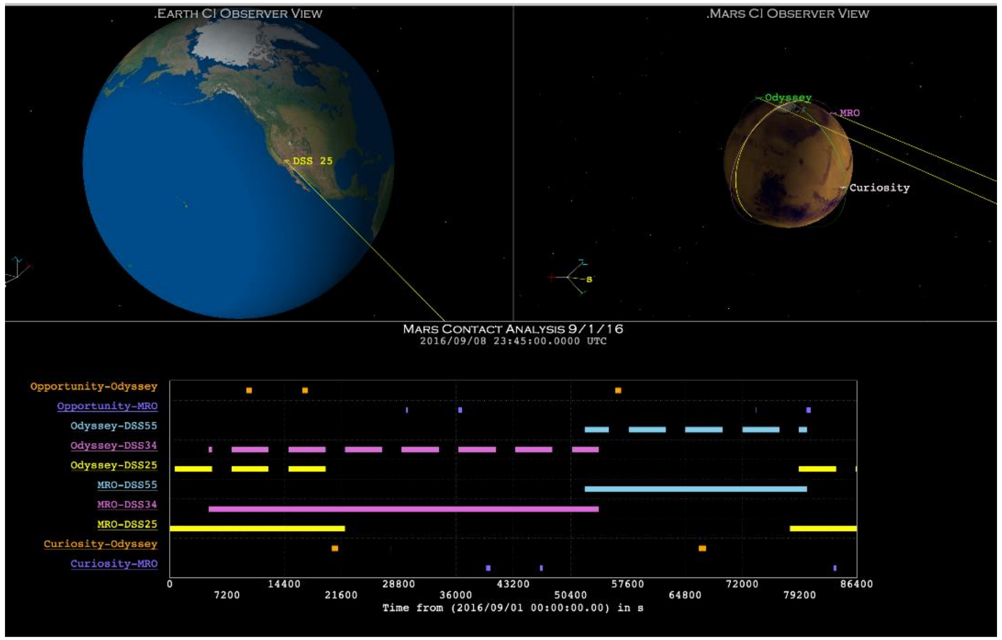
Evaluation of Classifier Complexity for Delay Tolerant Network Routing

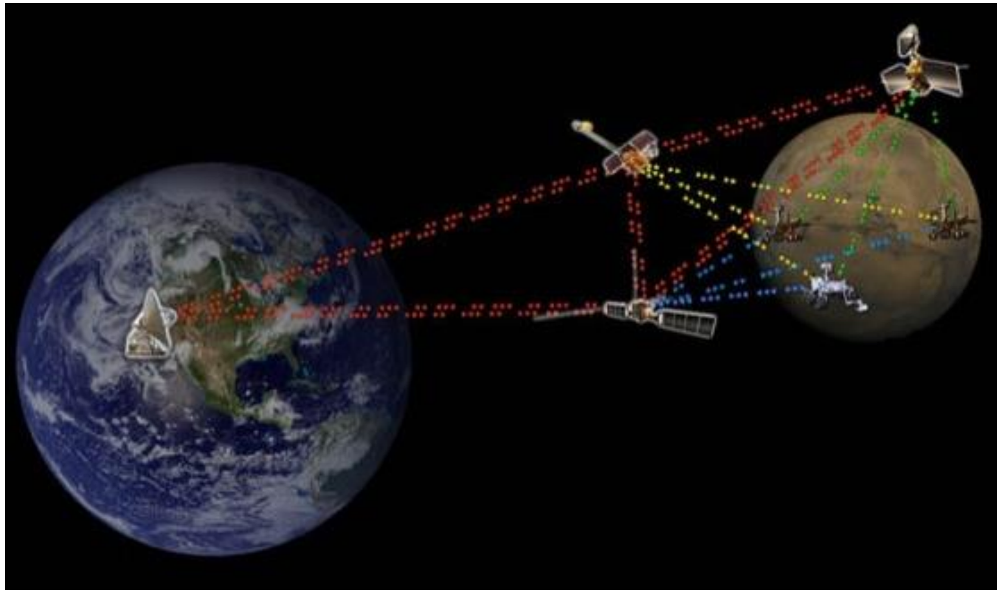
Rachel Dudukovich and Gilbert Clark
NASA Glenn Research Center

Christos Papachristou
Case Western Reserve University

Introduction

- Develop intelligent routing method for delay tolerant networks (DTN)
- Reduce number of bundles (protocol data unit) replicated in Epidemic style routing
- Opportunistic routing scenario versus deterministic
- Evaluate classifiers of varying complexity for delivery prediction
- Forward bundles to nodes with greatest likelihood of delivering message





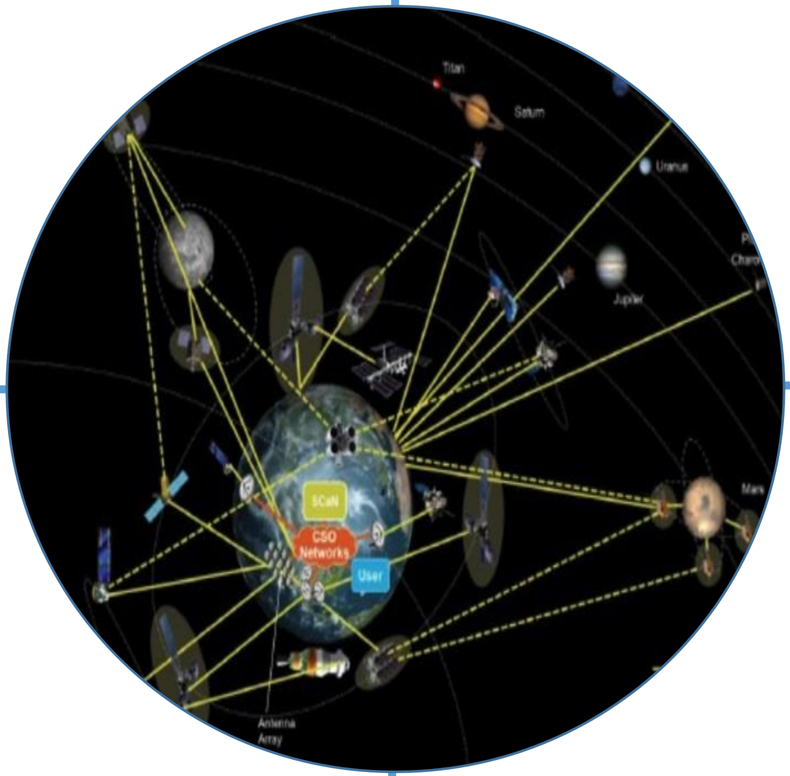
Motivation

Software Defined Networking

- Rule-based switches
- Control and data plane separation

Delay Tolerant Networking

- Common protocol layer
- Storage for long term disruptions



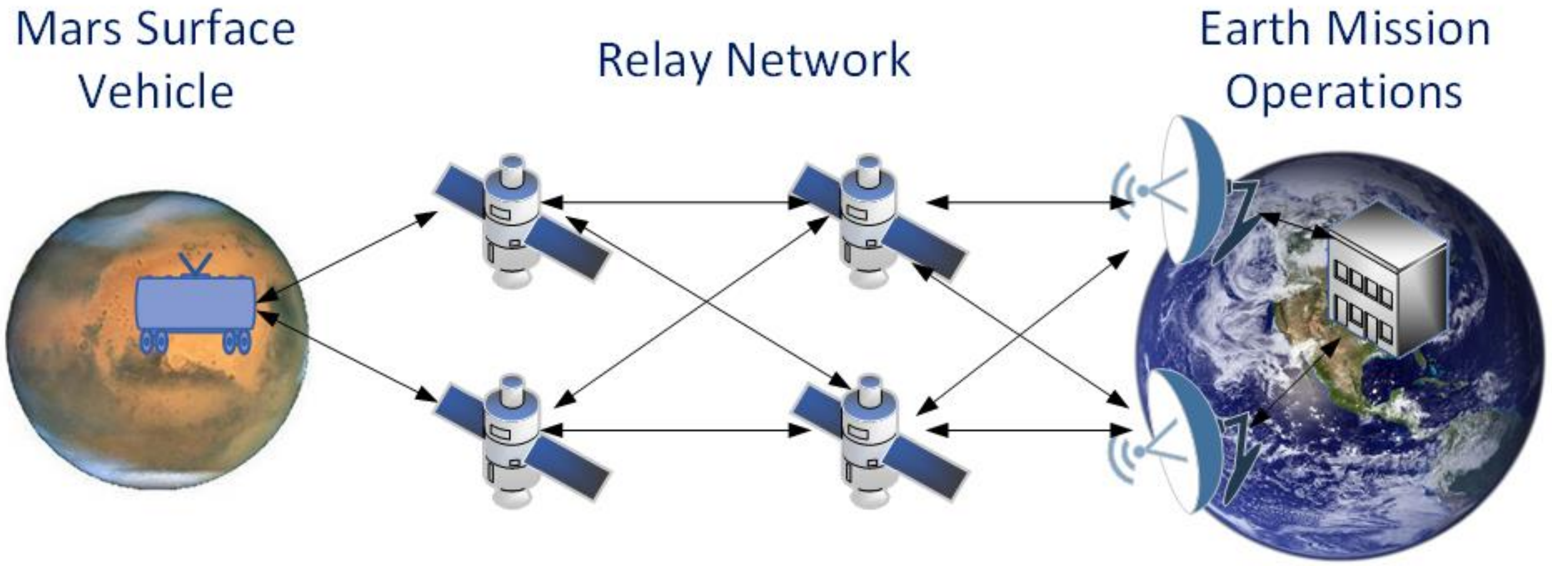
- Cognitive Networks
- Data driven approach to quality of service
- Reduce labor

