Project Notes **Spring 2018**

RL+MANETs

Professor Peter Stone *Students: Justin Lewis, Jacob Winick*

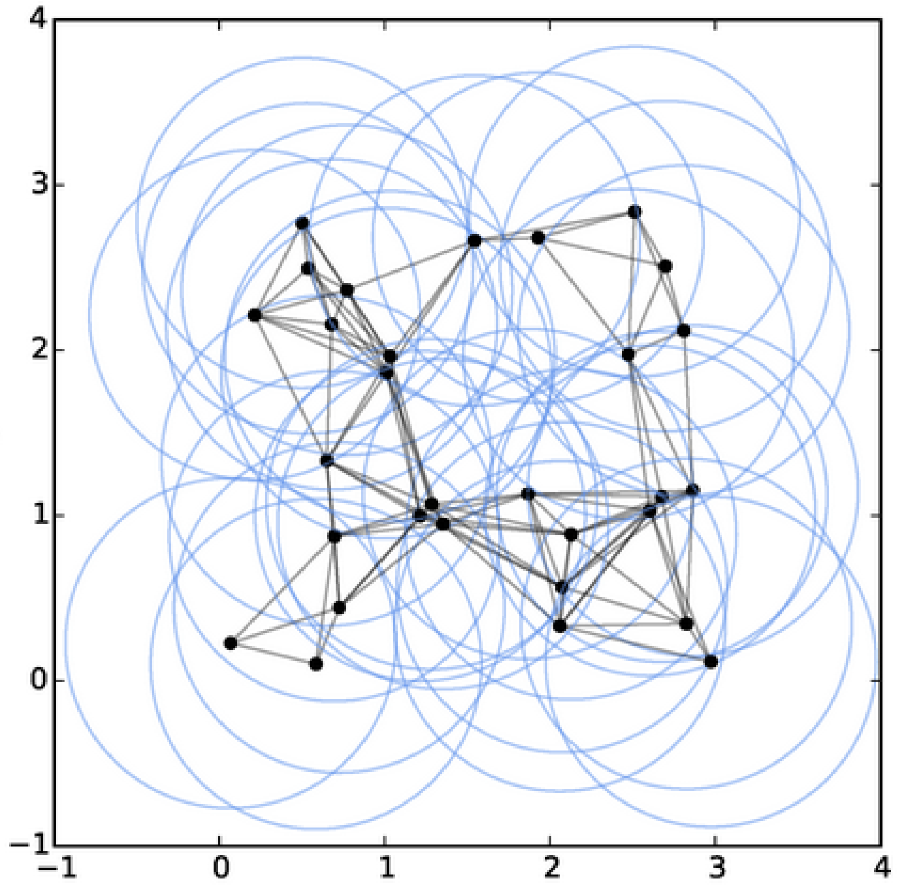
1 Research Problem Summary:

- *Overall goal:* apply Reinforcement Learning to packet route optimization for a class of Wireless Ad Hoc Network known as Mobilized Ad Hoc Networks (MANETs)
- Route optimization for MANETs is difficult
  - network topology is dynamic
  - generally decentralized
- Additional problem difficulties:
  - transmission cost / interference
  - power constraints
  - scalability

2 MANETs:

- decentralized network structures
- nodes are mobile:
  - can move in space
  - may join/leave any time
- *applications*:
  - mobile internet (e.g: project LOON)
  - mobile phone networks / SPANs
  - mobile sensor networks
  - vehicular networks / VANETs
  - military applications
  - smart cities

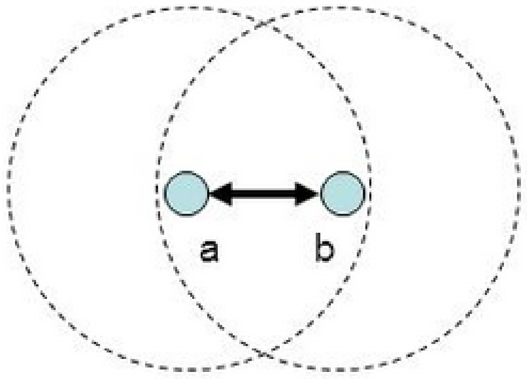
2.0.1 MANET Example:



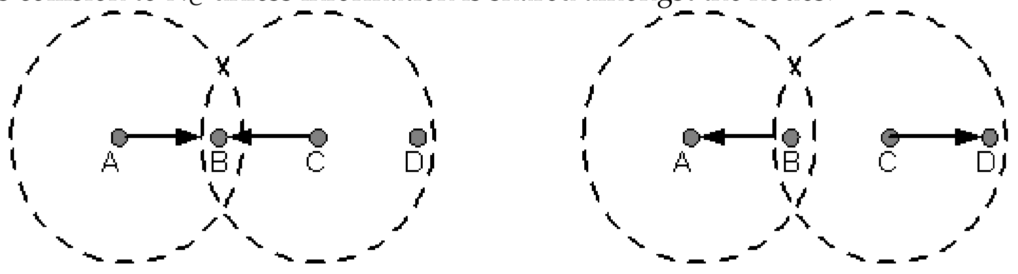
Range of communication demonstrated by surrounding circle

3 Communication Scheme:

Each device uses radio waves to communicate packets to other devices. The broadcasted message is transmitted between nodes over a certain frequency channel. If two devices use the channel simulatenously, a “collision” occurs. This nullifies both broadcasts.



This can cause complex interactions amongst the communication nodes. For example in the *"hidden node problem"* (see below), node N_A can see N_B, but cannot see N_C. If N_A and N_C both communicate to N_B simultaneously, then a collision will occur; furthermore, node N_A cannot attribute this collision to N_C unless information is shared amongst the nodes.



Hidden node problem Exposed node problem

These interference problems coupled with a decentralized control scheme can lead to inefficient routing. A large portion of time can be spent rebroadcasting messages which previously collided.

4 Traditional Solutions (abridged):

1. Reactive routing (AODV):

   - This method is reactive because nodes do not save information regarding the entire network topology. They only maintain connections to local neighbors.
   - Nodes periodically send "hello" messages to gain information about their immediate neighbors.
   - Whenever a node wishes to send a packet through the network, it sends a "route request" (RREQ) to find a path to the destination. This request is passed along the network following a fixed protocol until the destination is found. Then the information is passed back through the network to the sending node.

2. Proactive routing (OLSR):

   - This method is proactive as the nodes maintain a local "image" of the network's full topology.
   - Nodes periodically send "hello" messages to gain information about their immediate neighbors.
   - Additionally, nodes broadcast their local connections to the rest of the network through "flooding". Essentially, the information is spread to every connected node in the network by a fixed policy. "Flooding" can cause unnecessary transmission and cause packet collisions.
   - With a local image of the network, a version of shortest path algorithm is applied to find a suitable path to the destination.

5 Why Reinforcement Learning?:

- The "best" strategy given limited computation time and limited power usage is difficult to derive directly
- RL agents can learn how to best encorporate available information at a node to improve routing (while maintaining constraints such as power usage)
- RL agents can learn how to most effectively cooperate (direct analysis is difficult). Ex: how often and how much information should be shared amongst neighbors.
- The computation time complexity of a DRL solution could potentially be smaller (i.e. faster) than a more straightforward optimization algorithm (latency is important)

6 RL Problem Constraints:

1. Decentralized control
2. Multi-agent learning scheme
3. Full cooperation (protocol enforced)
4. Arbitrary intercommunication (with associated cost)

7 Physical / Environmental Constraints

1. Number of unique frequency channels is variable and will be analyzed to see achievability of specific data rates
2. Range of unique frequency channels is proportional to transmission frequency

8 RL Agent State Space:

- there are a number of potential state types which may prove useful in deriving the best policy. Below is a list of state classes and specifics.

