Research paper: Hanabaratti and Jogdand., 2011: 99-104.

DESIGN OF AN EFFICIENT RANDOM WALK ROUTING PROTOCOL FOR WIRELESS SENSOR NETWORKS

Kavita D.Hanabaratti and Rashmi Jogdand*

Dept. of Computer Science, KLS GIT, Belgaum
*Dept of MCA, KLS GIT, Belgaum
Corresponding Author: kavitagayad@gmail.com;rashmi25_mj@yahoo.com

ABSTRACT

Wireless Sensor Networks (WSNs) consist of small nodes with sensing, computation, and wireless communications capabilities. Energy awareness, power management and data dissemination considerations have made routing in wireless sensor networks a challenging issue. Low latency data delivery is an important requirement for achieving effective monitoring through wireless sensor networks When sensor nodes employing the duty cycle sending a message along the shortest path, however, does not necessarily result in a minimum delay. Our project studies the lowest latency path problem i.e the characteristics of path with minimum delay that connect source node to the sink under random duty cycling nodes, since low latency data delivery is an important requirement for achieving effective monitoring through wireless sensor networks. The paper propose a forwarding protocol based on biased random walks where nodes use only local information about neighbours and their next active period to make the forwarding decisions. This is referred as lukewarm forwarding. Analytical and simulation experiments make it possible to reduce the latency without increasing the number of transmissions needed to deliver the message to destination.

Keywords: Wireless Sensor Networks, Latency, duty cycle, random walk routing protocol

INTRODUCTION

A sensor network could consist of a large number of sensor nodes, each of which has a limited communication range and a limited energy supply in the form of a battery. Each sensor node, ni, periodically generates a data sample that needs to be communicated to the sink. As the sink may not be within direct communication range of ni, its sample has to be forwarded to the sink through intermediate sensor nodes using some routing protocols [1-2].The default data forwarding approach (assuming no data aggregation) is for each sensor node to forward its samples, as well as the samples it receives from other nodes, to its parent node in the spanning tree. All the samples travel up the tree and ultimately reach the sink. The spanning tree could be constructed in a breadth-first manner so as to minimize the length of the path (in terms of the number of intermediate nodes) from a sensor node to the sink. If all sensor nodes keep their wireless transceivers active all the time, the shortest path from a node to the sink is likely to also be the fastest path for data communication from that node to the sink. This is because each intermediate node can start trying to forward a message to its parent node as soon as it receives the message from one of its children. However, keeping the transceivers active at all times can quickly drain the energy supply of sensor nodes. This is because the transceiver is usually the most power-hungry component of the node. It consumes energy even when it is listening on the wireless medium. Duty cycling is a widely used approach to reduce energy consumption and prolong the life of a sensor network. Nodes keep their radio on for only a limited amount of time in each time cycle. With duty cycling, the active periods of sensor nodes could be staggered. The active periods of nodes could be scheduled such that within any duration of time T for each pair of neighbouring nodes, their active periods overlap long enough to enable at least one message to be sent by one node to another. Besides easing the requirement of tight clock synchronization, this staggering of active periods of nodes in a neighbourhood has the added benefit of easing contention for the wireless communication medium. Data forwarding under duty cycling nodes may result in higher end-to-end message delay. When the next hop node is idle (sleeping), a node has no option but to buffer the message. For some applications, however, reporting data sample promptly to the sink may be very important. Besides the higher cost resulting from the two-radio architecture, the signalling channel clearly cannot be itself subject to duty cycling, and thus, the overall energy efficiency is limited.

Existing System: Low latency routing under duty cycling nodes is addressed in several papers. There are two approaches suggested in the literature, coordinated sleeping mechanism and random sleeping mechanism. In coordinated sleeping, nodes communicate with each other to adjust their sleeping schedules to meet some requirement such as coverage. For example, Geographic Adaptive Fidelity (GAF) [3] allows unnecessary nodes to turn their radio off without affecting the level of routing fidelity. Each node uses a GPS-indicated location to associate itself with a point in a virtual grid. Nodes associated with the same point on the grid are considered equivalent in terms of the cost of packet routing. Such