Machine Learning

- Message ferry/relay
- Neighbor discovery
- Opportunistic routing

Mobile ad hoc Networks

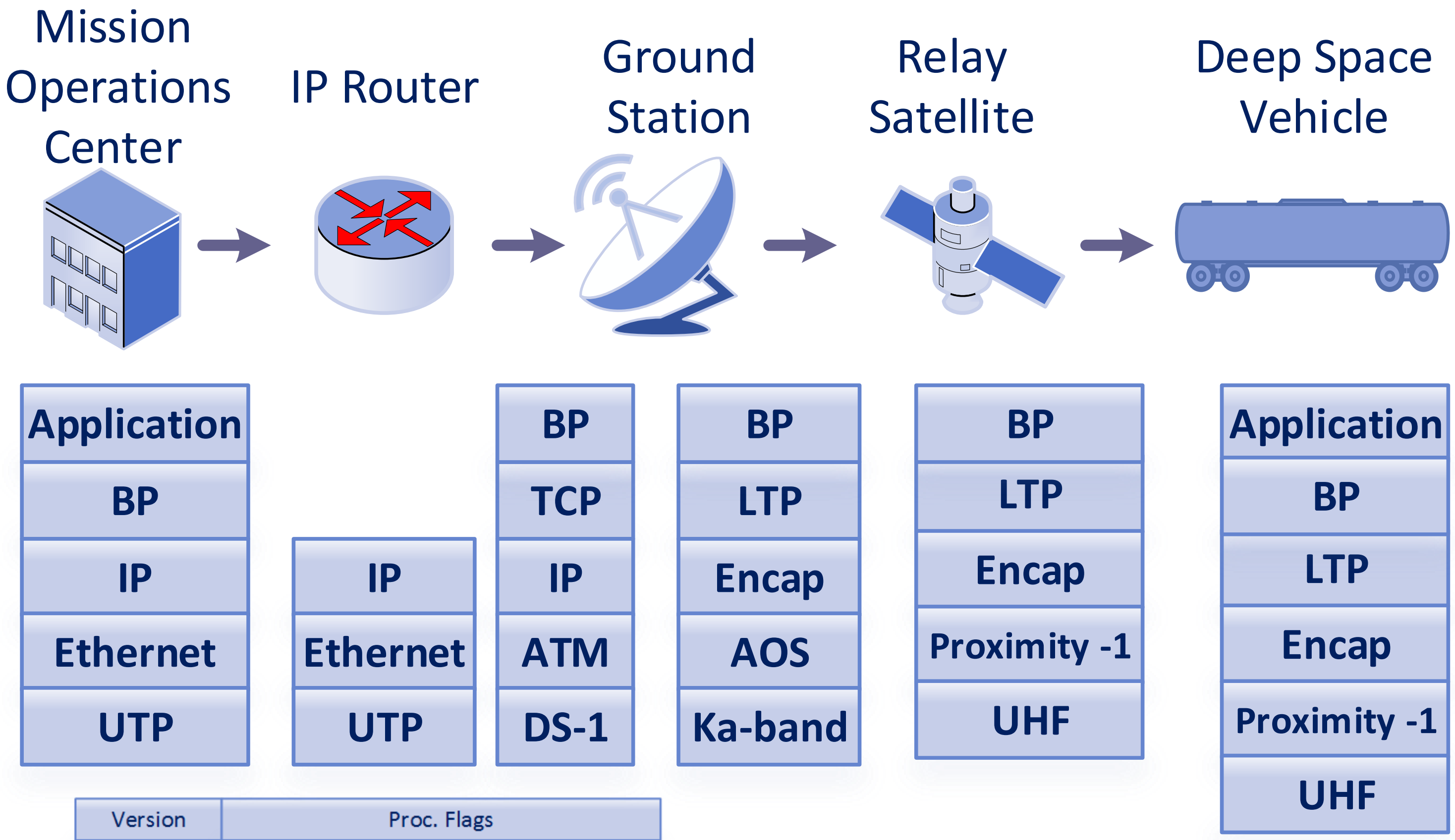
Routing Problem

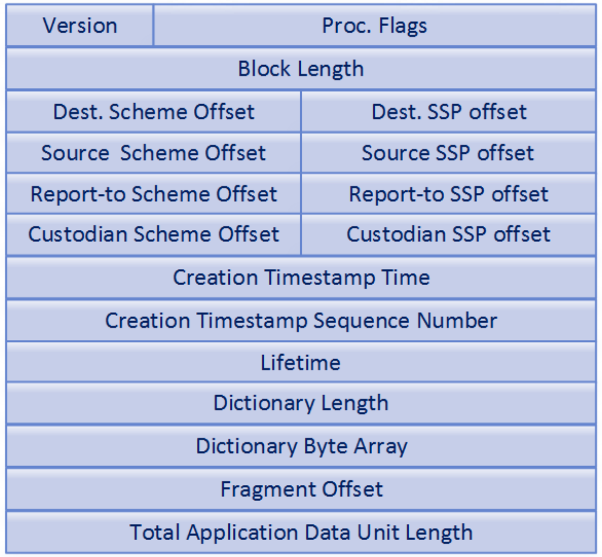


- Find efficient paths through complex network
  - Increasing number of nodes
  - Heterogeneous protocol stack
  - End to end path changing in time
- Opportunistic routing vs. scheduled contacts
- Data-driven approach
  - Delivery history
  - Retransmission attempts

Delay Tolerant Networking

- Architecture and protocols for networks with:
  - Long round trip times
  - Lack of continuous end-to-end path
  - Asymmetric and/or error prone links
  - Heterogeneous protocols
- Bundles are stored until they can be forwarded to a neighbor



