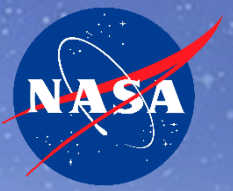
National Aeronautics and Space Administration



Microservice Architecture for Cognitive Networks

Dr. Rachel Dudukovich and Dr. Janette Briones

- **NASA Glenn - Cognitive Signal Processing Branch**

Alan Hylton and Gilbert Clark

- **NASA Glenn - Secure Networks, System Integration and Test Branch**

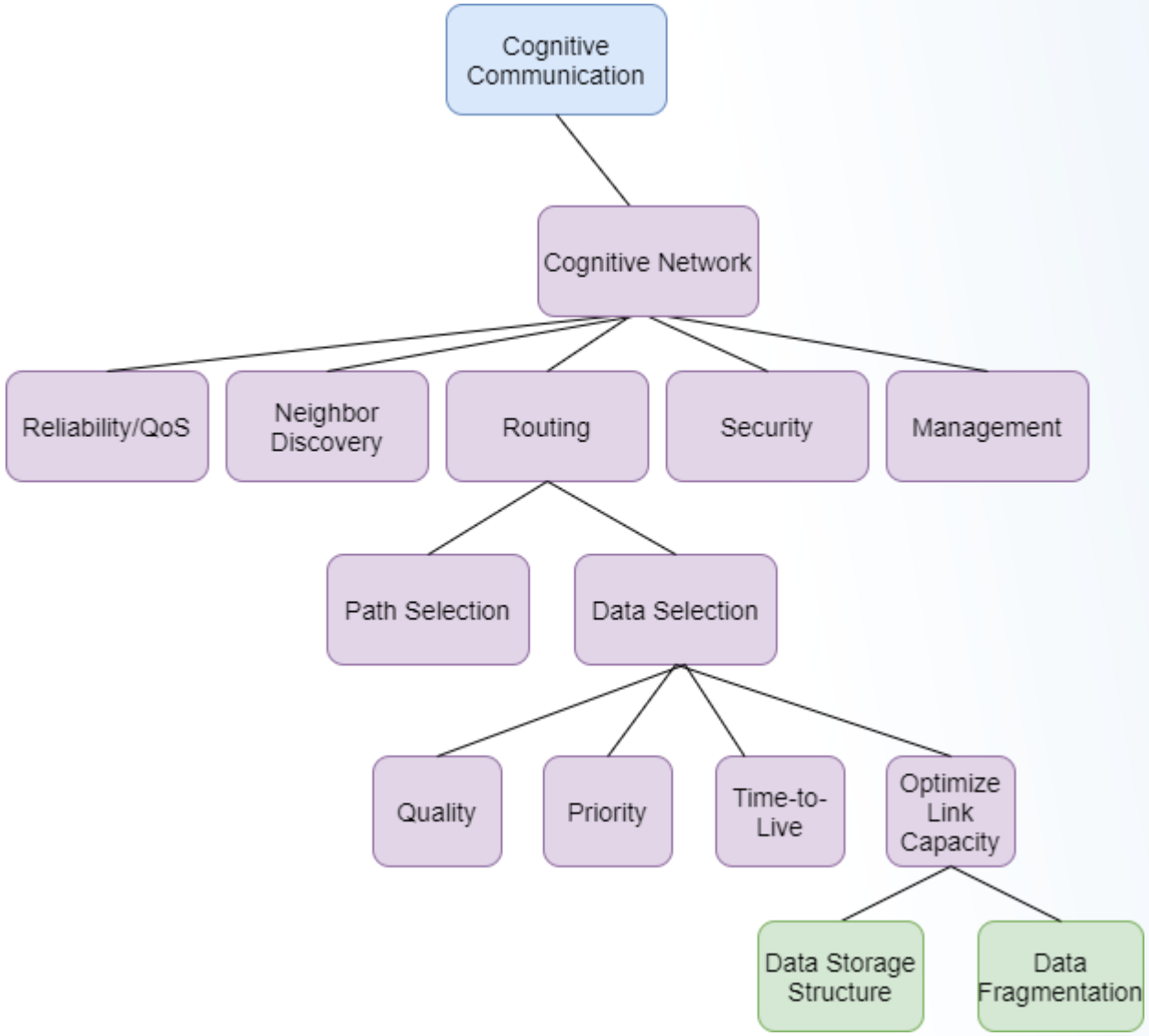
October 12, 2020

Outline

- Motivation
  - Introduction to cognitive networking
    - Efforts at GRC
  - Flight mission concepts
    - CubeSat
    - Lunar network
- Delay tolerant networking (DTN)
  - Software development efforts
  - Microservice concept
- Cognitive networking
  - AI/ML algorithms
  - Simulations
- Future work

