# Power Transmission and Channel Control Approach using Learning Automata in the Internet of Things

**Peilin Chen** ( peilin.chen@yahoo.com )
Xi'an University of Technology

**Research Article**

**Keywords:** Fog computing, Internet of Things, Learning automata, 6LoWPAN.

**Posted Date:** April 7th, 2021

**DOI:** https://doi.org/10.21203/rs.3.rs-386622/v1

# Power Transmission and Channel Control Approach using Learning Automata in the Internet of Things

**Peilin Chen**
[1] Department of Computer Science, Xi'an University of Technology, China
*Email:* peilin.chen@yahoo.com

***Abstract*—Energy consumption management and optimal use of node resources are key elements in the Internet of Things. In this study, a framework based on cognitive topology control that acts based on the $L_{R-I}$ learning automata and game on it, has been used to control power, channel and contention windows and to create preventive behavior in the network. Due to the limitations of nodes in the Internet of Things, the transfer of learning automata processing to the cloud, fog and edge has been investigated to increase lifespan, reduce memory consumption and increase processing power. The communication was done based on IPv6 protocol and IEEE 802.15.4 standard. The nodes also used the uIP lightweight protocol stack and the RPL lightweight routing protocol. In order to use the sixth version of the Internet Protocol in the IEEE 802.15.4 standard platform, the 6LoWPAN protocol has been used to compress and convert headers. Computing on fog nodes has also been used to perform game calculations on automata. Finally, the Cooja simulator was used in the Contiki operating system to evaluate the efficiency of the proposed method, which showed the superiority of the proposed method in energy consumption, memory usage and processing power compared to other methods that control power and channel.**

***Index Terms*— Fog computing, Internet of Things, Learning automata, 6LoWPAN.**

## I. Introduction

Today, the Internet of Things (IoT) is becoming more widespread than ever. Smart energy, smart agriculture, smart farming, smart house and building, smart transportation, smart health, smart city and smart living environment are just a few examples of these applications.

Things on the Internet of Things often have limiting properties. Things often have low processing power, low memory and limited energy source. The ultimate goal of this research is to provide a solution to increase quality of service parameters and improve energy consumption by considering these limitations. Due to the mentioned limitations, it is not possible to use common Internet protocols and routing protocols in these devices. Many devices on the Internet of Things use the IEEE802.15.4 standard for communication. On the other hand, the increasing development of the IoT and the proliferation of devices that require a network connection make the use of IPv6 to address this volume of devices mandatory. But the MTU on the IPv6 is 1280 bytes, and the maximum frame size in the Mac layer of IEEE 802.15.4 standard is 127 bytes. In addition, only 102 bytes of 127 bytes of frame size are allocated for IPv6 packages. If we consider the maximum security, only 81 bytes remain. For this purpose, the 6LoWPAN protocol was introduced for the use of IPv6 in the IEEE 802.15.4 standard. This protocol is actually an adaptive layer between the network layer and the Mac. One of the most important tasks of this layer is to compress and fragment the IPv6 package header and compress the UDP header. According to the content, nodes on Internet networks must support IPv6 and use the 6LoWPAN protocol if using the IEEE 802.15.4 standard.

The use of topology control for optimal use of limited resources of these devices can have a significant effect on performance improvement. Many topology controls operate on the basis of cooperation among nodes, and some do not consider the mobility of nodes. Cooperation among nodes leads to increased network traffic and reduced node performance. Others have tried to solve this problem by using cognitive and learning elements. But most of these control topologies do not take into account the limitations of the Internet of Things and are not applicable to it. Using edge, fog and cloud computing can overcome many of these limitations. Therefore, this paper presents a framework with the use of cognitive topology control based on learning automata for power and channel control. By controlling the power, the energy consumption of nodes is reduced, which this leads to an increase in their lifespan and efficiency due to the limitation in the energy source of nodes. Channel control is also done to allocate channels without collision with nodes and reducing network latency. On the other hand, automata processing has been transferred to the fog to reduce the workload on the processing core of the nodes, minimize energy consumption and use less memory of nodes. In implementing this topology control, IoT constraints are considered and the RPL lightweight protocol is used for routing and uIP protocol stack, which is a lightweight stack based on the TCP/IP protocol stack. The 6LoWPAN protocol is also used to support IPv6 in this protocol stack.

The rest of paper is organized as follows. In the section II, the proposed method is then described in section III and the efficiency of the proposed method and the results of simulations are presented in section IV. Discussion and conclusion are also included in section V.

## II. The Proposed Method

In this section, the specifications of the proposed topology control and learning automata are first described, then the automata processing with the proposed method and framework called EPLM (External Processor Learning Machine) is described. In this paper, an external processor (cloud, fog, or edge) is used for the first time for automata computations to control power and channel for IoT. Also, the type and content of messages that need to be exchanged, and the power and channel determination way by the game on the learning automata in the external processor, are presented precisely in a new way and with high efficiency and superior to the methods that have existed so far. The processing cycle and the relationships between them are shown in Fig. 1.

Two cognitive elements have been used in creating topology control, one for power control and the other for channel control and contention window. Initially, the probability of establishing a link for all links is estimated with BPST.

Then the transmission power, channel and contention window for the nodes are determined by playing on the learning automata and sent to the nodes. Power is then updated in the physical layer, channel and contention window in MAC layer. Finally, the nodes send the necessary messages to the fog processor to calculate the feedback so that the next round of the game can be performed on the automata and new values of power, channel and contention window can be specified. The automata used in this study is of LR-I, structure-variable type and has a limited number of operations (FALA). The game on the automata continues until it reaches Nash equilibrium.

With the launch of the RPL Routing Protocol, client nodes are detected and notified to an external processor. Then, part of memory is allocated to each nodes in the external processor and initial values are given to it (the first column on the left of Fig. 1).

The total number of client nodes is notified to the clients, and each node builds a totally zero vector with a length of the number of clients plus three. At the beginning of the vector, the letter N is used to mean neighbors. From now on, each node, according to the end of the inter-network address of its neighboring nodes, puts the appropriate index in vector 1 (meaning to be a neighbor). The last two values of the vector are first filled with the letter R and then the response of the environment to access the channel (0 or 1). Since now, we call this vector the information vector.

### A. Estimation of link establishment

You can see the link establishment estimation step briefly in in Algorithm 1.

The link establishment estimation is run for all possible links, but all its processing is done on an external processor (first row, Fig. 1). Eq. 3 has been used to use history and learning. The number of repetitions of the algorithm (m) depends on the next step of power control algorithm. This means that until the game on the automata reaches Nash equilibrium for power determination, the link establishment should be estimated. Environmental feedback for the first iteration is considered positive.

