The LeyLine Protocol

A Routing Protocol for Resource Constrained Wireless Sensor Networks

Muhammad Shayan
17100212@lums.edu.pk

A report detailing the work done as
Senior Year Project

under the supervision of
Dr Zartash Afzal Uzmi

1st May 2017

The LeyLine Protocol

A routing Protocol for Resource Constrained Sensor Networks

Muhammad Shayan

Abstract

We propose a layer 3 protocol that makes the use of ley-lines in order to ensure that the state that is required to be stored for routing is constant in nodes in a wireless sensor networks. These nodes are known to be resource constrained in terms of memory and energy therefore there is immense need for a routing protocol that requires minimum memory and energy consumption. Most existing routing protocols exhibit linear or polynomial growth in their routing tables which is not suitable for sensor nodes. Our routing protocol in its simplest form will require that these nodes maintain only two routing entries, the previous and the next node on the line. The resulting stretch from storing such a limited amount of entries will be O(N), however, the protocol allows for additional entries to be stored in the forwarding tables. These skip entries ensures that a suitable tradeoff is achieved between routing state and stretch resulting in minimum energy consumption. Apart from determining the amount of entries required to achieve the perfect tradeoff, we compare the performance of the ley line protocol with existing protocols (AODV) in terms of average number of hops, memory consumption and overhead. We accomplish this through TOSSIM simulations.

Contents

1 Background and Motivation **1**
- 1.1 Memory Constraints . . . 1
- 1.2 Low Power Consumption . . . 2
- 1.3 Failure of Traditional Protocols . . . 2
- 1.4 Random Deployment of Sensors . . . 2
- 1.5 Lack of focus on any to any routing . . . 3
- 1.6 Our work . . . 3

2 Design, Implementation and Testing **4**
- 2.1 Creation of a ley-line . . . 5
  - 2.1.1 Step 1: Selection of Starting Node . . . 5
  - 2.1.2 Step 2: Sending out invitation for next node . . . 5
  - 2.1.3 Step 3: Reaching a leaf node . . . 6
  - 2.1.4 Step 4: Back Propagation and Including Potential Nodes . . . 6
  - 2.1.5 Step 5: Starting node repeats the above process for another node . . . 6
  - 2.1.6 Step 6: Search for intersection node . . . 7
  - 2.1.7 Step 7:Build a line at the intersection . . . 7
  - 2.1.8 Step 8: End of the line building process . . . 7
- 2.2 Addressing scheme . . . 8
- 2.3 Shortcut Entries . . . 9
- 2.4 Testing . . . 9

3 Experimental Work and Conclusion **11**
- 3.1 Simulation Methodology . . . 11
- 3.2 Determining optimal number of entries . . . 12
- 3.3 LeyLine vs AODV . . . 13
  - 3.3.1 Overhead . . . 13
  - 3.3.2 Average Path Length . . . 13
  - 3.3.3 Memory Consumption . . . 13
- 3.4 Conclusion . . . 14

Chapter 1

Background and Motivation

Wireless Sensor Networks are expected to become an integral part of the Internet of Things – the next generation of the Internet. The next billion devices inter-networking with each other are expected to be low power tiny wireless devices like sensors and actuators. (1) These devices have inherent resource limitations that poses a challenge for all layers in the protocol stack. For the routing layer, there is a plethora of routing protocols available and there is a lack of awareness as to which protocols is to be used in which situation.

1.1 Memory Constraints

A wireless sensor network is comprised of nodes that are small devices with limited memory and storage space. Usually, a sensor node is composed of a RAM and a flash memory. A typical Telos node has only 10 kilobytes of RAM (2) . The small memory capacity limits routing table size, buffer capacity and other state. Moreover, usually there is not enough space to execute large programs after loading the operating system and the application programs (3). Therefore, conventional routing algorithms that require a large amount of expensive memory for table maintenance are hence deemed unsuitable for sensor networks (4) . Examples of such protocols include AODV (5) , OLSR (6) and BGP because even though they produce shortest paths of which are defined to have a stretch of 1 but they exhibit linear or polynomial growth rates of routing tables. Hence, any routing protocol that is proposed must take care not to store large number of entries in its implementation. For this purpose, the simplest form of our protocol proposes that only two entries be stored in the forwarding table so that the precious memory of the nodes can be utilized elsewhere.

