Fast-Converging Distance Vector Routing for Wireless Mesh Networks

Yangcheng Huang
Department of Computer Science
University College London
London WC1E 6BT, UK
yangchenghuang@ieee.org

Saleem Bhatti
School of Computer Science
University of St Andrews
Fife KY16 9SX, UK
saleem@cs.st-andrews.ac.uk

Abstract

A major concern about distance-vector routing protocols for wireless mesh networks is its slow convergence in the presence of link changes, which can potentially degrade network stability. This paper studies the impact of update intervals on network convergence and proposes a fast-converging distance-vector routing algorithm. Our simulation results have shown that the proposed algorithm could effectively reduce convergence latency and improve throughput without leading to a significant increase in control overhead.

1. Introduction

Wireless Mesh Networks (WMNs) are emerging as a key technology for next generation wireless networking. A WMN is characterised by dynamic self-organisation, self-configuration and self-healing. This means that they should be easy and fast to deploy; highly scalable with increase in the number of nodes; permit reliable and cost-effective network deployment under very diverse environments; and allow provision of better coverage and capacity to stationary and mobile users.

Existing routing protocols (including link-state routing, on-demand routing and source routing) tend to broadcast *network-wide* control messages in order to discover and maintain routes between end nodes. Therefore, these protocols can have high a message overhead when used in WMNs.

Distance-vector routing protocols like DSDV [9], however, have a lower overhead since the nodes running distance-vector protocols only exchange control messages with their *adjacent nodes*.

The Destination-Sequenced Distance-Vector (DSDV) routing protocol is based on Bellman-Ford Routing, and maintains in each node a routing table that lists all available destinations, the number of hops in each path and a sequence number. The node sends routing updates periodically to its neighbouring nodes, when the routing table changes (a full update) or when entries are modified (a partial update).

For wireless mesh networks the major concern for a distance vector protocol is its slow convergence in the presence of link changes, which can potentially degrade network stability. Due to the lack of central management of mesh networks and the unreliable nature of wireless links, it is expected that link breaks or link quality changes in mesh networks will not be rare events. In such environments, distance-vector routing [9] converges much slower than link-state routing and can potentially degrade network stability.

There is still no clear understanding of the factors that affect the convergence of distance vector protocols in wireless mesh networks. It is critical to investigate possible algorithms to improve convergence of distance-vector routing protocols.

This paper investigates the route convergence problem in the context of wireless mesh networks. Specifically, the impact of update intervals on network convergence is studied with a combination of model-based analysis and simulation-based performance evaluations. A fast-converging distance-vector routing algorithm is proposed to improve route convergence speed. Our simulation results have shown that the proposed algorithms could effectively reduce convergence latency, improve routing throughput without leading to a significant increase in control overhead.

There have been several adaptive routing approaches for wireless mesh networks. For example, Boppana et al [3] proposed an Adaptive Distance Vector (ADV) routing algorithm. Like DSDV [9], ADV exchanges route updates between neighbouring nodes. However, only the route entries of active nodes (nodes currently forwarding packets) are advertised, which reduces the size of route update messages. In addition, route updates are triggered only under certain conditions, such as route unavailability. Trigger thresholds are used to determine whether a "partial update" or a "full update" is advertised.

These adaptive approaches are usually targeted to reduce control overhead. In addition, the ADV algorithm has several shortcomings. Firstly, although adaptive to mobility conditions, the topology update advertisement and its performance are determined by "constant trigger thresholds" that need to be configured. Secondly, triggered by network events, the route update frequency might increase quickly with node mobility, which leads to larger overheads than periodic updates. Thirdly, since only partial route information is maintained, it might take longer for a new connection to find a valid route. Due to these problems, the performance of ADV might still be under question, especially when compared with other routing protocols such as AODV [8] and OLSR [5] under various factors such as node mobility.

The rest of the paper is organised as follows. Section 2 gives some background information on existing distance-vector routing algorithms. Section 3 presents an analytic study on the convergence latency of existing distance-vector algorithms. Section 4 gives the detailed description of the proposed algorithms. Section 5 introduces the simulation configurations used in this study. Section 6 presents our observations based on simulations using NS2. Conclusions are summarised in Section 7.