Motivation

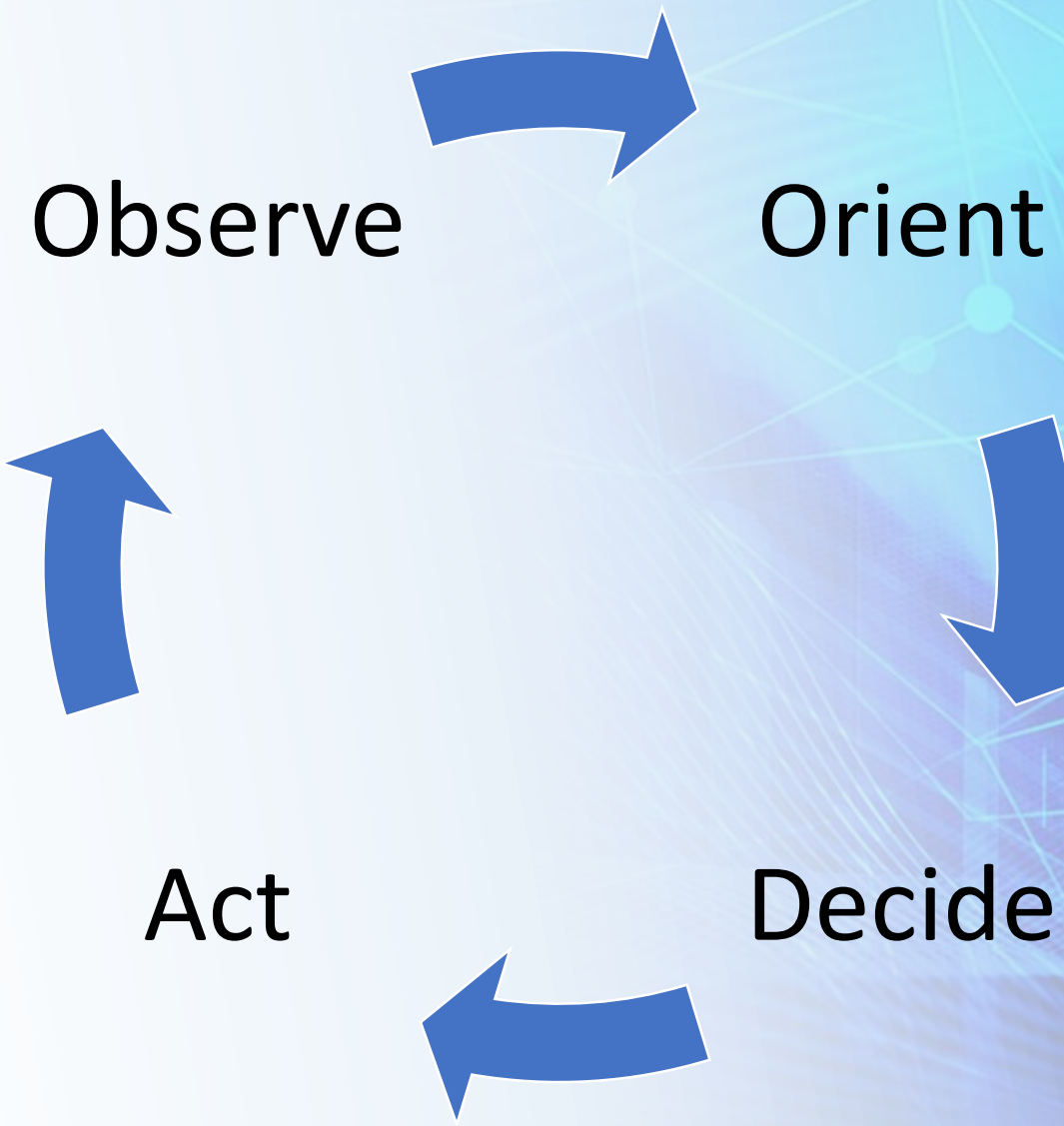
Cognitive Networking



- Understand areas where cognition could be applied to decision making process
- Understand what constitutes:
  - Automated / Algorithmic
  - Intelligent
  - Autonomous
  - Cognitive
- Intelligent : context aware

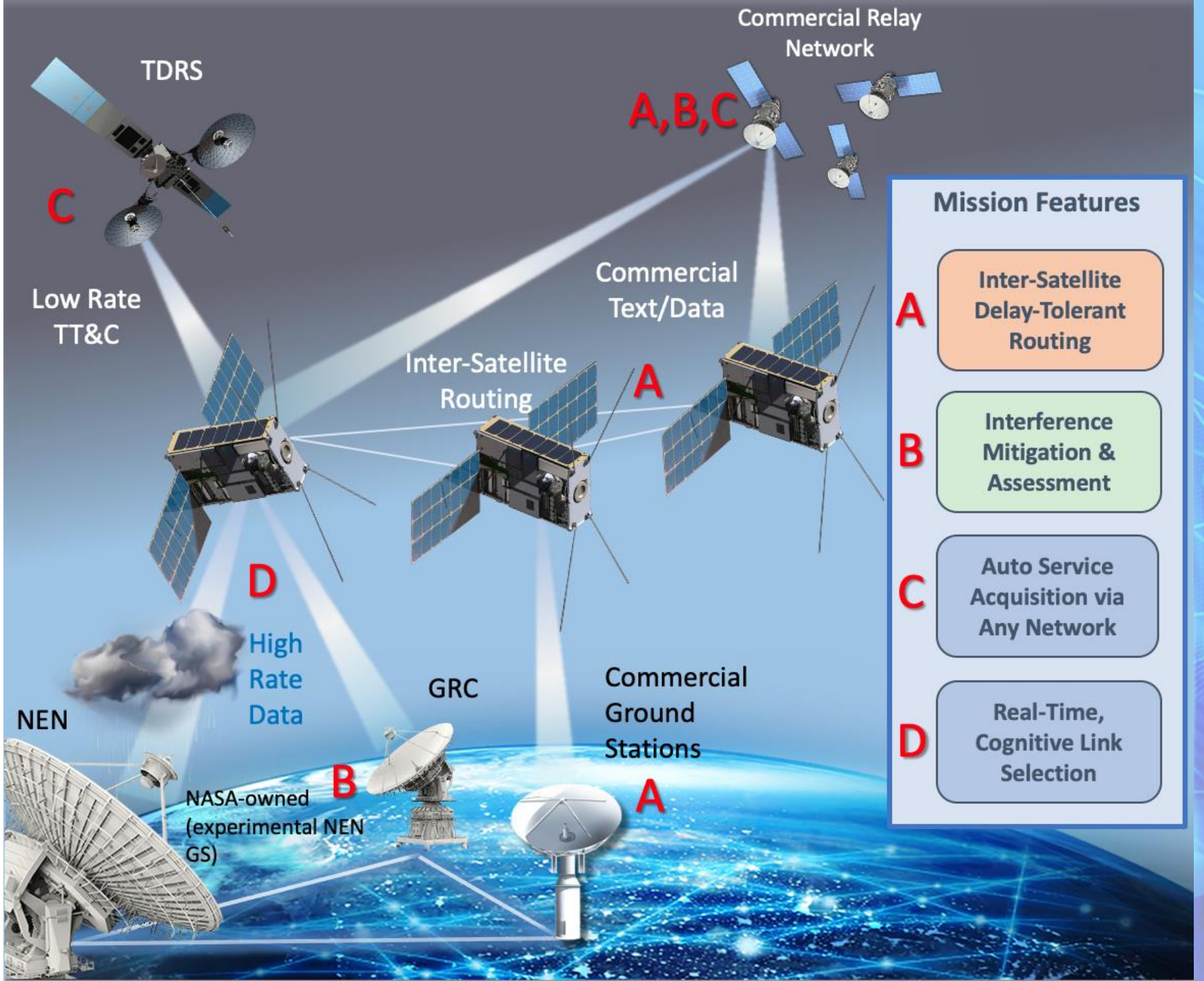
Defining Cognition

- Sense and observe environment
- Learn from previous actions
- Take action to improve objective
- React to changes in the environment
- Build on concept of OODA loop
  - Observe, Orient, Decide, Act
- How do traditional neural networks, other classification algorithms fit?