1.2 Low Power Consumption

Apart from the nodes facing a memory constraint, sensor networks also have a low power consumption requirement. The nodes are powered by batteries which generally supply limited power and are usually irreplaceable. Hence while traditional routing protocols tend to focus on the quality of service provisions, routing protocols must turn their attention primarily to power consumption. Hence it is essential that they possess in-built trade off mechanisms that give the end user the option of prolonging the lifetime of the network at the cost of lower throughput or higher transmission delay (7) . We wish to provide the best tradeoff in the Ley Line Protocol by determining the optimal entries in the forwardingtable that provide a reasonable stretch. This would lead to routing consuming the optimal amount of energy needed for delivery of data.

1.3 Failure of Traditional Protocols

It is well known that the traditional routing protocols of the Internet are insufficient to cater to wireless sensor networks. In the Internet, interior gateway protocols (IGP's) are used to route packets within the network of a company or organization. These types of protocols can be broadly classified into two categories - the Distance Vector Routing Protocols e.g. RIP and Link State Routing Protocols e.g OSPF. Both these types of algorithms use the well-known Bellman Ford or Djikstra algorithms to determine the shortest path from one node to all other nodes in the network considering the number of hops as the optimization metric. Although these algorithms are scalable, however, they are computationally expensive and introduce a large amount of overhead resulting in low energy efficiency (8). Therefore, there is a need for a routing protocol that is sensitive to the limited energy constraints of the nodes in a sensor network.

1.4 Random Deployment of Sensors

The placement of sensors is difficult to determine precisely and their deployment is random at best. They might be evenly distributed or densely deployed in one region compared to the other (9). Consequently, it is not possible to control the connectivity as for wired networks and therefore each sensor node does not know its position within a topology apriori. For routing purposes, it is essential nodes become aware of their neighbors automatically after deployment. Keeping in mind the memory and energy strain caused by such an overhead of discovering the neighbors, Ley-line aims to minimize this overhead by just requiring that a node is required to know a limited amount of

neighbors for routing purposes. Otherwise, a routing protocol that requires knowledge of just 1-hop neighbors can easily exhaust memory in a dense topology.

1.5 Lack of focus on any to any routing

Furthermore, most protocols proposed for wireless sensor networks assume that all routing that needs to occur is between the sink and all the source nodes. Although, this holds true for some cases, however, there is also a need for any to any routing because it is important for network management, network diagnosis, data-centric storage and generalizes all other specialized communication patterns (10). The existing any-to-any routing protocols require nodes to keep track of all neighbors and to keep state for routing paths that pass through a node. The high node density in many sensor networks make this requirement impossible to fulfill. The focus of existing protocols with the exception of a few is only on link constraints however sensor network protocols need to address the memory and power constraints too.

1.6 Our work

In order to address these problems, this project aims to implement and evaluate the Ley Line Routing Protocol. This protocol limits the amount of state per node to a constant number and then optimizes stretch within that limit. Ley Line works by building a single routing path, a logical line, which covers the entire network. Most links on the line connect neighbors, but some connect "remote" nodes. In its simplest form, each node only connects with two neighbors, the right and left nodes on the line, and the routing stretch can be O(n). Nodes can maintain additional routing entries, for routes through some other physical neighbors, which enable L2R to skip further along the line. These skip entries significantly reduce routing stretch. While it is possible for LeyLine to be used in different network settings, this project is focused solely on making it suitable for sensor networks.

Chapter 2

Design, Implementation and Testing

Ideally, in order to consume minimum memory, we would like our protocol to store just two entries. This means that we need to find a Hamiltonian Path in the connectivity graph of the nodes. However, finding a Hamiltonian path has been deemed to be an NP complete problem in computer science. Moreover, not all graphs have a Hamiltonian Path that covers all the nodes therefore we introduce the concept of a ley-line. A ley-line is like a hamiltonian path, but it covers all or some nodes of a given graph. A simple way to form a ley line is to start from any node and recursively include a random neighbor to the line until we cannot include anymore nodes. Since the leyline might only cover a subset of the graph, we introduce a node with degree ≥ 3 called an intersection node. Intersection nodes can be on two or more ley lines at the same time. Multiple ley lines can share the same path and then "fork", at intersection nodes, to go in different directions on a graph. It is straight forward to see how a combination of ley lines can cover all nodes of any connected graph. Figure 2.1 explains this concept visually.