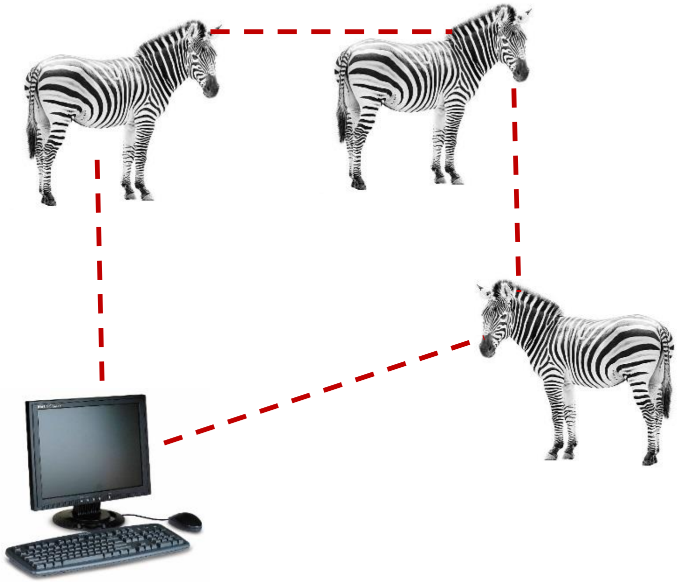


Primary block – processing and routing information for bundle

Payload Block – Contains application data

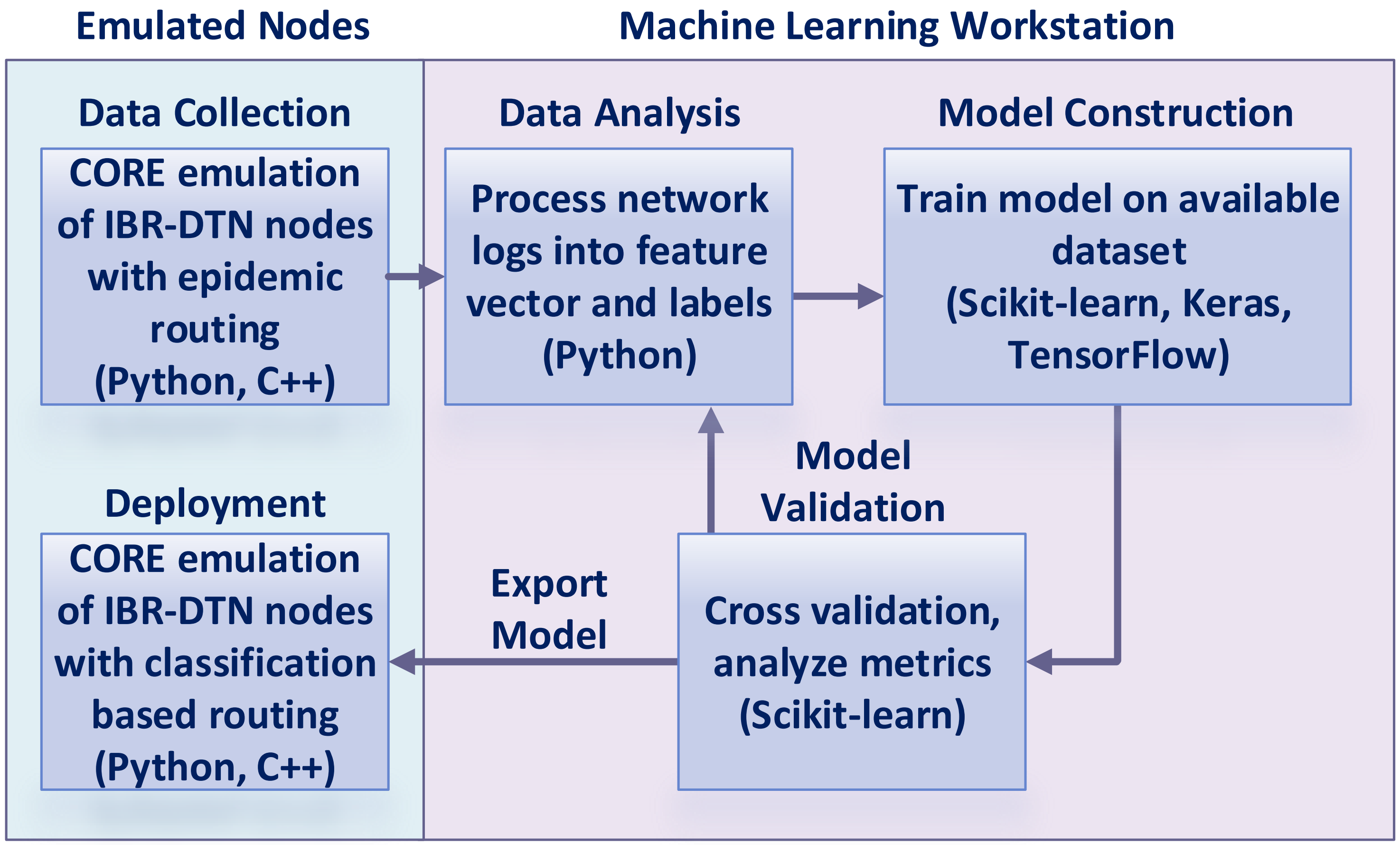
Opportunistic DTN Routing

- Nodes may discover each other
  - Exchange "handshake" information
- Epidemic routing: send any bundle the neighbor doesn't already have
- Best probability of delivery
  - Creates duplicate bundles in the network
  - Wastes transmission opportunities, data storage, processing
- How to determine best neighbors to get data to final destination?



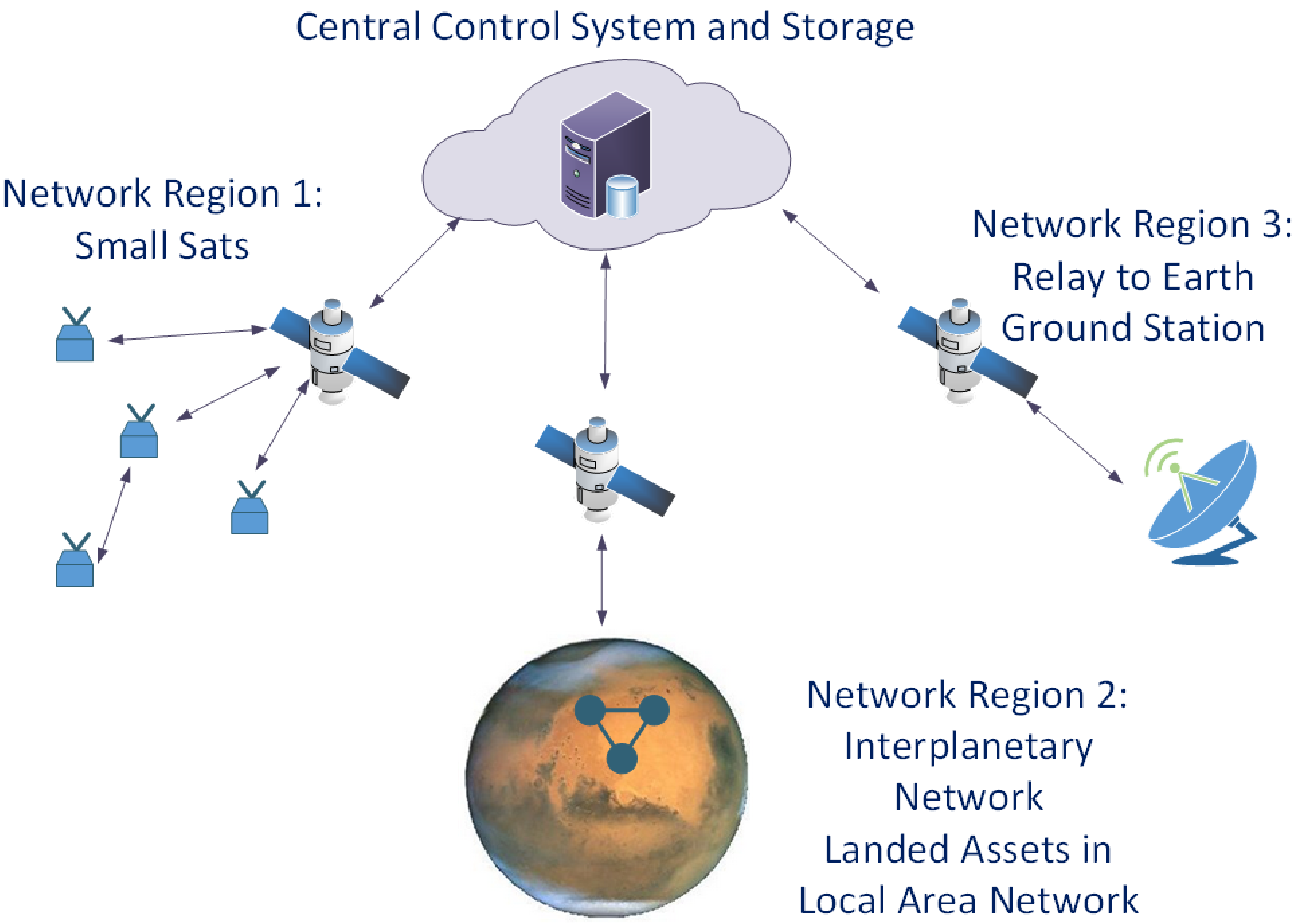
Approach

- Container based emulation
  - Run DTN protocols to generate data set and test routing module
- Feature vector:
  - Source id
  - Destination id
  - Time
  - Forwarded node
  - Delivery Success
- Derived from node system logs