Flight Mission Concept

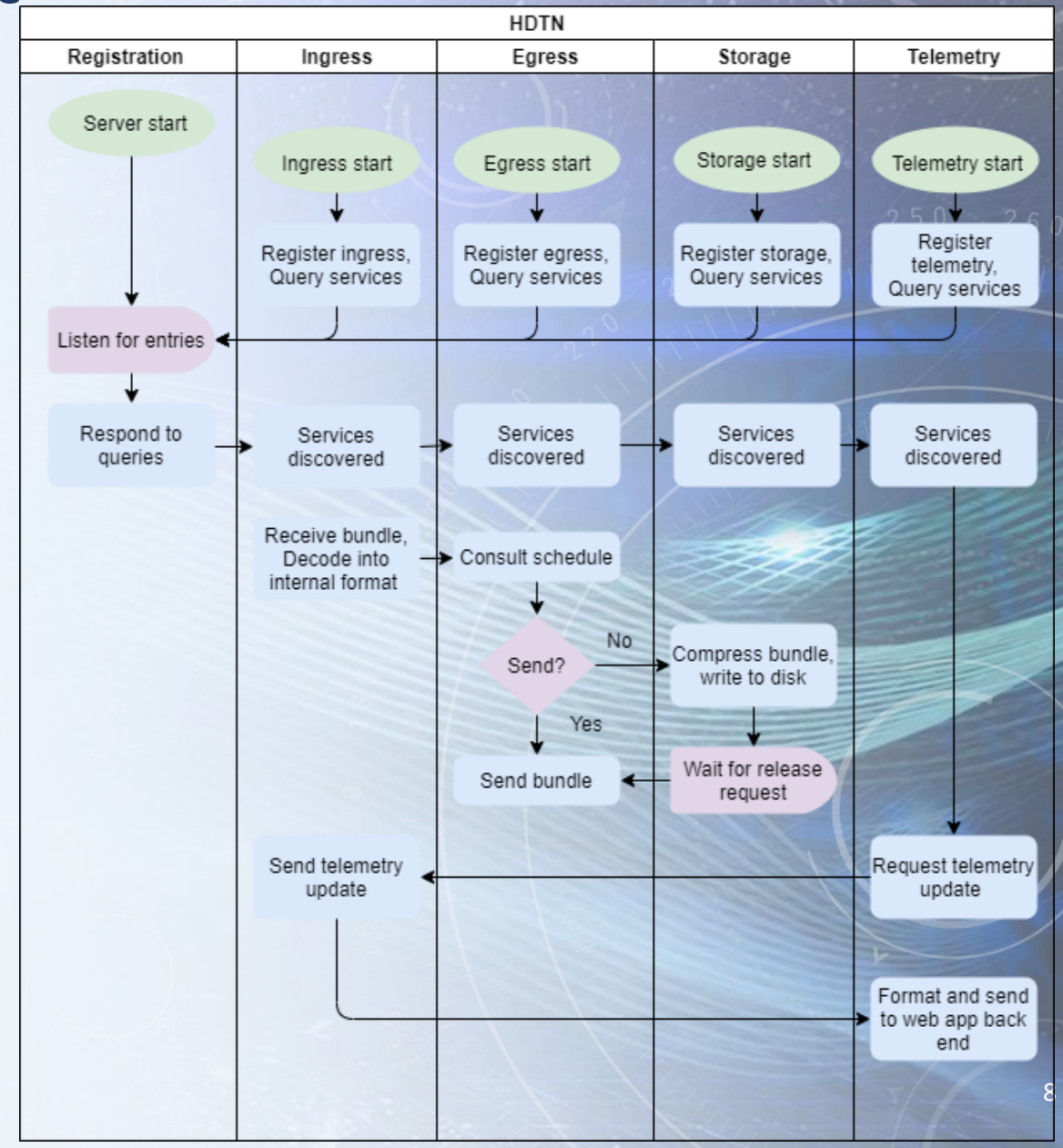
- Flight demonstration with 3 CubeSats
- Apply cognition to:
  - Networking
  - Scheduling
  - Link optimization
- Utilize commercial ground stations and services
- Extend concepts to lunar environment



Delay Tolerant Networking

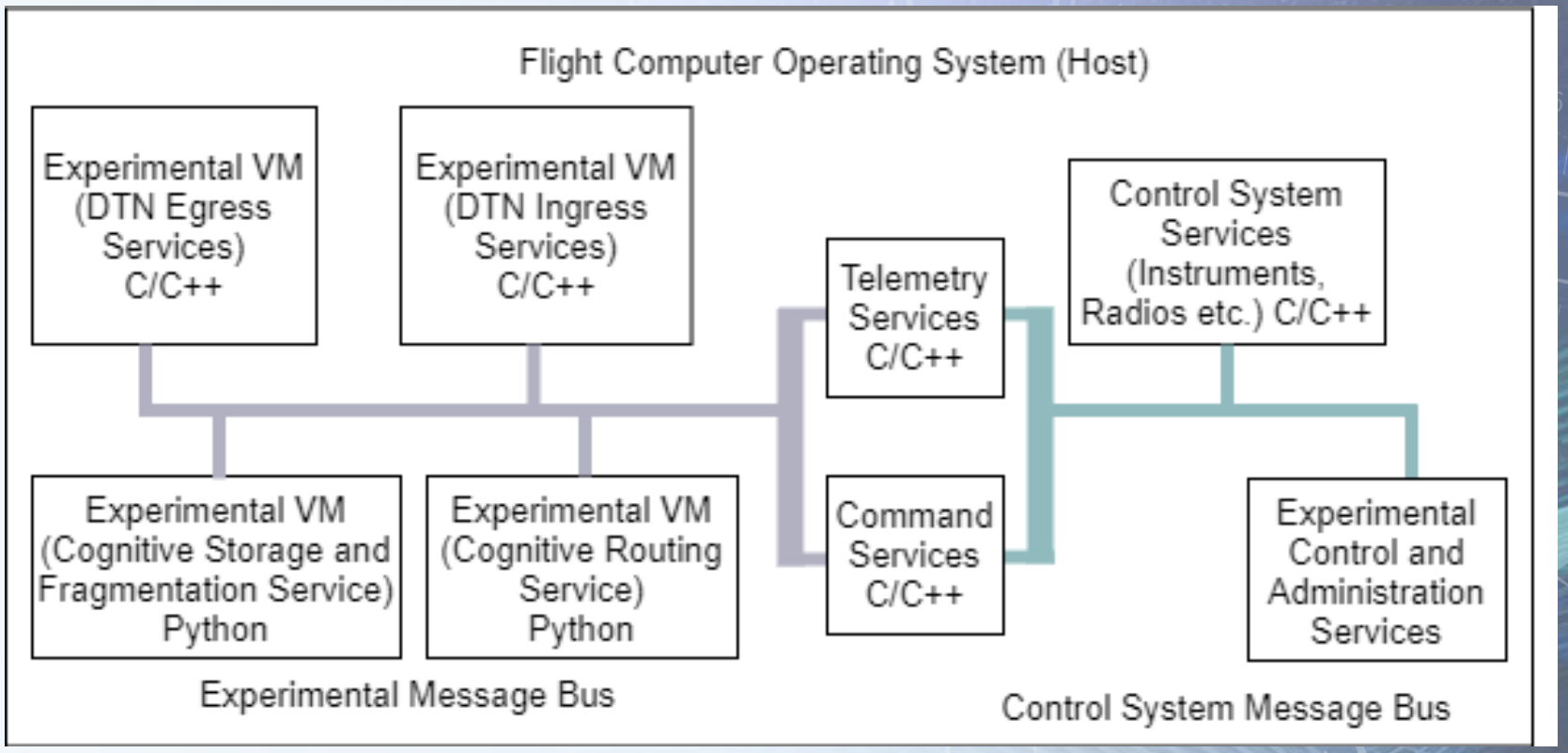
Software Development Effort

- Push for open source release
- Interest in collaborating with DTN community
- Within GRC teams, sharing code base and design concepts
- Recently released BPCodec
- https://github.com/nasa/HDTN-BPCodec
  - BP 6 and BP 7 encoding /decoding
  - BP 7 experimental, early work
  - Additional tools to send/receive bundles
  - Encoding/decoding benchmark
- Additional modules to be released relatively near term



