CHAPTER 4

TEMPORAL INFORMATION SYSTEMS

4.1 TEMPORAL EXTENSIONS

This chapter presents temporal extensions to the traditional information system (Rajam et al. 2008). The use of the temporal information system in a mobile node is then presented.

A Generic Temporal Information System (GTIS) $S$ is defined as a set of information tables, $S = \{T_i, 0 \leq i \leq n\}$ with each information table $T_i = \{U_i, A, V, \rho_i\}$ located at a time $t_i$ on the time axis. A time interval can also be considered instead of a time instance.

A special case is when $U_i = U$, i.e. the same universe of elements that appears in each information table. A particular element ($x \in U$), for each attribute ($a \in A$), would have a value $\rho_i(x, a)$, at each time $t_i$. For example, patient $x$ has fever at $t_i$, and does not have fever at $t_j$. This special case is when each $T_i = \{U, A, V, \rho_i\}$.

Here, this is treated as a single Temporal Information System $T$ defined as $T = \{U, A \cup \{a_t\}, V, \rho\}$, where $a_t$ is the time attribute with $V_{a_t}$ as a set of pairs $(t_i, t_{i+1})$, $0 \leq i < n$ with $t_0, t_1, \ldots, t_n$ as a sequence of time instances, $|V_{a_t}| = n$. 
For each elementary set that is formed from the set of attributes \( A \) (that is, the set of attributes without the time information), there are now \( n \) elementary sets, where the first elementary set consists of elements that occurred between time instance \( t_0 \) and \( t_1 \), the second elementary set of elements between \( t_1 \) and \( t_2 \), and the last elementary set of elements between \( t_{n-1} \) and \( t_n \). This can be pictured as vertical blocks of elementary sets along a time axis.

4.2 TEMPORAL INFORMATION AND TEMPORAL DECISION SYSTEM IN A MOBILE NODE

In traditional Rough Set Theory and VPRS, the value of the decision attribute of a new element (or unknown element or test case) is predicted, based on which elementary set it falls into. The elementary set into which it falls is determined by the values of the attributes of that element. However, in this mobile ad hoc routing application, the destination can possibly be reached through several different sequences of intermediary nodes. In other words, several different combinations of attribute values make it possible for the destination to be reached. That is, elements in several different elementary sets correspond to routes that lead to this particular destination. Thus several elementary sets play a role in identifying the best next hop for a particular destination. So, it is not possible to use a single elementary set to predict the value of the decision attribute. The union of these elementary sets is used. In other words, for a particular destination, the union is taken of all the elementary sets that correspond to valid routes from the current mobile node to the destination.

A stringent method of predicting the next hop is when all the elements in this union of elementary sets have the same value of the decision attribute, then this value is taken as the predicted next hop. In other words, all
known routes to this destination should have this particular node as the next hop. This can also be considered as that value of the decision attribute for which all these elementary sets are in its lower approximation. It is to be remembered that the decision attribute is a multi-valued attribute, and so the lower approximation is with respect to a value of the decision attribute.

Another method is to have the predicted next hop as that value of the decision attribute where the union of these elementary sets is in the $\beta$-positive region. The conditional probability is determined using the union of elementary sets, and not a single elementary set. The probability that a particular next hop occurs given that the route leads to a particular destination is taken as the conditional probability. This conditional probability should be greater than a threshold $\beta$. In other words, a large number of known routes to this destination have this particular node as the next hop.

In a Temporal Information System (TIS) for a mobile node, each element (corresponding to a route) has a particular value of the time attribute, that is, each element falls in a particular time interval. This is determined by the time stamp of the next hop of the route that corresponds to this element. The Temporal Decision System (TDS) in a mobile node is used to predict the next hop.

In each time interval, a predicted next hop is determined. An appropriate method (as explained earlier in this section) is used to determine the predicted next hop in each time interval. The predicted next hop for the TDS is then determined based on the number of time intervals in which it is the predicted next hop.
The predicted value of the decision attribute is determined from the TDS based on the probability of a particular value of the decision attribute being the predicted value in the different time intervals. This probability is the number of time intervals in which that value of the decision attribute is the predicted value divided by the total number of time intervals. The predicted value of the decision attribute is the value for which this probability is greater than a threshold $\beta'$. In other words, that particular next hop has been the predicted next hop in most of the time intervals.