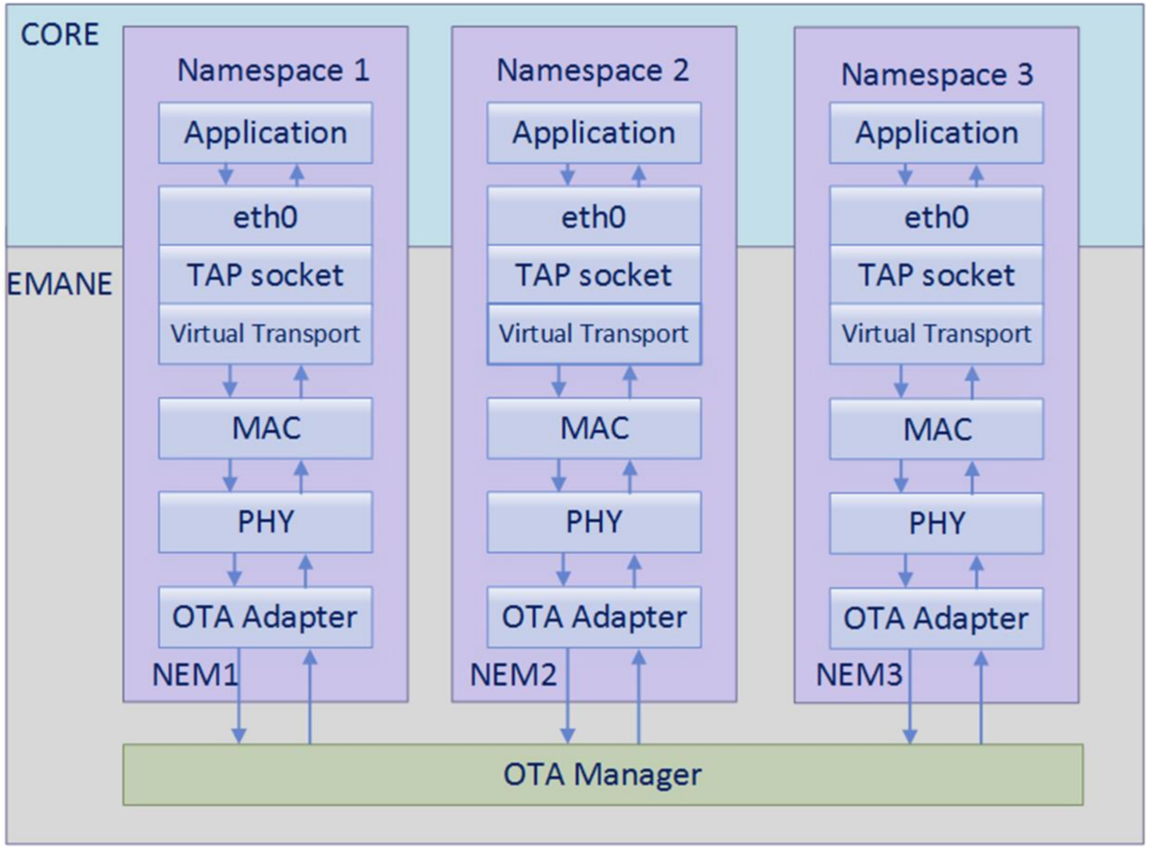
Network Architecture

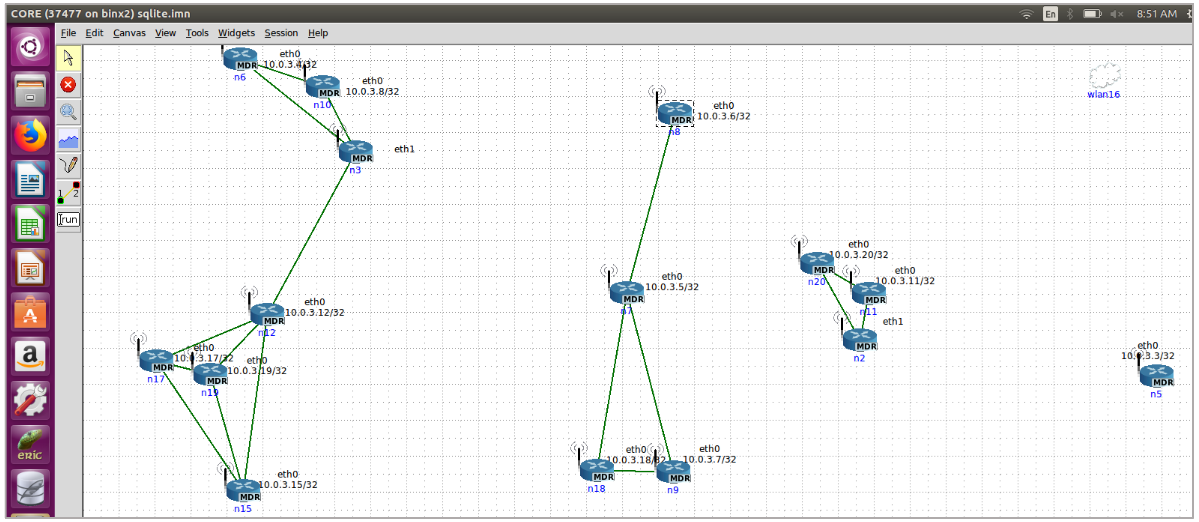
- Nodes save local network logs
- Forward to central processing node
- Data analysis (train and test) perform at central node
- Distribute model to nodes
- Allows for extensive computing resources not available to many flight computers



Emulation Environment

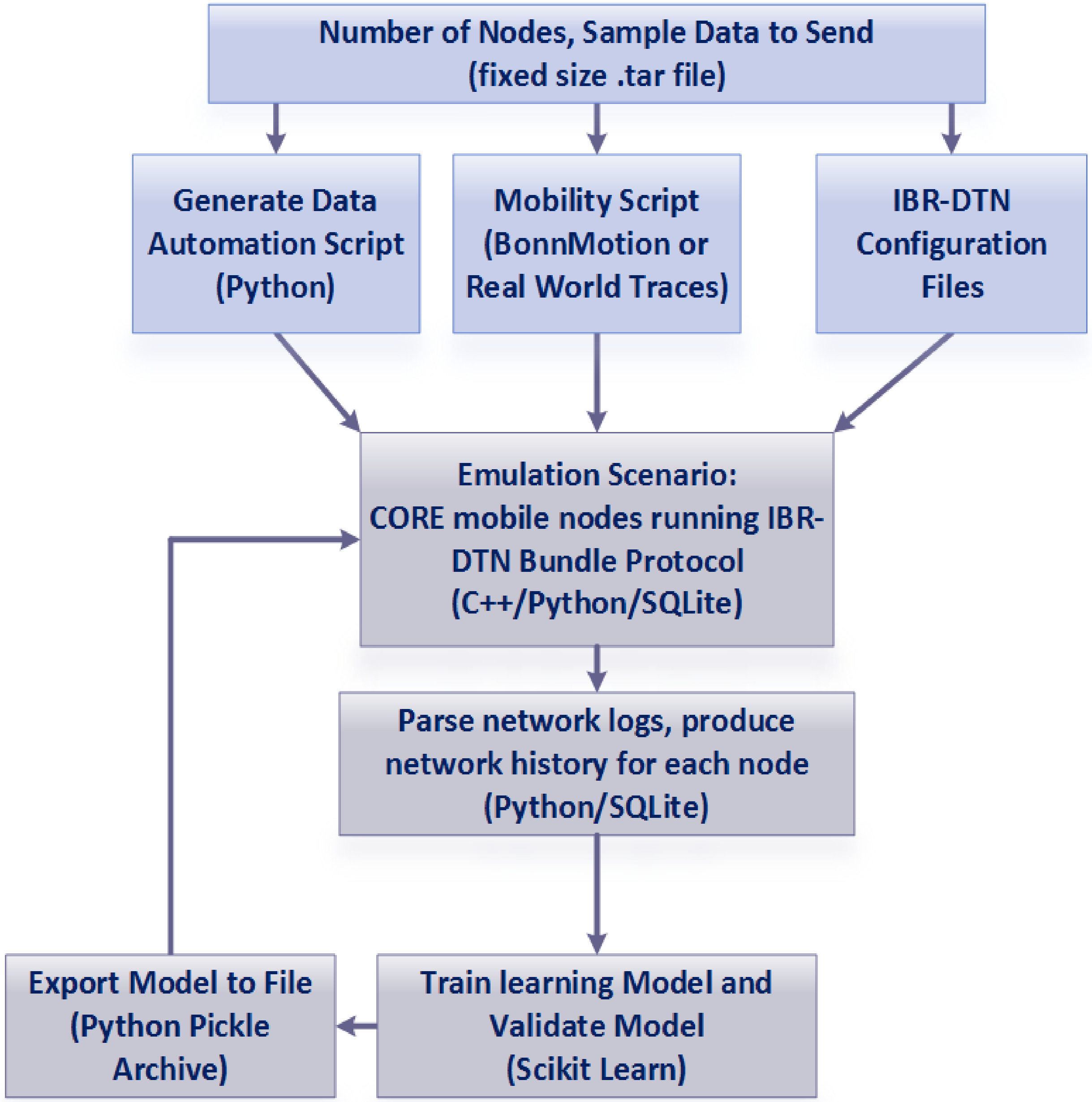
- Emulate multiple nodes on a single host
- All nodes share same OS, network stack and resources are isolated
- Uses Linux containers (LXC) and Ethernet bridging
- Emulates network layers 3 and above (network, transport, session, application)