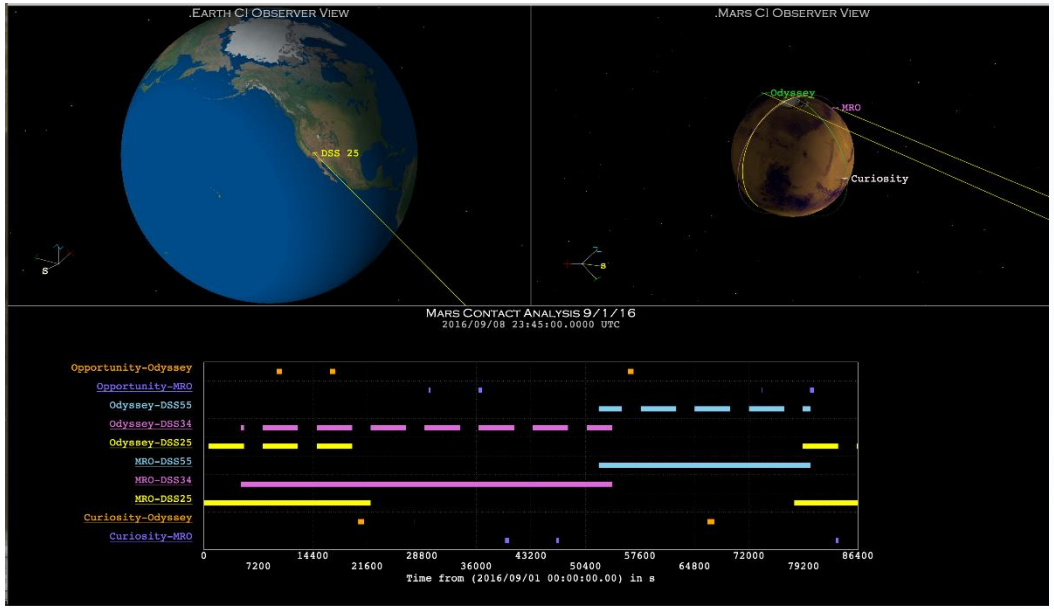
Microservice Concept

- Critical flight components are separated from experimental services
- Services exchange data using high performance asynchronous messaging library
- State is replicated versus shared to avoid use of locking concurrency mechanisms (mutex, semaphore, shared memory)
- A possible error in one service will not corrupt data/state for others



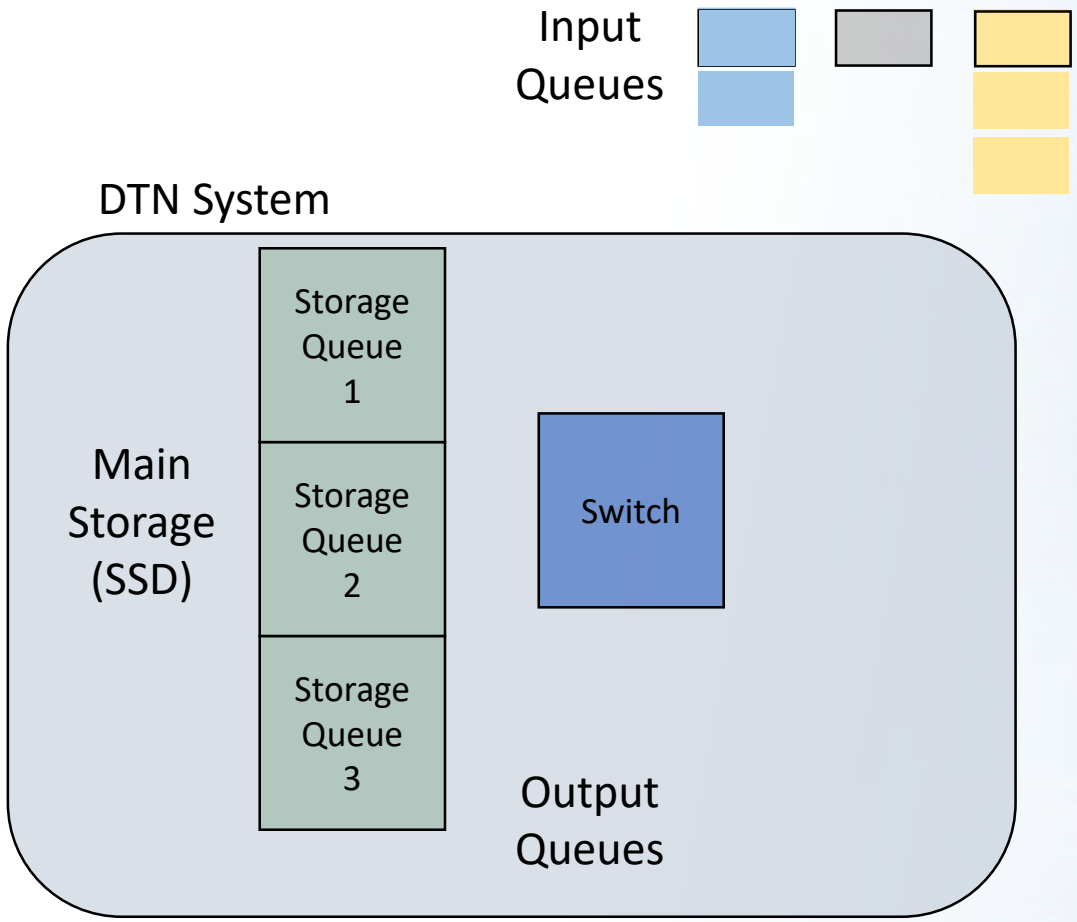
Contact Graph Routing

- Developed for space networks in which contact times between nodes are scheduled in advance and are well-known
- Uses knowledge base:
  - Sending node number
  - Receiving node number
  - Contact start and stop times
  - Data rate between nodes
  - Distance between nodes



- Computes Contact Capacity:
  - C=Data Rate*($T_{start} - T_{stop}$)
- Estimated Capacity Consumption
  - ECC=message size+overhead
- Residual Capacity:
  - $\sum C - \sum ECC$
- Find contact with best case delivery time to transmit a given message

