Chapter 2

Proposed Partner Choosing Algorithms

The *802.11 PCF* is a polling-based scheme, which has been introduced in Chapter 1. In PCF, PC performs the role of polling master. The PC usually resides in the AP. Once the AP gains control, it polls the associated stations on its polling list. In PCF, a station may transmit only if it gets polled. The polled station will transmit a frame after receiving the poll from PC. The *802.11 PCF* does not specify the actual polling algorithm and how to manage AP's polling list. Because we focus on the cooperative functioning in PCF, the weight/priority of every station is set the same and all the packet sizes are set the same. So we apply the simplest scheduling algorithm method, *Round Robin (RR)*. Applying RR, AP will query every station in turn to check whether a station has frames to send.

Observing the behavior in *802.11 PCF*, the polling processes could be divided into three different cases as Fig. 1.3 illustrates. *(1)* A wireless station will reply a normal data frame to PC if it has frames to send after receiving a poll from PC; *(2)* If a stations does not have any frames to send, it would transmit a *Null Acknowledgement* frame back to PC when getting a poll; *(3)* Due to the instability of the wireless radio, the ongoing transmission may suffer from poor channel condition. A station may not receive a poll or the acknowledgement from the PC correctly because the poll frame or the acknowledgement frame is lost. On the other hand, the polled station may not send the reply messages, such as a data frame or

Null Acknowledgement frame, to the PC successfully. According to *802.11 PCF*, if the data frame from a wireless station is not acknowledged, a station cannot retransmit until the station is polled by the PC again.

When sending a reply data frame to PC, the wireless station may suffer from low *SNR(Signal Noise Ratio)* or channel error. The reply data frame may fail to arrive the PC or get ruined in the middle of transmission. Due to the burst characteristic of wireless channel error, a wireless station may suffer from long delay until it can finally reach to the PC successfully. Also, if the polled station to the PC is under channel error or low SNR, it means that the reply data frame may be lost, then the polling frame is a waste.

The proposed method which deals with the problem described above is based on the concept of cooperative communications. The quality of service experiences severe variation due to channel instability, thereby the use of some type of diversity is necessitated. We proposed a new form of spatial diversity of PCF, in which diversity gains are achieved via the cooperation of wireless stations. Although a wireless station A is possible to fail to reach the PC, its neighbor station B may have a clear connection to PC. If station A can communicate with station B well, then station A may send its data frame copies to station B and ask B to send for it when station A is under bad channel. The action of station A sending data frame copies to station B is named *Whisper*. And station B is named station A's *Helper*. In PCF, the chance that a wireless station can $Whisper$ to its helper is when some other station is sending data frame to the PC. The *Whisper* process is shown like Fig. 2.1. If station A receives a poll frame from the PC but measures a low SNR, or station A is under bad channel currently, it can whisper to the station B when station C is sending its data to PC. Thus, next time when PC polls station A, station B can send A's data frame for A. The action of *Help* is illustrated as Fig. 2.2

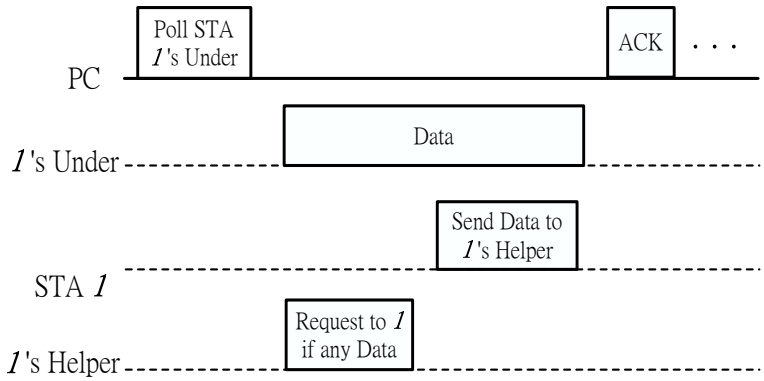

Figure 2.1: The Process of Whisper

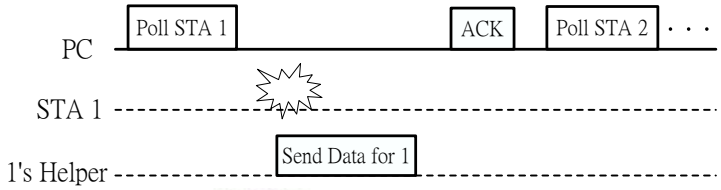

Figure 2.2: The Process of Help

2.1 Static partner choosing algorithm in PCF

When a wireless station enters a new radio coverage, it needs to do *authentication* and *association* procedures in the 802.11 standard. The *association request* message is as Fig. 2.3 shows. The format is actually the *Management frame*, and the practical content of *Association Request* is in the *Frame Body*. The *Association Request* information contains four parts in the original design : *Capability Information*, *Listen Interval*, *SSID*, and *Supported rates*. Additional fields, called *Location Information* is need in our method. The location of station is the basic parameter for our partner choosing algorithms. All the stations in the system must tell the PC their location. So that the PC can use the *Location Information* to take into the partner choosing algorithm to assign *Helper* to every station.

After gathering all the location information from all the stations, the PC computes all the distances between any two stations and the distances between PC and all stations. Then