4.2.1 Weighted Temporal Information System and Weighted Temporal Decision System in a mobile node

In Weighted Temporal Information Systems (WTIS) and Weighted Temporal Decision Systems (WTDS), weights $w_1, w_2, \ldots, w_n$ are assigned to the elementary sets between time instances $t_0$ and $t_1$, the elementary sets between $t_1$ and $t_2$, ... and the elementary sets between $t_{n-1}$ and $t_n$, respectively. The predicted value of the decision attribute is determined after associating weights with the time intervals. The predicted value of the decision attribute is that value of the decision attribute where the probability is greater than a threshold $\beta'$. The probability is the sum of the weights of time intervals in which that value of the decision attribute is the predicted next hop divided by the sum of the weights of all the time intervals.

When the more recent time intervals play a more important role, the weight of a more recent time interval is higher than the weight of a less recent time interval.
4.3 MOBILE AD HOC ROUTING USING WEIGHTED TEMPORAL INFORMATION SYSTEMS (WTIS)

In Temporal Information Systems, each elementary set is associated with a particular time interval. In Weighted Temporal Information Systems, elementary sets in different time intervals have weights. Here, more recent time intervals are assigned higher weights than the less recent time intervals.

The use of the WTIS to predict the value of the decision attribute has already been described in Section 4.2.1. This uses the predicted value of the decision attribute in different time intervals. The experiment described here uses a simple approach to determine the predicted value of the decision attribute in a particular time interval. A predicted value of the decision attribute in a time interval has atleast one element, with that value of the decision attribute, in the union of elementary sets. That is, the union is in the upper approximation for that value of the decision attribute.

4.3.1 Routing based on WTIS

Here, the route cache of the mobile node is used as the WTIS. Routes that are learnt and used are added to the cache of the mobile node. When routes are added, the time stamp of each link is added along with the routes. However, unlike DSR, even if the same route is present in the cache earlier, the new route is added with the new stamp stamps. So, the cache now has the same route multiple times, but with different time stamps.

In the source node, initially, as in DSR, a shortest route in the route cache if available, is placed as the source route in the data packet. If not
available, route discovery is done.

Then in the source node, and in any intermediate forwarding node, the WTIS is used to determine the best next hop (using Algorithm 5). If the next hop is found, and does not result in a loop, the data packet will be forwarded to this next hop. If this next hop is different from the one in the source route that is already in the data packet, this new next hop is appended to the source route in the data packet at the current node and the route is invalidated by setting a flag in the data packet.

If a next hop cannot be determined from the WTIS, or if the next hop results in a loop, if the source route in the data packet has not been invalidated earlier, the data packet is forwarded according to the source route. Else, a route discovery is done.

The total time is divided into time intervals. The list of next hops to the destination that are present in the route cache is found. For each possible next hop, from the current time interval till the initial time interval, a weighted sum of the number of times that the particular next hop is used is found. More weight is assigned if the next hop has been used in the recent past. That is, the weights assigned decrease for earlier time intervals.

The ratio of the weighted sum of the usage of the node to the total weight is found. The node for which the ratio is greater than a threshold $\beta'$ is chosen as the next hop.
Algorithm 5 Finding a next hop based on WTIS

```plaintext
findWeightBasedHop()
{

Find all possible next hops that will lead to the destination from this node;

foreach possible next hop nh do
    timeInterval = currentInterval;
    weightedSum = 0;
    weight = maxWeight;
    totalWeight = 0;

    while timeInterval >= 0 do
        if nh is used as a nexthop in timeInterval then
            weightedSum = weightedSum + weight;
        end
        timeInterval = timeInterval - 1; //previous timeInterval
        totalWeight = totalWeight + weight;
        weight = weight - 1;
    end
    ratio[nh] = weightedSum/totalWeight;
end

Find the nexthop nh for which the value of ratio is the maximum and return;
}
```
Table 4.1: Comparison of DSR and TIME_WT for varying number of connections