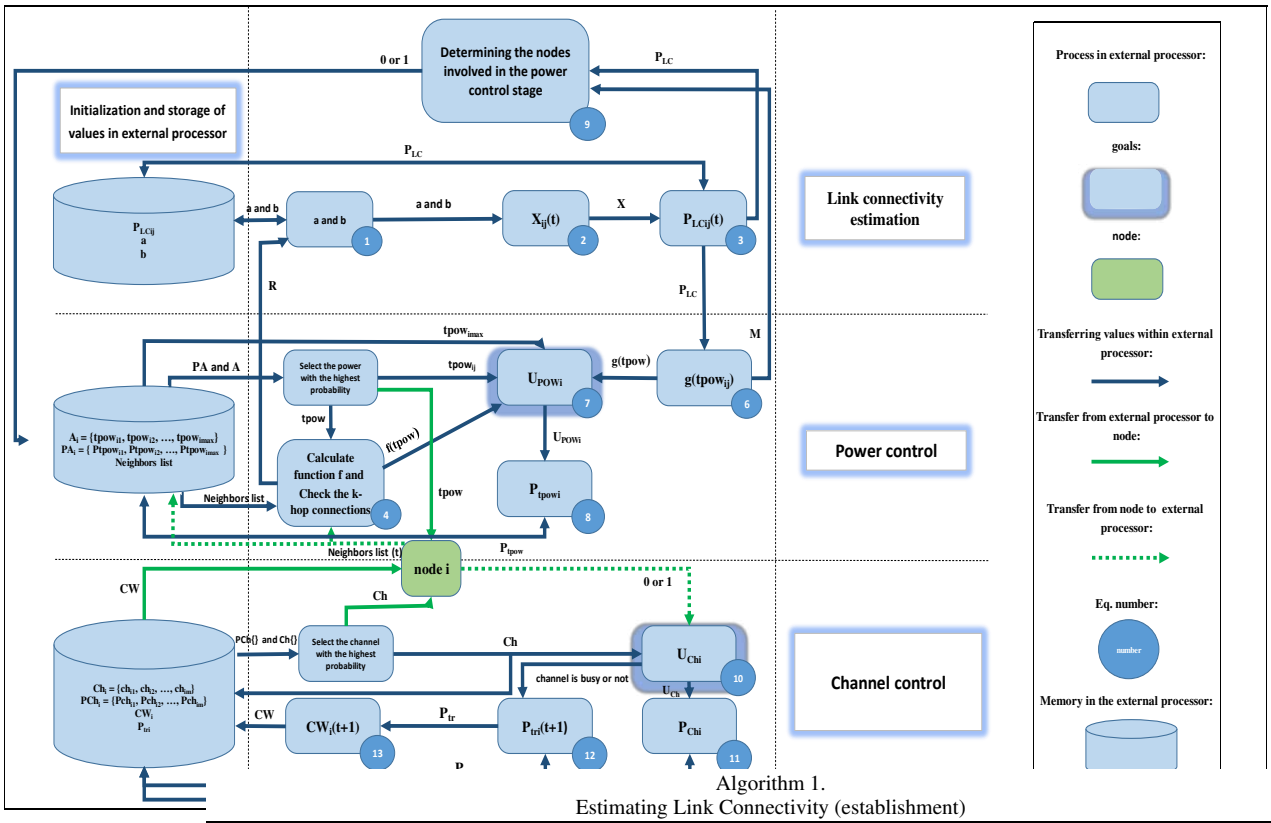
### *B. Determination of the transmission power*

The power determination algorithm for i-th automata, corresponding to i-th node in the network, is shown in algorithm 2. This game has been used to allocate optimal power to nodes.

The phrase inside the first parentheses in Eq. 7 is used to score lower powers. The result of the link establishment estimation algorithm is also used in the g function to give higher value in the utility function to the powers that lead to maintaining more stable links.

In order to reduce the volume of automata calculations, reduce the messages sent to the node and increase the efficiency, in the interruptions between the repetitions of the power determination algorithm, the following Equation is checked to determine the nodes participating in the next iteration.

$$\left|\sum_{l=1}^{n} \frac{P_{LC(il)}}{n} - M'\right| \leq Th \qquad \text{(Eq. 9)}$$




Algorithm 1.
Estimating Link Connectivity (establishment)

Algorithm 2
Game on Learning Automata for Power Control

Given: R (environment response = link connectivity in previous round), λ (learning parameter), m (number of iterations)
FOR each link
Step 1. Initialization
Put: $P_{Lcij} = 0$ // Probability of link connectivity Between nodes i and j
Put: a = b =1 //a is number of times link connectivity exists and b indicating the opposite
Put: m = 0
FOR m = 0 to m
Step 2: Link connectivity estimation
(Eq. 1) Phase 1: If R(t) = 1 THEN Put $a_i(t) = a_i(t-1) + 1; b_i(t) = b_i(t-1)$
ELSE Put $a_i(t) = a_i(t-1);\ b_i(t) = b_i(t-1) + 1$
(Eq. 2) Phase 2: $X_{ij}(t)$ is calculated using $\frac{\int_0^{X_{ij}(t)} v^{a-1}(1-v)^{b-1}dv}{\int_0^1 u^{a-1}(1-u)^{b-1}du} = \frac{95}{100}$ // $X_{ij}$ is probability of link connectivity reward
(Eq. 3) Phase 3: Put $P_{LCij}(t) = P_{LCij}(t-1) + \lambda(X_{ij}(t) - P_{LCij}(t-1))$
END FOR
END FOR
END

Given: $A_i = \{tpow_{i1}. tpow_{i2}. \ldots . tpow_{imax}\}$ (power action set (r actions)), n (Number of node i neighbors at instant t which are connected to node i based on $tpow_{ij}$), w, $P_{LC(il)}$ (probability of the link connectivity of the nodes which are connected to node i based on $tpow_{ij}$), λ (learning parameter)
FOR each node i
DO:
Step 1. Initialization
Put: $P_{tpowij}(t) = 1/r \quad \forall j=1 \ldots r$ // Probability of actions (r actions)
WHILE Nash equilibrium is reached
Step 2: The beginning of the game for power adjustment
Phase1: IF probabilities are equal THEN select the action with the maximum power (to prevent interruption)
ELSE select the $j^{th}$ action ($tpow_{ij}$) from action set based on the probability value of that action
END IF
(Eq. 4) Phase 2: IF k_hop connectivity retained THEN set the function f to 1
ELSE set f to zero
END IF
Remove any nodes that were not in the list of neighbors in the last w iterations
(Eq. 5) Put $M = \frac{\sum_{l=1}^{n} P_{LC(il)}}{n}$
(Eq. 6) Put $g(tpow_{ij}) = M - \sqrt{\frac{1}{n}\sum_{l=1}^{n}(P_{LC(il)} - M)^2}$
(Eq. 7) Put $Upow_i = \left(1 - \frac{tpow_{ij}}{tpow_{imax}}\right) \cdot g(tpow_{ij}) \cdot f(tpow_{ij})$
// Calculation of utility function

(Eq. 8) Phase 3: Put $P_{tpowij}(t+1) = P_{tpowij}(t) + \lambda * (U_{powi})(1 - P_{tpowij}(t))$
FOR all $l \neq j$
Put $P_{tpowil}(t+1) = P_{tpowil}(t) - \lambda * (U_{powi} - 0.1)P_{tpowil}(t)$
END FOR
END WHILE
END FOR
END OF GAME

Where n is the number of neighboring nodes for the node i (direct neighbors), $P_{LC(il)}$ is the probability of a link establishment between node i and its neighbors, M′ is the mean probability of a link establishment between node i with its neighbors, and *Th* is the threshold based on network conditions.
If the mean difference in the probability of establishment link between node i and its neighbors in two consecutive iterations exceeds the threshold, it is more likely that either higher or lower power is allocated, which will lead to network outages. Therefore, only nodes for which the difference in mean probability exceeds the threshold are included in the next iteration of the power algorithm.

As shown in Fig. 1, the selected power by automata is declared to node by a message from the external processor, and the node uses that transmission power to transfer the packages. Then the node completes the first part of the data vector and sends it to the external processor by completing the channel determination step and completing the data vector. To calculate the f function and feed the link estimation step, two vectors containing neighbors' data are given to block No. 4. The vector sent by node is the neighbors based on the selected power, but the other vector is the w neighbors of the previous iteration. *w* has been used to prevent the removal of a neighboring node due to incorrect power selection. The higher the node speed and network dynamics, the lower the value of w is considered, because the nodes exit the neighborhood more quickly, and the higher the value of w, the lower the efficiency and incorrect zeroing of the function f and the efficiency function.

### *C. Determination of the channel and contention window*

This step has been used to allocate the channel without collision with the nodes and reducing the delay and increasing the successful delivery rate of the package. The channel and contention window determination step is summarized in Algorithm 3. The number of iterations of this algorithm depends on the number of nodes requested to access the channel.