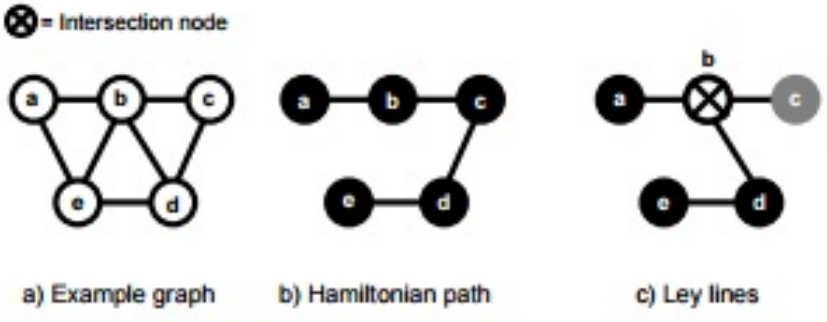

FIGURE 2.1: Hamiltonian Path vs Ley-Lines

2.1 Creation of a ley-line

Once, we had defined the concept of a ley-line to use in solving our problem, we came up with an algorithm to implement this concept using *nesC* for *TinyOS*. nesC requires using an event-driven programming model to write our applications. This made the job of implementing the protocol quite easier since it enabled us to define the receipt of different messages as events and define actions to be taken by the node as soon as such a message is received. The following steps describe in detail how a ley-line covering all nodes is developed using our implementation.

2.1.1 Step 1: Selection of Starting Node

The first step of our protocol requires the selection of a random node to start the creation of a ley-line. In order to simplify the process and reduce the overhead involved in a distributed selection process, we decided to assign the responsibility to a node that is assigned a TOSNODEID of 1 at the time of deployment. The TOSNODEID is unique to every node and is used to send messages from one node to the next. Fig 2.2 illustrates this step.

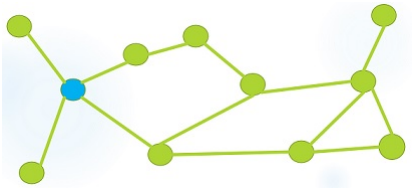

FIGURE 2.2: Selection of starting node

2.1.2 Step 2: Sending out invitation for next node

In order to add the next node on the line, the starting node sends out a broadcast asking for potential nodes to be the next nodes on the line. The first node to reply back is the one to get included on the line.

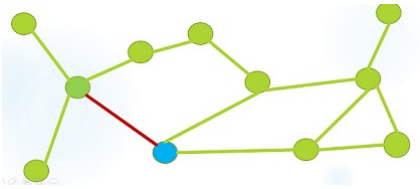

FIGURE 2.3: Sending out invitation for next node

2.1.3 Step 3: Reaching a leaf node

The process of adding nodes to the ley-line continues until we reach a leaf node that doesn't receive a reply from any of its nodes. This node is designated as a leaf-node on the line that has been developed so far.

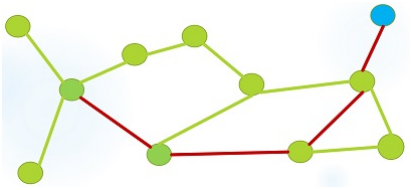

FIGURE 2.4: Reaching a leaf node

2.1.4 Step 4: Back Propagation and Including Potential Nodes

A message is propagated back to the starting node along the line that has already been built. Along the way, the nodes also ensure that any potential node that can be included on the line is included. A node qualifies to be included if it has two neighbors that occur successively on the ley-line. Before sending out the back propagation message, the node sends a broadcast asking its neighbors to send a request to be included on the line to its successor on the line. If the successor receives such a request, the node that sends the request is included on the line when the back propagation message is received by the successor.