<table>
<thead>
<tr>
<th>No. of Nodes</th>
<th>Pkt. delivery ratio (%)</th>
<th>Avg. delay (secs)</th>
<th>Normalized control overhead</th>
<th>Avg. hopcount</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DSR</td>
<td>TIME_WT</td>
<td>DSR</td>
<td>TIME_WT</td>
</tr>
<tr>
<td>5</td>
<td>99.77</td>
<td>99.81</td>
<td>0.0222</td>
<td>0.023</td>
</tr>
<tr>
<td>10</td>
<td>98.43</td>
<td>97.2</td>
<td>0.2234</td>
<td>0.4343</td>
</tr>
<tr>
<td>20</td>
<td>63.7</td>
<td>67.14</td>
<td>3.5151</td>
<td>3.1693</td>
</tr>
<tr>
<td>30</td>
<td>47.64</td>
<td>49.67</td>
<td>5.5828</td>
<td>4.7106</td>
</tr>
<tr>
<td>40</td>
<td>37.89</td>
<td>40.65</td>
<td>6.0473</td>
<td>5.8553</td>
</tr>
</tbody>
</table>

4.3.2 Performance evaluation varying the number of connections

The simulation environment is as explained in Section 2.4.2. Initially, the mobile nodes are spread randomly over the network. Nodes move in a field with dimensions 1500 m × 300 m with a maximum speed of 2 m/sec. The pause time is 20 seconds. The number of nodes is kept fixed as 50. The number of communicating source-destination pairs is varied from 5 to 40. Data packets are transmitted at a rate of 4 packets/sec. The size of the time interval is taken as 40 seconds. The value of β′ used is 0.5. Simulations are run for 1000 simulated seconds.

Each value in the table and graphs is the average of the values got in 10 simulation runs corresponding to 10 different scenarios. For each scenario, the communicating source-destination pairs and the initial positions of the nodes are changed randomly. The proposed protocol used in this section is referred to as TIME_WT. The results are given in Table 4.1.

The packet delivery ratios for DSR and TIME_WT are nearly similar for 5 and 10 connections. When the number of connections is increased from
20 to 40 there is a slight improvement in the packet delivery ratio from 5% to 7% for TIME_WT (Figure 4.1).

The normalized control overhead for TIME_WT is higher (not better) than that for DSR when the number of connections is 5 and 10. With increase in the number of connections from 20 to 40, there is an average improvement of 14% to 22% over DSR (Figure 4.3).

The average hop count and the average end-to-end delay for TIME_WT are not as good as that for DSR when the number of connections is 5 and 10. But when the number of connections is increased from 20 to 40, it is seen that there is a slight improvement of about 2% in average hop count and of about 5% in average end-to-end delay over DSR (Figure 4.4, Figure 4.2).

![Comparison of packet delivery ratio of DSR and TIME_WT for varying number of connections](image)

Figure 4.1: Comparison of packet delivery ratio of DSR and TIME_WT for varying number of connections

Figures 4.5, 4.6, 4.7, 4.8 show the percentage improvement of TIME_WT over DSR_{glb-loc} in packet delivery ratio, average end-to-end delay, normalized control overhead and average hop count respectively. It is seen that TIME_WT performs better than DSR_{glb-loc} in the three parameters except aver-
Figure 4.2: Comparison of average end-to-end delay of DSR and TIME_WT for varying number of connections

age hop count. But the average hop count of TIME_WT is better than DSR.

So, for 50 nodes, TIME_WT is better than DSR_{glob-loc} and is also better than DSR in most of the metrics studied. This shows that the introduction of the notion of time has given improvement in performance.
Figure 4.3: Comparison of normalized control overhead of DSR and TIME_WT for varying number of connections

Figure 4.4: Comparison of average hop count of DSR and TIME_WT for varying number of connections
Figure 4.5: Percentage improvement of TIME_WT over DSRglob-loc in packet delivery ratio for varying number of connections

Figure 4.6: Percentage improvement of TIME_WT over DSRglob-loc in average end-to-end delay for varying number of connections
Figure 4.7: Percentage improvement of TIME_WT over DSR_{glob-loc} in normalized control overhead for varying number of connections

Figure 4.8: Percentage improvement of TIME_WT over DSR_{glob-loc} in average hop count for varying number of connections
4.4 MOBILE AD HOC ROUTING USING $\beta$-POSITIVE REGIONS IN WTIS

4.4.1 Routing based on $\beta$-positive regions

The experiment described in this section determines the predicted next hop as that value of the decision attribute where the union of these elementary sets is in the $\beta$-positive region, as described in Section 4.2.

