

UNC Charlotte MANIAC Challenge Strategy

Minsu Huang, Fan Yang, Yu Wang

*Department of Computer Science
University of North Carolina at Charlotte
Charlotte, North Carolina 28223, USA
{mhuang4,fyang4,yu.wang}@uncc.edu*

I. INTRODUCTION

The MANIAC Challenge aims to better understand cooperation and interoperability in mobile ad hoc networks. According to the rules of the MANIAC challenge, each team will have two laptops to participate the competition and the organizers will provide four reference nodes. Together all laptops will form a mobile ad hoc network. The reference nodes will randomly send packets to each team. Every time a node receives a packet intended to it, it will gain ten points, and to forward a packet for another node it will consume one point. When the competition finish, the team has the highest point will be the winner.

To achieve higher points, each team need to forwarding fewer packets for other teams and receiving more packets destined to themselves. In the last MANIAC Challenge, since the MANIAC API did not provide any mechanism to monitor neighbors' behaviors, the best strategy is dropping all packets passing through to avoid point deduction. However, if every team behaves like that, no packets can be transmitted in such non-cooperative network. This year, the MANIAC Challenge API allows hosts with supported network cards to observe traffics that are within range. This enables hosts to monitor the behavior of neighboring hosts and see what they are sending and resending. Therefore, simply dropping all packets may not be the optimal strategy anymore.

II. ANALYSIS -- A SIMPLE GAME

In the competition, each team is selfish and want to maximize its score by increasing the amount of its own received packets and avoiding packet forwarding for the others. Such competition with selfish users may be modelled and analyzed by game theory.

We first consider a simplest game where only two teams compete and each team only has one node. As shown in Figure 1, suppose reference node 1 (ref1) send a packet to node B via node A, and reference node 2 (ref2) send a packet to node A via node B. So each node (A or B) faces a choice to forward (F) or drop (D) the packet. We can list all of the possible combinations of their strategy and the corresponding payoffs as in Table 1. The payoffs are calculated under MANIAC rules.



Figure 1. a simple forwarding game with two teams.

Table 1. Strategy form of two teams.

A/B	F	D
F	9,9	-1,10
D	10,-1	0,0

As shown in Table 1, if both A and B choose to forward packet for its counterpart, they both gain 9 points. However, according to game theory, the statement both forward (F, F) is not a Nash equilibrium and it can not hold. When A chooses to forward, if B forwards, it will gain 9 points, and if it drops, it will gain 10 points. Thus, B prefers to choose “drop”. When A chooses to drop, if B forwards, it will lose one point, and if B drops it will lose nothing. Thus B still prefers to choose “drop”. So whatever is A’s choice, B will choose “drop” as the dominating strategy. The same holds for A. Therefore, both A and B will choose to drop other’s packet. The strategy combination of (D, D) is the Nash equilibrium, because no player has an incentive to change. This is a classical game of “Prisoner’s dilemma”. Notice that if both A and B have multiple packets, the same result can be derived where dropping all packets is the dominating strategy.

The above example is only the simplest static case of the game. However, in reality the competition is much more complicated and dynamic. (1) There are multiple teams participating the game; (2) The forwarding game repeats for multiple rounds; (3) In each round, the participants (forwarding node, its neighbors, destinations, routes) are dynamically changing. Therefore, it is very hard to model such game and derive the optimal strategy.

In such complex game, it is not clear for us that simply dropping all packets is still the optimal strategy. Notice that if a node drops a packet, some of its neighbors will know that it is not cooperative, hence they may not forward packet for it in the future rounds. This may results in lower gain of points.

III. OUR STRATEGY

Since we failed to model the competition using game theory, we adopt a simple reputation-based approach which includes two major components: reputation calculation and forwarding strategy.

Reputation Model: First, we maintain a list of nodes who we learned via either routing table, neighbor list, or overhearing messages. We identify each node by its IP and MAC address, and keep updating this list. For each node u in the list, we record a reputation $r(u)$, which is initially set to 0. This reputation value is only a local reputation from our observations, since we can only hear from our neighbours. It is calculated and updated using partial information of the network when we receive a packet. However, as the competition goes on, we hope to cumulate enough information to have reasonable evaluations on nodes' reputations.

Reputation Calculation: Each time we hear a packet, we learn some information of certain nodes' behaviors. We divide the information we can detect into the following cases.

- We only know that a neighbor u has transferred a packet to another node v (where v is not our team member and not the destination of the packet), but don't know whether it has forwarded or redirected the packet. Then we add the corresponding node's reputation $r(u)$ by *one* point, since it at least transfers packets.
- We detect a node u has forwarded a packet to the right destination which is not our team member. The we add its reputation $r(u)$ by *five* points because it is doing great favor to other nodes.
- We get a packet destined to ourselves from u , then add *ten* points to its reputation $r(u)$, since it is clearly a friend to help.
- We detect a node u has dropped a packet (*i.e.*, we heard that u received the packet but did not send it out), then we lower its reputation $r(u)$ by *ten* points.
- We detect a node u has redirected a packet (*i.e.*, we heard that u received the packet whose destination is our neighbor but u sent to other nodes), then we lower its reputation $r(u)$ by *five* points.

The number of points in above cases could be changed during further study.

Forwarding Strategy: We investigate the tradeoff between forwarding packets and dropping packets by designing a new randomized forwarding strategy based on the collected reputations. Each time we are required to forward a packet, we set a probability p to forward or redirect it. We set p and decide whether *forward* or *redirect* according to the following different situations.

- If we have a packet going toward our team member, we will *forward* it without any hesitation (*i.e.*, $p=1$).

- If we are required to forward a packet whose destination is right the next hop node, we *drop* it ($p=0$), otherwise a rival will gain ten points.
- If the next hop is a node with bad reputation (a simple threshold may be used) and it is not the destination, we will set p accordingly bigger and *forward* it with probability p , because the next hop node may drop it with high probability.
- If the next hop is a node with good reputation and it is not the destination, we set a high probability p to *redirect* the packet to the neighbor with worst reputation. By doing so, nobody can tell whether we *redirect* or *forward*, hence we can keep our good reputation while prevent the packet towards the destination.

The above is just a brief description of our current strategy, and we will further modify and refine it during our implementation. We need to study the tradeoff between dropping/redirecting more packets and keeping a better reputation among our rivals for further gain. Another information may be considered during making the forwarding decision is the route length. We will integrate such information when we calculate the probability p . The longer the remaining route is, the larger the probability p will be.

IV. STATUS OF IMPLEMENTATIONS

Currently, we already installed all needed softwares and MANIAC API in two laptops and made them MANIAC ready for the competition. We are still in the process of implementing of our strategies by writing the codes of both reputation calculation and forwarding strategy modules. After the implementation, we will build a simple simulation environment to verify our strategy on a single machine.