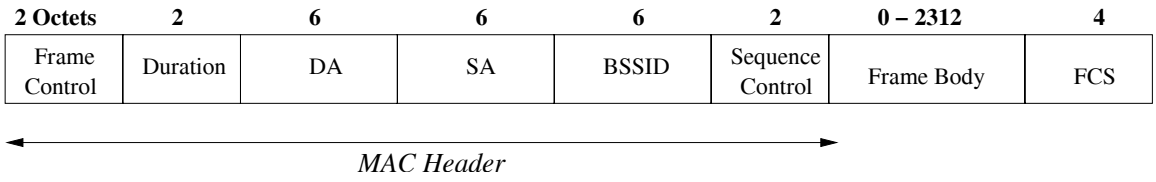

Figure 2.3: The Format of 802.11 Association Request Message

PC starts constructing the *helper structure* for every station. The helper choosing method is shown as Fig. 2.4. The system setup assumes that when the PC transmits a polling frame or an ACK(acknowledge) frame to some station, the transmission range is big enough to reach every station in the system. And the PC will let the station know the location of itself. When the station replies its data frame back to the PC, it will use the PC's location information to compute and automatically tune its transmission power in order that the transmission coverage just covers to the PC. So that during the period when the station sending data frames to the PC, some other stations can *Whisper* to their *helpers*. As the figure illustrates, when station C is sending its data frame to the PC, station A which is not in the *Area 1* can whisper to station B because *Area 2* does not overlap *Area 1*. So the method put the element (*Helper*, *Under*) into station A's helper structure. By the method running through all the stations, the PC establishes the whole *helper structure* for all stations. The form of the *helper information* of every station is illustrated like Fig. 2.5. The *Under* field indicates station N can *Whisper* during station 3, or 4, or 5, etc. And the *helper_info_num_* means the number of *Under*. The *Helper Information* is a list structure. There is a *helper* list for every *Under*. It means station N can *Whisper* to station 1 or 2 *Under* station 4. And the *helper_num_* indicates the number of helper for every *Under*.

In the original thoughts, for every station, single helper is considered. If a station has more than one helpers, then it must have chances to send its data frame copies to all the helpers. Besides, when the PC polls this station, the reply data duplication may happen if more than one helpers have its data frame copies of the station. There is no way to control which helper should help if all of them have the data frame copies. If the station chooses one

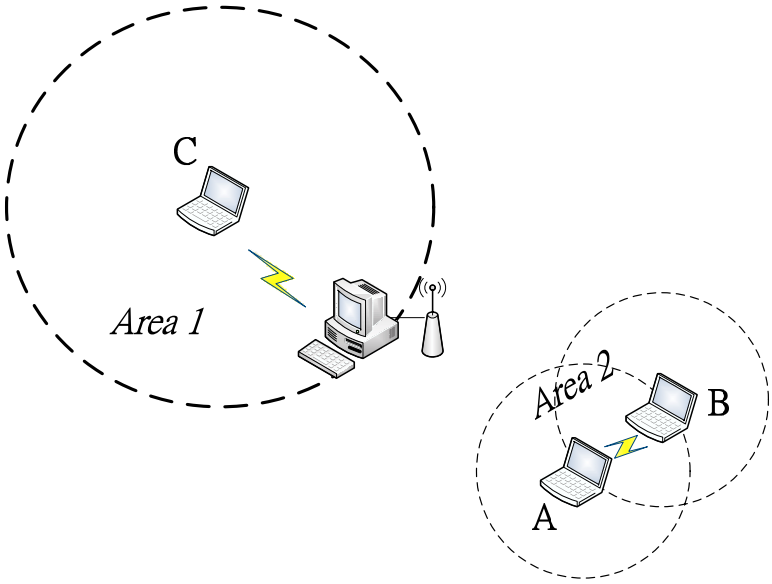

Figure 2.4: How to Choose Helper

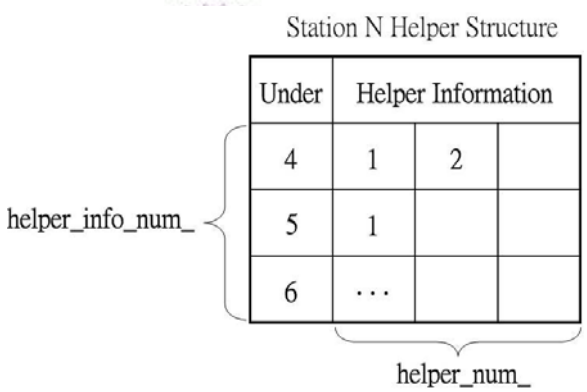

Figure 2.5: The Form of The Helper Structure

station among its helpers to send data frames for it, the station must have a clear information that which helper is currently under good channel condition between the PC so that this helper could help. But a station cannot access the information of other stations, so that single helper is considered. And the helper choosing is held in the beginning. By applying the helper choosing algorithm, under the best condition, every station has a (*Helper*, *Under*) pair and the helper assignments are distributed. For example, station A has a helper station

B under station D (*B*, *D*) and a helper station C under station E (*C*, *E*). And station F has a helper station B under station D (*B*, *D*). If station A chooses the pair (*B*, *D*) to be its helper pair, then station F will have nothing to choose. Station F will have no helper. To prevent from this kind of thing that some station may not be able to have a helper, we have to make a arrangement to let other stations to give up and choose another helper. Thus, the station may have the chance to have a helper. In this case, a partner choosing algorithm is needed. A station with small *helper_info_num_* means it has fewer opportunities to get a helper, because its candidates are few and it might be no one left when its turn to choose. And a station with big *helper_info_num_* is likely to have a helper because it has more candidates in its list. So we should let the stations with small *helper_info_num_* choose their helpers first. When a station decides a (*Helper*, *Under*) pair, the PC will insert this pair into PC's list called *decided_helper_*. In the same *Under*, more than one pairs of *Whisper* are permitted. If there is another pair that wants to *Whisper* in the same *Under*, then it will be allowed if its *Whisper* transmission range named *Area 3* does not overlap *Area 1* and *Area 2*. In this case, there might be more than one pairs of stations *Whisper* in an *Under*.

And the helper choosing algorithm is listed as followed.

1. Construct the *helper structure* of every station.

2. Sort the helper structure of every station by their *helper_info_num_* , from small to big.

3. Choose the helper structure of the station with the smallest *helper_info_num_* . By applying the chosen *decided_helper_* list, delete the element that would interfere with the existing (*Helper*, *Under*) pairs.

4. From the rest of the helper structure of the station after the selection in Step 3, among the helpers if there are still some stations not yet assigned to be the helper of other stations, then randomly choose helper among the helpers that haven't been the helpers of other stations. If every station in the helper list has already been assigned to be the

helper of other stations, then randomly choose helper among all of them in the helper list.