equivalence is exploited to keep some nodes located in a particular grid area in sleeping state in order to save energy. In SPAN [4], coordinator nodes stay awake and perform routing of packets while other redundant nodes are allowed to sleep to conserve energy. The coordinators are elected in a distributed manner and the role of the coordinator is rotated among different nodes to introduce fairness. Unlike these protocols, our proposal does not require any location information support. Based on block design combinatorics, Zheng et al. suggest an Asynchronous Wakeup Protocol (AWP) [5] in which nodes wake up according to a Wakeup Schedule Function (WSF). These WSFs have the interesting property that any shifted version of a WSF would still have one or more active period of nodes overlapped. This reduces the complexity involved in synchronization of all nodes and avoids the problem of network partition due to different sleep-and-wake schedules of nodes. In random sleeping mechanisms, sensor nodes enter the sleeping mode randomly and independently from each other. For example, the Randomized Independent Sleeping (RIS) mechanism [6] assumes that sensors are synchronized globally and time is divided into slots. At the beginning of each time slot, a sensor independently decides to remain awake or enter sleeping mode with probability of p and 1-p, respectively. When RIS is used as the underlying sleeping mechanism, a node has to buffer its message until the next hop becomes awake. Random Asynchronous Wakeup (RAW) [7] employs random wake-up schedules and allows a node s to forward the message to any active neighbor that belongs to a Forwarding Candidate Set, which is defined as the set of neighbours that is closer to the sink by at least a given value than the node s itself. RAW is designed for high-density networks and its simple criteria do not allow a message to increase its distance from the destination. Holes are not allowed in the network. Our proposal can work under any network density. Moreover, in order to reduce latency, sometimes a message has to move away from the destination. Yet another orthogonal technique for supporting duty cycling is based on using a dedicated signalling band for waking up a node on-demand, e.g., PAMAS [8-11]. However, this solution is more costly since it requires two-radio architecture. Our protocol uses only one radio by design.

Problems with Existing System: There are several kinds of flaws which are associated with the existing system and all these can be highlighted as follows:-

1) RAW is designed only for high-density networks and its simple criteria do not allow a message to increase its distance from the destination. Holes are not allowed in the network.
2) PAMAS solution is more costly since it requires two-radio architecture

Problem Definition: The project deals with the lukewarm potato forwarding of messages, which is a compromise between the shortest path forwarding and hot potato forwarding. Hot potato forwarding is a greedy approach to fast propagation of a message toward the sink: a node, on receiving a message, forwards it to the neighbor that is the first to become active. In essence, the message is like a hot potato that the node wishes to get rid of at the earliest possible opportunity. However, this greedy approach assumes no knowledge of network topology and may also fail to yield the fastest propagation of a message to the sink. In this proposal, the node with the message is still interested in quickly passing the message to a neighbor. However, it is not as hasty as it would be under hot potato forwarding. If it knows that its parent on the shortest path forwarding tree will wake up within a threshold amount of time, the node prefers to forward the message to the parent. Otherwise, it forwards the message to the neighbor that is the first to become active

Theoretical background: Theoretical background highlights some topics related to project work. The description contains several topics which are worth to discuss and also highlight some of their limitation that encourage going on finding solution as well as highlights some of their advantages for which reason these topics and their features are used in this project.

Duty Cycle: It considers a time slot model, where data transmission requires one time slot. For two neighbours to communicate, they must be active during the same time slot. The duty cycle of each node is determined by the superimposition of two kinds of active slots, called random active slots and scheduled slots. The random active slots are d time slots apart from each other, where d is a random value with uniform distribution in the range [1...K]. In other words, if a node is active at time k, then the next random active slot occurs at slot k + d with probability 1/K, 1 <=d <= K. Then, K is the maximum time distance between two consecutive active time slots. A scheduled active slot is an extra active slot that models the fact that a node can switch its radio on to communicate with one of its neighbors. It remarks that a scheduled slot is determined by the sending node and not by the receiving one, i.e., the slot does not correspond to any wake-up functionality. Consider Figure 1, where the sink node is depicted as a gray node, and suppose that node A needs to send a data sample to the sink. The right-side part of the figure shows a possible activity pattern. An empty box with full line represents a random active slot whereas scheduled active slots are depicted as boxes with dashed lines. It can be seen that the delay associated

with the shortest path is D= $t_{A,B} + t_{B,D} + 1$ (the sink is always active). The lowest latency occurs when the message is routed through the longer path A, B, E, F, G whose delay D` = $1 + t_{A,C} + t_{C,E} + t_{E,F} + t_{F,G}$ is less than D.