The routing protocol is similar to that of the previous section. The next hop is chosen using the notion of threshold $\beta$ ($\beta$-positive regions) as described in Algorithm 6.

Only next hops that will lead to the destination are considered. Within each time interval the ratio of the number of routes with a particular next hop and will lead to the destination to the total number of routes that will lead to the destination is found.

A weighted sum of the number of times this ratio is greater than $\beta$ is found. More weight is assigned if the next hop is used in the recent past. Weights assigned keep decreasing for earlier time intervals. The node for which the ratio of this weighted sum to the sum of the weights is greater than a threshold $\beta'$ is taken as the next hop.

4.4.2 Performance evaluation varying the number of connections

The parameters used for the simulation in this set of experiments are the same as that used for the experiments in Section 4.3.2. Initially, the
**Algorithm 6** Finding next hop based on $\beta$-positive regions

```java
findVPRSWeightBasedIHop()
{
    Find all possible next hops that will lead to the destination from this node

    foreach possible next hop nh do
    {
        timeInterval = currentInterval - 1;
        weightedSum = 0;
        weight = maxWeight;
        totalWeight = 0;

        while timeInterval >= currentInterval - k do
        {
            nhopCount = the number of routes with next hop nh and will lead to the
destination in timeInterval;

            totalCount = the number of routes that will lead to the destination in
timeInterval;

            ratio1 = nhopCount/totalCount;
            if ratio1 > $\beta$ then
                weightedSum = weightedSum + weight;

            end

            timeInterval = timeInterval - 1; //previous timeInterval
            totalWeight = totalWeight + weight;
            weight = weight - 1;
        }

        ratio[nh] = weightedSum/totalWeight;
    }

    Find the nexthop nh for which the value of ratio is greater than $\beta'$;
}
```
mobile nodes are spread randomly over the network. Nodes move in a field with dimensions 1500 m × 300 m with a maximum speed of 2 m/sec. The pause time is 20 seconds. The number of nodes is kept fixed as 50. The number of communicating source-destination pairs is varied from 5 to 40. Data packets are transmitted at a rate of 4 packets/sec. The size of the time interval is taken as 40 seconds. The values of β, β’ used are 0.6, 0.5 respectively. Simulations are run for 1000 simulated seconds. The simulation environment is as explained in Section 2.4.2.

Each value in the table and graphs is the average of the values got in 10 simulation runs corresponding to 10 different scenarios. For each scenario, the communicating source-destination pairs and the initial positions of the nodes are changed randomly. The proposed protocol used in this section is referred to as VPRS_WT. The results are given in Table 4.2.

The packet delivery ratios for VPRS_WT is less than that for DSR for 5 and 10 connections. When the number of connections is increased from 20 to 40 there is a slight improvement in the packet delivery ratio from 2% to 6% (Figure 4.9).

The average end-to-end delay for VPRS_WT is similar to that of DSR when the number of sources is 5. But when the number of connections is increased from 10 to 40, it is seen that there is an improvement of about 6% in average end-to-end delay over DSR (Figure 4.10).
Table 4.2: Comparison of DSR and VPRS_WT for varying number of connections

<table>
<thead>
<tr>
<th>No. of Nodes</th>
<th>Pkt. delivery ratio (%) DSR</th>
<th>VPRS_WT</th>
<th>Avg. delay (secs) DSR</th>
<th>VPRS_WT</th>
<th>Normalized control overhead DSR</th>
<th>VPRS_WT</th>
<th>Avg. hopcount DSR</th>
<th>VPRS_WT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>99.77</td>
<td>95.57</td>
<td>0.0222</td>
<td>0.044</td>
<td>0.1799</td>
<td>0.3461</td>
<td>2.486</td>
<td>2.79</td>
</tr>
<tr>
<td>10</td>
<td>98.43</td>
<td>92.63</td>
<td>0.2234</td>
<td>0.253</td>
<td>0.7045</td>
<td>1</td>
<td>2.9162</td>
<td>3.35</td>
</tr>
<tr>
<td>20</td>
<td>63.7</td>
<td>64.57</td>
<td>3.5151</td>
<td>2.961</td>
<td>2.7863</td>
<td>2.4009</td>
<td>7.7483</td>
<td>7.57</td>
</tr>
<tr>
<td>30</td>
<td>47.64</td>
<td>49.23</td>
<td>5.5828</td>
<td>4.578</td>
<td>3.6473</td>
<td>3.023</td>
<td>9.719</td>
<td>9.74</td>
</tr>
<tr>
<td>40</td>
<td>37.89</td>
<td>40</td>
<td>6.0473</td>
<td>5.691</td>
<td>5.2105</td>
<td>4.2388</td>
<td>13.396</td>
<td>12.94</td>
</tr>
</tbody>
</table>

