Mobile Ad-Hoc Cognitive Packet Networks

Erol Gelenbe, Ricardo Lent
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816
{erol,rlent}@cs.ucf.edu

Abstract

This paper investigates an extension to Mobile Ad-Hoc Networks (MANET), of “Cognitive Packet Networks” (CPN) which offer smart Quality of Service (QoS) driven routing for peer-to-peer connections. An Ad-Hoc CPN (AHCPN) uses smart packets to discover and maintain routes and neighbors. Because these smart packets give priority to the use of a reinforcement learning based algorithm over flooding to propagate, the resulting routing overhead introduced by AHCPN is small.

1 Introduction

The sheer size and diversity of the Internet in distinct types and number of users, servers, nodes and links, will not easily allow it to cater to individual QoS needs. In this paper we propose a Mobile Ad-Hoc Network path establishment and routing algorithm inspired by the “Cognitive Packet Network” (CPN) approach.

CPN provides best-effort QoS to end-to-end connections using smart packet routing. CPN uses smart packets (SP) for finding routes based on QoS, Acknowledgements Packets (AP) to bring route information and measurements back to the nodes and source, and source routed dumb packets (DP) to carry payload. CPN uses reinforcement learning (RL) to route SP based on the users specific “QoS goal” which is a numeric function to be minimized (e.g. delay, or jitter, or loss, etc) or maximized (e.g. packet dispersion to avoid eavesdropping). RL based routing determines the best next hop for SP at each node. Dumb packets, which constitute the bulk of the traffic, are source routed using routes discovered by the SP; AP are also source routed (from destination to source).

QoS is a challenging issue in wireless mobile ad-hoc networks (MANET), for numerous civilian and military applications including video and voice that will increasingly be integrated into the MANET framework. The CPN algorithms approach to discovering paths in a network that can offer best-effort QoS to selected users provides a promising avenue for research. However, the CPN approach needs to be adapted to this specific framework to create possibly distinct source-to-destination and destination-to-source paths, and deal with frequently changing of connectivity conditions while avoiding excessive overhead.

2 Cognitive Packet Networks and Related Work

This paper investigates an extension to Mobile Ad-Hoc Networks (MANET), of Cognitive Packet Networks (CPN) [6, 8] which offer smart Quality of Service (QoS) driven routing for peer-to-peer connections. CPN has smart packets (SP), dumb packets (DP), and acknowledgements (AP). DP carry payload and are source routed (i.e. packet route is stored in the packet and provided by the source node) using routing information generated by SP. SP are sent out to look for routes to a destination; the rate at which they are sent is a fraction of the rate at which DP are sent. SP also collect measurement data as they visit nodes. AP carry back measurement data to be used in QoS based routing and deposit it in mailboxes at nodes; they also bring route information which has been discovered by SP back to the source nodes to be used by subsequent DP. A small six node wire-line CPN test-bed has been implemented [8] in a framework that also supports ordinary IP, as well as QoS driven QoS CPN routing, and we are currently deploying a 32-node wireline test-bed.

Existing proposals for routing in ad-hoc mobile networks can be classified as suggested by Royer and Toh [21] in table driven protocols (proactive) and source-initiated on-demand driven protocols (reactive). While proactive protocols require global information about the network to maintain routing information for every possible source-destination pair, reactive protocols initiate a route discovery only when it is needed by a node. Proactive protocols are in general more expensive in terms of use of network resources than reactive protocols,
since they require a periodic transmission of routing tables by the nodes. Destination-Sequenced Distance-Vector routing (DSDV) [18], Wireless Routing Protocol (WRP) [13], Clusterhead Gateway Switch Routing (CGSR) [2] are examples of proactive protocols where the control packet overhead grows as $O(n^2)$. Global State Routing (CSR) [1] takes the idea of link state routing to reduce routing messages by sending them only on link changes. FishEye State Routing (FSR) [17] is an attempt to reduce the still high routing overhead of CSR by using different update frequencies depending on the distance to other nodes.