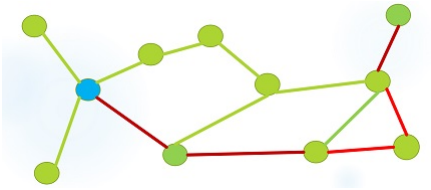

FIGURE 2.5: Back Propagation and including potential nodes

2.1.5 Step 5: Starting node repeats the above process for another node

Steps 1-4 are carried out after the starting node chooses another one of its neighbors. The line that has been built at the end of this step is called the core LeyLine.

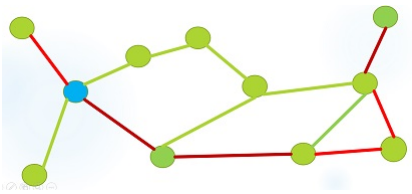

FIGURE 2.6: Repeat process in other direction

2.1.6 Step 6: Search for intersection node

Since its impossible for one line to cover all nodes, we need to search for potential intersection node to serve as intersection points for the ley-lines. Therefore, a Search for Intersection(SIN) message is propagated from one leaf node to the other. On the way, if a node has any undiscovered neighbors that node is selected to become the intersection node. A broadcast message is sent out by the node on receiving the SIN message to identify any undiscovered neighbors.

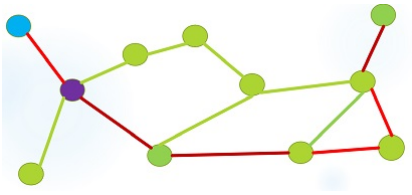

FIGURE 2.7: Search for intersection nodes

2.1.7 Step 7:Build a line at the intersection

Steps 1-6 are repeated with the intersection node becoming the starting node. The intersection node propagates the SIN Packet forward only after the line building process is complete at the intersection node.

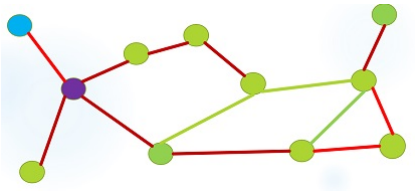

FIGURE 2.8: Build a line at the intersection

2.1.8 Step 8: End of the line building process

When the SIN packet reaches the leaf node of the leyLine the line building process has reached its end. The end-result can be said to be a depth-first spanning tree that has been developed in a distributed manner.

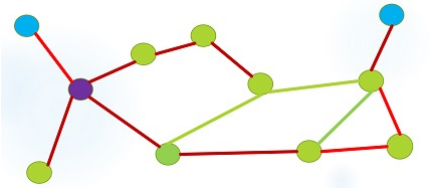

FIGURE 2.9: End of Ley-Line Building

2.2 Addressing scheme

Once the Ley-Line is built, nodes would need to make routing decisions when helping a packet on its way to the destination. The nodes would only be able to do this if the addresses are assigned in an increasing order from one end of the core Ley-Line to another. Therefore, we have also implemented a distributed addressing scheme as part of the protocol.

When a leaf node receives a SIN packet marking the end of the Ley-Line development, the SIN packet makes the leaf node aware of the number of nodes in the network. This is because while the line was being developed the messages being exchanged by the nodes were keeping track of the number of nodes added to the Ley-Line.

Equipped with the address space and the number of nodes, the leaf node finds the addressing interval by just dividing address space with the number of nodes. The leaf node assigns itself an address equal to the interval and sends a message to the next node on the line that consists of the address interval and the address of the leaf node. The next node calculates its own address and assigns it to itself. The message is propagated along the ley-line and every node is assigned an address used to make routing decisions.

When the Assign Address (AA) Message reaches an intersection node, the node ensures that addresses are first assigned in an increasing order to the nodes on the lines for which it acts an intersection point. Afterwards, it assigns an address to itself before propagating the message forward on the core LeyLine. This ensures that the spanning tree developed acts as only one virtual line for routing purposes. This is illustrated in the figure below.