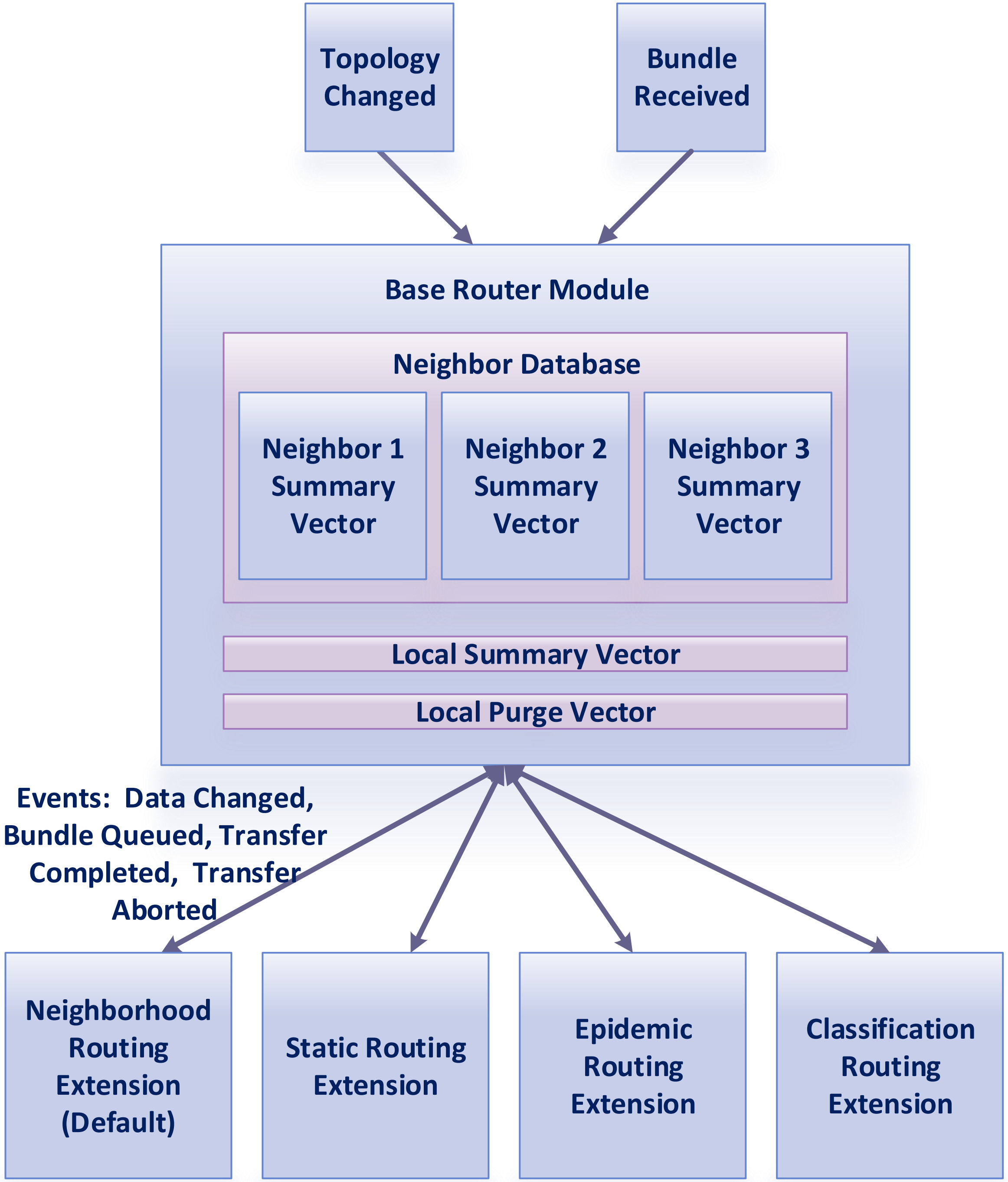
Tool Chain

- Configure CORE emulated nodes
  - IBR-DTN configuration
  - Automate node motion
  - Automate transfer of random messages
- Compile network logs
  - Format into feature vector and labels
- Train and validate model
- Export model



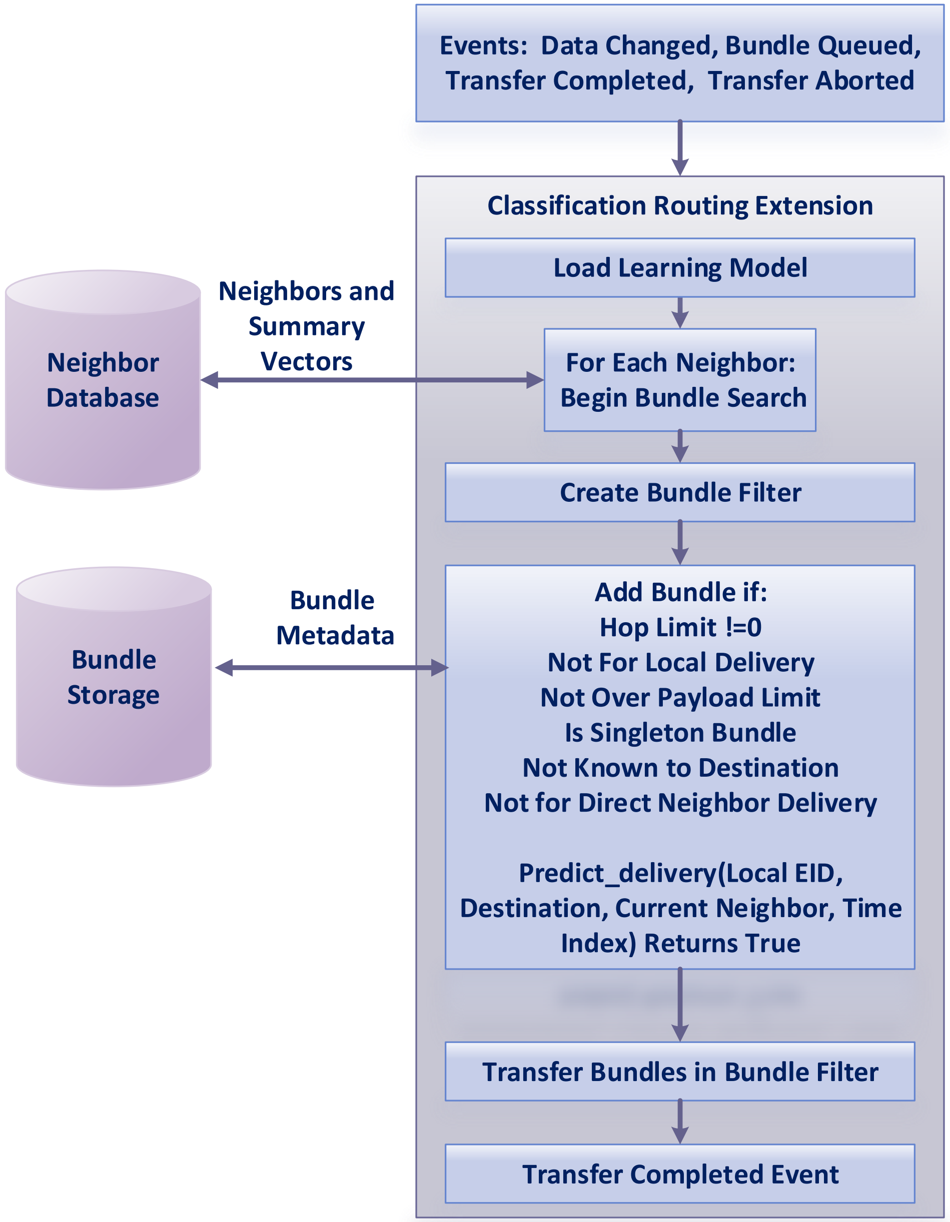
Base Router

- Manages database of neighbors and known bundles
- List of local bundles and recently purged bundles
- Routing extensions implement routing decisions for particular algorithm