2. Distance Vector Routing

The Destination Sequenced Distance-Vector (DSDV) [9] routing protocol is a table-driven proactive routing algorithm based on the Bellman-Ford routing algorithm. Each node in the network maintains a routing table that records distance vectors, i.e. the number of hops to all of the possible destinations within the network and the corresponding next-hop nodes.

The main improvement made to the basic Bellman-Ford algorithm is the loop-free property by marking nodes with sequence numbers, which are used to distinguish stale routes from new ones. A DSDV update packet contains an unique sequence number (SN). The transmitter assigns this SN, and the receiver selects the packet with the highest SN, i.e. the most recent route. The route labelled with the most recent sequence number is always used. In the event that two updates have the same sequence number, the route with the best metric (i.e. lowest number of hops) is used in order to optimise the path. An advantage of DSDV is that in relatively stable networks like a Wireless Personal Area Network (WPAN), incremental updates are sent to avoid extra traffic. Its main disadvantage is that in fast changing networks, like mobile networks, as the number of incremental update packets increases rapidly, then full dumps are preferred, or DSDV requires *bidirectional links* to operate, so that links are treated as symmetric in terms of metrics and so routing state is reduced.

- *Topology update strategies:* DSDV requires each node to advertise its own routing table by broadcasting its entries to each of its current neighbours *locally*. In order to reduce the amount of state information carried in each update and help alleviate the potentially large amount of topology update traffic, DSDV employs two types of update packets.

  Full updates carry all available routing information and might require multiple network protocol data units (NPDUs). Full updates can be transmitted relatively infrequently when no movement of mobile nodes occurs.

  Incremental updates carry only information that has changed since the last full update. The size of incremental updates is smaller than that of full updates, and therefore can fit into a standard network protocol data unit (NPDU). When movement becomes frequent, the size of incremental updates increases, and approaches the size of a NPDU. Then, a full update can be scheduled so that the next incremental update will be smaller.

- *Neighbour sensing schemes:* new neighbours can be detected by exchanging periodically the routing tables. There are two proposals for link breakage detection: either by use of the layer-2 protocol, or by use of a time-out (if no routing table updates have been received for a period from an existing neighbour). When a link to a next hop is broken, any route through that next hop is immediately assigned a metric of ∞ so that it should not be selected for data delivery.

3. Convergence Issue of Distance Vector Protocols

Here, we analyse the route convergence latency L, i.e. the period from the occurrence of the change (i.e. when route inconsistency occurs) to the time the nodes in the network update their route state repositories (i.e. converging to route state consistency again).

A node running a distance-vector protocol detects link changes and updates its route tables. These link changes are advertised by exchanging route tables between the neighbouring nodes. Therefore, the route convergence latency, L, is the sum of link detection latency, L_d, and route advertisement latency, L_a.

We assume that:

- The message exchange events in each node are independent. The intervals of message broadcasting in any two nodes conform to a uniform random distribution.

- The change event of links is an independent, identically distributed Poisson process with arrival rate λ.

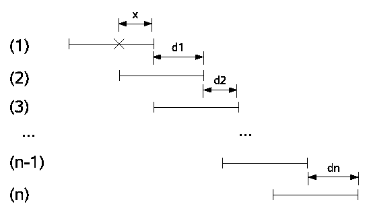


Figure 1. Convergence of Distance Vector Protocols

The assumption is reasonable, if the node degree is small and the nodes are moving randomly so that the process of route change is totally random.

Let r be the refresh interval. Let d_i ($i = 1, 2, 3...$) be the time gaps between successive messages sent by neighbouring nodes (as shown in Fig 1).

Our previous work [7] has revealed the expected link detection rate as follows[1].

$$E(L_d) = r + \frac{e^{-r\lambda} - 1}{\lambda} \tag{1}$$

Using the assumption, $d_i \sim U(0, r)$:

$$L_a = d_1 + d_2 + ... + d_m \tag{2}$$

Therefore:

$$E(L_a) = m\frac{r}{2} \tag{3}$$

From this we can see that, the route advertisement latency, L_a, is directly proportional to the refresh interval, r.

We calculate the route convergence latency as follows.

$$\begin{aligned} E(L) &= E(L_d) + E(L_a) \\ &= r + \frac{e^{-r\lambda} - 1}{\lambda} + m\frac{r}{2} \end{aligned} \tag{4}$$

In summary, the analysis above shows that the update interval has a significant impact on route convergence. Specifically, the convergence speed of routes is approximately inversely proportional to the route update intervals. So, tuning the update interval may result in improvements in routing performance.