5. Insert the chosen (*Helper*, *Under*) pair for the station in the *decided_helper_* list.

6. Choose the station with the next small *helper_info_num_* and return to Step 3.

After every station finishing choosing their (*Helper*, *Under*) pair and the PC inserting all the pairs into its *decided_helper_* list, the PC will then broadcast the *decided_helper_* list within the *BEACON* frame in the beginning. All stations decapsulate this frame and know their own *Helpers* and *Unders*. More important, all stations will get to know the information about which stations they should help, named their *Helpee*. Then they will ask their *Helpee* to send their data frame copies to them during the specific *Under* transmitting data frames to the PC, so that they can help their *Helpees* to send their data frames to the PC for them during the next pollings of their *Helpees*.

Fig. 2.6 is an example of the scenario. There are six stations in the scenario, and a PC resides in the AP. The AP is in the middle of the map and its location is (10, 10). And the six stations are located at (5, 0), (0, 0), (0, 5), (15, 20), (20, 20), (20, 15) respectively. And the helper structures are constructed as Fig. 2.7 to Fig. 2.9. Due to the characteristic of this scenario, the best example of *helper* distribution is like Fig. 2.10. In Fig. 2.10, there is exactly one *Whisper* in every *Under*. And the algorithm described above will achieve this kind of distribution.

However, after simulation we found the static partner choosing algorithm can be improved in some ways. In the static partner choosing algorithm, the *Helper* and *Under* of a station is chosen in advance. So the *Whisper* might not work when bumping into the following listed situations :

1. When the *Helper* is under bad channel.

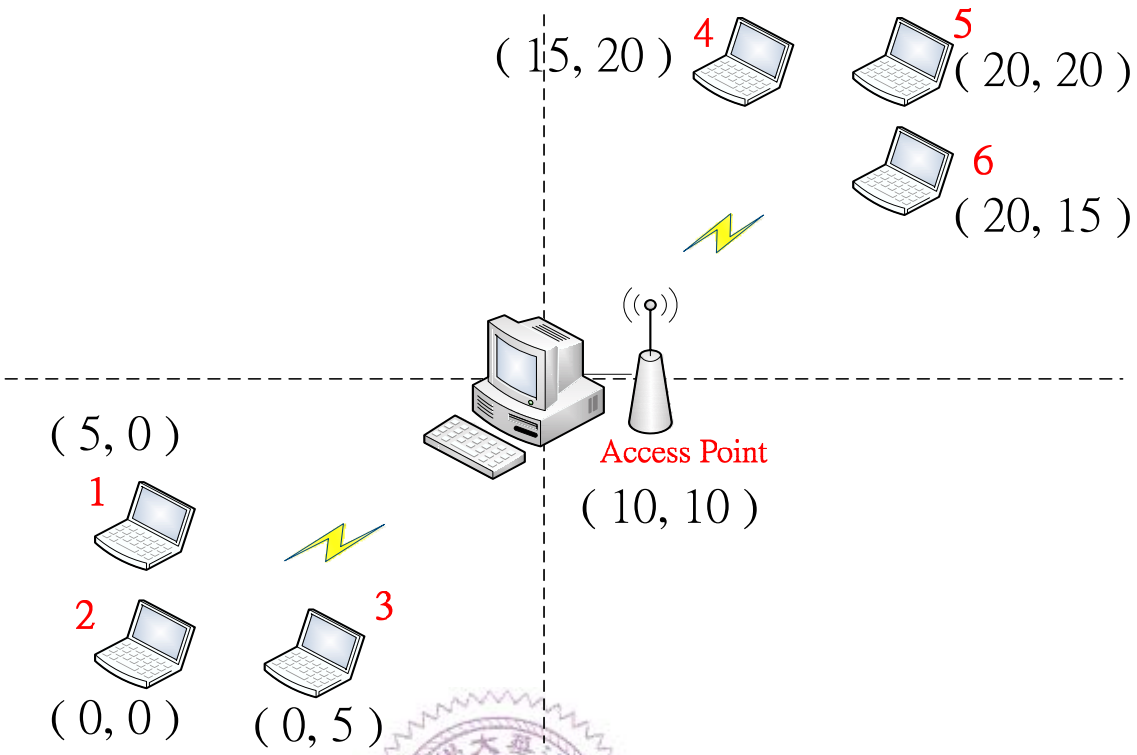

Figure 2.6: An Example Scenario

2. When the *Under* is under bad channel. It means that the station would not have the chance to *Whisper* to its helper.

3. When the channel condition between the station and its helper is bad.

There are still some other reasons that the *Cooperation Processes* might not work. For example, when the PC is under bad channel, then the polling frames or the ACK frames can not be sent successfully. Thus, the stations lose the chances to *Whisper* to their helpers and their helpers lose the chances to send data frames for them. In this case, no improvements can be made. So this case is not under our considerations.

A new algorithm must be constructed. The performance can be improved if the (*Helper*, *Under*) pair is dynamically assigned according to the updated channel condition information gathered by the PC.