Low Latency Path: Estimate the delay (number of slots) of a message routed from the origin node to the sink along the fastest possible path, D_{min}. To calculate such an average, it uses a flooding algorithm. Once a node receives the message, it sends the message to its neighbours as soon as they wake up. The minimum delay is the time elapsed from when the message is generated until the first copy arrives at the sink. Measured the delay for nodes at different distances and made the following observation. From the figure 2 it concludes that the shortest path is not always the fastest path. There always existed a path in the delay which is much better the shortest path.

Lukewarm Forwarding Protocol: The protocol assumes that the clocks of the sensors are synchronized. The protocol works according to a time slot model, where slots are synchronized. Moreover, it requires that a sensor knows the identity of its next hop nodes (space look-ahead) and that it is able to predict the next active slot of its neighbours (time look-ahead).The packet transmission requires one time slot, i.e., medium access protocol, collisions, etc., are all such that they can be managed within a time slot. Slot synchronization and prediction are discussed later in this section. Let F denote the forwarding node, G a generic neighbor of F, and P the parent node, i.e., the neighbor of F which is on the shortest path from F to the sink. Moreover, let W_{TP} be the forwarding delay through P, namely the time elapsed from when F received the message until P will wake up. The key idea of the protocol is that F forwards the message to P only if W_{TP} is not "too long." Otherwise, F sends the message to the first neighbor that becomes active.

The forwarding rule is then as follows:

- If $W_{TP} < T$, then F sends the message to P.
- Otherwise, F sends the message to the first node G that becomes active (ties are broken at random).

The main intuition behind the proposal is to tune the greediness of the forwarding strategy by regulating the maximum waiting time, T, that a node can tolerate before data forwarding along the shortest path is abandoned. By setting T properly, messages can be forwarded in a deterministic fashion along the shortest path (T = K, where K is the maximum idle period), in random fashion (T = 0), or according to a combination of both (0 < T < K). For T = 0, the protocol follows a hot potato forwarding strategy. And, since active slots are random, the message undergoes a random walk. By increasing T, the forwarding strategy becomes less greedy and passes from a hot strategy to a lukewarm one. The message is now forwarded as a biased random walk, where the bias level represents the probability of the message being routed via the shortest path. Thus, another way of looking at the forwarding strategy arising when T varies is by casting message forwarding as a biased random walk. When T= K, the random walk has the maximum bias level, whereas it is completely unbiased when T= 0. For 0 < T < K, the bias level varies between these two extreme cases.

Slot Synchronization: As nodes share a common clock, the duration of a time slot is the same at each node. Hence, to achieve slot synchronization, nodes must agree on the slot boundaries. This can be achieved in the following way. The sink node acts as master node by periodically sending a short synchronization signal, say, every K time slots, which serves as reference point for slot alignment of its neighbor nodes. A node n maintains slot synchronization by listening for short synchronization packets sent by the parent of n, pn, when it wakes up. In general, if the slot boundaries of a node n are not aligned, then n will keep the radio on until it receives the synch signal from pn (the sink acts as parent of all its neighbouring nodes). After that, the node enters the duty cycle mode of operation. Note, however, that slot synchronization, while important for the algorithm to properly work, is not critical in the sense that an unsynchronized node can quickly recover. More specifically, assume that node n has to send a message to a node x.Assume that due to slot misalignment, node n will not see x active at the predicted time slot. In this case, node n can simply keep its radio on until one of its neighbours becomes active and then send the message to it. Clearly, this deviation from the decided node has no impact on the correctness of the protocol, as random forwarding is embedded into the protocol's logic. In order to resynchronize, n continues to keep the radio on until it receives the synchronization signal from p_n. After that, the node enters the duty cycle mode again.