Algorithm 3
Game on Learning Automata for Channel Control

Given: $ch_i = [ch_{i1}. \ldots . ch_{im}]$ (channel action set), C (constant value), λ (learning parameter), α (between 0 and 1)

FOR each node i
Step 1. Initialization
Put: $P_{chij} = 1/m \quad \forall j=1 \ldots m$ // Probability of actions (m actions)
Put: $P_{Tri}(0) = 0.5$ //Channel access priority
WHILE Nash equilibrium is reached
Step 2: The beginning of the game for channel allocation
Phase1: Node i waits for $CW_i$ (x.C in first iteration that x is number between 0 and 1)
Phase 2: IF probabilities are equal THEN the channel is randomly selected
ELSE select the $j^{th}$ action ($ch_j$) from action set based on the probability value of that action
(Eq. 10) Phase 3: Based on the response received from the environment:

IF $ch_i$ is free THEN set $Uch_i$ to 1 // Calculation of utility function
ELSE set $Uch_i$ to zero
(Eq. 11) Phase 4: Put $P_{chij}(t+1) = P_{chij}(t) + \lambda * U_{chi} * (1 - P_{chij}(t))$ // Update Probability of actions
FOR all $m \neq j$ Put $P_{chim}(t+1) = P_{chim}(t) - \lambda * (U_{chi} - 1/2) * P_{chim}(t)$
END FOR
(Eq. 12) IF channel is busy THEN put $P_{Tri}(t+1) = P_{Tri}(t) + \lambda * (1 - P_{Tri}(t))$
ELSE Put $P_{Tri}(t+1) = P_{Tri}(t) - \lambda * (P_{Tri}(t) - \alpha)$ //Update channel access priority
(Eq. 13) Phase 5: Put $CW_i = (1 - P_{Tri}) * C$ //Update contention window
END WHILE
END FOR
END OF GAME

Eq. 12 and 13 are used to increase the priority and chance of access to the channel for nodes that have more traffic to prevent package buffer filling and increased latency. $\alpha$ is also used to prevent node deprivation from accessing the channel.

As shown in Fig. 1, the channel and contention window in the external processor are specified and declared to node. The node then accesses the channel according to the contention window. And depending on whether the channel is free or busy, completes the last value of the data vector and the vector is sent to an external processor. Each node operates until a message is received from an external processor, based on the latest power, channel, and contention window (initially according to the initialization in step 1).

## III. Simulation results

Simulations were performed with the Cooja simulator and Z1 motes in 2800 seconds. All motes use the IEEE 802.15.4 standard, IPv6 protocol. The 6LoWPAN protocol is used to compress the IPv6 package header. RPL protocol is also used for routing and UDP protocol is used for package transfer. The proposed method (EPLM) is compared with the improved TCLAB method, which is simply called TCLAB. Changes have been made to the structure of TCLAB method and the new method has been compared with higher efficiency. The simulation is done based on the transfer of processes to fog, but because the framework provided for the cloud and edge can also be used, it is referred to as an external processor. A processor with 1.8 GHz frequency is also used as a fog processor.

First, memory consumption is compared for two methods. Then, by changing the parameters 1- number of neighbors, 2- percentage of package delivery and receiving success in transmission range, 3- transfer range and distance from server and 4- mobility and speed of node movement. Total energy consumption (mean of each node) and quality of service parameters such as end-to-end delay and successful package delivery rate were compared for the two methods.
Some of the initializations .

The alpha value is determined to be 0.25, and threshold are also determined in proportion to the dynamics and allowable velocities of the nodes in that network. The number of authorized channels is considered to be 10, and the authorized power to be selected by each node in the simulation are considered to be 3, 7, 11, 15, 19, 23, 27, and 31, respectively that the allowable hardware values and corresponding values are 025 dBm to 0 dBm. These values are approximately in the range of 1 mW to 1 Watt.

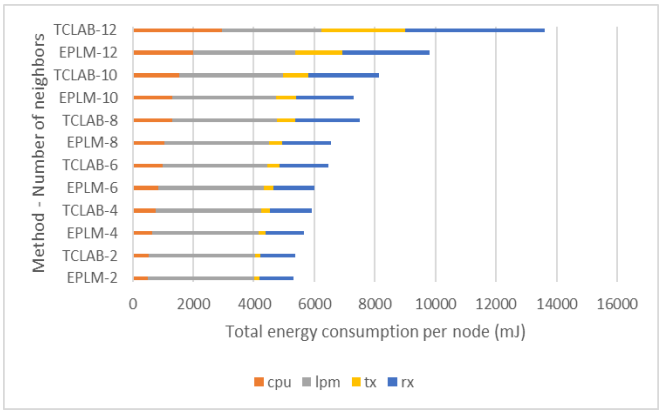
### *A. Memory consumption*

The nodes in the network that run TCLAB have higher memory usage than EPLM due to the automata execution within them. The memory usage according to the code of each node is as follows:

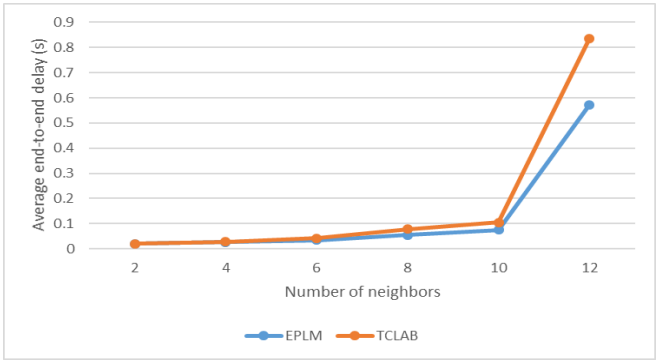
At TCLAB, nodes consumed 50,740 bytes of 92 KB of flash memory and 6,634 bytes of 8 KB of RAM.

These values for EPLM are 48,308 bytes of flash memory and 6,316 bytes of RAM. Therefore, using EPLM saves more than 2.5% (2.64%) of flash memory consumption compared to TCLAB. Also, using EPLM will save about 4% (3.97%) of RAM usage.

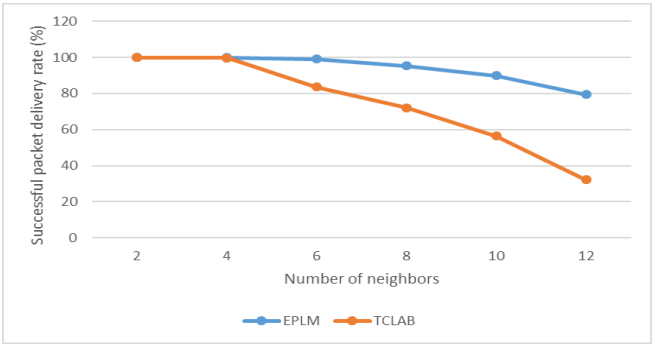
### *B. Evaluating the effectiveness of the proposed method*




(a) Energy consumption




(b) end-to-end delay




(c) Successful packet delivery rate

Fig. 2 Comparison of energy consumption, successful packet delivery rate and end-to-end delay of EPLM and TCLAB with different Probabilities of success in message transmission

*1) Investigating the effect of number of neighbors on efficiency*

Simulation in this part is performed for 2, 4, 6, 8, 10 and 12 neighbors for each node, and the successful delivery rate of the package in the transfer range is considered to be 98%.

The number of neighbors is a factor influencing the efficiency of both methods. In both methods, the volume of packages varies according to the total number of nodes within the network (data vector). The number of nodes also affects the size of automata calculations (effective at the link connectivity estimation stage). Therefore, increasing the number of nodes further affects the efficiency of TCLAB, because automata processing in TCLAB is done in TCLAB within the node, but in EPLM, at the fog node. The simulation result also confirms this by increasing the CPU energy consumption in TCLAB more than EPLM.

In addition, increasing the number of neighbors increases the number of messages in TCLAB and reduces its efficiency. But increasing the number of neighbors does not increase the number of messages in the EPLM (although RPL messages increase) and only affects the size of the sent packages. According to Fig. 2-a, the growth of energy consumption is evident in both methods (in sending and receiving), which is due to the increase in the volume of packages and the increase in the number of RPL messages. However, due to the increase in the number of messages because of the increase in the number of neighbors in TCLAB, the growth of energy consumption is higher than EPLM. Also, due to the greater number of messages in TCLAB and the intrinsic constraint

of nodes, there is a significant difference in end-to-end delay and successful package delivery rate compared to EPLM (Figures 2-b and 2-c).

*2) Investigating the effect of success rate of sending and receiving packages in the range of sending and receiving, on efficiency*

For various reasons, such as noise in the environment and the network in which the nodes operate, the probability and percentage of success of sending (tx) and receiving packages (rx) in the range of sending and receiving may be less than 100%. In order to measure this on efficiency of both methods, different simulations with 100, 80, 60 and 40% have been performed for the success of sending and receiving packages. The number of servers for all simulations is 2 and the number of neighbors (for each client node) is 6. For better measurement, the number of neighbors is not considered low, and having two servers for optimal performance in a model where the probability of success of sending and receiving is 100%, to more obviously show the effect of reducing the probability of success sending and receiving.