[1]We assume no packet loss. Please refer to [7] for more details.

One simple method to improve the route convergence speed is through reducing the route update interval. However, increasing the frequency of message dissemination in this way increases control overhead and resource consumption (e.g. bandwidth, CPU, battery power and memory). So, it is necessary to find efficient methods in tuning update intervals in order to improve route convergence speed without introducing excessive overhead.

4. Fast-Converging Distance-Vector Routing

In this section, we propose a fast-converging distance-vector algorithm (FCDV) to adapt dynamically the route update intervals. It has been inspired by control-theoretic adaptive mechanisms similar to those widely adopted in the Internet, e.g. Additive Increase Multiplicative Decrease (AIMD) of TCP, which is used to adjust sending rates in response to network congestion. Our approach in this algorithm uses a *Additive-Increase Multiplicative-Decrease (AIMD)* controller to adapt the route update interval, r, to the conditions of node mobility and data traffic.

Briefly, the refresh interval r is divided by a factor α ($\alpha > 1$) if node mobility or packet drop rate increases, and otherwise incremented by a factor β. By aggressively reducing r when the packet failure rate and the network change rate increase, the routing algorithm improves route convergence, which reduces data packet drops and increases route availability. Whenever the network change rate decreases, the routing algorithm lowers the refresh frequency conservatively until it finally reaches a steady state. Please refer to Algorithm 1 for more details.

Algorithm 1 *Fast-Converging Distance-Vector Routing*

```
Input: $r_0 < \frac{1}{\beta}$
  $r \leftarrow r_0$
  loop
    if $(rt\_chg\_cnt > prev\_chg\_cnt)$ then
      if $(\frac{rt\_chg\_cnt - prev\_chg\_cnt}{prev\_chg\_cnt - prev2\_chg\_cnt} > 1)$ then
        $r \leftarrow \max(\frac{r}{\alpha}, r_{min})$
      end if
    end if
    $r \leftarrow \min(\frac{r}{1 - r * \beta}, r_{max})$
  end loop
```

5. Performance Analysis

We integrate our proposed algorithms with the DSDV implementation which runs in version 2.9 of NS2 [1] and uses the ad-hoc networking extensions provided by CMU [2]. The detailed configuration is shown in Table 1.

Table 1. MAC/PHY Layer Configurations

MAC Protocol	IEEE 802.11
Radio Propagation Type	TwoRayGround
Interface Queue Type	DropTailPriQueu
Antenna Model	OmniAntenna
Radio Radius	250m
Channel Capacity	2Mbits
Interface Queue Length	50

We use a network consisting of n nodes: $n = 20$ to simulate a low-density network, $n = 50$ to simulate a high-density network. Nodes are placed in a 1000×1000 m^2 field. All simulations run for 100s. The radio range (radio radius) was 250m.

We use the Random Trip Mobility Model, "... a generic mobility model that generalizes random waypoint and random walk to realistic scenarios ..." [4] and performs perfect initialisation. Unlike other random mobility models, Random Trip reaches a steady-state distribution without a long transient phase and there is no need to discard initial sets of observations. Manhattan Mobility Model is also used under different scenarios.

The mean node speed, v, ranges between 1m/s to 30m/s. For example, when the mean node speed is 20m/s the individual node speeds are uniformly distributed between 0m/s and 40m/s. The average node pause time is set to 5s.

A randomly distributed Constant Bit Rate (CBR) traffic model is used which allows every node in the network to be a potential traffic source and destination. The rate of each CBR traffic is 10kb/s. The CBR packet size is fixed at 512 bytes. There are at least $n/2$ data flows that cover almost every node.

For each datum point presented, 100 random mobility scenarios are generated. The simulation results are thereafter statistically presented with the mean of the metrics and the errors. This reduces the chance that the observations are dominated by a certain scenario which favours one protocol over another.

In each simulation, we measure each CBR flow's throughput and control traffic overhead and then calculate the mean performance of each metric as the result of the simulation.

Throughput is considered as the most straight-forward metric for MANET routing protocols [6], and is computed as the amount of data transferred (in bytes) divided by the simulated data transfer time (the time interval from sending the first CBR packet to receiving the last CBR packet).