Prediction of the Active Slot: In order to predict the next active time slot, all nodes use the same Pseudorandom Number Generation (PRNG) algorithm. Slots are numbered from 0 to K - 1.Each node n uses a local PRNG for determining the time elapsed between two consecutive active time slots, which is initialized with a local seed. A node also maintains a "copy" of the PRNG used by each of its neighbours, initialized with the seed they are using. Each time a node n wakes up, n sends a short synchronization packet containing its local seed s_n and a newly generated random number, Sn, picked uniformly in the

range [1 . . .H] , where H >> K is a multiple of K, say, H =100K. The goal of S_n is twofold. First, it is used for calculating the (relative) time of the next active slot of n, which is given by 1 + (S_n mod K). Second, an active neighbor m of n, uses S_n to "synchronize" its copy of n's PRNG. For this, m uses such a copy to generate new numbers until the same number S_n is obtained. Note that as the activation pattern of nodes is random, any pair of nodes eventually become active at the same time, thus, nodes eventually learn each other's seed.

SYSTEM ARCHITECTURE

System architecture is the conceptual design that defines the structure and behaviour of a system. An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system components or building blocks and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. The System architecture is shown in figure 3.

The system consists of following sub modules.

- Config Panel: User can create the wireless sensor network and the configuration by using the Config Panel.
- Wireless Sensor Network Simulator: This module simulates the wireless sensor network. Nodes can send data packet to sink node using multi hop routing.
- Visualizer: This module displays the activities at the sensor node and the data packet flow in a neat GUI.
- Routing Engine: This module helps in finding the shortest path to sink from any source node.
- Lukewarm Forwarder: This module implements the lukewarm forwarding protocol for forwarding the packets in the fastest pat

System Flowchart: refer figure 4.

Data Flow Diagram of the system: A context-level or level 0 data flow diagram shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization (figure 5-6).

IMPLEMENTATION

Module Implementatio: This project named biased random walk routing protocol consists of two major parts which are mentioned below .The project has been implemented using Matlab on Windows XP platform. The implementation of these two algorithms is described below.

Duty Cycle Distribution Algorith: The epidemic style algorithm for calculating the minimum delay.

Data: Activation patterns, initial node, sink, topology

Result: Minimum delay, Initialization: *delay ← 0;*while *Sink not infected do* for *any active uninfected node m do* if *a neighbor n of m is infected then n infects m (m can infect starting from the* next round), end, end, *delay ← delay + 1;*

Routing Engine: Routing Engine learns the shortest path from any node to the sink node by using the following algorithm. Sink node forwards hello packet with hop count value as 0. Each node receiving it forwards the hello packet by incrementing the hop count and its id in the message. Each node receiving the hello packet remembers the shortest hop count to sink and the node through which it has go through. So when hello packet reaches all nodes, each node has the information of next node in the shortest path to the sink node.

Lukewarm Forwarder Protocol: Each node knows about the next node in the shortest path to the sink node. When packet arrives for routing to the node, lukewarm forwarder decides to route using the following algorithm.

T: Wait Time , N: Shortest Path Node, If (N can become active within T)
{Store packet and forward when N becomes active}, else
{X ←Find neighbor node which is becoming active in shorter time forwardPacket(X)}

RESULT ANALYSIS

The result analysis can be done for the 3 cases (Table 1). The below snapshot shows the graph of threshold vs delay or the latency, from the snapshot it can be conclude that as the value of threshold increases the value delay decreases (figure 7). The graph which shows the plot of threshold versus delay (figure 8). Snapshot showing the dropping message (figure 9). Snapshot showing the message that TTL is expired after pressing the route button (figure 10).

CONCLUSION

The project addressed the lowest latency path problem and provided a heuristic protocol with low delay. In a sensor network where sensor nodes are duty cycling, following the shortest path from a source to a destination for delivering a message does not result in minimizing the latency of message transfer. This is basically due to the buffering delay required to forward message along the path.The project also analyzed this delay gap that tends to increase with the number of sensors nodes forming the system. This study led to the design of the lukewarm potato forwarding protocol that can be tuned to minimize the delay of a path from a distance to a destination. The versatility of this solution lies in the fact that by varying the value of a single parameter, the protocol changes its level of warmness (aggression) passing from the classical hot potato scheme to the cold deterministic shortest path forwarding. Although deviating from the shortest path requires additional transmissions, and hence, higher energy consumption, this increase is compensated by a lighter duty cycle. The experiments show that it can obtain the same delay performance by consuming less energy or, conversely, getting a lower delay by spending the same amount of energy while obtaining the same data delivery delay as shortest path routing.