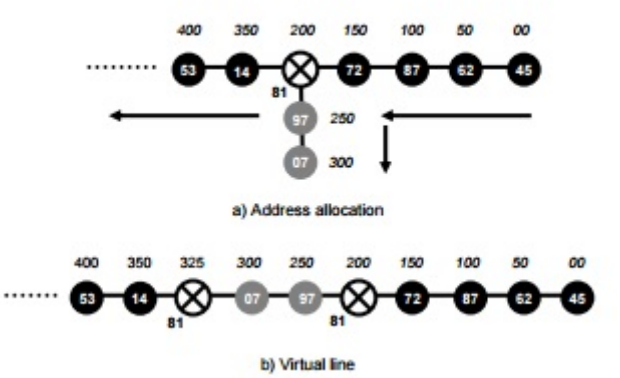

FIGURE 2.10: Layer-3 addressing

2.3 Shortcut Entries

The process of routing after storing such a limited number of entries would result in a routing stretch of O(N). Keeping in view the low power consumption of sensor networks in mind, this stretch would be infeasible for sensor networks. Therefore, to reduce the routing stretch, we provide each node with a fixed number of shortcut entries of one hop neighbors to use while routing. The figure below shows how nodes can use shortcuts to reduce the stretch. Determining the optimal number of entries to reduce the routing stretch remains to be determined through experiments.

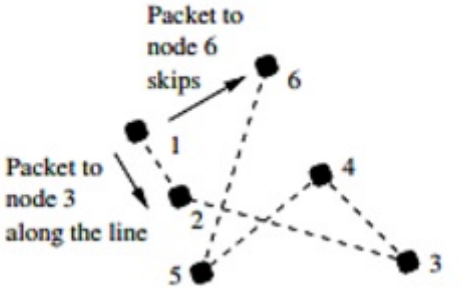

FIGURE 2.11: A panther is always watching

2.4 Testing

A thorough testing of the protocol was carried out in a simulated environment. The simulators used for carrying out the testing were TOSSIM and Cooja. The simulators allowed us to test the protocol on 10 different random deployments of 300 nodes. The

results of the testing showed that our protocol was able to cover all the nodes in the resulting Ley-Line.

Chapter 3

Experimental Work and Conclusion

In this section, we evaluate the efficiency and scalability of LeyLine by comparing it with AODV. AODV is the most popular shortest path routing algorithms that consumes a lot of memory but is extremely efficient when it comes to stretch. We use TOSSIM simulations, a packet-level simulator to present our TOSSIM simulations allow us to study the performance in more realistic large-scale wireless network. We use the AODV implementation [25] in TinyOS for TOSSIM simulation. However, before comparing our protocol with AODV we must first determine the optimal number of skip entries required by our protocol to perform at its best in the simulated conditions.

3.1 Simulation Methodology

In order to study the performance of the protocol, we deploy N nodes where $10 \leq N \leq 100$ in an area of $200m^2$. Every node has a fixed communication range R i.e it cannot communicate with any node outside R. It is also assumed that there is no packet loss, collision or network congestion. We use the following performance metrics to quantify the efficiency and robustness of the routing protocol.

- Average Stretch: The ratio of number of hops it takes on average for a message to be delivered to its destination with the length of the shortest path.
- State stored: The number of entries stored by the routing tables for routing.
- Overhead: The amount of control messages that need to be sent to set up the forwarding tables for routing.

For every value of N, we compute the above metrics by taking the average value of 10 different N-nodes topologies.

3.2 Determining optimal number of entries

In order to determine the optimal number of routing entries for our topologies, we counted the number of hops it took on average for a message to be delivered from a certain source to destination. We varied the number of shortcut entries from 0 to 20. Our results below show that the optimal number of entries in the forwarding table is 5-10 because increasing the number of entries from this point onwards does not result in significant savings in stretch.