Classification Routing Extension

- Load learning model
- Search for possible neighbors/bundles
- Check epidemic criteria
  - Do not replicate bundles known to neighbor
- Predict delivery



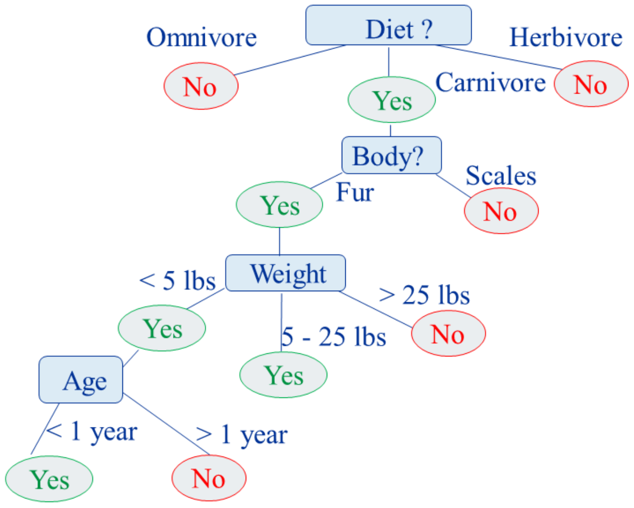
Decision Tree & Random Forest

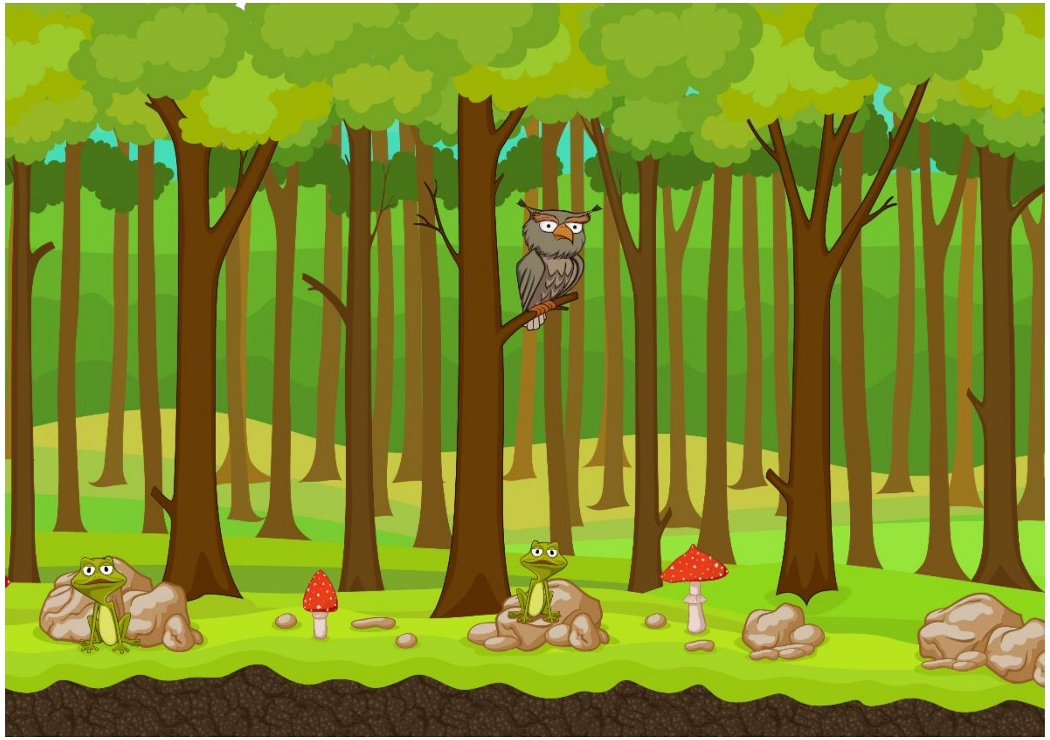
- Decision Tree

$$Entropy(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in Val(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

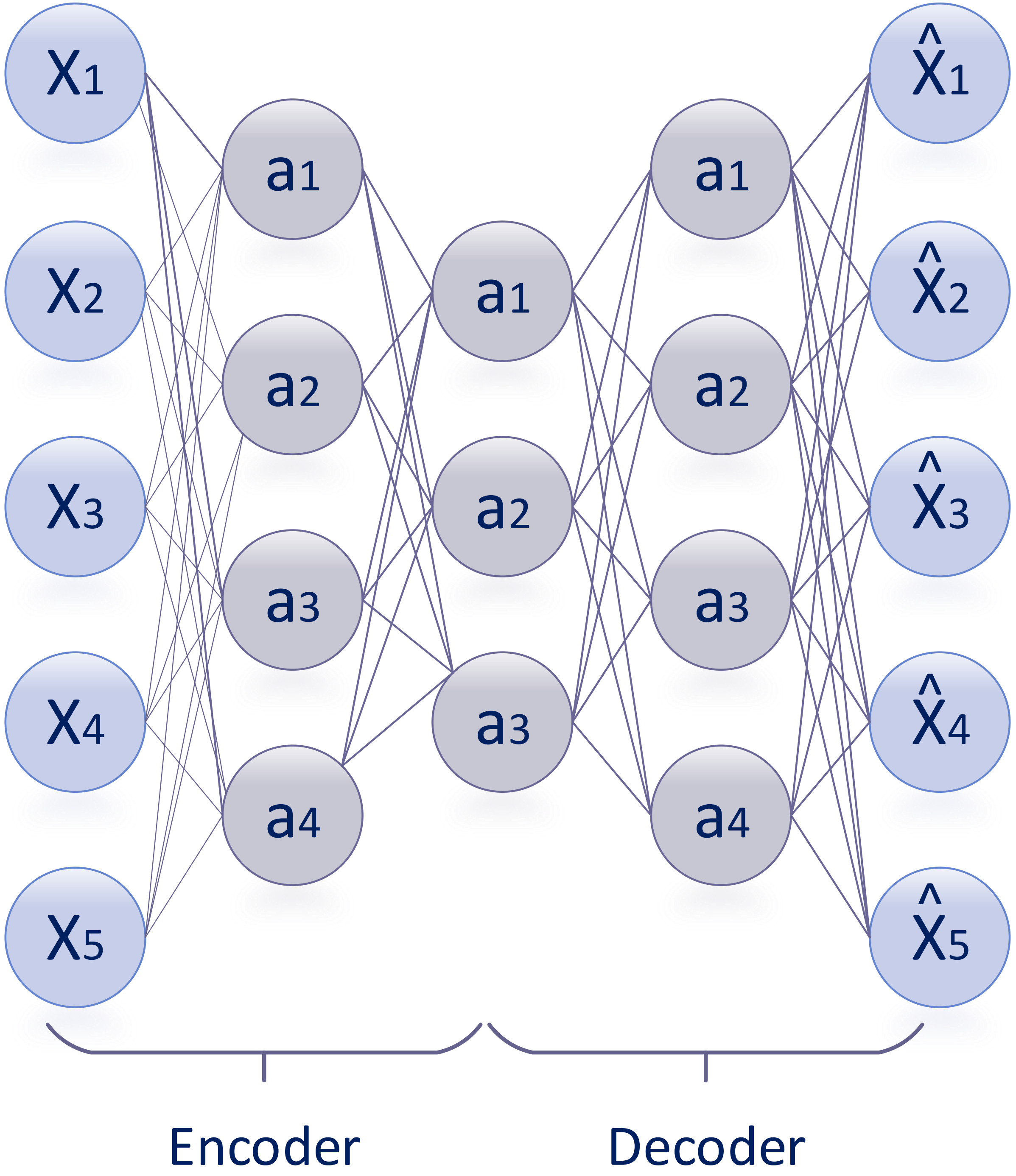
- *O(n×m log m)* for n attributes, m training samples
- Random Forest
  - Ensemble of decision trees
  - Each tree based on subset of data
  - Take the most common answer
  - Controls overfitting
- *O(M(n×m log m))* for M trees, n attributes, m training samples