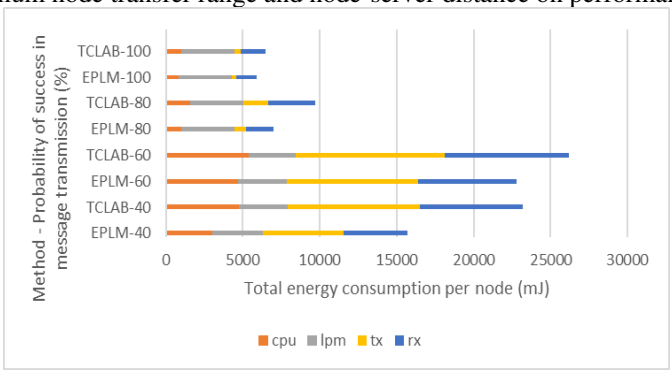
According to Fig. 3, the superiority of EPLM in energy consumption (Fig. 3-a), end-to-end delay (Fig. 3-b) and successful package delivery rate (Fig. 3-c) with different probabilities of success in sending and receiving, compared to TCLAB is still standing. The reason for these superiorities is similar to the reasons given in part 4-B-1.

With the increase in the likelihood of success in sending and receiving in both methods, the rate of successful sending of the package has also increased with gradual growth; but in terms of energy consumption and end-to-end delays, we're witnessing a sharp change in the chances of success in sending and receiving to more than 60%.

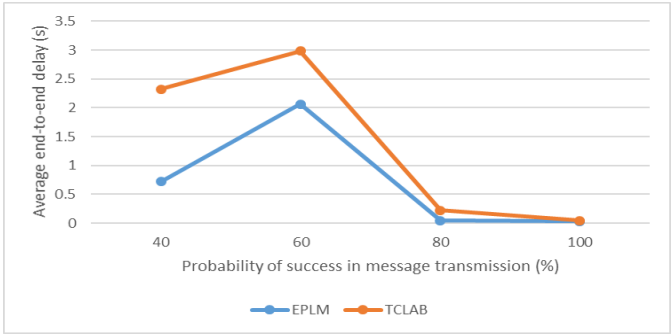
According to Fig. 3, the use of EPLM in networks with a low probability of success in sending and receiving is much more effective than TCLAB, because in low probabilities, the difference between the two methods is very large in the end-to-end delay and energy consumption, and as the probability increases, this difference decreases numerically, but the EPLM remains superior.

*3) Investigating the effect of maximum transfer range and node-server distance on performance*

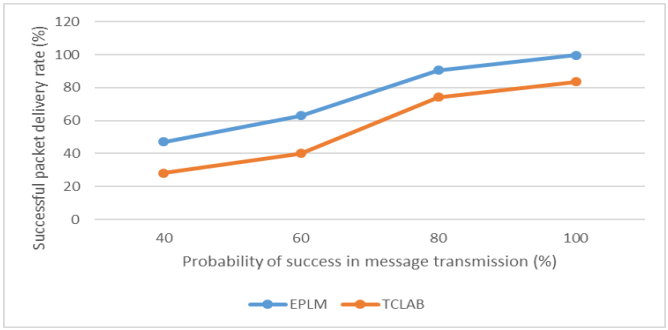
In this part, the effect of maximum node transfer range and node-server distance on performance is measured.




(a) Energy consumption



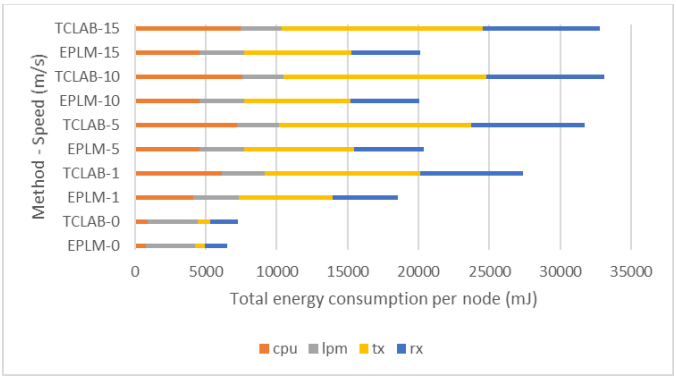

(b) end-to-end delay



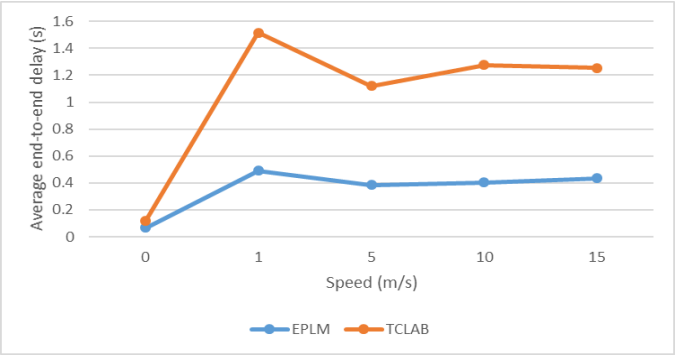
(c) Successful packet delivery rate

Fig. 3 Comparison of energy consumption, successful packet delivery rate and end-to-end delay of EPLM and TCLAB with different Probabilities of success in message transmission

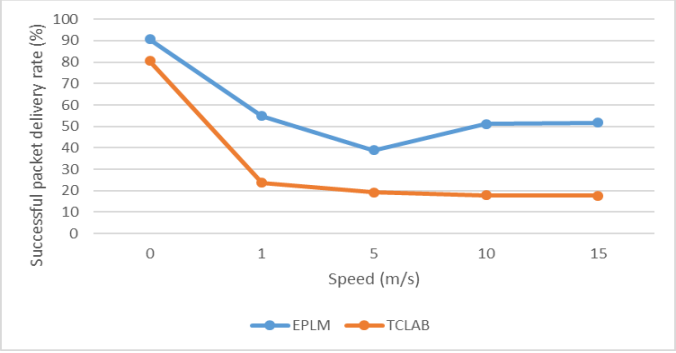
The number of neighbors is 6, number of servers is 2 and the probability of sending and receiving packages is 1. Simulation is performed for maximum ranges of 25, 50, 75, and 100 m (distances).
The results of simulations showed that changing the maximum transfer range (distance from the server), by keeping the distance between client and neighbor nodes constant, has little effect on energy consumption, latency and successful packet delivery rate. Because the calculations of the core do not differ and there is no change in sending and receiving messages between neighbors (because the number and distance of neighbors remain constant and only the distance of the server has changed). But changing the maximum transfer range in networks where nodes are moving will be effective due to the change in the number of neighbors (also in the stationary state, if the number of neighbors increases with increasing transfer range). Therefore, changing the data transfer range alone is not a very effective factor, unless it leads to a change in the number of neighboring nodes.




