

**EIBPDeps.ipynb**

a. 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 MTP Nodes:*** Disables traditional IP-based forwarding on EIBP nodes (prefixes) using sysctl commands.

**Topology\_info.ipynb**

This Python script extracts and saves network details including node name, interface index, name, operational state, address, alternative names, local address, and prefix length into the "eibp\_ip\_addr\_filtered.txt" file. If no qualifying entries are found (ifindex >= 3), it notifies that no valid information was discovered.

**EIBPStart.ipynb**

We use this script to upload the new code, compile it at the FABRIC nodes and run EIBP code in all the Fabric nodes.

* ***Initialization of variables*** - Function is used to initialise slice, configuration file name and code directory. We need to execute it only once when we start the server/ python notebook.

We also need to execute it if we change the code directory location or slice name or configuration file name, otherwise we don’t need to execute it every time.

* ***Access the Slice***This function is used to access the slice and the manager from the Fablib.py. We don’t need to execute this every time.
* ***KILL tmux session MNLR (if required)***This function needs to be executed almost every time if we are already running an MNLR session in tmux. This process kills the tmux MNLR session on each node.
* ***Delete the Log from a Prior Test (if required)***This function needs to be executed if there are already existing log files. To delete log files we need this function.
* ***Delete MNLR\_Code in the slice(if required)***
* ***Upload the Source Code***  
  This function needs to be executed once to upload the C MNLR code on all the nodes. If we made any changes in the C code then only we need to execute this cell.
* ***Compile the Code***Reading command files and running MNLR code on each node using tmux.
* ***Reading command files and running the MNLR code on each node using tmux***This code reads a configuration file ('config.txt') with node names and associated commands, storing them in a dictionary. It then uses this data to create and execute new tmux sessions for each node with the specified command using a manager's function.
* ***KILL tmux session MNLR***This function needs to be executed almost every time if we are already running an MNLR session in tmux. This process kills the tmux MNLR session on each node. Here we are repeating this step again to collect files.
* ***Collect Log results***  
  This code creates a directory for log files if it doesn't exist and then downloads log files named in the format "EIBP\_{name}.log" from multiple destinations with prefixes "c," "d," and "a" into the specified log directory.It also adds the node name to the downloaded log files' names. Now that the nodes have logged updates to their respective log files, they need to be downloaded to be analysed.
* ***Delete the Log from a Prior Test***This function needs to be executed if there are already existing log files. To delete log files we need this function.  
    
  When do we need to download log files?  
  For sanity check. To check if every node is communicating properly.if we introduce a failure to check the log files for metric collection

**EIBPTest.ipynb**

We will collect by failing an interface. So MNLR has to be running in the nodes so the failure and its impact is recorded in the log files by MNLR.

* ***Initialization of variables*** - Function is used to initialise slice, configuration file name and code directory. We need to execute it only once when we start the server/ python notebook.

We also need to execute it if we change the code directory location or slice name or configuration file name, otherwise we don’t need to execute it every time.

* ***Access the Slice***This function is used to access the slice and the manager from the Fablib.py. We don’t need to execute this every time.
* ***Delete the Log from a Prior Test (if necessary)***This function needs to be executed if there are already existing log files. To delete log files we need this function.
* **EIBP Initial Convergence**This code reads a configuration file ('config.txt') with node names and associated commands, storing them in a dictionary. It then uses this data to create and execute new tmux sessions for each node with the specified command using a manager's function.
* ***EIBP Reconvergence Testing - Take the Interface down***This code brings DOWN a network interface specified by {INTF\_TO\_FAIL} across multiple nodes and then retrieves IP addresses from the specified nodes in parallel using Python's manager.executeCommandsParallel function.
* ***KILL tmux session MNLR***This function needs to be executed almost every time if we are already running an MNLR session in tmux. This process kills the tmux MNLR session on each node. Here we are repeating this step again to collect files.
* ***Collect Log results***  
  This code creates a directory for log files if it doesn't exist and then downloads log files named in the format "EIBP\_{name}.log" from multiple destinations with prefixes "c," "d," and "a" into the specified log directory.It also adds the node name to the downloaded log files' names. Now that the nodes have logged updates to their respective log files, they need to be downloaded to be analysed.
* ***Bring the Interface Back up***  
  This code brings UP a network interface specified by {INTF\_TO\_FAIL} across multiple nodes and then retrieves IP addresses from the specified nodes in parallel using Python's manager.executeCommandsParallel function.

**EIBPAnalysis**Run all of the cells in the notebook. This will generate a result from the log files for convergence time, control overhead, churn rate.  
  
Convergence time: time taken from 1 failure node to the last node heard about it.

Control Overheard: Size of the failure message overall

Churn rate : how far the error message propagates.

IF\_DIRECT\_DOWN: nodes detect interface is down

IF\_TIMER\_DOWN:dead timer is down or expiring.