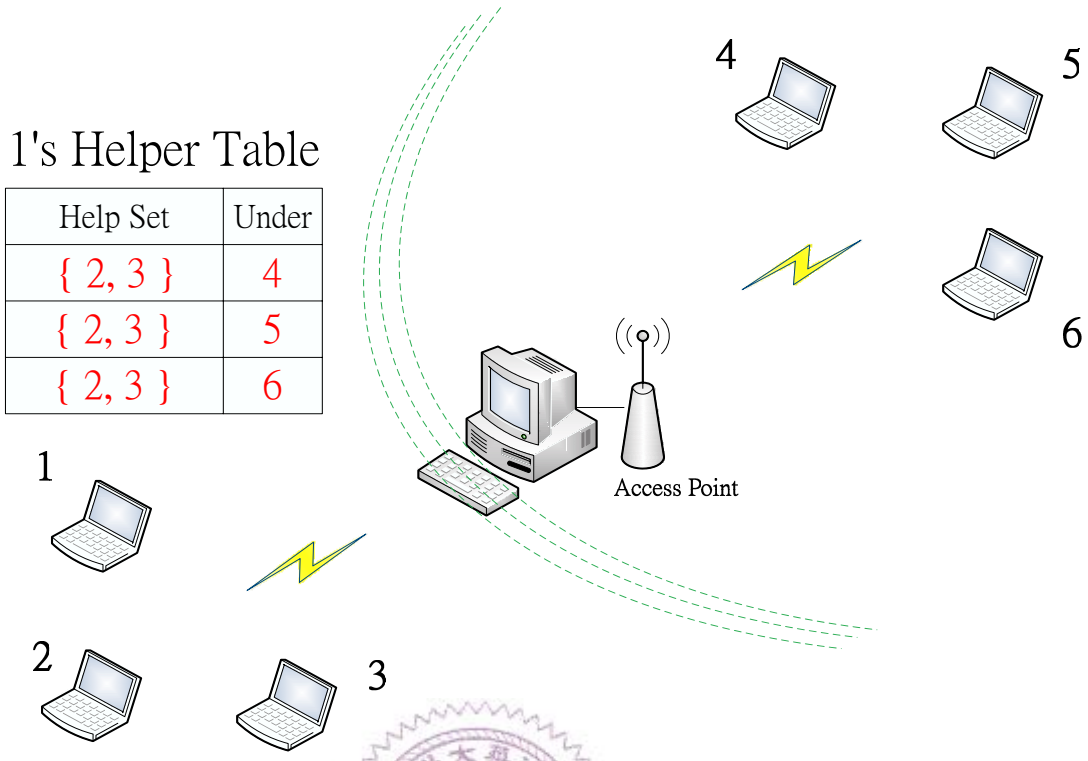

Figure 2.7: The Helper Structure of Station 1

2.2 Dynamic partner choosing algorithm in PCF

PCF is a centralized controlling scheme. The algorithm described above is running in the PC, too. The cooperation performance would decrease because of some reasons, like the chosen *Helper* station cannot *help* because it faces channel error condition too; or the channel between the station and its *Helper* is bad so that it cannot ask for help; or the station cannot send its data frame copies to its *helper* because its *Under* has a bad channel condition between the PC too, even when the station can communicate with its *Helper* well. By the updated station channel condition information gathered by PC, there are some modifications can be made.

In the static partner choosing algorithm, the (*Helper*, *Under*) pair is chosen in the beginning and will not be changed. In the *Dynamic Partner Choosing Algorithm*, the (

2.2. Dynamic partner choosing algorithm in PCF

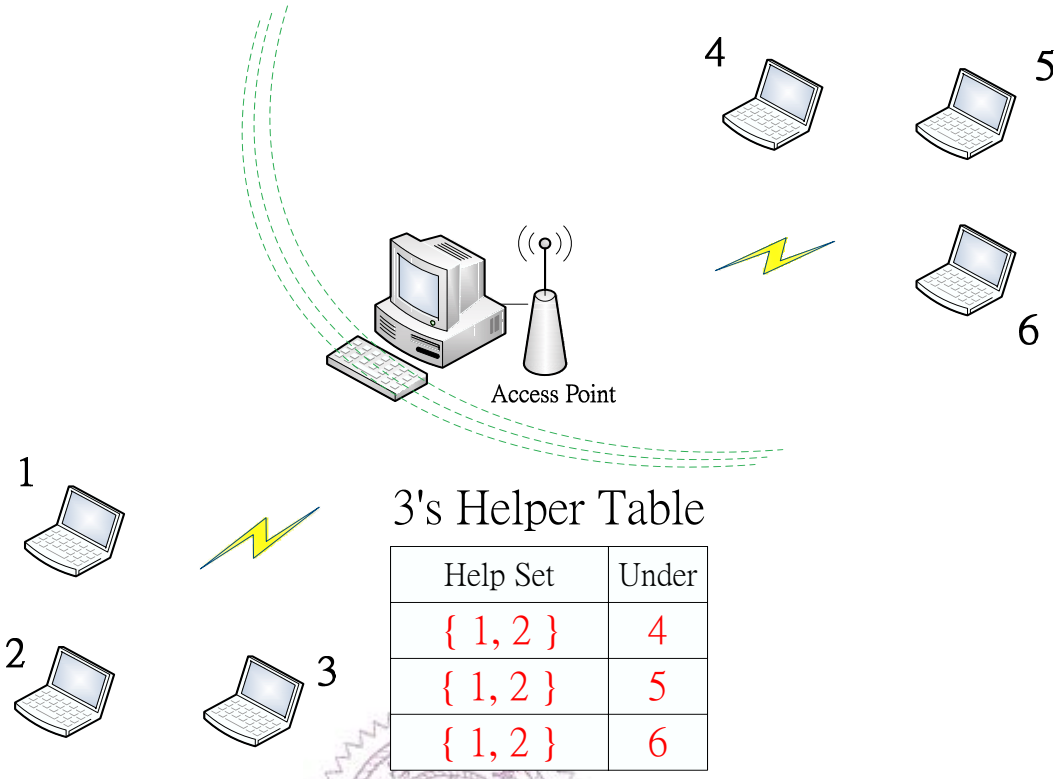

Figure 2.8: The Helper Structure of Station 3

Helper, *Under*) pair is dynamically assigned by the PC. The helper structure is constructed as the static partner choosing algorithm describes. But the (*Helper*, *Under*) pair of every station is not selected in advance. The PC will assign a (*Helper*, *Under*) pair to a station after the PC gets to know that the station is under a bad channel. Thus, a station has a *Helper* only when this station has troubles sending its data frames to the PC by itself at that time. The assigned (*Helper*, *Under*) pair in the *decided_helper_* list of the PC will be removed once the PC has the idea of the station retrieving a good channel between the PC. When the *Helper* or the *Under* of the station turns to be in bad condition, the PC will then select another pair of (*Helper*, *Under*) for this station from its helper structure and the pair must not interfere with the existing pairs in the *decided_helper_* list.

An example of the process of the dynamic partner choosing algorithm is illustrated.