1. Routing Information:
   - Packet source / destination
2. Environment Information:
   - Current estimate of topology of network
   - Strength of network connections
   - Physical location and heading of nodes (possible?)
3. System Information:
   - number packets in local buffer
   - battery state
4. History
   - time of day
   - time elapsed between certain actions

9 RL Agent Action Space:

- similar to the state space, there are a number of distinct actions which can be considered.
- I would imagine that the action space should be structured as a sequence of decisions within time slots (viewed as a time series)

1. Packet Related

    - Forward packet to neighbor (on specific channel)
    - Drop packet
    - Save packet to buffer
    - WAIT (do nothing)

2. Environment related

    - Sense channels (local connections)
    - Measure local node locations (may not be physically feasable)

3. Cooperation related

    - Proactively / reactively share information with nodes in network
    - Proactively / reactively request information from network

10 Reinforcement Signal / Reinforcement Protocol

- the reinforcement signal should be related to the following:

    – Quality of service (QoS)
    – Transmission slowdown / delay
    – Power usage
    – Transmission collisions

- How the reinforcement signal should be exactly formulated will require more considereation.

- The way in which the reinforcement signal is provided to an agent is also an open question.

11 Initial RL Formulation:

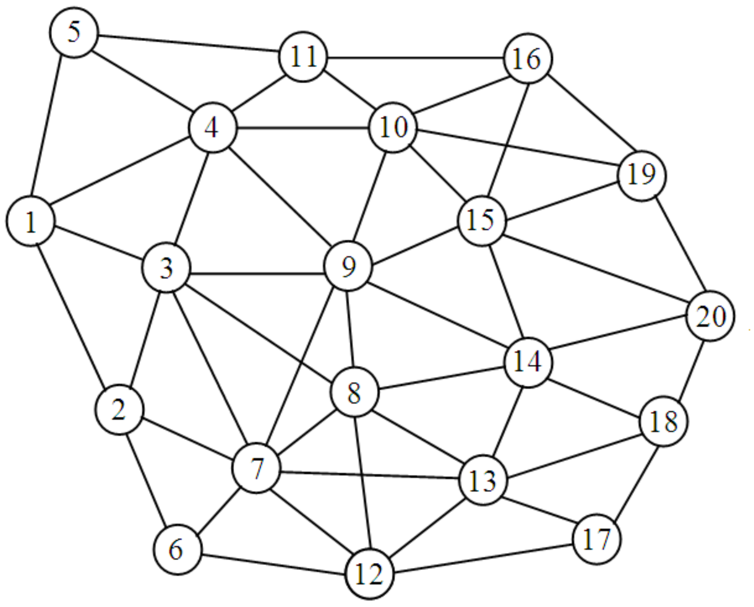
Assume constraints from sections 6 and 7. The following is an initial attempt at formulating the state space, action space, and learning mechanism.

11.1 State Space

Let the state space be broken into two primary sections as described below:

1. Environment state

   Let a node's estimate of the true environment state be represented by an undirected graphical model $G(V, E)$ with vertices V and edges E.

   Each vertex $v \in V$ represents an agent within the assumed collaborative scheme. (Let v_ℓ represent the local agent). Each edge $e(v_i, v_j) \in E$ represents the communication channel connecting two nodes.

   Each vertex $v_i \in \{V / v_\ell\}$ will hold the following information:

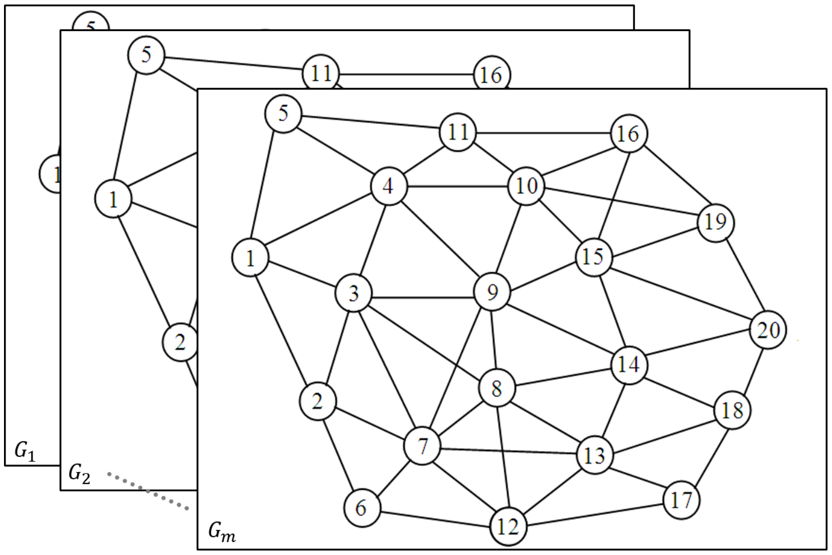
   - node ID
   - flag if v_i is a source or destination of a packet in buffer
   - counter for time elapsed since v_ℓ directly communicated with v_i last (only for neighbors)
   - counter for time elapsed since v_ℓ received topology information from v_i
   - information regarding collisions between v_ℓ and $v \in \{V / v_\ell\}$

Each edge $e \in E$ will will hold the following information:

- channel quality characteristic
  - could be simple: number representing channel strength
  - could be complex: vector-valued characterization
- uncertainty metric for channel information (freshness counter)

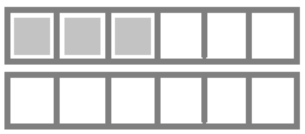
Assuming each node can communicate over multiple frequency channels $f_1, f_2, ...$ for each link $e(v_i, v_j)$, the above formulation can be extended to a vector where each entry characterizes a sub-channel for each frequency $f_1, f_2, ...$

Additionally, it may be beneficial to maintain a history of n_ℓ's topology estimates as a more dynamic state space. I.e. $\vec{G} = \{G_1(V_1, E_1), ..., G_m(V_m, E_m)\}$



2. Local buffer state

   The buffer state of v_ℓ will represent the node's local packet buffer. It is assumed the buffer is dynamic in that the agent may send packets without respecting the order they were received.



Each entry in the buffer will include the following information:

- flag for packet presence
- packet source, destination
- time spent traveling through network
- time spent in buffer
- estimate to reach destination (if destination location is known)

11.2 Action Space

Similar to the state space, the action space can be divided into two sections as described below:

1. Communication Actions

   At each discrete time instance t the agent will choose among the available actions. These actions all utilize the same communication medium and are thus mutually exclusive.

   - Sense subset of neighboring nodes $S(N(n_\ell)) \in N(n_\ell)$
   - forward prioritized packet to neighbor $n_j \in neighborhood(n_\ell)$
   - share topology estimate $G(V, E)$ with node subset $S(V) \subseteq V$ (this would not directly scale well, but it can be pared down)
   - WAIT/do nothing

2. Buffer Scheduling

   Buffer scheduling will ultimately prioritize packets within the buffer for sending. The buffer scheduler could potentially operate semi-asynchronously from the communication portion of the agent.

11.3 Time-Domain Scheduling

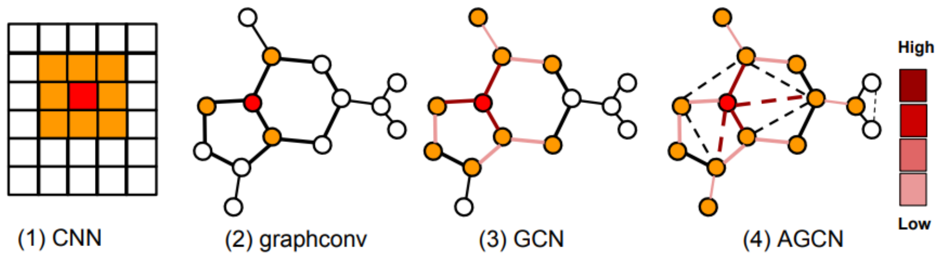
In order to avoid collisions between two transmitting nodes, a time-domain scheduling algorithm should be utilized. Such an algorithm will attempt to schedule node transmissions in time to minimize the number of collisions. Even with a centralized controller, scheduling for ad hoc networks is NP-hard (Data Networks, Bertsekas and Gallager)