Complex Network



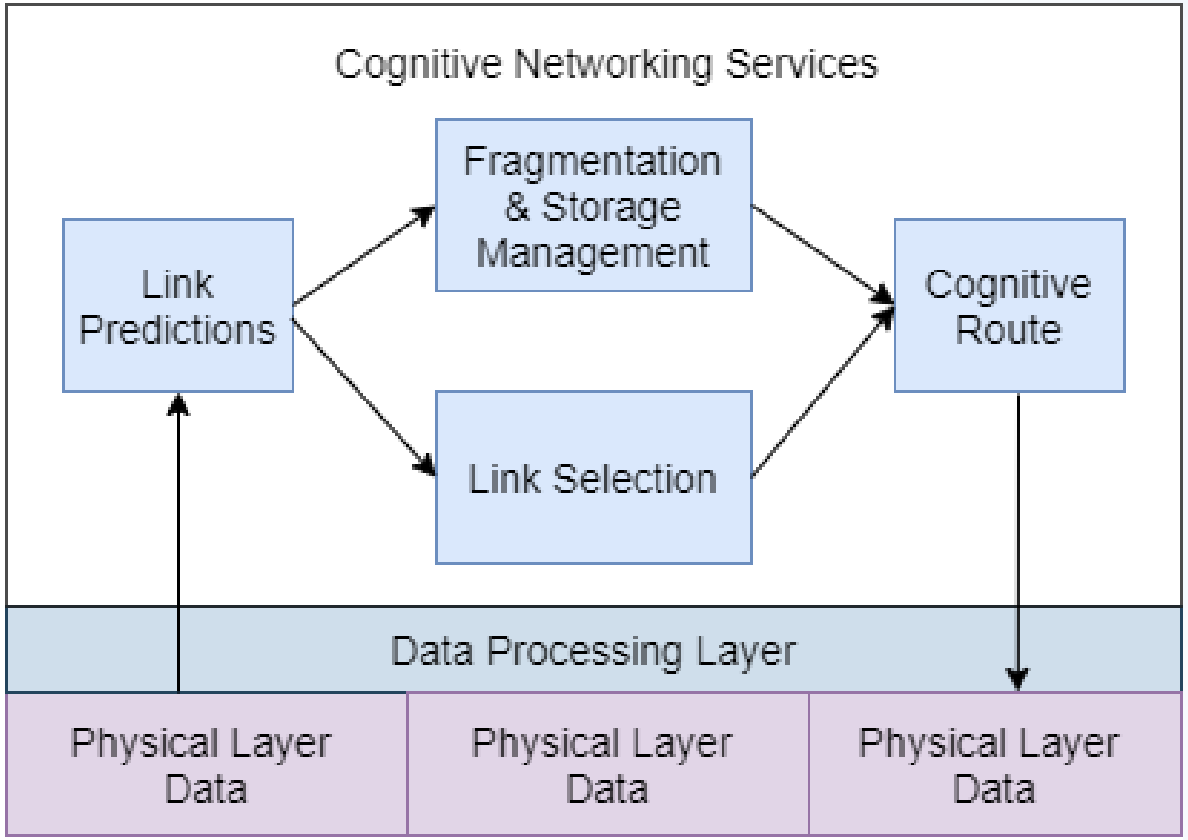
- Network increases in complexity as number of nodes increases
- Even more complicated to add:
  - Data Priority
  - Custody Transfer
  - Optimize throughput
  - Data Expiration
  - Reliable protocols
- How can ML/AI aid in decision making process?

Cognitive Networking

Algorithm Summary – Cognitive Networking Area

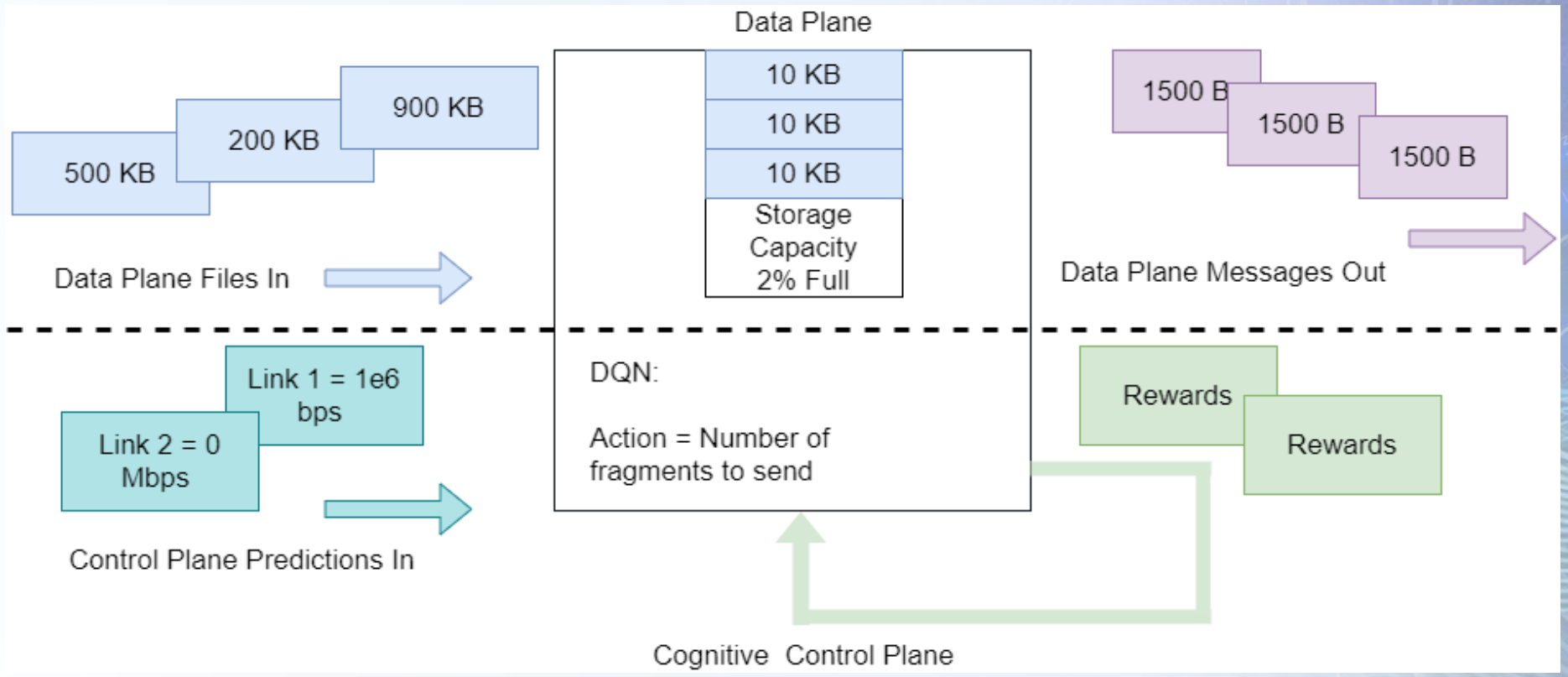
Algorithm	Application	Network Scenario	Learning Type/ Location	Learning Architecture	Description	Network Layer
Spiking Neural Network /CSG	Routing, LTP Segment Size	Deterministic	Online, in flight	Distributed	Neuromorphic learning element	Bundle Layer
Temporospatial SDN (TS-SDN)	Routing, Antenna Pointing	Deterministic	Orbital calculations and antenna modeling done on ground or central node	Centralized	Add time-dynamic modeling of network elements to SDN control plane	SDN Control Plane/ Application Layer
Q-Routing	Routing	Opportunistic	Online (RL), in flight	Distributed	Basic Reinforcement Learning, attempt to minimize delay/ number of hops/shortest path	Bundle Layer/Application Layer
K-means Clustering	Routing/ Data Analysis/ Network Management	Deterministic, Opportunistic	Offline, Ground or central node	Centralized	Discover features within the data	Clustering can be based on node location
Autoencoder	Routing	Opportunistic	Offline, Ground or central node	Develop model on central node, distribute to nodes	Classification of nodes, reduce need for feature engineering	Bundle Layer
Random Forest/Ensemble Approaches	Routing	Opportunistic	Offline, Ground or central node	Develop model on centralized node, distribute to nodes	Ensemble approach to minimize overfitting and other drawbacks of decision tree and other algorithms (Bayesian learning, etc.)	Bundle Layer
Deep Q Network	Fragmentation/ Storage	Deterministic, Opportunistic	Online, in flight	Distributed	Objective function for RL approximated by NN	Bundle Layer
Advantage Actor Critic	Fragmentation/ Storage	Deterministic, Opportunistic	Online, in flight	Distributed	2 parallel RL agents	Bundle Layer