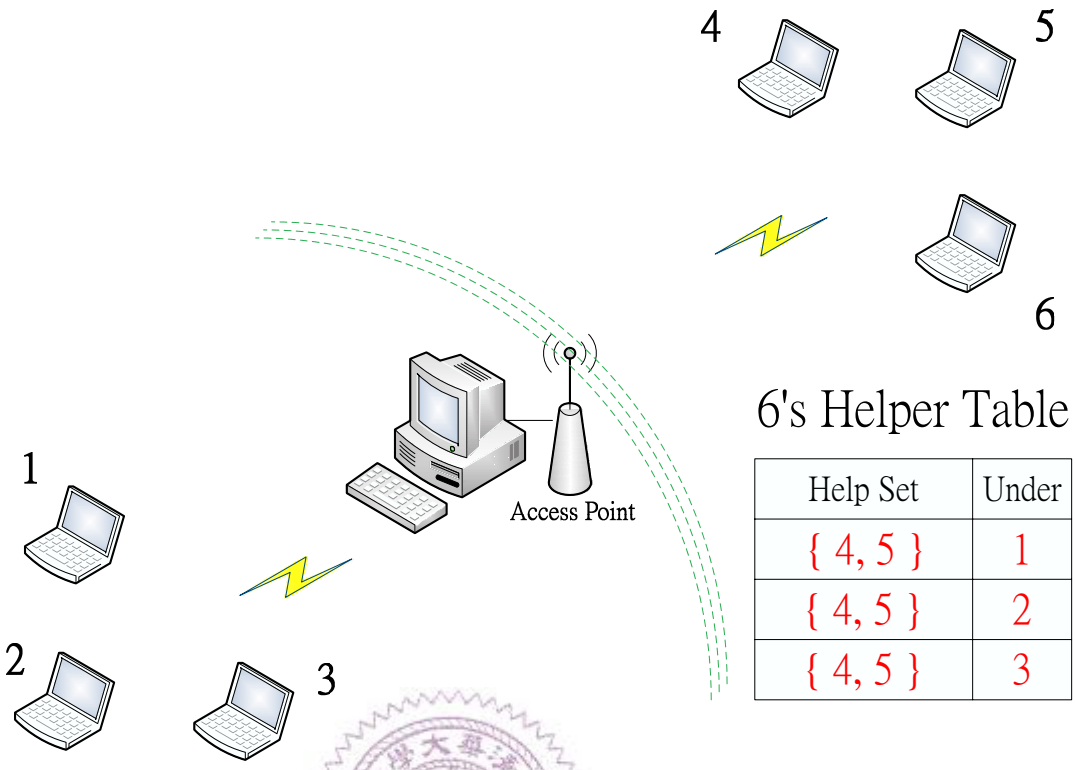

Figure 2.9: The Helper Structure of Station 6

The example scenario is like Fig. 2.6, too. The scheduling is round-robin, so PC polling the stations in the marked number order is assumed. The helper structure is constructed in the PC as described in the previous section. From station 1 to station 6 in this example, after polling station 6 the PC will go back from station 1 again and again. The figures from Fig. 2.11 to Fig. 2.15 is numbered in their occurrence order. In Fig. 2.11, when the PC polls station 1, station 1 suffers from a bad channel condition and cannot reply to the PC. Because there is no other station is in bad condition, the PC randomly chooses a (*Helper*, *Under*) pair, which is { 2, 4 } this time. And the PC inserts { 2, 4 } into the *decided_helper_* list. After that, when the PC polls station 6 as Fig. 2.12 shows, station 6 receives PC's polling frame with low SNR. This means it might be a risk for station 6 to send data frames to the PC. In this case, station 6 would not take this risk and would give up sending data frame to the PC. When the *Send Timer* of the PC expires, the PC chooses a (*Helper*, *Under*)