(a) Energy consumption




(b) end-to-end delay



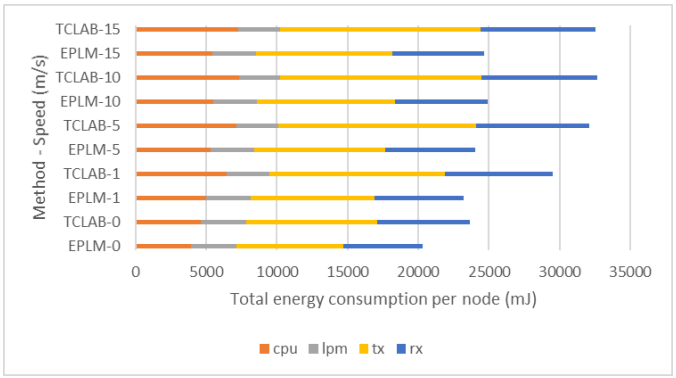

(c) Successful packet delivery rate

Fig. 4 Comparison of energy consumption, successful packet delivery rate and end-to-end delay of EPLM and TCLAB with different speeds (with 20 client nodes)

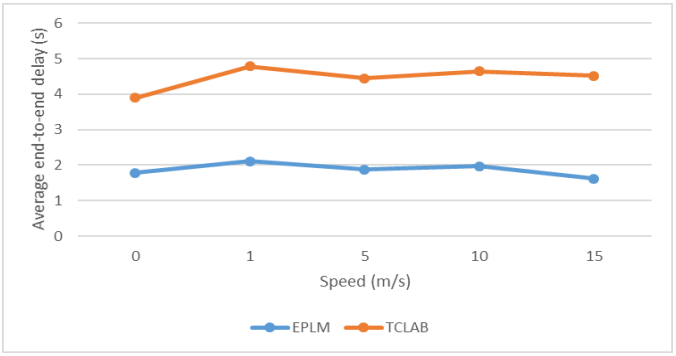
On average, EPLM outperformed 8.5% in energy consumption, 18.86% in successful package delivery rate, and 25.99% in end-to-end delay compared to TCLAB in total simulations in this part.

*4) Investigating the effect of mobility and velocity of nodes on efficiency*

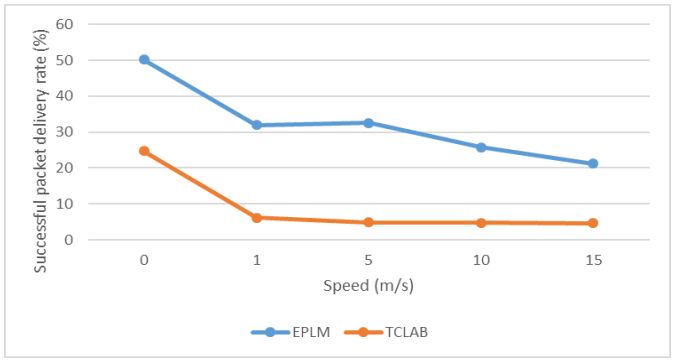
In many Internet of Things networks, nodes are moving. Mobility and velocity of movement are factors that affect the efficiency of Internet of Things networks. In order to conduct a more comprehensive and realistic test for comparing both methods and to investigate the effect of mobility and velocity, simulations with different mobility speeds and high number of client nodes have been performed. For all simulations, the number of client nodes is considered 20 or 45 and the number of server nodes is considered 9. For all simulations, the nodes are randomly arranged. Bonnmotion's program is used to create Random Waypoint model to create random motion. Random order and random motion have been considered the same in both methods. But in order to get a more realistic result without dependence on random ordering and movement, a number of simulations with different movements and arrangements are created for each method, and finally the average results are used for comparison. The probability of successful sending and receiving is 98% and the network space is considered 500 m * 500 m. The simulation results with 20 client nodes are shown in Fig. 4 and with 45 client nodes in Fig. 5.




(a) Energy consumption




(b) end-to-end delay




(c) Successful packet delivery rate

Fig. 5 Comparison of energy consumption, successful packet delivery rate and end-to-end delay of EPLM and TCLAB with different speeds (with 45 client nodes)

Due to the output of the simulations, the mobility of nodes has a significant effect on their efficiency. But changing the speed does not make a big difference in efficiency (compared to mobility and immobility). In other words, efficiency has experienced a greater change between the state of nodes' mobility or not and not to change the speed. If we leave the immobility mode aside (zero speed), there will be no difference by increasing the mobility of the nodes, the end-to-end delay (Fig. 4-b and Fig. 5-b) and

the successful package delivery rate (Fig. 4-c and Fig. 5-c), but the energy consumption of the nodes is increased (Fig. 4-a and Fig. 5-a).

EPLM has maintained its superiority over TCLAB, according to simulations conducted in this section. In addition, by increasing the size of network and the number of nodes, there has been an increase in performance differences, especially at successful package delivery rates. This indicates a major weakness in TCLAB, the use of which can only be justified for small networks, because as the size of the network and the number of nodes increases, the information of neighbors, which is a key element in the correct automata processing, is lost in an unacceptable way. Another important conclusion that can be drawn from simulations in this part is that the efficiency of both methods decreases with increasing network size, which is rooted in the limitations of devices on the Internet of Things. Finally, according to the results, TCLAB is very sensitive to network size and node mobility, and this sensitivity is much lower for EPLM.

For better comparison and independence of the results on a certain speed, other simulations are performed for both methods with variable speeds between 1 and 15 m/s. It should be noted that for each method, a number of simulations are performed and the mean results are used.

The results show that 8030 mJ energy consumption is more in TCLAB. In other words, TCLAB consumes about 33% energy more than EPLM. Also, the delay in TCLAB is 2.9 times that of EPLM and the successful package delivery rate is 0.18 times that of EPLM.

## IV. Discussion and Conclusion

Given the limitations in memory, energy sources, and processing power of devices used in the Internet of Things, it is vital to provide solutions to improve resource utilization. On the other hand, maintaining quality of service standards is essential to have efficient networks. To this end, several methods have been proposed to control power and channels to reduce energy consumption and improve service quality standards, some of which were mentioned in section II. Methods that have used learning, outperformed, but often neglect the inherent limitations of objects. Using an external processor (cloud, fog and edge) is a way to improve resource usage and network performance, unfortunately, there has been no research on the Internet of Things for power and channel control simultaneously.

In order to evaluate the efficiency of using an external processor to control power and channel, a new framework was provided based on learning automata and an external processor, and the efficiency of this method was compared with the best method that did not use an external processor (edge processing). Performance comparisons were made in several sections, and the superiority of the proposed method (EPLM) in memory consumption, power consumption, and processor utilization was proven. The results also showed an improvement in quality of service criteria when using an external processor (fog computing).

The results of simulations showed that if the automata was executed on the nodes, the quality standards of the service would be drastically reduced and the energy consumption would be greatly increased. This decline in quality makes it impossible to use the methods presented in section II for the limited nodes available on the Internet of Things. While using fog computing saves about 33% in energy consumption and reduces latency 3 time, and more importantly, the successful package delivery rate will improve up to 5 times. These results are given by assuming to have 1-step ($k = 1$) information about a neighboring, which also improves performance by increasing k, due to less error in detecting stable links. But increasing k is costly if an external processor is not used. However, the proposed method assumes that information is sent in one step, and if an increase in k is required, calculations in an external processor can be performed without applying any additional costs to the client nodes. This fact makes the use of EPLM even more preferable.

However, if a cloud, fog or edge processor is used, this transfer delay must also be calculated and considered based on the type of external processor and the facilities available. Also, the cost of using an external processor should be compared with the improvement in the network used, and the cost of replacing the energy sources of the nodes (different depending on environmental conditions) and the efficiency required by them (depending on the function of each node) should also be considered to decide on the use of EPLM in a specific network. However, due to the sensitivity of the rapid response to the environment in proposed method (especially if the nodes are mobile), the use of edge and fog computing is preferred over cloud computing due to better transfer speeds.

## References


[1] S. K. Gawali and M. K. Deshmukh, “Energy autonomy in IoT technologies,” in *Energy Procedia*, 2019, vol. 156, pp. 222–226, doi: 10.1016/j.egypro.2018.11.132.

[2] D. Sikeridis, E. E. Tsiropoulou, M. Devetsikiotis, and S. Papavassiliou, “Energy-Efficient Orchestration in Wireless Powered Internet of Things Infrastructures,” *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 2, pp. 317–328, 2019.

[3] X. Liu, Z. Qin, Y. Gao, and J. A. McCann, “Resource Allocation in Wireless Powered IoT Networks,” *IEEE Internet Things*, vol. 6, no. 3, pp. 4935–4945, 2019.

[4] N. Li, J. C. Hou, and L. Sha, “Design and analysis of an MST-based topology control algorithm,” *IEEE Trans. Wirel. Commun.*, vol. 4, no. 3, pp. 1195–1206, 2005.

[5] X. M. Zhang, Y. Zhang, F. Yan, and A. V Vasilakos, “Interference-Based Topology Control Algorithm for Delay-Constrained Mobile Ad Hoc Networks,” *IEEE Trans. Mob. Comput.*, vol. 14, no. 4, pp. 742–754, 2015.