Figure 4.9: Comparison of packet delivery ratio of DSR and VPRS_WT for varying number of connections

The normalized control overhead for VPRS_WT is higher (not better) than that for DSR when the number of sources is 5, 10. With increase in the number of connections from 20 to 40, there is an average improvement of 14% to 19% over DSR (Figure 4.11).

The average hop count for VPRS_WT is not better than that for DSR when the number of sources is 5, 10. But when the number of connections is increased from 20 to 40, it is seen that there is a slight improvement of about 4% in average hop count over DSR (Figure 4.12).
Figure 4.10: Comparison of average end-to-end delay of DSR and VPRS_WT for varying number of connections

It is seen from Figure 4.13, Figure 4.14, Figure 4.15 and Figure 4.16 that the performance of VPRS_WT is better than DSR and DSR_{glb-loc} in packet delivery ratio, normalized control overhead and average end-to-end delay but not as good as TIME_WT in few of the cases. So, it is clearly seen that the introduction of time has improved the performance in the protocols TIME_WT and VPRS_WT.
Figure 4.11: Comparison of normalized control overhead of DSR and VPRS_WT for varying number of connections

Figure 4.12: Comparison of average hop count of DSR and VPRS_WT for varying number of connections
Figure 4.13: Comparison of packet delivery ratio of DSR, DSR_{glob-loc}, TIME_WT and VPRS_WT for varying number of connections

Figure 4.14: Comparison of end-to-end delay of DSR, DSR_{glob-loc}, TIME_WT and VPRS_WT for varying number of connections
Figure 4.15: Comparison of normalized control overhead of DSR, DSR_{glob-loc}, TIME_WT and VPRS_WT for varying number of connections

Figure 4.16: Comparison of average hop count of DSR, DSR_{glob-loc}, TIME_WT and VPRS_WT for varying number of connections
4.5 TEMPORAL INFORMATION SYSTEM AND HISTORY-BASED CACHING

4.5.1 History-based caching and application scenarios

In scenarios like an academic scenario or an office scenario, the movement of mobile nodes can be repetitive. If the history of the movement of the nodes is known, then the future positions of the nodes can be predicted.

So, when mobile nodes regularly repeat their mobility pattern, it is useful to cache routes based on this pattern. The caching strategy presented here uses the long term history of the previous movement behaviour of mobile nodes and is termed as history-based caching (Rajam and Siromoney 2002). History-based caching plays a role in scenarios where mobile nodes regularly repeat their mobility patterns. However, even caching strategies based on short term mobility prediction could also be of use in these scenarios.

A simple academic scenario is considered here for the studies. Several students attend classes in a university. Each student has a mobile node. At a certain time, for a fixed duration, he is at a particular location, till this class session is over. He then moves on to his next class and is available at that fixed location for a duration of time. His class schedule is based on a weekly timetable. On the same day the next week, he is likely to be at the same location. Each student has a mobility pattern of his own. Scenario files are generated based on this academic scenario and are used as the node movement patterns.

Figure 4.17 shows an example of the positions of a particular mobile node, say, $m_j$ at different time slots on different days of a week. The mobile node
is likely to be in the same positions next week on the same day. For example, $m_j$ is likely to be in position $X4, Y4$ in time slot 1 on all Mondays.