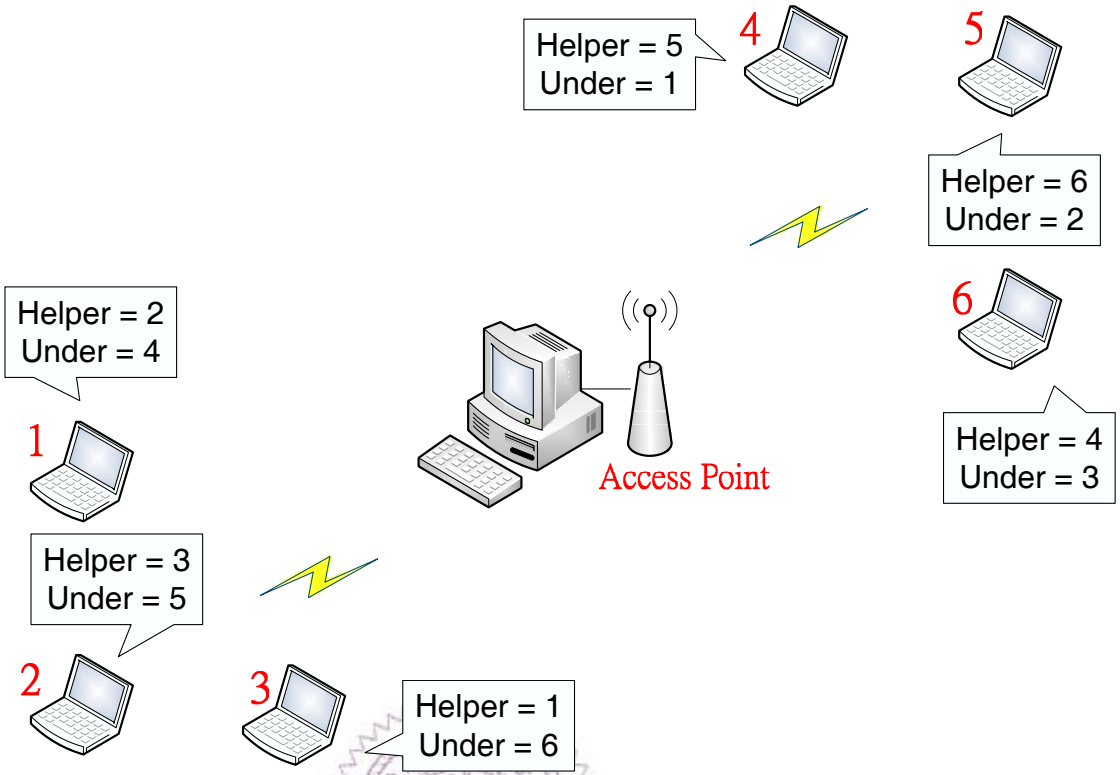

Figure 2.10: The Best Condition After Choosing (Helper, Under) Pair

pair for station 6. Observing the helper structure of station 6, station 1 is one of the *Under* candidate. But the PC has already known that station 1 is now in bad state. So the (*Helper*, *Under*) pairs which the *Under* is station 1 are eliminated from the candidate list. In the same way, the PC removes all the pairs that includes stations with bad state in the helper structure of station 6. If there are still some candidate pairs left in the helper structure, the PC will then randomly choose a pair for station 6, which is { 5, 2 } this time. When next time PC polls station 2, station 6 will *Whisper* to station 5. If their *Whisper* successfully, then station 5 will help station 6 to send 6's data when next time PC polls station 6. But if the *Whisper* between station 6 and 5 is not successful because it might go through a bad channel, like Fig. 2.13 represents, PC receives no data from station 5 when PC polls station 6. The PC will assume that station 6 does not have a clear connection with station 5. And because the

burst characteristic of channel error, the *Whisper* process between station 6 and 5 afterwards might not succeed either. Therefore, the PC will select another (*Helper*, *Under*) pair for station 6. Observing the helper structure of station 6 and removing the elements that contain stations which are currently in bad state first. Second, remove the elements that contain station 5 since the channel between station 6 and 5 is currently poor. Then choose in the rest of the helper structure of station 6, and this time the PC picks { 4, 2 }. This way station 6 will *Whisper* to station 4 during PC polling station 2. The dynamic partner choosing algorithm also deals with the problems about the *helpers* or *unders* turning to bad state. After a period of time, it is station 2's turn to have troubles connecting the PC like Fig. 2.14 shows. What the PC will do first is selecting a good condition (*Helper*, *Under*) pair. Next, the PC will check the *decided_helper_* list to see if any station has station 2 to be its *helper* or *under*. If yes, then the PC will delete this (*Helper*, *Under*) pair from the *decided_helper_* list and re-choose (*Helper*, *Under*) pairs for the victim stations. As the Fig. 2.14 illustrated, if station 2 turns to bad status, station 1 and 6 will become victims because they chose station 2 to be their *helper* or *under*. Then the PC will dynamically change helper or under for them like Fig. 2.15. Station 1 chooses { 3, 4 } this time after removing all the bad state stations or the stations that has been selected by other stations. And station 6 chooses the pair { 4, 3 } this time. Thus, the performance is sure to be enhanced.

For the dynamic partner choosing algorithm,the key differences from the static partner choosing algorithm are listed as followed :

- The (*Helper*, *Under*) pair is assigned by PC to a station only when the PC gets to know that this station cannot reply its polling. The selected (*Helper*, *Under*) pair will be insert into the *decided_helper_* list. The *decided_helper_* list is broadcasted by PC in every polling. So the involved stations will have ideas of which station I should help and which station helps me. This (*Helper*, *Under*) pair will be eliminated from this list when the station is capable of sending data frames to the PC by itself.

- When a station turns to bad state, the PC will not only choose a (*Helper*, *Under*) pair

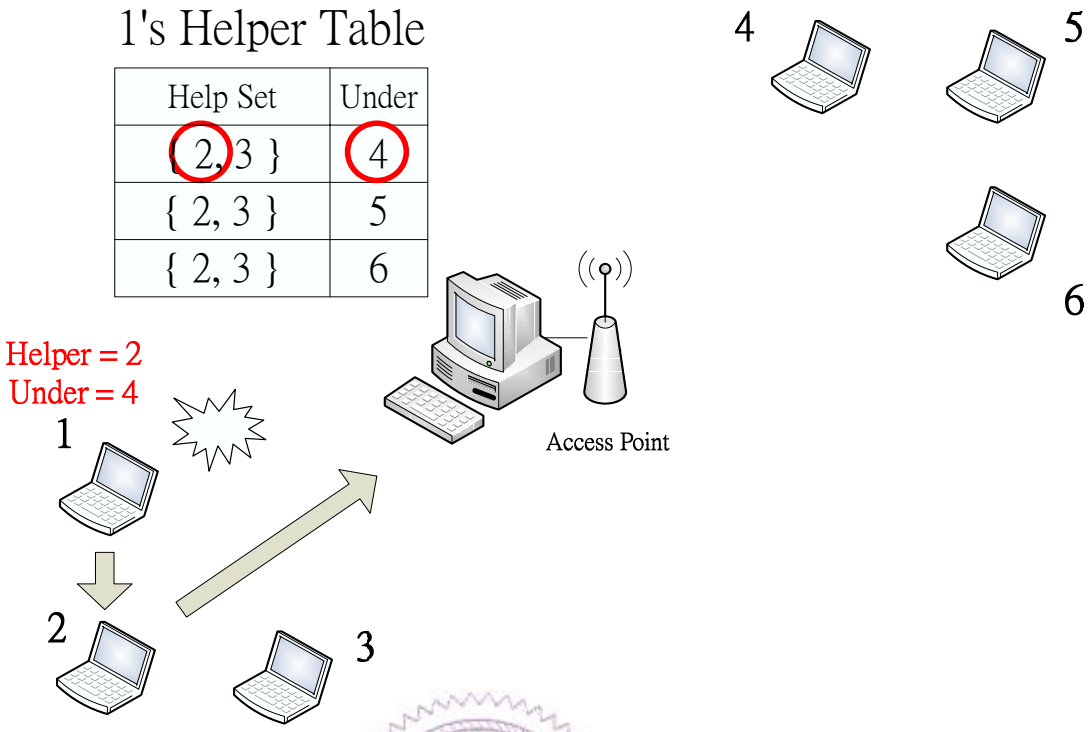

Figure 2.11: Station 1 Has a Bad Channel and Choose a Helper

for this station, but also check the *decided_helper_* list to see if this station is *helper* or *under* of other stations. If this station indeed involves in the (*Helper*, *Under*) pair of other stations, we call victims, PC will randomly select another good (*Helper*, *Under*) pair for the victims from their helper structure and delete the old victim (*Helper*, *Under*) pairs from the *decided_helper_* list.