Most on-demand protocols use a variation of flooding, sending a route request packet to find a route from a source to a destination. The flooding is typically restricted by requiring that each node process each request packet at most once. This approach produces a control packet overhead that grows as $O(n)$. In order to accelerate route discovery, many on-demand routing proposals stop the route search process when the request packet encounters a node with knowledge of a route to the destination: DSR [9, 10], AODV [20, 19], LMR [3]. However, this condition does not reduce the overall complexity because it only prevents the continuation of that particular replica of the route request. A way to reduce route discovery complexity is shown in Location Aided Routing (LAR) [12] and geoTORA [11], where the propagation area of the flooding is restricted by using location information of the nodes, but a Global Positioning System (GPS) is required at each node.

Despite route discovery is triggered only when needed in reactive protocols, some source-initiated on-demand driven proposals still require an expensive periodic transmission of a control packet [23] to maintain updated information about neighbors and their link status. ABR [22] and SSR [5] use periodic beacons to determine the link stability of a node in order to use this information in the routing process. The Temporally Ordered Routing Algorithm (TORA) [14, 15, 16] runs over the Internet Manet Encapsulation Protocol (IMEP) [4] which in turn use the beacon approach as well.

3 Random Neural Networks and the Reinforcement Learning algorithm

A RNN is “an analytically tractable spiked random neural network model whose mathematical structure is similar to that of queueing networks” [7]. The model is based in nonlinear mathematics, but it has a product form solution as in many simpler queueing network models.

Similarly to models of biological neurons, RNN use both excitatory and inhibitory connections to connect the artificial neurons.

The probability that the $i$th neuron in the network is excited is given by the state $q_i$:

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)}$$

where

$$\lambda^+(i) = \sum_j q_j w^+_{ji} + \lambda_i \quad \lambda^-(i) = \sum_j q_j w^-_{ji} + \lambda_i$$

Here $w^+_{ji}$ is the rate at which neuron $i$ sends "excitation spikes" to neuron $j$ when $i$ is excited, and $w^-_{ji}$ is the rate at which neuron $i$ sends "inhibition spikes" to neuron $j$ when $i$ is excited. For an $n$-neuron network, the network parameters are $n \times n$ weight matrices $W^+ = \{w^+_{ji}\}$ and $W^- = \{w^-_{ji}\}$. The values for the weight matrices are acquired from the data, and they represent the knowledge stored in the RNN.

3.1 Reinforcement learning-based routing

Routing decisions are made by a RNN constructed using as many nodes as decisions outcomes. After convergence of the neural network, the winning neuron provides the next-hop for the Smart Packet.

Let the neurons be numbered $1, \ldots, n$. Thus for any
decision $i$, there is some neuron $i$. Decisions in this RL algorithm with the RNN are taken by selecting the decision $j$ for which the corresponding neuron is the most excited, i.e., the one with the largest value of $q_j$. Note that the $i$th decision may not have contributed directly to the $i$th observed reward because of time delays between cause and effect.

The routing goal $G$ is defined as a function to be minimized. In CPN, we have used the following goals so far: transit delays, packet loss, and transit jitter. These goals can be combined to create a more complex goal.

The inverse of the goal is the reward $R = G^{-1}$, which can be measured from the data. The reward at time $t$, determines the decision threshold $T_i$ as follows:
\[ T_l = \alpha T_{l-1} + (1 - \alpha)R_l \]

where \( R_l, l = 1, 2, \ldots \) are successive measurements for \( R_l \), and \( \alpha \) is some constant \( 0 < \alpha < 1 \), typically close to 1.

Suppose that we have now taken the \( l \)th decision which corresponds to neuron \( j \), and that we have measured the \( l \)th reward \( R_l \). Let us denote by \( r_l \) the firing rates of the neurons before the update takes place. We first determine whether the most recent value of the reward is larger than the previous "smoothed" value of the reward which we call the threshold \( T_{l-1} \). If that is the case, then we increase very significantly the excitatory weights going into the neuron that was the previous winner (in order to reward it for its new success), and make a small increase of the inhibitory weights leading to other neurons. If the new reward is not better that the previously observer smoothed reward (the threshold), then we simply increase significantly the inhibitory weights leading to the previous winning neuron (in order to punish it for not being very successful this time). This is detailed in the algorithm given below. We compute \( T_{l-1} \) and then update the network weights as follows for all neurons \( i \neq j \):

