This version is meant for the phase 2 of the SDN work: Joint SDN synchronization and placement.

This means the major things are as represented in the slides I sent earlier, including:

*State space* 🡺 Concat of *State space for synchronization* and *state space of placement*.

Action space is similar, i.e., lets consider: 4 neigbhors and sync. Budget 2, and 2 placement locations, then🡺 *Action space* looks like 🡺 1 0 0 1 0 1, where first 4 terms for sync action and last 2 for placement action.

**Note: This version does not have changing state or action space since it is not transfer learning-based version**

Start with the two files,

1. SDN\_sync for shortest path routing task
2. SDN\_sync\_LB.py for load balancing task

These files set up the DRL including learning agents and the game to be played, with the environment defined by *config.environment* (config defines most parameters in the DRL), and notice the value for shortest path routing is **sdnSync\_SP**, a class defined in **sdnSyncGameDictEncoded**.

I suggest using go to definition features in vs\_code or equivalent on your IDEs

Inside sdnSyncGameDictEncoded, class sdnSync, I have tried to remove a lot of unnecessary variables but please note that some may still be remaining.

In line 139 of sdnSyncGameDictEncoded, self.config\_atlernator as defined in main files like SDN-sync.py are used to choose between two methods. If False, placement and sync. Rewards are alternatingly improved, if not, they are improved together as defined in the slides.

*Self.network.update\_controller* links to the class that defines and evolves the network as DRL is learning/ game is played.

from underlyingNetwork\_creator\_v3 import NetworkState\_SP is pointing to this file.

We have discussed underlyingNetwork\_create\_v3 multiple times but do let me know if particular functions bother.

**Please ignore load\_balancing functions for now.**