- When a station retrieves a good channel between the PC, PC will remove its (*Helper*, *Under*) pair from the *decided_helper_* list. Also, since the station can join the helping process now, PC will check if this station can help other stations which are under bad channel but has no one to be their *Helper* or *Under*. The new selected (*Helper*, *Under*) pairs will be insert into the *decided_helper_* list.

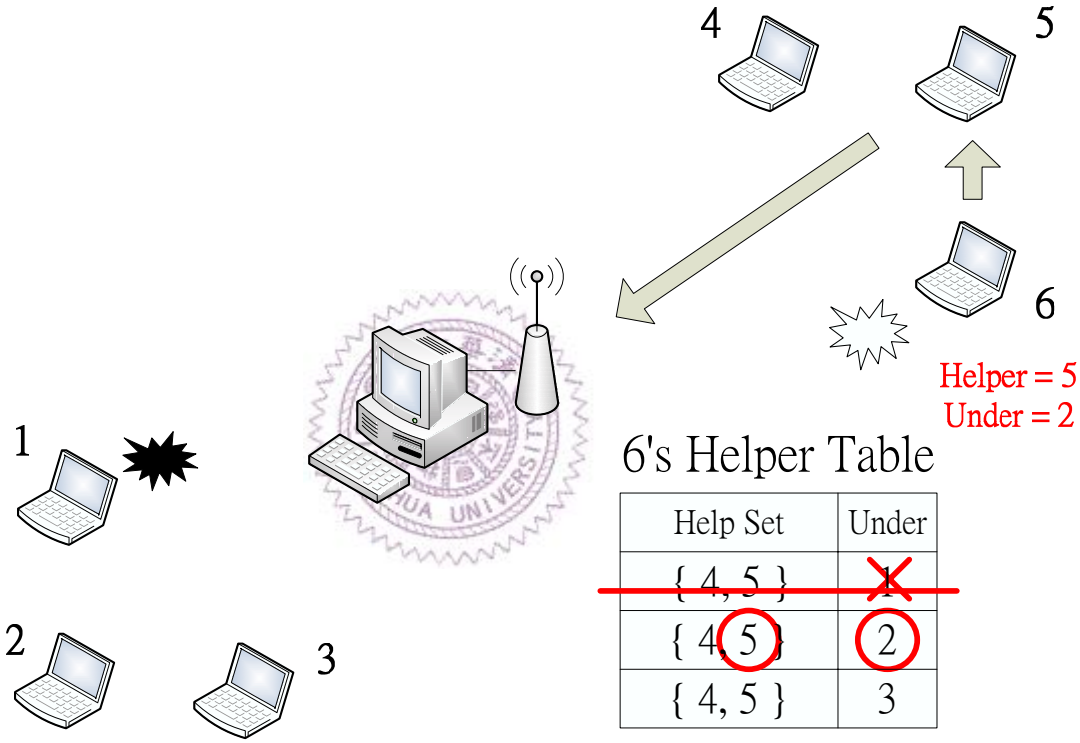

Figure 2.12: Station 6 Has a Bad Channel and Choose a Helper

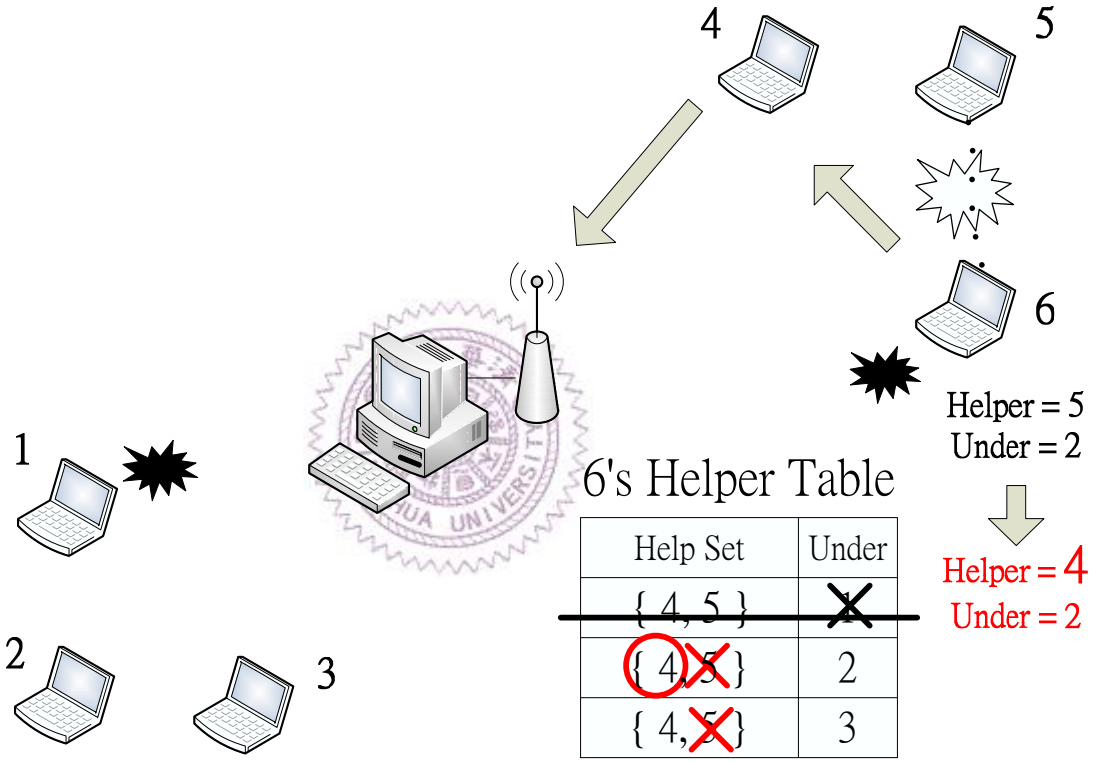

Figure 2.13: The Channel Between Station 6 and 5 Becomes Poor

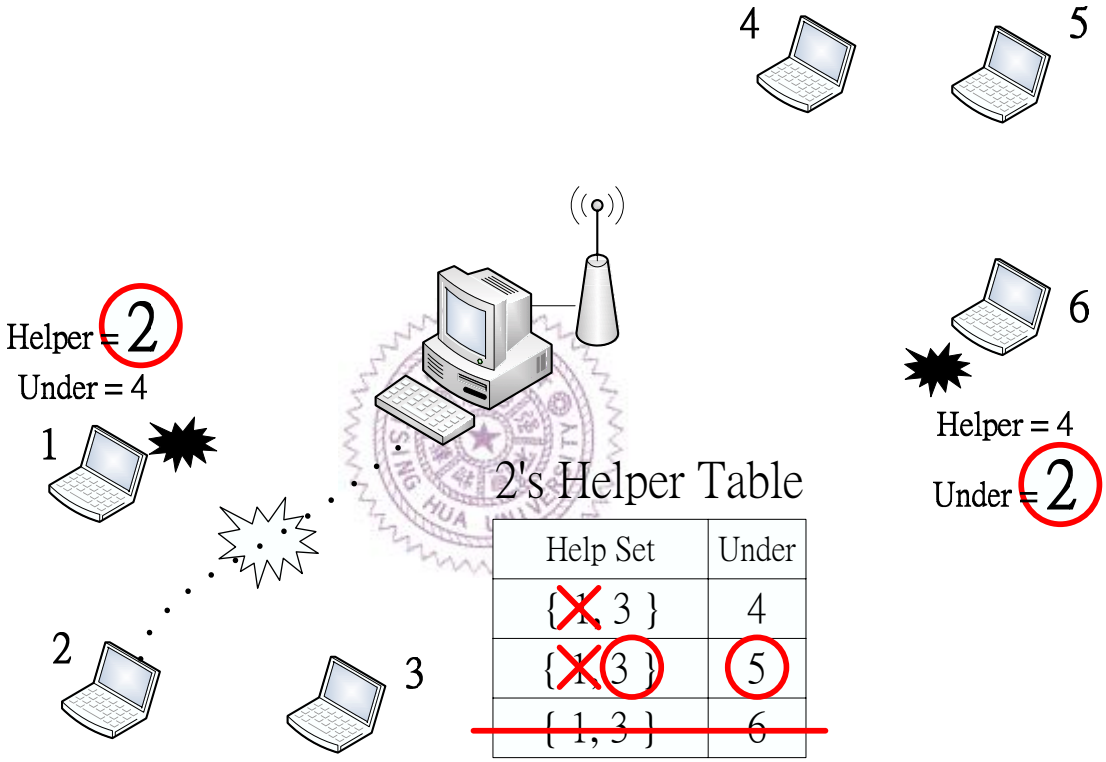

Figure 2.14: Station 2 Turns to Bad State

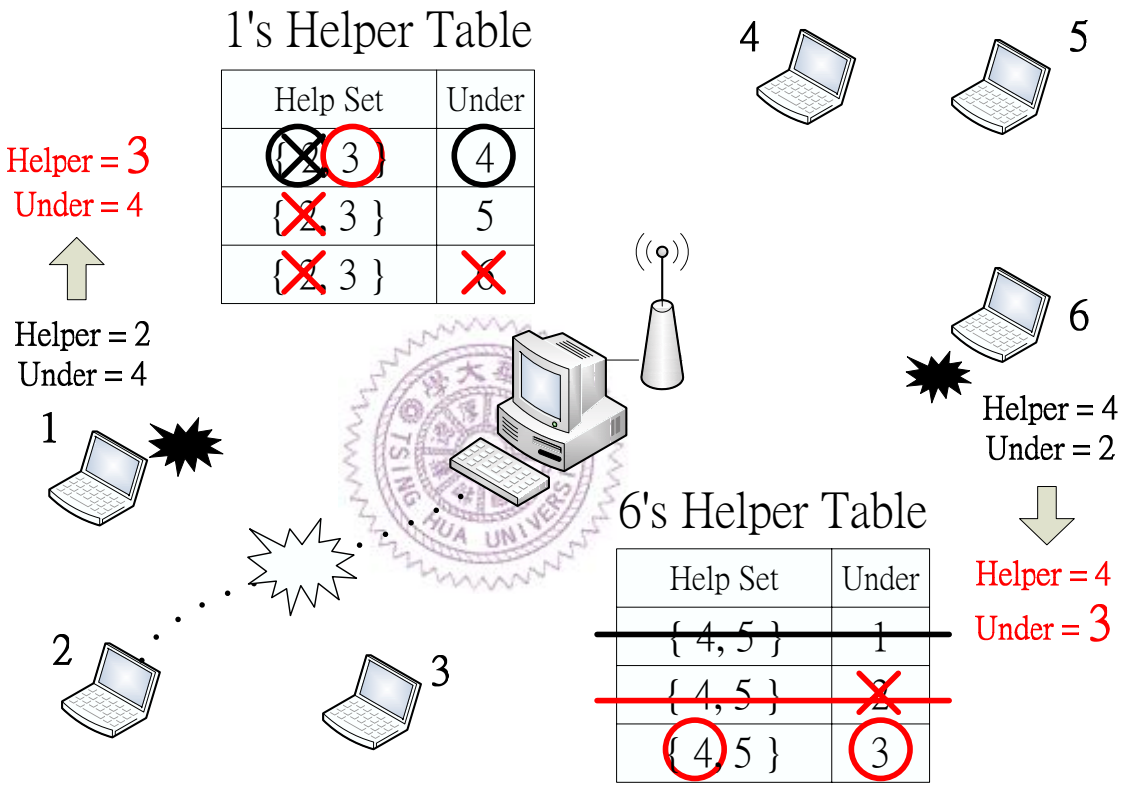

Figure 2.15: Station 3 and 6 Have to Change Their (Helper, Under) Pair