Autoencoder

- Neural network based
- Attempt to reconstruct input dataset by developing encoding and decoding functions that minimize the error between the input data and reconstructed data
- Complexity ~ *O((N+F)DI)* where N is number of nodes, D is size of hidden layer, I is number of iterations, F is dimension of feature vector
- Reduce amount of manual feature engineering required



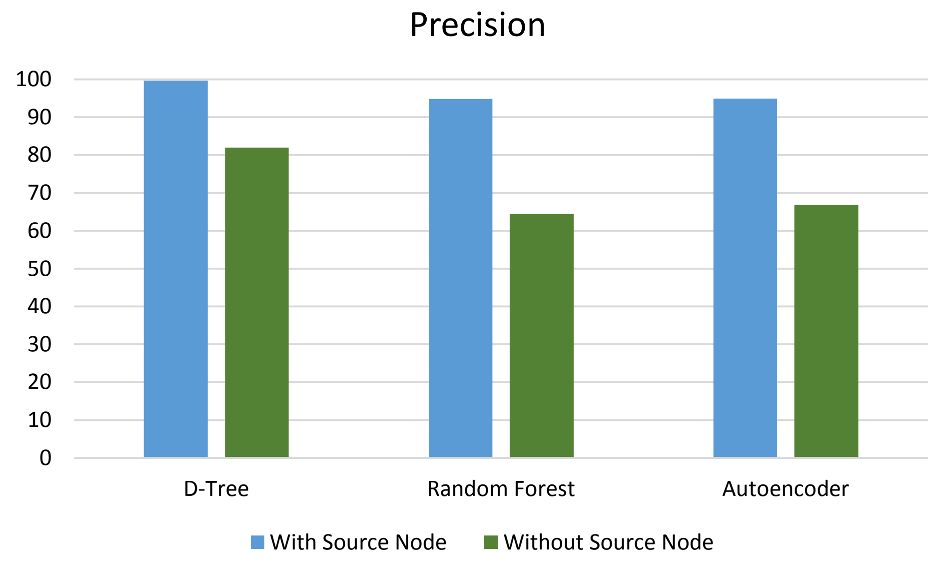
Training Times and Sample Size

- Training set 37,984 samples
- Validation set 16,280 samples

Model Training Times	
Algorithm	Training Time (s)
Decision Tree	0.05
Random Forest (100 trees)	0.77
Autoencoder (50 iterations)	323.36

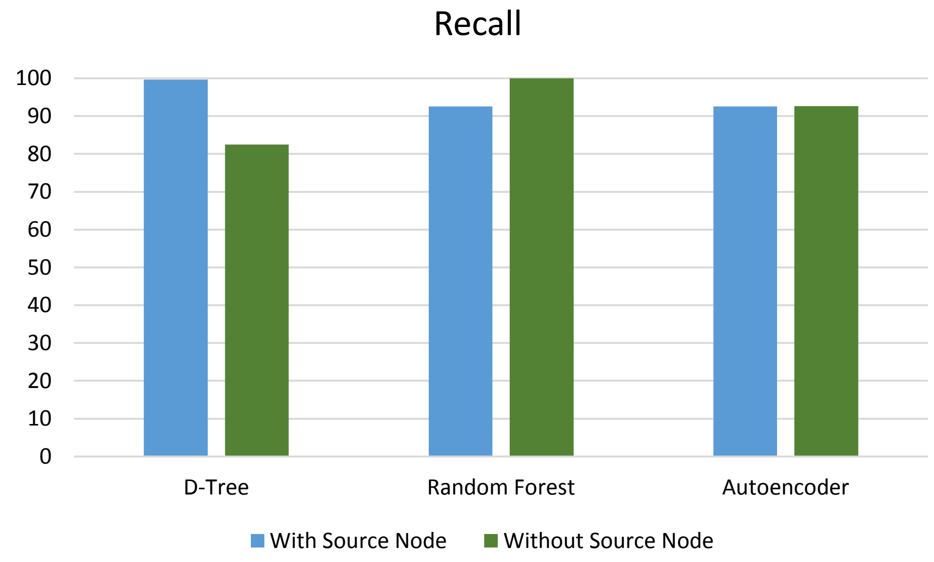
- System specs:
  - Lambda stack for Ubuntu 18.04
  - NVIDIA RTX 2070
  - Intel Core i7-8750H
  - 16 GB DDR4 RAM

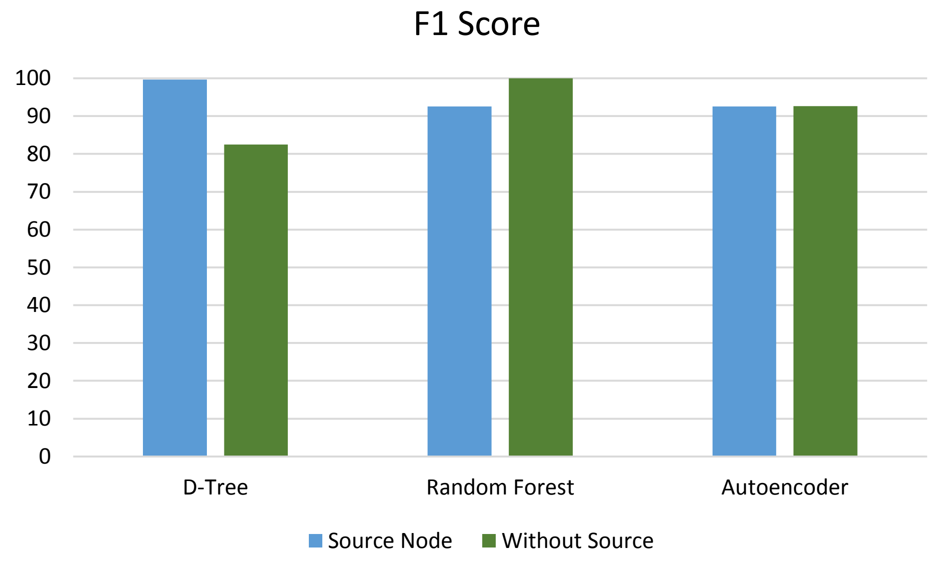
Machine Learning Metrics



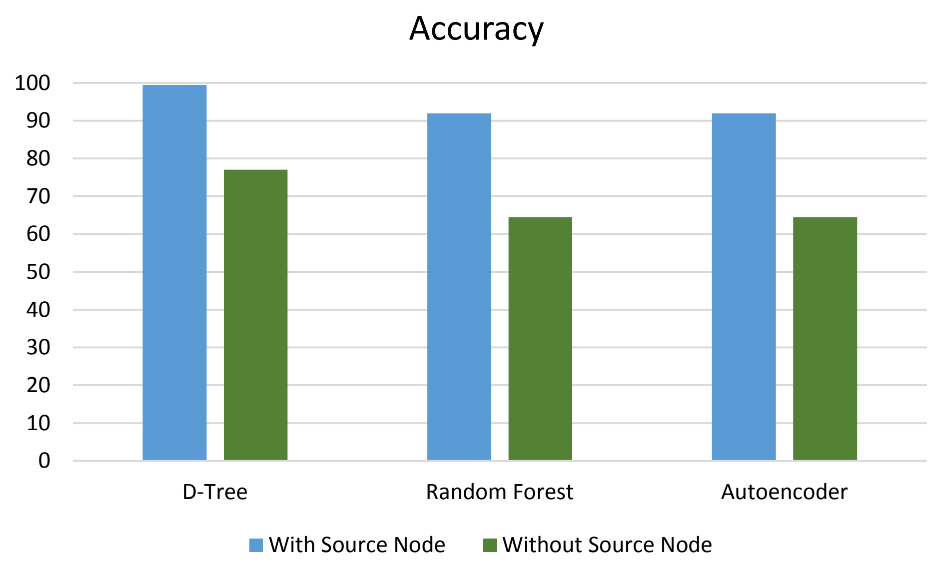
- Precision= $\frac{t_p}{t_p+f_p}$
- Recall= $\frac{t_p}{t_p+f_n}$
- F1 score= $\frac{2(p\times r)}{p+r}$