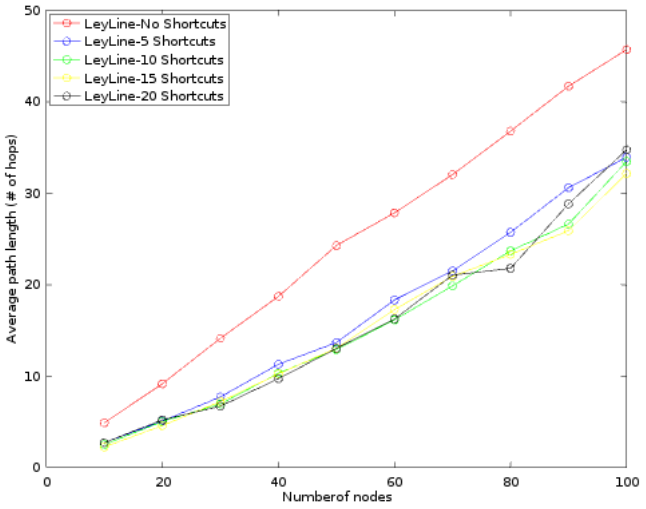

FIGURE 3.1: Variation of pathlength with the number of shortcuts

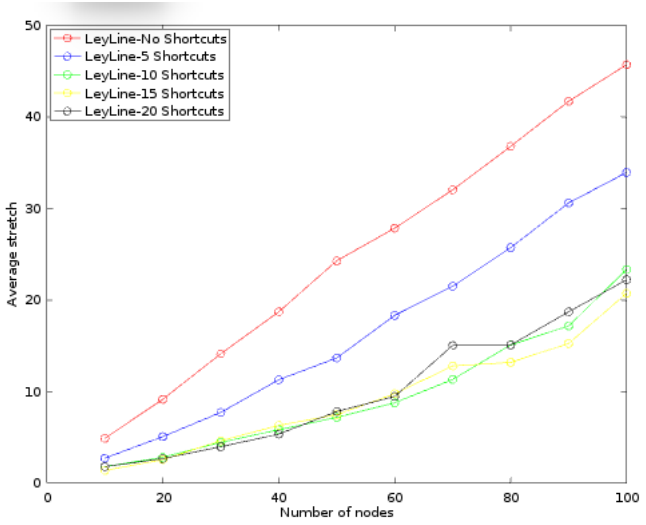

FIGURE 3.2: Variation of pathstretch with the number of shortcuts

3.3 LeyLine vs AODV

In this section we look at the performance of the LeyLine protocol compared to the Ad-hoc on demand distance vector routing protocol (AODV). We compare their performance in terms of overhead, stretch and memory consumption.

3.3.1 Overhead

Before routing can take place in either protocol, the routing tables must be filled so that the nodes can make routing decisions. A comparison of the number of messages it takes to populate its routing tables shows us that LeyLine takes half as much messages as AODV.

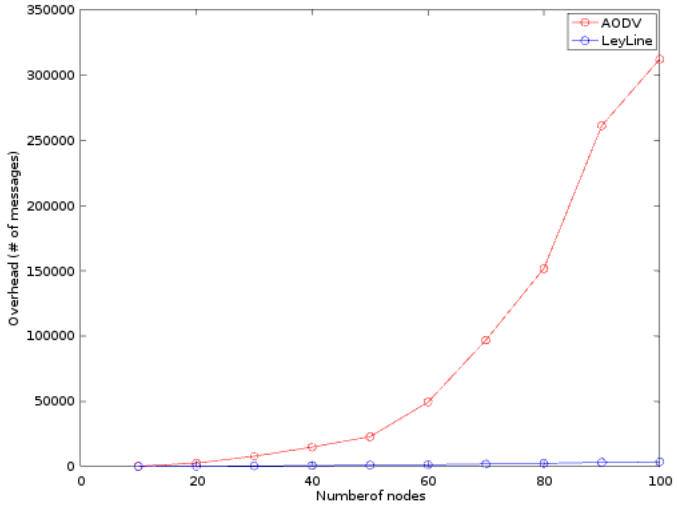

FIGURE 3.3: Determining optimal number of entries

3.3.2 Average Path Length

In order to determine which protocol consumes less energy, we look at the number of hops it takes on average to deliver a message. Our results show that AODV performs way better than LeyLine in terms of stretch for $N \leq 100$.