Microservice Based Cognitive Router

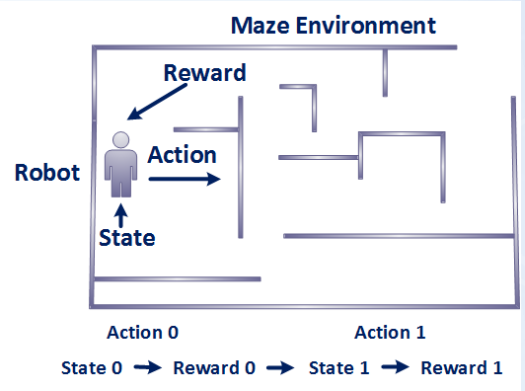


- Envision elements of the cognitive system divided as hierarchical tasks
- Can be implemented as processes, virtual machines, containers etc.
- Multiple layers from physical links, network protocols and the cognitive/control processes
- Input data and feedback/rewards create control loop across layers

Fragmentation and Storage Service



Reinforcement Learning



Action	↑	↓	→	←
Start	0	0	1	0
Idle	0	0	0	0
Correct Path	0	0	0	0
Wrong Path	0	0	0	0
End	0	0	0	0

- Q-Learning
- Agent, Environment, State, Action, Rewards
- Optimal action-value function is estimated by iterative updates of Bellman equation

$$Q_{i+1}(s,a) = \mathbb{E}[r + \gamma max_{a'} Q_i(s',a')|s,a]$$

- "Reinforcement Learning: An Introduction", Sutton and Barto, 1998
- This might take a long time to actually converge
- Needs discrete, finite state and action spaces
- Can estimate with a linear function approximator, non-linear function or neural network
- Q Network

Deep Q-Learning

- Try to replace Q-table with a model (Neural Network)
- In practice can be unstable and divergent
- Family of algorithms with lots of additional techniques to attempt to improve on the basic idea
- Double Q-Learning, dueling networks, prioritized experience replay
- Bellman equation gets modified to use with neural network in loss function
- Minimize sequence of loss functions $\mathcal{L}(\theta)$ where θ is the NN weights
- $Y(s,a,r,s') = r + \gamma \max_{a'} Q_{\theta^-}(s',a')$
- $\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim U(D)}[(Y(s,a,r,s') - Q_\theta(s,a))^2]$
- U(D) is uniform distribution over replay memory D
- “Playing Atari with Deep Reinforcement Learning”, DeepMind Technologies, 2013

Actor-Critic

- The experience replay used by DQN stores the agent's data in memory to be batched or randomly sampled from different time steps
- Requires more memory and computation for each real interaction
- Instead asynchronous advantage actor-critic (A3C) asynchronously executes multiple agents in parallel on multiple instances of the environment
- Doesn't require specialized hardware (GPUs), can be easily run on standard multi-core CPU
- Each actor-learner runs in its own thread
- Multiple actor-learners can explore different parts of the environment and use different exploration policies
- "Asynchronous Methods for Deep Reinforcement Learning", Google DeepMind, 2016

Advantage Actor Critic (A2C)

Critic estimates the value function

- Action-value (Q value) or state-value (V value)

Actor updates the policy distribution in the direction suggested by the critic

Advantage function : Given the state, how much better is a specific action compared to the average action

- $A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t)$

Does not need two neural networks to find this because of the Bellman equation

- $Q(s_t, a_t) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1})]$
- $A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)$

Now only one neural network estimates the V function

Actor finds the policy gradient of $A(s_t, a_t)$

- Updates the probability distribution of actions so that actions with higher expected reward have a higher probability value for an observed state
- $\nabla_\theta J(\theta) \sim \sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t|s_t) A(s_t, a_t)$

Simulated Scenario

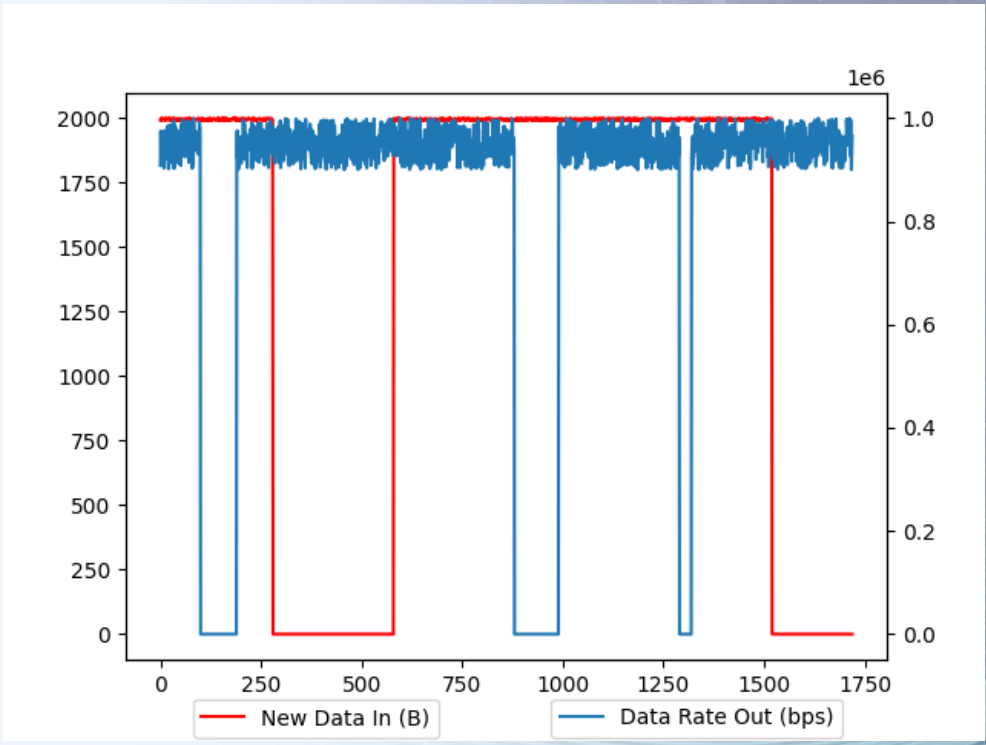
Generate input data

- Number of bytes of new data coming in per time step
- Predicted data rate output per time step (bps)

