To Do list:

- ☑ (1) Think about network architectures to integrate, what type of protocols to use, etc.
- ☑ (2) Figure out how to simulate my chosen architecture

(1)
Some thoughts on network architectures for traffic intersections

- Thinking about a Content Addressable Network (CAN) for intersection sensors

Given the fact that traffic lights are in a fixed position, we can probably leverage the spatial locality of CANs to manage updates and state information efficiently.

Some research on CAN locality property: Each node is only aware of its neighbours

- This simplifies routing, reduces overhead in a controlled environment
- Note: traffic control systems require low-latency communication
- CANs are scalable and decentralised but routing performance needs to be evaluated under high update frequencies of the learning model given the fact there will be real-time requirements and expectations
- Since the nodes (intersection nodes) are static, this gives some stability which plays tot the strength of CANs
- A foreseeable issue here is if this project is ever expanded to include actual smart vehicles, we might be cooked. Too many things happening at once, too many nodes trying to familiarise their neighbours, etc.
- We gotta be able to handle churn btw - this might be a game-ending L

Pivot to a more layered architecture?

- Local CAN is okay for just closely spaced intersections (immediate, short-range updates)
- Kademlia or Chord have better lookup features and can provide improved performance and fault-tolerance
- PIVOT TO KADEMLIA
    - Read Kademlia paper [1]
    - Also read Emily Martin's RS2 Kademlia paper (Thank you Emily)
- Also want to look into gossip-based protocols
    - Read the following papers:
        - Gossip-based Peer Sampling [2]
        - Gossip-Based Computation of Aggregate Information ***Key paper right here*** [3]
- One more note: with the federated learning integration, the network architecture needs to support efficient aggregation of model updates
    - Low latency and reliable message delivery

(2)

- Looked into OPNet simulation and modelling - too difficult to set up, couldn't register for Riverbed
- Did research on OMNeT++
    - Looks good potentially
    - Downloaded/installed - all is well - green across the board
    - Built sample project - simple node to node packet connection

Now that I have OMNeT++ setup and running on my computer, need a new TO DO

TO DO:

- [ ] Define objectives / requirements of my project

Objectives/Requirements

- [x] Familiarise myself more with OMNeT++ and its frameworks
    - [x] Find a framework for integrating Kademlia simulations easily (can also use Java extensions) - found OverSim - looks good
    - [x] Run through the rest of the technical documentation - done, very good stuff here
- [x] Figure out how to build traffic/city layout and simulation
    - [x] OMNeT++ is insanity. Figured out how to import OSM files from openstreetmap.org and have it outputted in the GUI.
    - [x] Figured out how to dynamically generate "cars" which are rectangles that traverse across found roads
    - [ ] Need to figure out how to have rectangles 'stop' at intersections. I briefly read in the documentation of a project that someone built regarding traffic lights and traffic light controllers. Could be the move - check back in later.
- [ ] Design architecture for actual network simulation
    - [ ] Design the network topology
    - [ ] Communication protocols (want Kademlia/gossip/anything else I think would be good)
    - [ ] Federated learning integration
        - [ ] Plan how learning updates will be aggregated and shared across the network. Probably will use info from [3]
- [ ] Develop Simulation Model
    - [ ] Create the NED files to define the network nodes and connections
    - [ ] Integrate traffic light control logic
    - [ ] Data exchange based on the P2P protocols
    - [ ] Integration of the actual inflow-outflow equations
- [ ] Specific Scenarios Simulations (lol SSS 🐍)
    - [ ] Peak (rush hour) vs. normal conditions (initial thought: maybe gather data from City of Victoria?)

- [ ] Parameter tuning (message frequency, node latency, learning update intervals, etc.)

- [ ] Testing / Iterative Development and Design
  - [ ] Just keep hammering it until it’s put together I guess
- [ ] Data Collection and Logging
  - [ ] OMNeT++ has a really good data collection and logging system so I’m not really stressed here
- [ ] Finally, document everything along with proof of concept for final presentation

References

[1] Maymounkov, P., & Mazieres, D. (2002). *Kademlia: A Peer-to-peer Information System Based on the XOR Metric*. New York University. https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

[2] Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., & Van Steen, M. (2004). *Gossip-based peer sampling*. distributed-systems.net. https://www.distributed-systems.net/my-data/papers/2007.tocs.pdf

[3] Kempe, D., Dobra, A., & Gehrke, J. (2003). *Gossip-Based Computation of Aggregate Information*. IEEE-Explore. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1238221&tag=1