3.3.3 Memory Consumption

AODV computes the shortest path from the source to the destination therefore its routing table increases linearly with the number of nodes. On the other hand, LeyLine

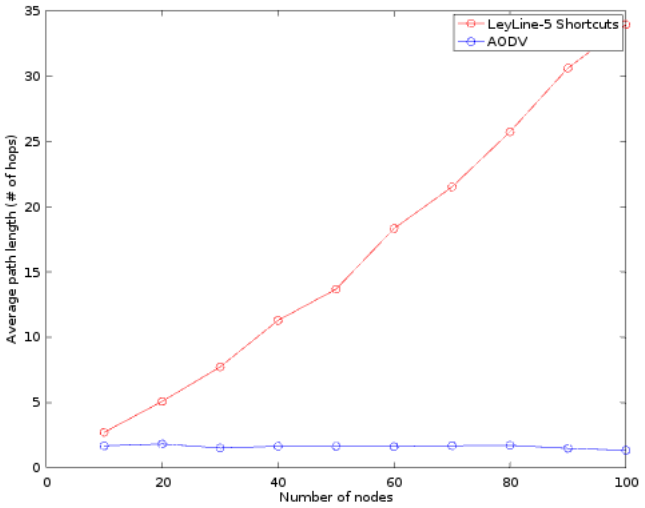

FIGURE 3.4: Average Path Length: LeyLine vs AODV

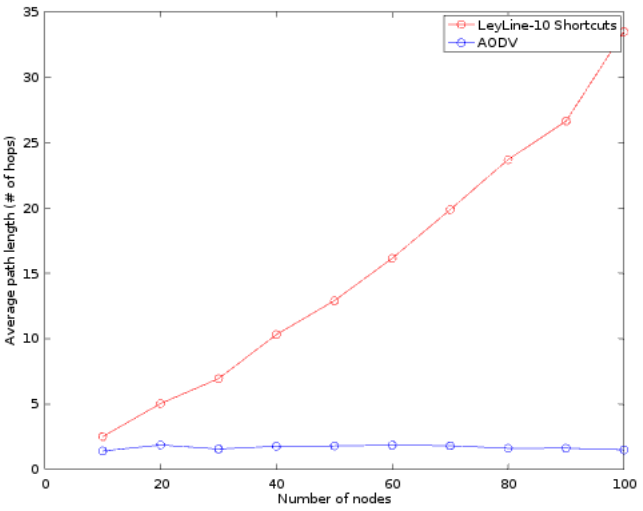

FIGURE 3.5: Average Path Length: LeyLine vs AODV

only stores a constant number of entries in its routing table. This is illustrated in the figure below.

3.4 Conclusion

LeyLine's overhead and memory consumption is more favorable when compared to a traditional benchmark (AODV). This shows that LeyLine enables sensor nodes to populate their routing tables with a lower overhead, have smaller routing tables. Hence, Leyline enables nodes to conserve memory in the routing process.