[6] P. Santi, “Topology control in wireless ad hoc and sensor networks,” *ACM Comput. Surv.*, vol. 37, no. 2, pp. 164–194, 2005.
[7] D. M. Blough, M. Leoncini, G. Resta, and P. Santi, “The K-Neigh Protocol for Symmetric Topology Control in Ad Hoc Networks,” in *4th ACM international symposium on Mobile ad hoc networking & computing*, 2003.
[8] R. Wattenhofer and A. Zollinger, “XTC: a practical topology control algorithm for ad-hoc networks,” in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, 2004.
[9] D. Sikeridis, E. E. Tsiropoulou, M. Devetsikiotis, and S. Papavassiliou, “Wireless powered Public Safety IoT: A UAV-assisted adaptive-learning approach towards energy efficiency,” *J. Netw. Comput. Appl.*, vol. 123, pp. 69–79, 2018.
[10] J.-S. Leu, T.-H. Chiang, M.-C. Yu, and K.-W. Su, “Energy Efficient Clustering Scheme for Prolonging the Lifetime of Wireless Sensor Network With Isolated Nodes,” *IEEE Commun. Lett.*, vol. 19, no. 2, pp. 259–262, 2015.
[11] S. M. Jameii, K. Faez, and M. Dehghan, “AMOF: adaptive multi-objective optimization framework for coverage and topology control in heterogeneous wireless sensor networks,” *Telecommun. Syst.*, vol. 61, no. 3, pp. 515–530, 2016, doi: 10.1007/s11235-015-0009-6.
[12] K. Miyao, H. Nakayama, N. Ansari, and N. Kato, “LTRT: an efficient and reliable topology control algorithm for Ad-Hoc networks,” *IEEE Trans. Wirel. Commun.*, vol. 8, no. 12, pp. 6050–6058, 2009.
[13] H. Nishiyama, T. Ngo, N. Ansari, and N. Kato, “On Minimizing the Impact of Mobility on Topology Control in Mobile Ad Hoc Networks,” *IEEE Trans. Wirel. Commun.*, vol. 11, no. 3, pp. 1158–1166, 2012.
[14] N. Shirali and S. Jabbedari, “Topology control in the mobile ad hoc networks in order to intensify energy conservation,” *Appl. Math. Model.*, vol. 37, no. 24, pp. 10107–10122, 2013.
[15] H. B. Salameh, “Rate-Maximization Channel Assignment Scheme for Cognitive Radio Networks,” in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, 2010.
[16] C. Di., B. Zhang, Q. Liang., S. Li, and Y. Guo, “Learning Automata-Based Access Class Barring Scheme for Massive Random Access in Machine-to-Machine Communications,” *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6007–6017, 2019.
[17] D. P. Paramo, L. T. Oquendo, V. Pla, and J. M. Bauset, “Deep Reinforcement Learning Mechanism for Dynamic Access Control in Wireless Networks Handling mMTC,” *Ad Hoc Networks*, vol. 94, 2019.
[18] S. Gheisari and E. Tahavori, “CCCLA: A cognitive approach for congestion control in Internet of Things using a game of learning automata,” *Comput. Commun.*, vol. 147, pp. 40–49, 2019.
[19] W. Rehan, S. Fischer, and M. Rehan, “Anatomizing the robustness of multichannel MAC protocols for WSNs: An evaluation under MAC oriented design issues impacting QoS,” *J. Netw. Comput. Appl.*, vol. 121, pp. 89–118, 2018.
[20] Z. Beheshtifard and M. R. Meybodi, “Learning Automata Based Channel Assignment with Power Control in Multi-Radio Multi-Channel Wireless Mesh Networks,” *J. Telecommun. Syst. Manag.*, vol. 5, no. 139, 2016.
[21] R. S. Komali, R. W. Thomas, L. A. Dasilva, and A. B. Mackenzie, “The price of ignorance: distributed topology control in cognitive networks,” *IEEE Trans. Wirel. Commun.*, vol. 9, no. 4, pp. 1434–1445, 2010.
[22] P. Rahmani, H. H. S. Javadi, H. Bakhshi, and M. Hosseinzadeh, “TCLAB: A New Topology Control Protocol in Cognitive MANETs Based on Learning Automata,” *J. Netw. Syst. Manag.*, vol. 26, pp. 426–462, 2018.

# Figures

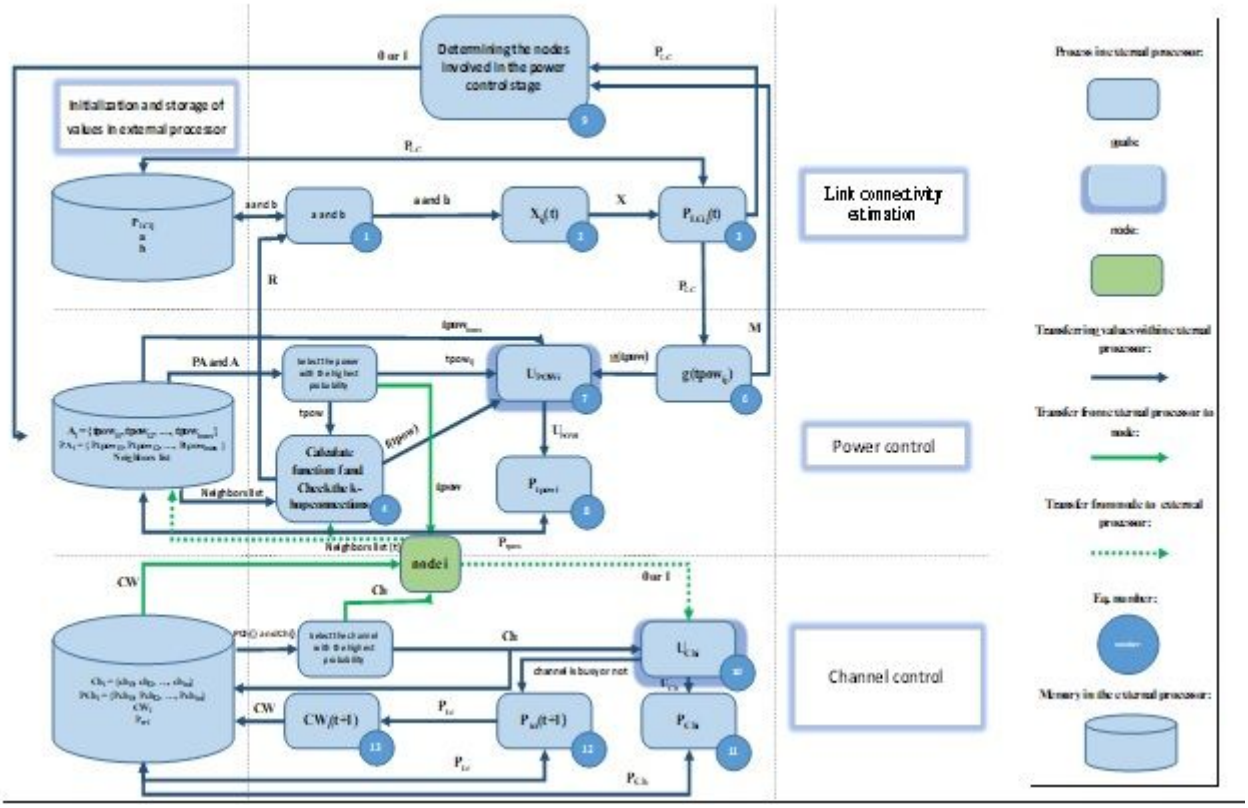
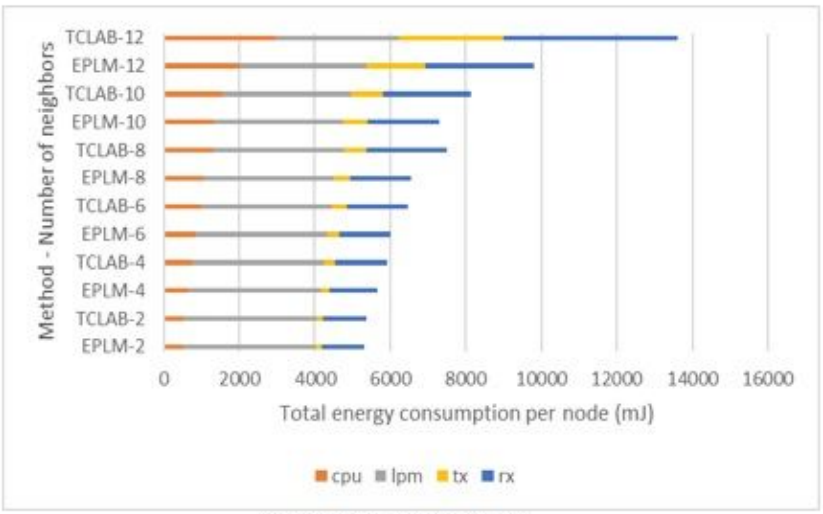
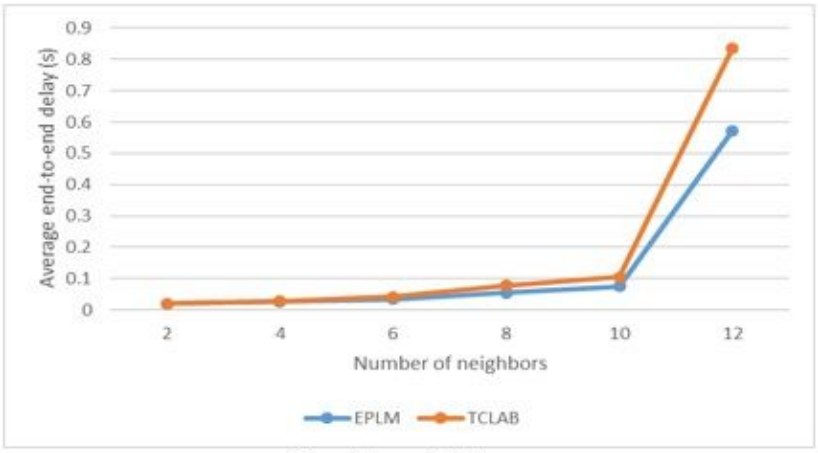