Random access scheduling protocols for MANETs exist and could be used in order to simplify the scope of the problem. I am unsure of how well these scheduling algorithms perform; however, I plan to investigate how scheduling may benefit from learning algorithms.

11.4 Learning Method

The state and action space for the above constructions seems very large (possibly intractably large). I would imagine that the state and actions spaces can be pared down by simple assumptions (such as reducing the size of the input topology graph by only preserving full information for a subset of the graph)

If the state and action spaces remained large, I would think that some functional approximation method would be advised. The envisioned RL algorithm would utilize work on Graph Convolutional Neural Networks. By preserving the graphical structure of the network toplogy, the full information quality of the data would be preserved.



The specific formulation of the deep Q-network using graph convolutional neural networks is left open. Additionally, I am still considering what paradigm should be considered for the multi-agent interaction and how the reinforcement signals should be shared.

The reinforcement signal itself, as stated in section 10, will be related to some QoS metric / transmission slowdown/ or successful transmission percentage.

12 References:

-RL for Routing in MANETs

[1] Y.-H. Chang, T. Ho and L. P. Kaelbling, "Mobilized ad-hoc networks: A reinforcement learning approach".

[2] K. Maneenil and W. Usaha, "Preventing malicious nodes in ad hoc networks using reinforcement learning," in 2005 2nd International Symposium on Wireless Communication Systems.

-RL for Routing

[3] J. A. Boyan and M. L. Littman·, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach".

[4] G. Stampa, M. Arias, D. Sánchez-Charles, V. Muntés-Mulero and A. Cabellos, "A Deep-Reinforcement Learning Approach for Software-Defined Networking Routing Optimization".

-Time Domain Scheduling

[5] A. Köse, N. Evirgen, H. Gökcesu, K. Gökcesu and M. Médard, "Nearly Optimal Scheduling of Wireless Ad Hoc Networks in Polynomial Time," 2017.

[6] Y. Xu, W. Chen, Z. Cao and K. B. Letaief, "A Distributed Random Access Protocol with Enhanced Routing in Time-Slotted MANETs," in IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference, 2008.

-Graph Convolutional Network / Graph Signal Processing

[7] R. Li, S. Wang, F. Zhu and J. Huang, "Adaptive Graph Convolutional Neural Networks," 9 1 2018.

13 Random Thoughts:

- nodes can “sniff” packets and infer the graph structure
- nodes could proactively probe the network to learn its structure (causal graph inference)
- sharing information with neighboring nodes comes at a cost
  - how can the agent’s local information be shared at different sizes?
  - essentially a variable-sized embedding?
- perhaps it is wise to share some information at times which are less busy? (offline learning)
- it seems that the model complexity of each node in the network could be variable and dependent on its position in the network
  - less connected nodes probably are less important in the global optimization and thus require less information
  - spend power most efficintly based on network position
- if a core model is shared amongst the agents, then it may improve coordination
  - each node can anticipate the actions of the others
  - maybe some portion of the model is shared and another is not (role assignment)
- what about fairness?
  - for example, in a mobile phone network the users closest to the bay stations will receive higher traffic loads
  - perhaps these methods make more sense in settings where fairness isn’t as relevant (team strategy vs. mutual cooperation)
- perhaps some information is always passed along as part of the packet header? Inspiration from ant colony optimization.
- perhaps packet size can be accounted for in the scheme (some messages might be considerably smaller and require less time to transmit