Considering the broadcast nature of the control message delivery, the overhead is calculated by summing up the size of all the control packets *received* by each node during the whole simulation period.

6. Observations

In this section, we compare the routing performance of the proposed adaptive routing algorithms with that of a standard proactive routing protocol, and present the observations under the variation of selected parameters, such as node velocity and node density.

6.1. Impact of Update Intervals on Routing Performance

From Fig 2(a) and Fig 3(a) we can see that, the average throughput decreases with the increase of update intervals. This matches well our analytic results in Section 3.

6.2. Dynamic Update Intervals

As shown in Fig 4 and Fig 5, the proposed FCDV algorithm performs as well as standard DSDV with smaller update interval (i.e. $intVal = 1$) but with a much lower overhead. This demonstrates clearly the benefits of our *FCDV* algorithm described in Section 4. The control overhead is reduced because the route update interval is adjusted automatically to a large value when there are no network changes.

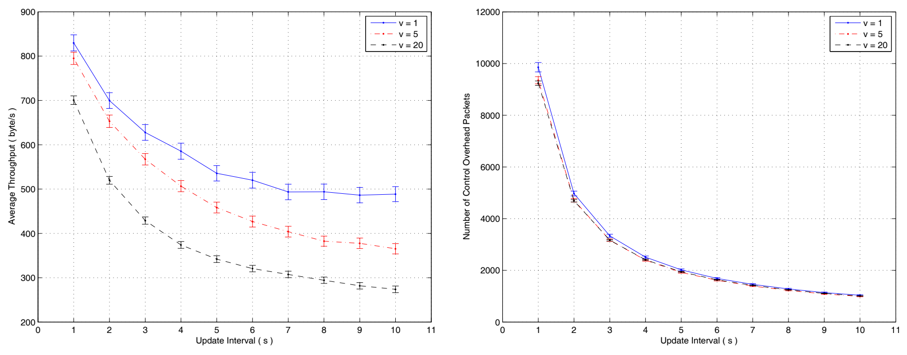
Compared to standard DSDV with larger interval (i.e. $intVal = 2$), FCDV shows good adaptability to node mobility. With the increase in node mobility, the throughput drop of FCDV is less significant than standard DSDV with larger interval. For example, as shown in Fig 5(a), when the node velocity increases from 10m/s to 20m/s, FCDV has a $\sim$10% performance drop, while standard DSDV (i.e. $intVal = 2$) has a drop of $\sim$20%. This matches our expectations. When the link change rate increases, the *FCDV* algorithm increases the route update rate so that route inconsistency can be recovered very quickly after it occurs.

On the other hand, FCDV has more overhead than standard DSDV with larger interval (i.e. $intVal = 2$). As shown in Fig 4(b) and Fig 5(b), the overhead of FCDV is $\sim$40% more than standard DSDV with larger interval. However, such overhead increase introduced by FCDV is still much smaller than the reducing update interval to $1s$ with DSDV.

To summarise, the simulation results show that *FCDV* improves the standard DSDV proactive routing algorithm in terms of the balance of throughput and overhead under the conditions simulated.

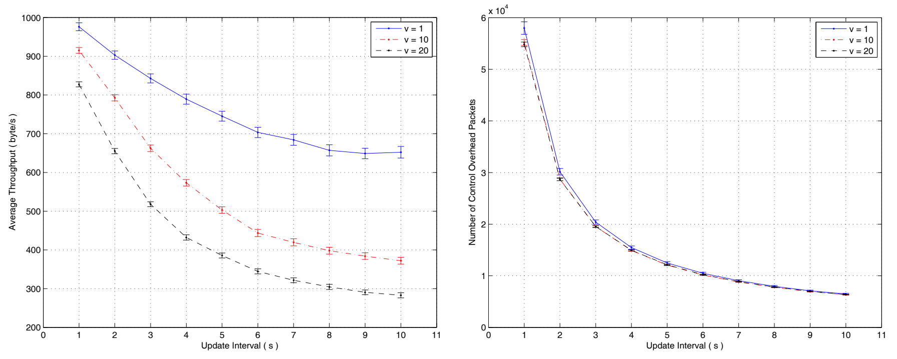
7. Conclusions

This paper investigates the route convergence problem in the context of wireless mesh networks. Especially, the impact of update intervals on network convergence is studied. A fast-converging distance vector algorithm is proposed to