**Figure 1**

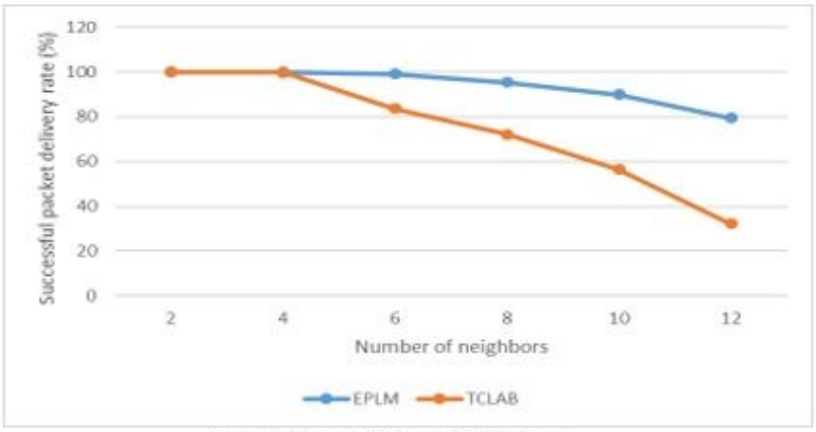
Processing cycle in EPLM




(a) Energy consumption



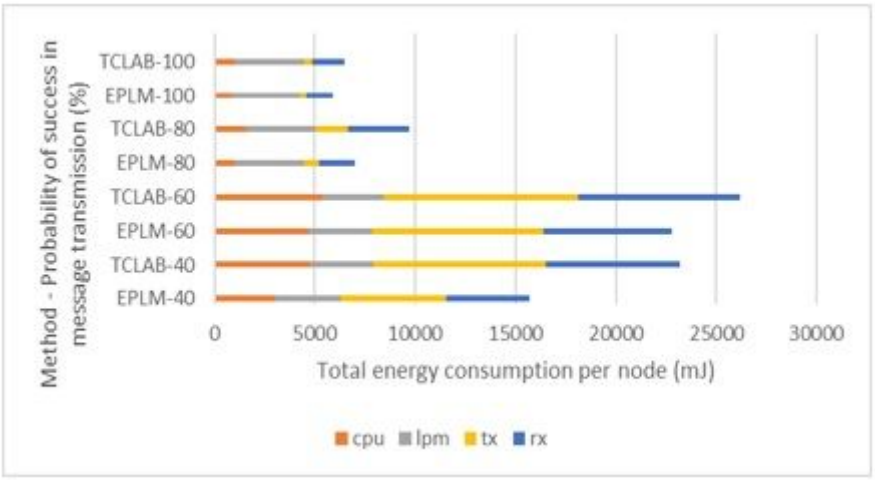

(b) end-to-end delay



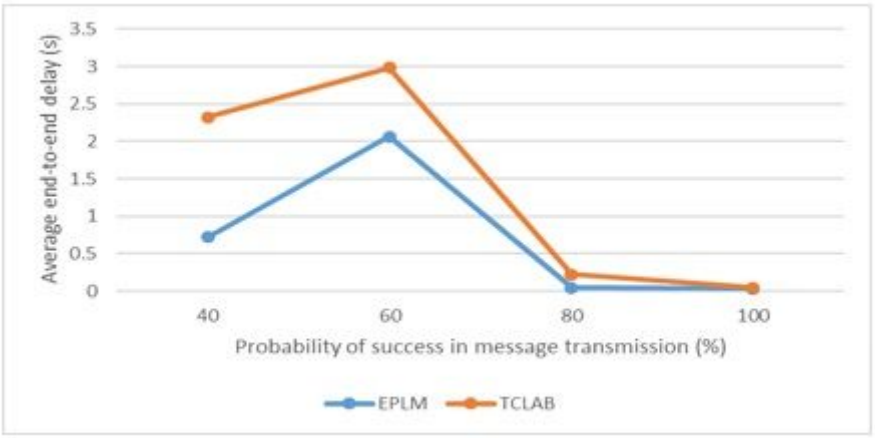

(c) Successful packet delivery rate

## Figure 2

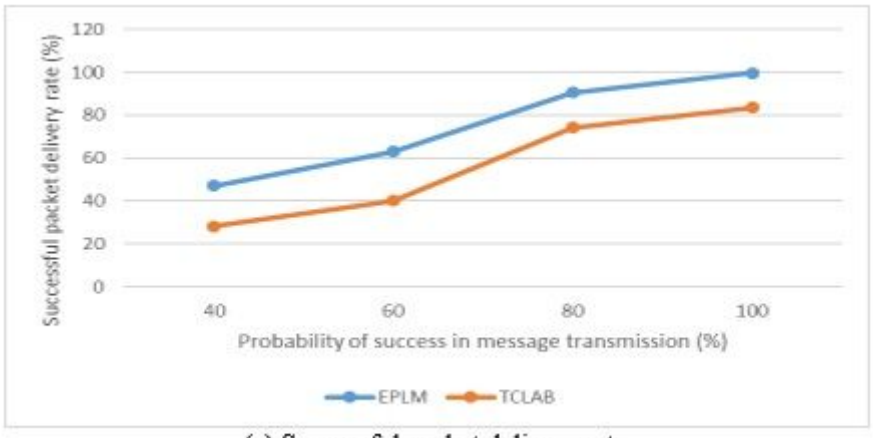
Comparison of energy consumption, successful packet delivery rate and end-to-end delay of EPLM and TCLAB with different Probabilities of success in message transmission