The delay of the proposed protocol is in the middle between the shortest path's one and the ideal case. This result confirms that if delay is a concern, classical shortest path routing is not the best choice, since it is very far from the achievable limit. The proposed protocol reduces the delay at the cost of extra hops. It is interesting to note that the lowest delay is obtained through a path shorter than the one. This means that there is still room for improvement, e.g., making a node aware of the topology and activity pattern of its 2 hops neighbours. Planning to extend the lukewarm protocol for calculating the no of transmissions of the packets before the packet reaches to the destination.

REFERENCES

1. Robert Beraldi and Robert Baldoni, 2009. Lukewarm *Potato Forwarding*: A Biased Random Walk Protocol for Wireless Sensor Networks" IEEE 2009.
2. Al-Karaki, J.N and A.E. Kamal. 2004. Routing techniques in wireless sensor networks: a survey, Wireless communication IEEE..
3. Xu, Y. Heidemann, J and D. Estrin, 2001. "Geography-informed Energy Conservation for Ad-hoc Routing", In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking 2001, pp. 70-84.
4. Chen, B. Jamieson, K. Balakrishnan, H and R. Morris, 2002. SPAN: An energy efficient coordination algorithm for topology maintenance in ad hoc wireless networks, ACM Wireless Networks, 8: 481494.
5. Zheng, R. Hou, C and S. Lui, 2003. "Asynchronous wakeup for ad hoc networks," Proc. ACM MobiHoc 2003, pp. 3545.
6. Kumar, S. LAI, T. H and J. Balogh, 2004. "On k-coverage in a mostly sleeping sensor network", In Proceedings of MobiCom04 (Philadelphia,PA). 144158.
7. Paruchuri, V, Basavaraju, B. Durresi, A and R. Kannan. 2004. "Random Asynchronous Wakeup Protocol for Sensor Networks" in Proceedings of BroadNets, San Jose, CA, October 25-29, pp.710-717.
8. Singh. S and C. S. Raghavendra, 1998. "PAMASPower aware multiaccess protocol with signaling for ad hoc networks," ACM SIGCOMM *Computer Communication Review*, 28: 526.
9. Avin, C and C. Brito, 2004. "Efficient and robust query processing in dynamic environment using random walk techniques", Proc. of the third international symposium on Information processing in sensor networks, Berkeley, California, USA.
10. Boyd, S. Ghosh, B. Prabhakar, B and D. Shah, 2005. "Gossip and mixing times of random walks on random graphs", Proc. of SIAM ANALCO 2005, Vancouver, Canada.
11. Dolev, S, Schiller, E and J. Welch. 2002. "Random walk for self-stabilizing group communication in ad hoc networks", Proc. of the 21st PODC 2002, Monterey, California.