- If \( T_{l-1} \leq R_l \)
  \[ w_{ji}^+ \leftarrow w_{ji}^+ + R_l, \]
  \[ w_{ji}^- \leftarrow w_{ji}^- + R_l/(n-2), \quad k \neq j, \]
- else
  \[ w_{ji}^+ \leftarrow w_{ji}^+ + R_l/(n-2), \quad k \neq j, \]
  \[ w_{ji}^- \leftarrow w_{ji}^- + R_l. \]

Then we re-normalize all the weights by carrying out the following operations, to avoid obtaining weights which indefinitely increase in size. First for each \( i \), we compute
\[ r_i^+ = \sum_{m=1}^{n} [w^+(i,m) + w^-(i,m)] \]
and the re-normalize the weights with
\[ w^+(i,j) \leftarrow w^+(i,j) \times \frac{r_i^+}{r_i^+}, \quad w^-(i,j) \leftarrow w^-(i,j) \times \frac{r_i^+}{r_i^+}. \]  \[ (3) \]

Finally, the probabilities \( q_i \) are computed using the nonlinear iterations 1 and 2, leading to a new decision based on the neuron with the highest probability of being excited.

4 A CPN approach to Mobile Ad-Hoc Routing

In the proposed Ad-Hoc Cognitive Packet Network (AHCPN) approach we are considering the following issues: (a) a dynamic way to discover and maintain routes and neighbors (nodes that a node can reach in a single hop); (b) resources are typically scarce in mobiles, in particular energy availability, therefore the routing overhead must be small; (c) a high packet loss is to be expected because of the unreliability of the transmission medium; (d) unidirectional or asymmetrical links may exist in the network.

Route discovery and maintenance is performed by Smart Packets (SP), while Dumb Packets (DP) transport IP packets, and Acknowledgement Packets (AP) distribute routing information and delivery confirmation. Each packet is composed of three sections: a header, a Cognitive Map (CM), and a payload for DP. As in CPN, CM is an area where the packet transport network information collected as it travels.

Each node keeps a Mailbox (MB) where it stores information about each source-destination pair that it is aware of. This information is updated using the CM from AP. For the operation of the RNN/RL, each node also maintains a table with the weights for each RNN, that is computed from the information stored in the MB. Source nodes maintain the source route for each discovered path.

In AHCPN, neighboring nodes are discovered simply by listening to the radio channel. Each arriving packet is scanned for its source address. An entry is updated in a Neighbors’ Table (NT) at each node along with a time-to-live value.
Route discovery is triggered on demand, in AHCPN. When a node needs to find a new route, it sends out an SP by either using the reinforcement learning based algorithm to determine the best next-hop, or just broadcasting the packet. We use broadcast as a last-resource because it is expensive in terms of resources involved. A broadcast is used only when not enough information is available at the node to run the RNN/RL algorithm properly. Each node, after processing the first arriving SP, records the source address and packet identification value to avoid processing more than once the same request. Note that initially the process is exactly as a limited flooding ($O(n)$) for a network with $n$ nodes, but it quickly reduces up to $O(\log(n))$ as the nodes learn about the network topology and status, and SP are able to run the RNN/RL algorithm. In a low mobility scenario, our approach would perform similarly to CPN.

When a node moves out of range, other nodes will update their tables by removing its entry after the time-to-live period. They also remove all routes in which the moving node is involved. Furthermore, in order to maintain a consistent view of the network, a low percentage of broadcast SP are introduced to update MB, in a similar way CPN use a round-robin approach.

In order to deal with unidirectional and asymmetrical links, Smart AP (SAP) replace AP in the earlier version of CPN; SAP begins with a source route which is the shortest right-to-left reverse path of its corresponding SP. The reserve route is established by taking a SP's route, examining it from right (destination) to left (source), and removing any sequences of nodes which begin and end in the same node: e.g. the path $\langle a,b,c,d,a,c,g,h,c,l,m \rangle$ will result in the reverse route $\langle m,l,b,a \rangle$. Note that the reverse route is not necessarily the shortest reverse routes, nor the one resulting in the best QoS.

Contrary to an AP in “conventional” CPN, if an SAP arrives to a node from which the next node is not accessible anymore, then the SAP will take on the behavior of a SP to find its way to the source node, using RL to seek its next-hop. In this case, mailboxes are updated later with the information from the SAP carried in the next DP.