(a) Energy consumption



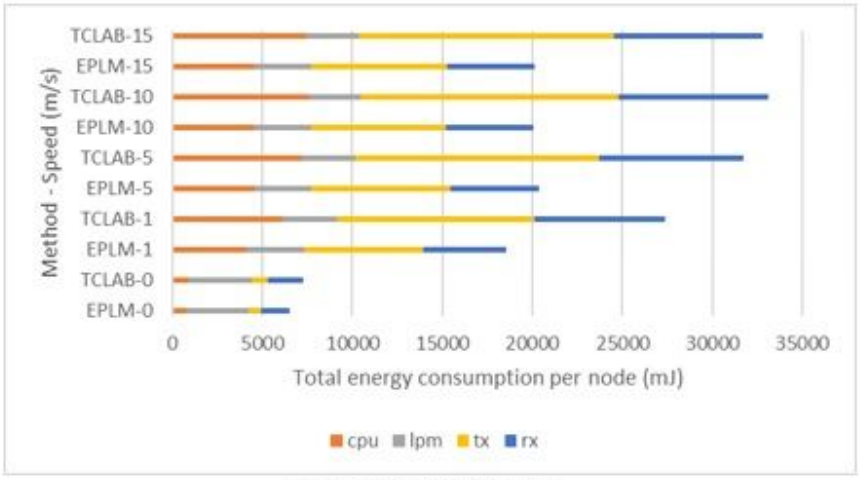

(b) end-to-end delay



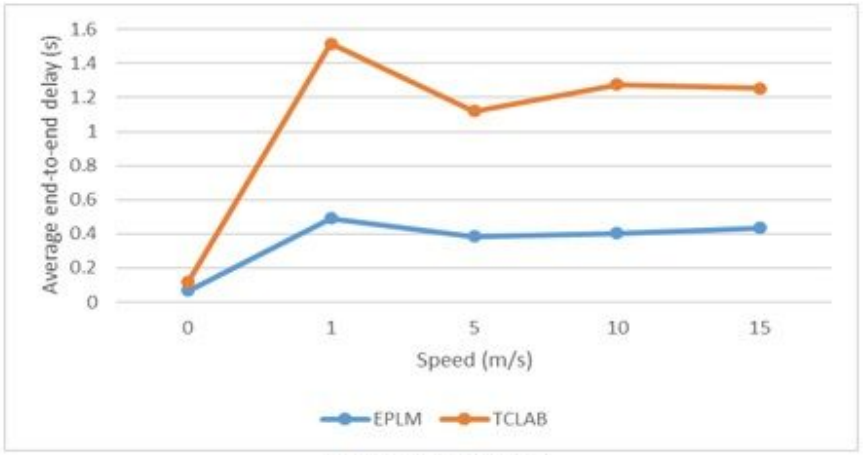

(c) Successful packet delivery rate

**Figure 3**

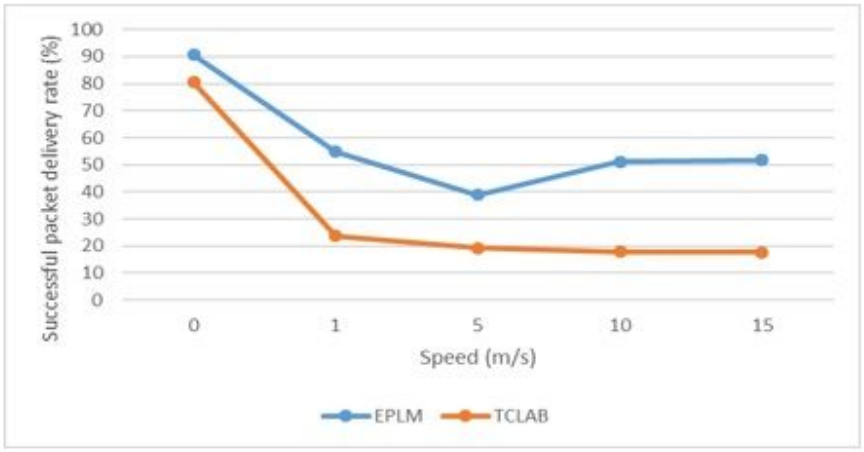
Comparison of energy consumption, successful packet delivery rate and end-to-end delay of EPLM and TCLAB with different Probabilities of success in message transmission




(a) Energy consumption



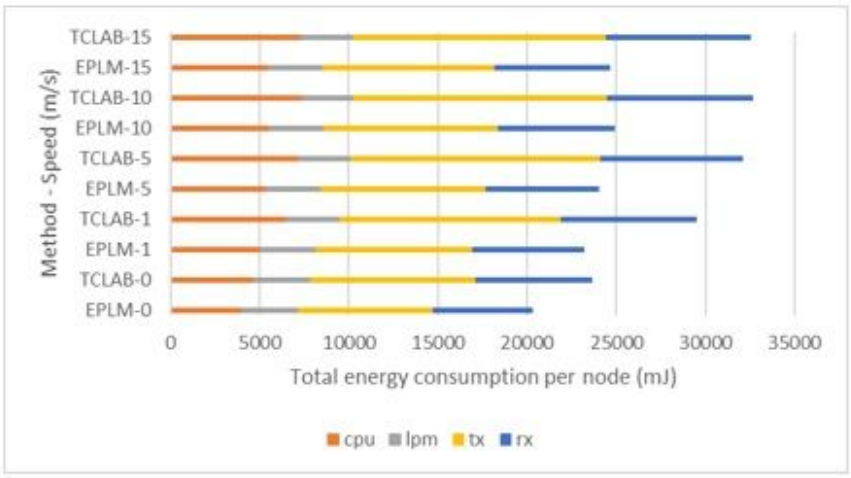

(b) end-to-end delay



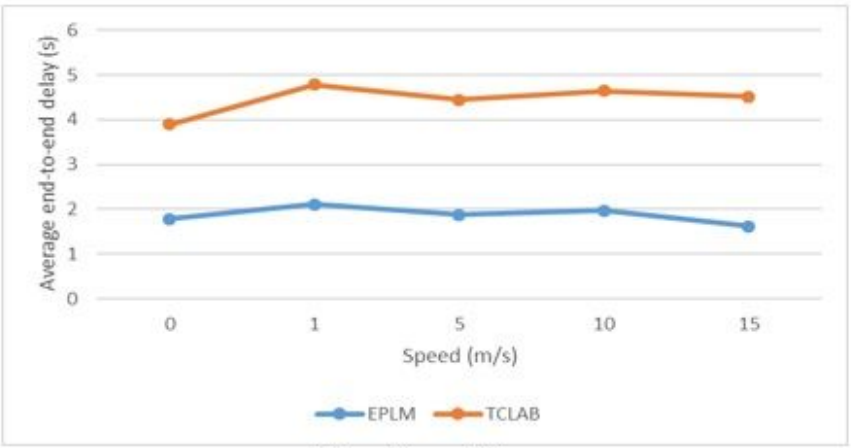

(c) Successful packet delivery rate

## Figure 4

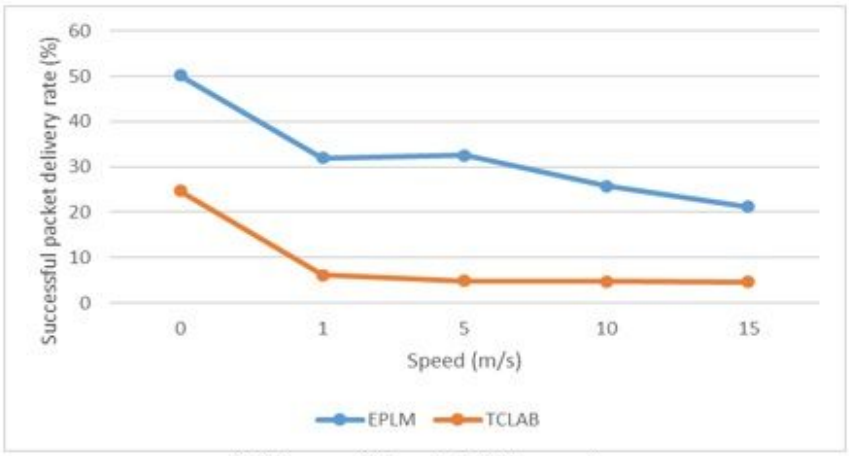
Comparison of energy consumption, successful packet delivery rate and end-to-end delay of EPLM and TCLAB with different speeds (with 20 client nodes)




(a) Energy consumption




(b) end-to-end delay




(c) Successful packet delivery rate

## Figure 5

Comparison of energy consumption, successful packet delivery rate and end-to-end delay of EPLM and TCLAB with different speeds (with 45 client nodes)