t_p= # true positives, f_p = # false positives
f_n = # false negatives



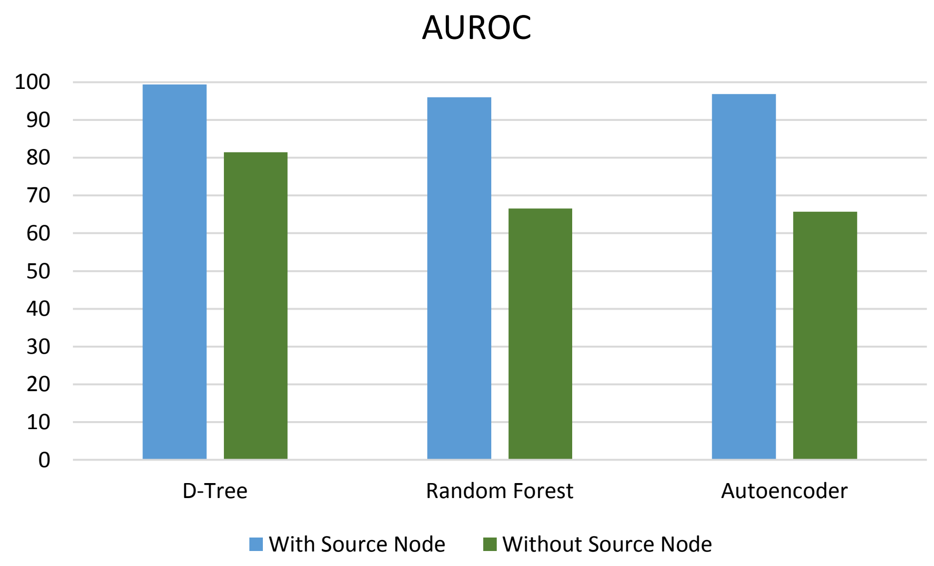


Machine Learning Metrics

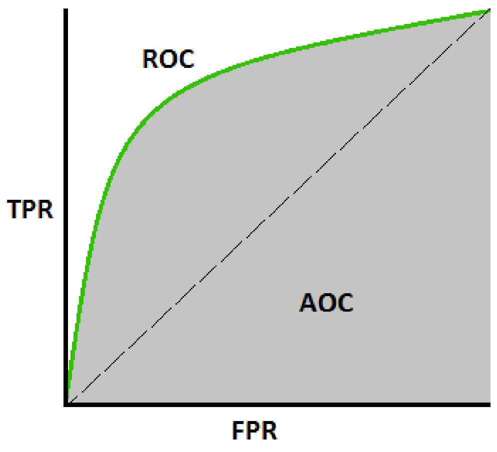


Accuracy=$\frac{1}{n}\sum_{i=0}^{n-1}[\![\hat{y}_i = y_i]\!]$

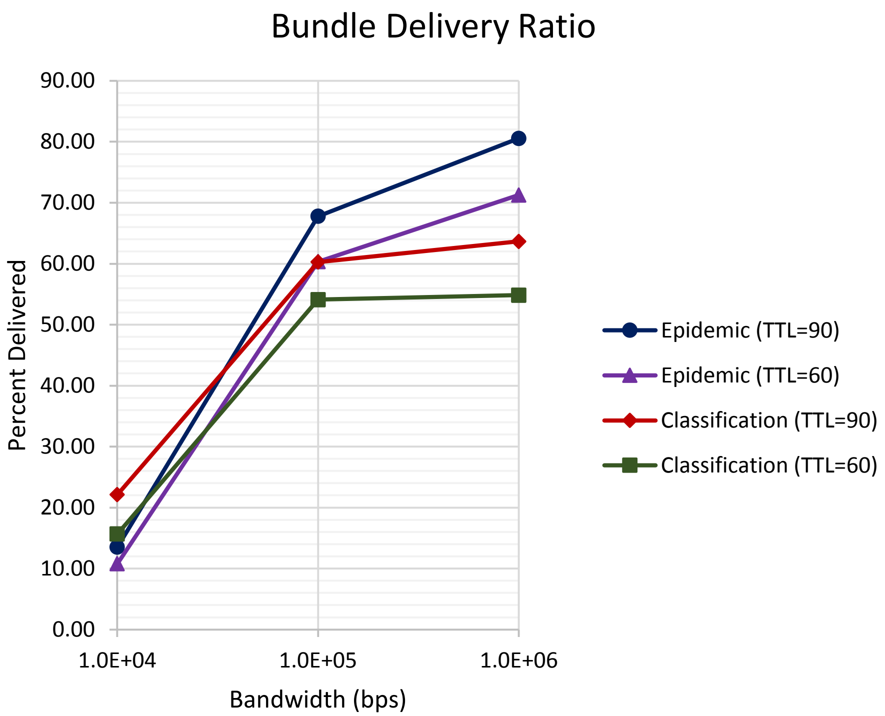
$\hat{y}_i$ =predicted value, y_i= true value
n= number of samples

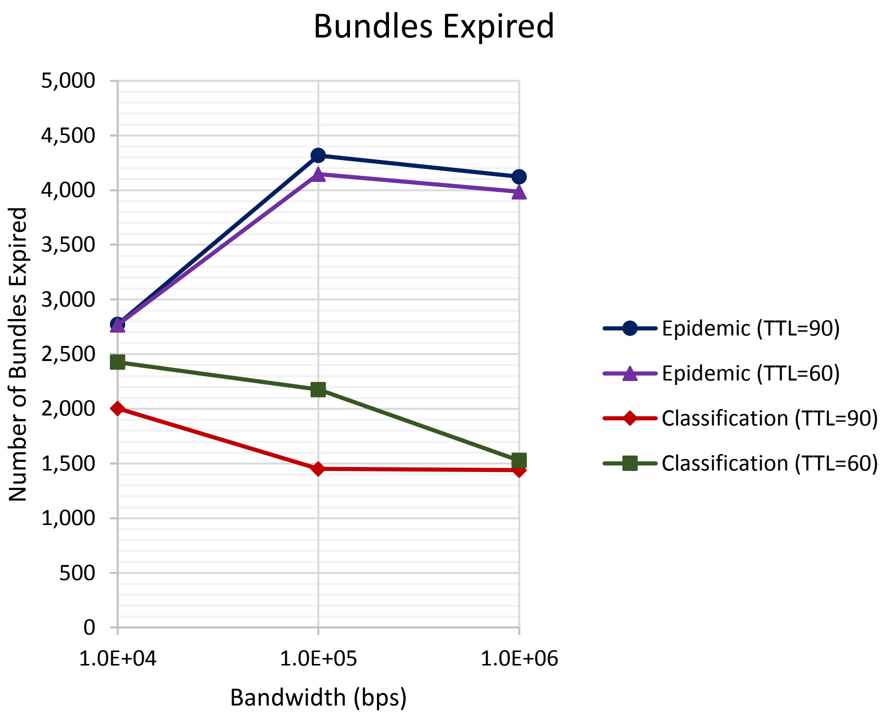


Area Under the Receiver Operating Characteristic Curve



Routing Metrics





Bundle delivery ratio $= \frac{bundles\ delivered}{bundles\ created}$

Conclusion and Future Work

- Decision tree performed reasonably well for simple dataset
- Future work to expand feature vector with buffer capacity, node location, retransmission attempts
- Additional methods such as reinforcement learning
- Apply a variety of techniques to specific aspect of routing problem
- Software defined networking architecture
  - Clear delineation between control and data plane