Figure 1 Simple example of longer yet faster path

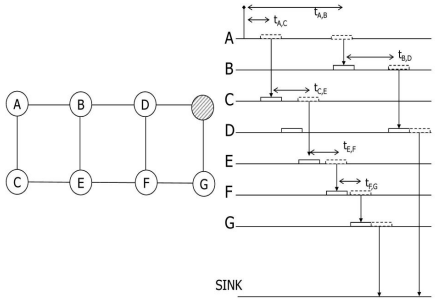

Figure 2 graph of average vs distance of longer yet faster path

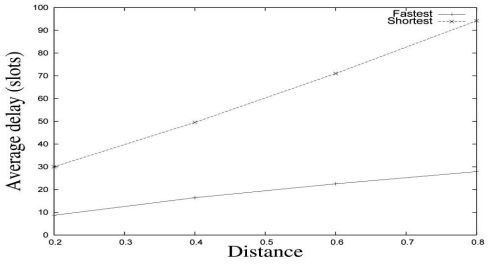

Figure 3 System Architecture

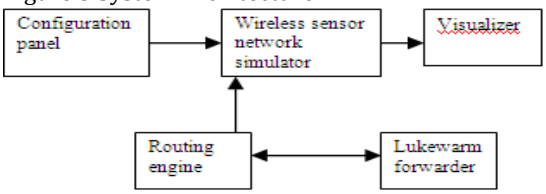

Figure 4 System flowchart

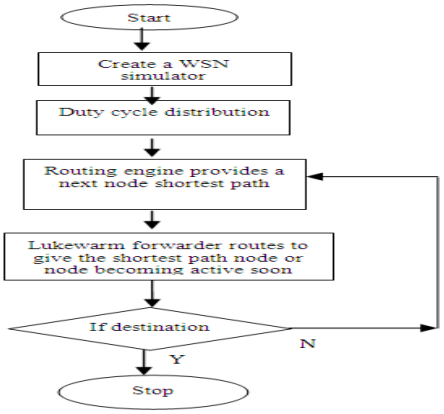

Figure 5 Level 0 Data flow diagram

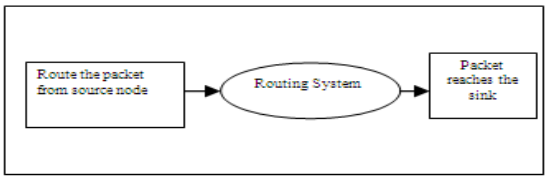

Figure 6 Level 1 Data flow diagram

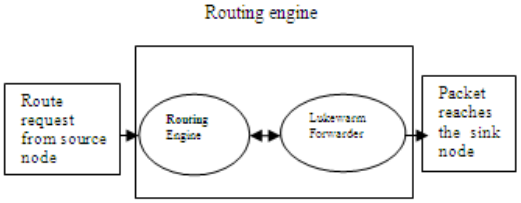

Figure 7 Threshold vs Delay

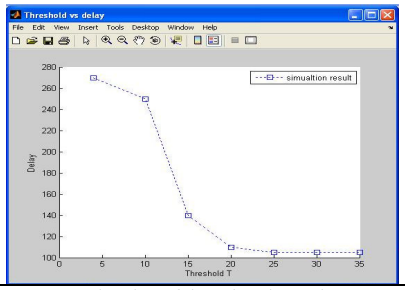

Figure 8 Threshold vs Delay

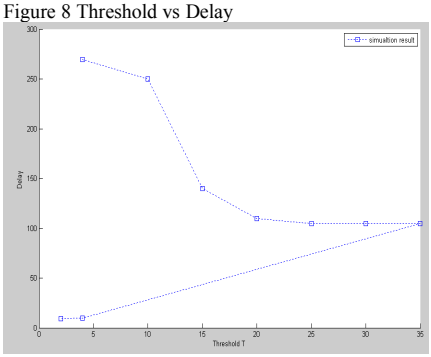

Figure 9. Message showing of dropping the packet

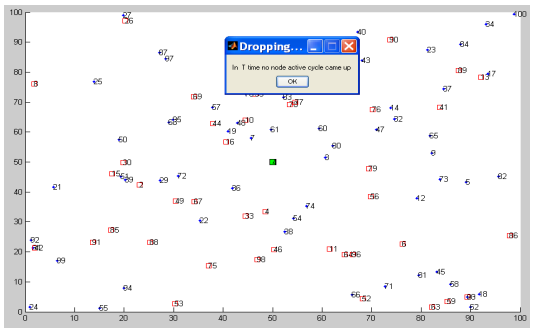

Figure 10 Message showing that TTL is expired

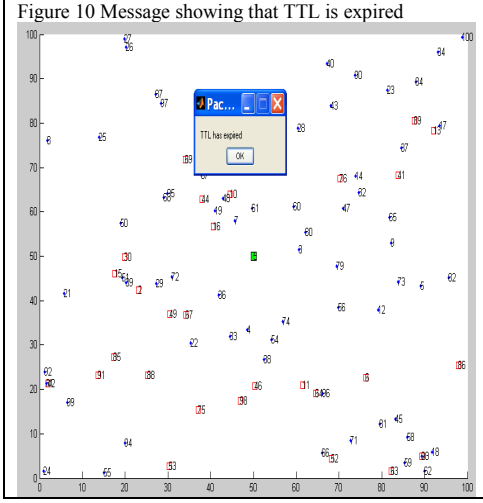

Table 1 .Result analysis

Case no	Tests done	remarks
1	minimum waiting time (T) > minimum idle time (K)	The path from source to sink exists
2	T=K.	The packet drops
3	TTL< T	TTL expired

****