Calculate data storage per time step

- `Current_stored=current_stored + new_data – (num_to_send*fragment_size)*data_rate_out`

Action will be the number of fragments to send



Parameter	Value
Max Storage	1 GB
Max Data In	2 KB per time step
Max Data Out	1 Mbps
Fragment Size	1500 B

OpenAI Custom Gym Environment

Initialize:

- Action space = number of fragments to send
- Observation space = last 5 time steps of new data in (bytes), predicted data rate out (bps) and current storage percent full

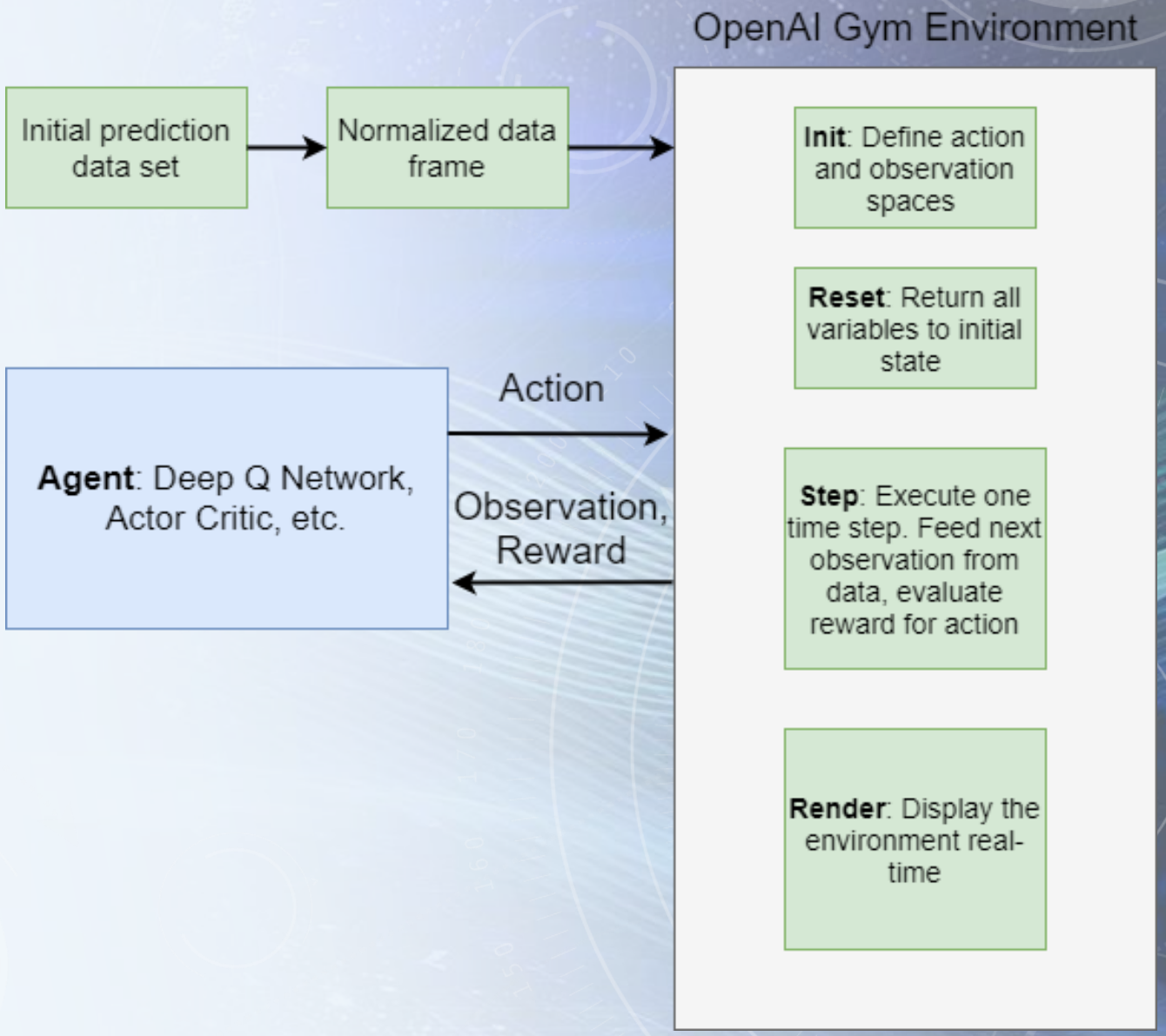
Step: Get the next observation, take action, calculate new storage capacity, get reward

Rewards for correct behavior:

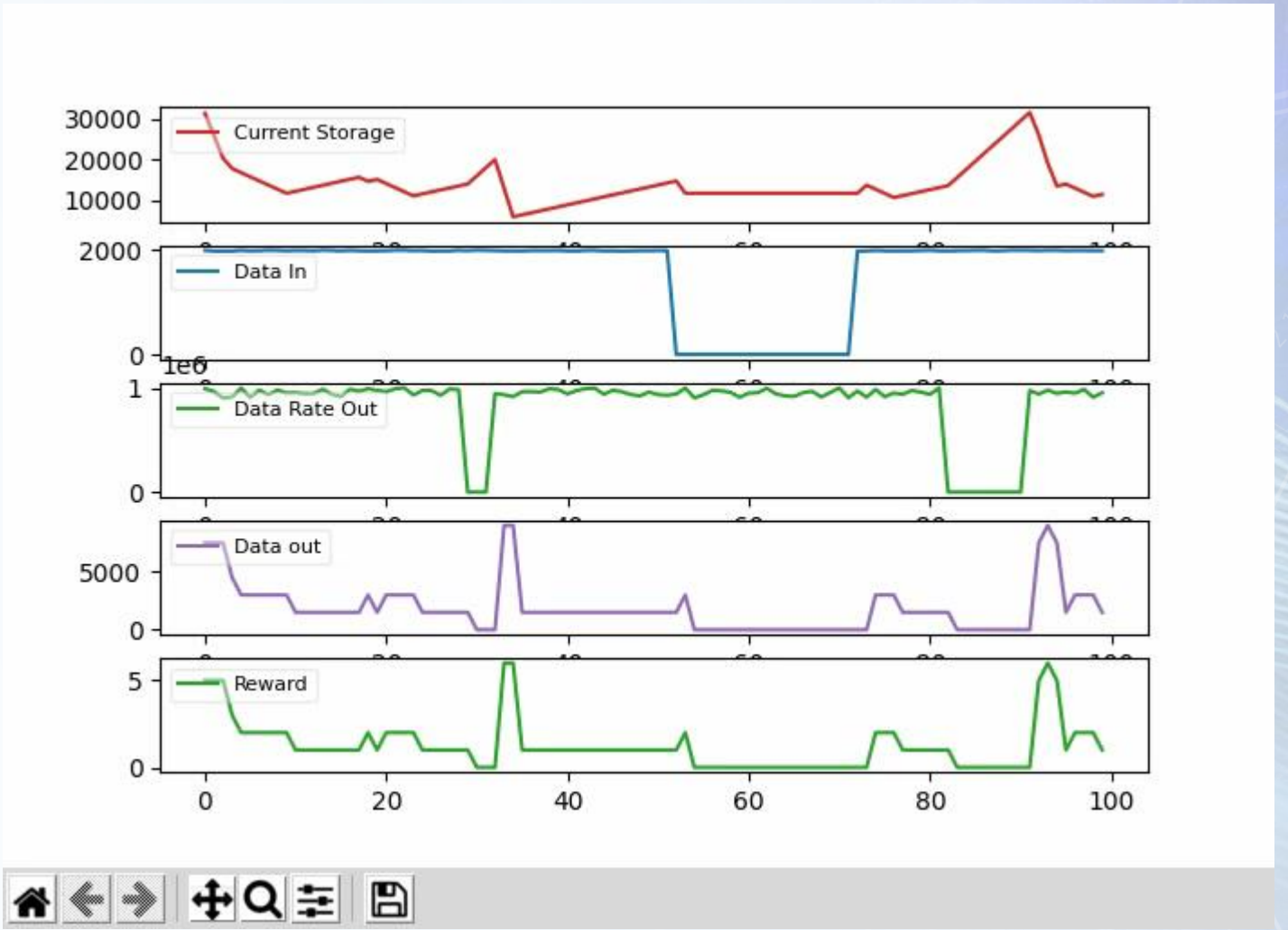
- Data rate = 0 and action = 0, reward = 0
- Action >0, total bytes to send <= max possible to send based on data rate and <= total stored bytes, reward = number to send