The use of energy information in the nodes, and link-quality information of the links is still under research. We plan to introduce these parameters in the routing goal of the random neural network. The global routing process would be able to maximize these quantities while minimizing delay, loss and jitter. Our proposal and results will be presented in short time.

5 AHCPN Testbed

We are implementing a test-bed to show the effectiveness of our proposed approach using PCs running Linux. To support the transport of IP packets, we have developed our algorithm as a virtual device driver for Linux. We have named this interface as CPN-IP exchange, (CIX). After assigning an appropriate IP address to the CIX interface, automatically we allow the IP layer to access AHCPN simply by using its regular routing mechanism. Note that the presence of a CIX interface in a computer does not prevent IP using other devices. In fact, in our implementation, both IP and CIX may use the same real devices.

Figure 1-a shows a initial topology that we are using to develop our test-bed. The network consists of both desktop and laptop computers with both wire interfaces (Ethernet, 10 Mbps) and wireless (IEEE 802.11b) in ad-hoc mode interfaces. The wireless devices adjust automatically their transmission rate to 1, 2, 5.5 or 11 Mbps depending on the link quality of the connection.

We allow CIX to use all available network interfaces in the nodes, except for network interface eth0 in node 192.168.2.1 that is connected to the Internet using an official IP number. We have programmed this node also to make address translation for the private IP network that we are using in our test-bed, so it serves as a default router for all other nodes to access the Internet. In addition to Ethernet interfaces, node 192.168.2.101 also has a wireless interface, and we use it as a gateway for the laptops to access the wired network.

One single IP address is assigned to each node (assigned to the CIX interface). Nodes 192.168.2.20, 192.168.2.30, and 192.168.2.40 are laptop computers so they are free to move.

Energy information at each node is obtained from the Advanced Power Management (APM) layer residing in the Linux kernel. A signal-to-noise ratio for every arrival is obtained from the wireless network interface, and is useful to calculate the link quality with the sending node. We plan to introduce both pieces of information in the routing goal function. For the time being, experiments are performed trying to minimize only the round-trip delay of the dumb packets.

Figure 1-b shows a floor plan of the Computer Science Building at UCF where our test-bed is located. We report below the results of a preliminary test using two of the laptop computers, and the desktops.

One laptop (198.168.2.20) was located across the hallway in point B, making sure that its location allows contact with the wired network in A. Points along the segment BC have a degrading connection to A as we tested. At point C there is no contact at all. The second laptop (198.168.2.30) was initially located at point A starting a connection with node 192.168.2.1 (ping, rate 1 packet/sec). Progressively we move the laptop from
point A to point B and then to point C. After staying for about 10 sec in C, we return the laptop to A using the same path. Movements were at walking speed of 1.5 meters/sec approximately.

Figure 2 shows the round-trip delay and the path of each individual packet for the forward transmission of ICMP (ping) packets. The reply ICMP packets showed a similar behavior. A packet loss is indicated by an “x” near the horizontal axis. To track better packet losses, we turned off retransmissions for this experiment. There is an initial loss of packets as shown, while the mobile attempted to find the initial route and neighbors. From packet count 75 to 100 there was a high packet loss because the mobile was traveling from B to C. After packet count 100, the mobile switched routes to use the second laptop as a gateway. In the way back to point A, the mobile switched back to direct connection to the wired network at packet count 175 (because it offered lower delay).


6 Conclusions

We have proposed an extension to Cognitive Packet Networks to operate in an ad hoc wireless network. Smart packets use both a reinforced learning based algorithm and broadcast to propagate for discovering routes and neighbors. By prioritizing the use of the reinforcement learning algorithm over flooding, we make AHCPN to require a very low number of control packets to operate. In AHCPN, the number of nodes involved in the discovery process is initially as high as $O(n)$, but it quickly drops close to $O(\log(n))$ as the nodes learn about
the network, and more smart packets are able to use the reinforcement learning approach. Energy constrains and link quality information are to be included in the routing algorithm as part of our current research, in addition to extensive experiments to show how effective and useful is this approach.

References


IEEE PERSONAL COMMUNICATIONS, April 1999.

[22] C-K Toh. A novel distributed routing protocol to support ad-hoc mobile computing. In Proceedings of 
ACM/IEEE, October 1996.