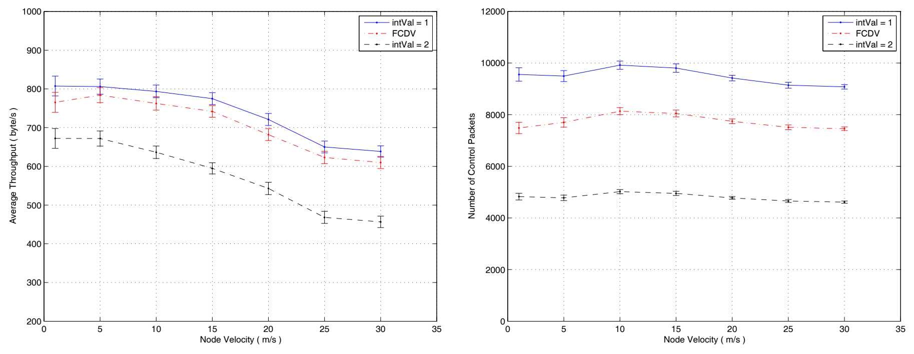
(a) Throughput (b) Traffic Overhead

Figure 2. Impact of Update Intervals on DSDV Performance (*n*=20)



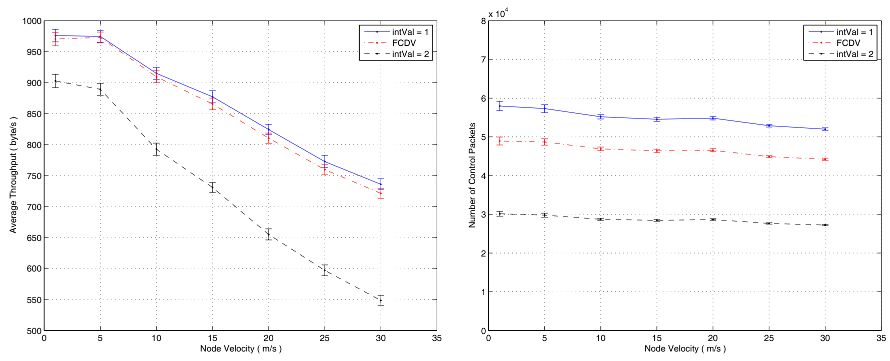
(a) Throughput (b) Traffic Overhead

Figure 3. Impact of Update Intervals on DSDV Performance (*n*=50)



(a) Throughput (b) Overhead

Figure 4. FCDV Performance (*n*=20 α=2 β=0.1)



(a) Throughput (b) Overhead

Figure 5. FCDV Performance (n=50 α=2 β=0.1)

improve route convergence speed. Our simulation results have shown that the proposed algorithm could effectively reduce convergence latency, improve routing throughput without leading to a significant increase in control overhead.

One potential problem with the proposed FCDV algorithm is the configuration of its parameters (i.e. α & β). We are currently working on a self-tuning distance-vector algorithm, which requires no manual configuration.

This work focuses on convergence issue caused by node mobility. Further analysis of FCDV (such as scalability) will appear in our future work.

The original data, the source code and the scripts used in this study are all available from the authors on request.

References

[1] Ns2 website. http://www.isi.edu/nsnam/ns/.

[2] The rice monarch project: Wireless and mobility extensions to ns-2. http://www.monarch.cs.rice.edu/cmu-ns.html.

[3] R. V. Boppana and S. Konduru. An Adaptive Distance Vector Routing Algorithm for Mobile Ad Hoc Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1753–1762, Anchorage, Alaska, USA, April 2001.

[4] J. Y. L. Boudec and M. Vojnovic. Perfect simulation and stationarity of a class of mobility models. In *Proc. IEEE Infocom Conference*, Miami, USA, 2005.

[5] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, a. Qayyum, and L. Viennot. Optimized link state routing protocol. In *IEEE INMIC Pakistan*, 2001. Best paper award.

[6] S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evauluation considerations. Request for Comments 2501, IETF, January 1999.

[7] Y. Huang and S. Bhatti. Model Based Analysis of Soft State Signaling Protocols. In *Proceedings of the London Communications Symposium*, London, UK, September 2005.

[8] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (aodv) routing. Request for Comments 3561, IETF, July 2003.

[9] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *SIGCOMM*, New York, NY, USA, 1994.