Rewards for incorrect behavior:

- Total stored bytes < total bytes to send , reward = -(max fragments to send)
- Data rate = 0 and action > 0 , reward = -(max fragments to send)
- Else reward = 0



OpenAI Gym DTN Environment for Reinforcement Learning



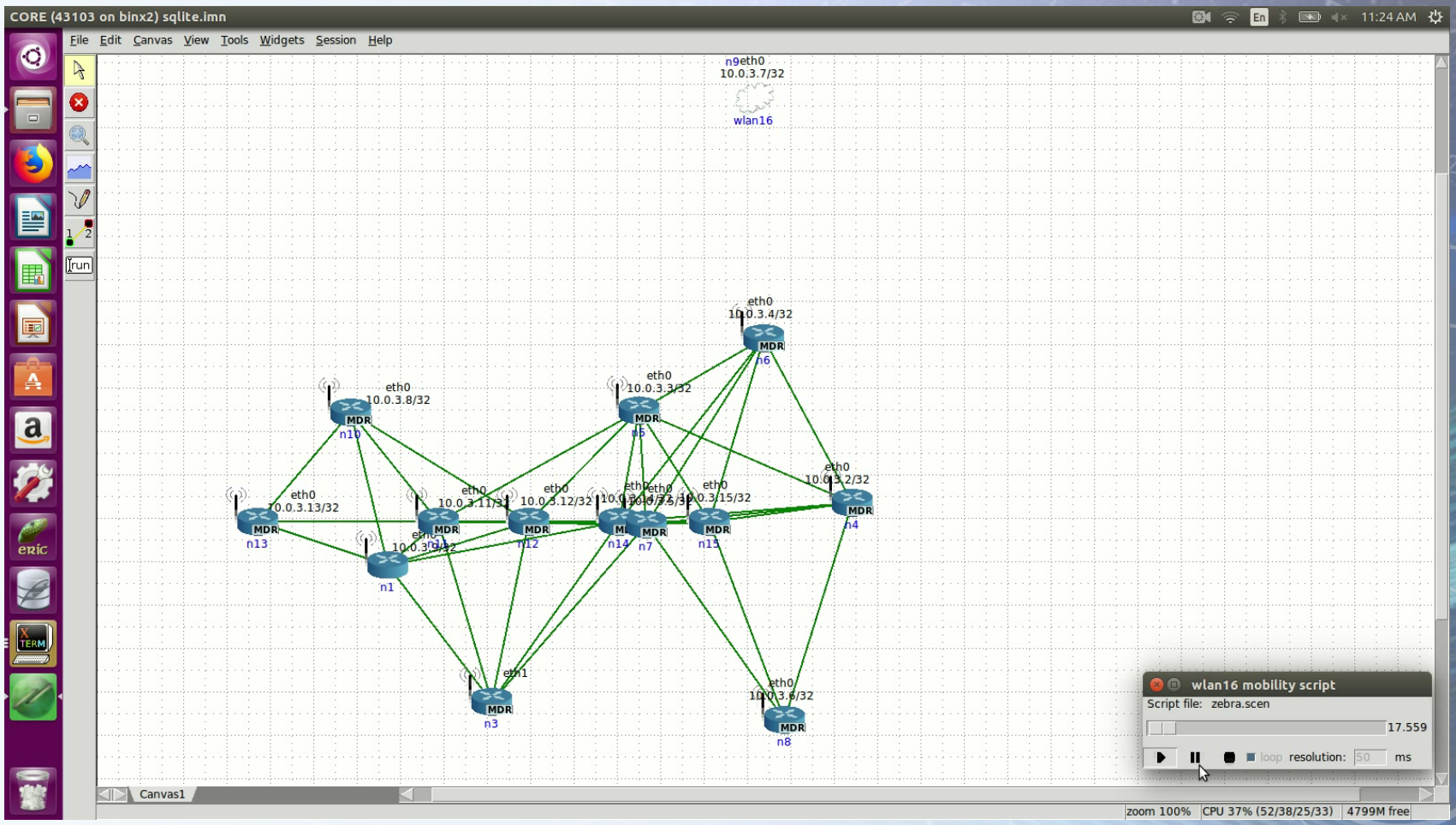
Deep Reinforcement Learning

Algorithm	Training Time (s)	~ CPU Usage %	~ Memory Usage
DQN	52.968	30	230 MiB
A2C	20.127	20	216 MiB

- Actor-critic claims to give a reduction in training time that is roughly linear in the number of parallel actor-learners
- 2 Actor-learners so training time is cut down by half
- Memory and CPU slightly less but this could be more dramatic for a bigger dataset
- Actor-critic exhibited more stable behavior
- Actor-critic fits into microservice concept of separate parallel processes which perform simplified tasks = better performance

DTN Emulation with CORE

- Linux containers used to emulate DTN nodes
- Developed routing algorithms based on Q-routing, common classification algorithms: Decision Tree, Random Forest, Autoencoder
- Configurable node mobility models : Random walk, GPS traces



Future Work

- Evaluate high performance commercial systems
  - IBM Watson Machine Learning Accelerator
  - Amazon Web Services SageMaker
- Applications and test cases
  - Embedded systems/IoT
  - Large-scale data science
- Space communication data sets
- Ground test bed
- Flight-like software development