<table>
<thead>
<tr>
<th>Time Slot</th>
<th>Sunday</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Saturday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slot 1</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td></td>
</tr>
<tr>
<td>Slot 2</td>
<td>(X4, Y4)</td>
<td>(X1, Y1)</td>
<td>(X1, Y1)</td>
<td>(X1, Y1)</td>
<td>(X1, Y1)</td>
<td>(X1, Y1)</td>
<td></td>
</tr>
<tr>
<td>Slot 3</td>
<td>(X4, Y4)</td>
<td>(X2, Y2)</td>
<td>(X2, Y2)</td>
<td>(X3, Y3)</td>
<td>(X2, Y2)</td>
<td>(X3, Y3)</td>
<td></td>
</tr>
<tr>
<td>Slot 4</td>
<td>(X5, Y5)</td>
<td>(X1, Y1)</td>
<td>(X1, Y1)</td>
<td>(X2, Y2)</td>
<td>(X5, Y5)</td>
<td>(X3, Y3)</td>
<td></td>
</tr>
<tr>
<td>Slot 5</td>
<td>(X5, Y5)</td>
<td>(X2, Y2)</td>
<td>(X2, Y2)</td>
<td>(X2, Y2)</td>
<td>(X2, Y2)</td>
<td>(X2, Y2)</td>
<td></td>
</tr>
<tr>
<td>Slot 6</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td>(X4, Y4)</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.17: Example of positions of a mobile node for a week**

Let mobile node $m_i$ be the source node and $m_j$ be the destination. When $m_i$ learns routes to $m_j$, the time slot and the day are also stored along with the routes, so that in future, when a data packet is to be sent on the same day and the same time slot, the particular route can be used.

It is seen easily that the Temporal Information System defined in Section 4.1 fits this scenario. Each row in the information table corresponds to a route and the value of the attribute $a_r$, $V_{a_r}$ is a particular time slot on a particular day when this route was valid. $|V_{a_r}| = n$. Here, the value of $n$ is the number of time slots in a day times the number of days in a week.

Now, for each elementary set that is formed from the set of attributes $A$ (that is, the set of attributes without the time information), there are $n$ ele-
mentary sets, where the first elementary set consists of elements that occurred in time slot 1 on Sunday, the second elementary set of elements that occurred in time slot 2 on Sunday, and the last elementary set of elements that occurred in time slot 6 on Saturday.

In Weighted Temporal Information Systems explained in Section 4.2.1 the elementary sets in different time slots have weights.

In routing based on WTIS, more recent elementary sets were given higher weights than less recent elementary sets. The academic scenario can also be considered as a special case of the WTIS model, with a particular nature of weighting. In this academic scenario, less recent time intervals are assigned higher weights than the more recent time intervals.

If $n$ is the number of time slots in a week times the number of days in a week, then the $n + 1$th time slot previous to the current slot is given a higher weight than the slots from the current slot till the $n$th slot. Similarly, if $n'$ is the number of slots in a day, then higher weights can be assigned for slots that are multiples of $n'$ from the current slot.

4.5.2 Routing based on history-based caching

Routes are cached based on the day and time that the route was used. The cache is organized (divided) into a primary (short term) cache and a secondary (long term) cache. The primary cache is a path cache which caches full paths used by the node for sending data packets. Each cache entry in the secondary cache has a destination, a table of all possible time intervals in a week with the route to the destination during that interval.
Whenever a node uses a route to any destination it caches the route in the primary cache and the secondary cache. In the secondary cache, it adds the route in the table entry corresponding to the day and time at which the path was used for the corresponding destination. Full paths are cached. When a new route other than the one present in the secondary cache is found for a time and day, then the old route is replaced with the new route.

If the primary cache becomes full, a path from the cache is removed and the new path is put in its place. The victim path to be removed is chosen in the order in which the path was put into the cache. In the secondary cache a single path is maintained for particular time on a particular day for a destination. If a new route is learnt, the old route is replaced with the new route.

When a node needs a route to a destination, it first searches the primary cache and then the secondary cache. If a route for the specified destination is not available in both the caches, then it performs a route discovery similar to DSR. If there are more routes in the primary and the secondary, the shortest route is used. While searching in the secondary cache, it searches for a route to the destination for the current day and time.

4.6 SUMMARY

The notion of time was introduced in the information systems proposed earlier and presented as a generic temporal information system. The notion of assigning different weights to elementary sets in different intervals of time was discussed. The proposed protocol TIME_WT chooses a next hop based on weighted elementary sets. The performance of TIME_WT was found to be better than DSR and DSR_{glob-loc}. 
Another proposed protocol VPRS_WT used the threshold mechanism from VPRS in temporal information systems to find the next hop. The performance of VPRS_WT was better than DSR and DSR_glb-loc but not as good as TIME_WT.

Caching of data based on history was seen to be helpful in scenarios, such as academic scenario, where mobile nodes repeat their movement patterns. The proposed weighted temporal information system was seen to fit this scenario.