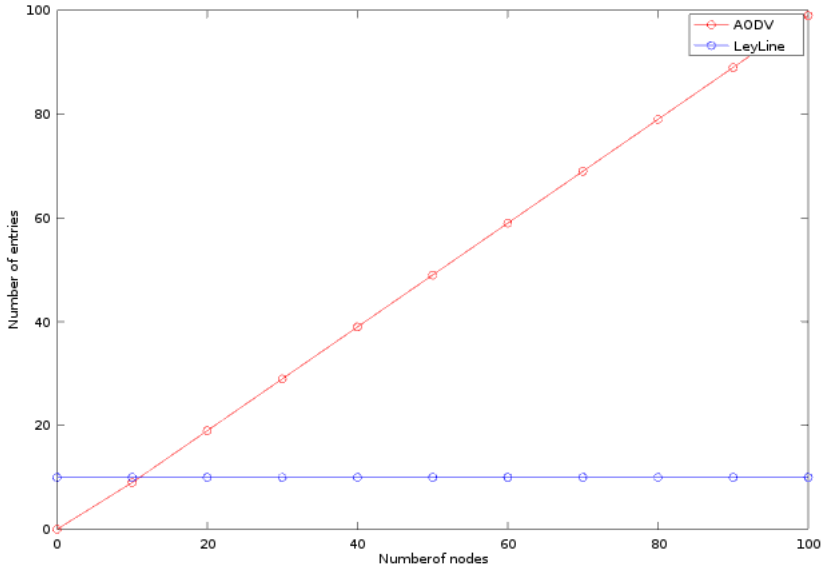

FIGURE 3.6: Memory Consumtion: LeyLine vs AODV

References

1. Making the world (of communications) a different place. Clark, D., et al. ACM SIGCOMM Computer Communication Review, July 2005.
2. MTM-CM5000-MSP 802.15.4 TelosB Mote module. advanticsys. [Online] [Cited: October 3, 2016.] http://www.advanticsys.com/shop/mtmcm5000msp-p-14.html
3. S. Obaidat, Muhammad and Misra, Sudip. Principles of Wireless Sensor Networks.
4. A survey of routing algorithms for wireless sensor networks. Datta, N.Narasimha and K., Gopinath. s.l. : Journal of Indian Institute of Science
5. Perkins, C.E, Royer, E.M. Belding and Das, S.R. Adhoc on-demand distance vector (AODV) routing. . s.l. : IETF Network Working Group, RFC 3561, 2003.
6. Clausen, T and Jacquet, P. Optimized link state routing protocol (OLSR) . s.l. : IETF Network Working Group, RFC 3626, 2003.
7. Wireless sensor networks: a survey. Akyildiz, I.F., et al. 38, s.l. : Elsevier Computer Networks, 2002.
8. Labrador, Miguel A and Wightman, Pedro M. Topology Control in Wireless Sesnor Networks. 2009
9. Langendoen, K, Baggio, A and visser, O. Murpyh Loves Potatoes: Experiences from a pilot sensor network deployment in precision agriculture. Rhodes, Greece : 14th Int Workshop on Parallel and Distributed Real-Time Systems, Apr 2006.
10. Routing requirements for low power and lossy networks. Culler, D. and Vasseur, JP. s.l. : IETF Networking Group, 2007

11. Sarkar Amit, T. Senthil Murugan, "Routing protocols for wireless sensor networks: What the literature says?", Alexandria Engineering Journal, vol. 2, August 2016.

12. M. Maroti. Directed flood-routing framework for wireless sensor networks. In 5th ACM/IFIP/USENIX Intl. Middleware Conf., 2004.

13. F. Ye, G. Zhong, S. Lu, and L. Zhang. A robust data delivery protocol for large scale sensor networks. In 2nd Workshop on Information Processing in Sensor Networks (IPSN '03)

14. Y. Zhang and M. Fromherz. Constrained flooding: A robust and efficient routing framework for wireless sensor networks. In 20th IEEE Intl. Conf. on Advanced Information Networking and Applications (AINA '06)

15. M . Thorup and U. Zwick, "Compact routing schemes," in Proc. SPAA Jul. 2001, pp. 1–10.

16. G. Toussaint, "The relative neighborhood graph of a finite planar set", Pattern Recognition, pp. 261-268, 1980.

17. B. Leong, S. Mitra, B. Liskov, "Path vector face routing: Geographic routing with local face information", Proc. IEEE ICNP, 2005-Nov

18. F. Kuhn, R. Wattenhofer, Y. Zhang, A. Zollinger, "Geometric ad-hoc routing: Of theory and practice", Proc. ACM PODC, pp. 63-72, 2003.

19. Y.-J. Kim, R. Govindan, B. Karp, S. Shenker, "On the pitfalls of geographic face routing", Proc. 3rd ACM/SIGMOBILE DIALM-POMC, pp. 34-43, 2005.

20. P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks", Proc. ACM SIGCOMM, pp. 35-42, 1988

21. Z. J. Haas, M. R. Pearlman, P. Samar, The zone routing protocol (ZRP) for ad hoc networks, Jul. 2002

22. A. Post, D. Johnson, Self-organizing hierarchical routing for scalable ad hoc networking, 2004

23. R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, I. Stoica, "Beacon vector routing: Scalable point-to-point routing in wireless sensornets", Proc. NSDI, pp. 329-342, 2005-May

24. Yun Mao, Feng Wang, Lili Qiu, Simon S. Lam, Jonathan M. Smith, S4: Small State and Small Stretch Routing Protocol for Large Wireless Sensor Networks, NSDI'07

25. http://www2.engr.arizona.edu/ junseok/AODV.html