This paper is a preprint (IEEE "accepted" status).

A. Popovici and V. Stangaciu, "Evaluation of a Time Efficient Medium Access Policy: GTDMA," *2023 27th International Conference on System Theory, Control and Computing (ICSTCC)*, Timisoara, Romania, 2023, pp. 283-288, doi: 10.1109/ICSTCC59206.2023.10308457.
keywords: {Time division multiple access;Wireless sensor networks;Communication channels;Media Access Protocol;Control systems;Real-time systems;Energy efficiency;TDMA;slot-stealing;real-time;sensor network, simulation},

DOI. 10.1109/ICSTCC59206.2023.10308457

Evaluation of a Time Efficient Medium Access Policy: GTDMA

Alin Popovici
Department of Computer and Information Technology
Politehnica University Timisoara
Timisoara, Romania
popovici.alin@gmail.com

Valentin Stangaciu
Department of Computer and Information Technology
Politehnica University Timisoara
Timisoara, Romania
valentin.stangaciu@cs.upt.ro

***Abstract*—Time-Division Multiple Access (TDMA) medium access policy is widely used in wireless sensor networks where predictability is mandatory in communication. Such a policy is ideal for real-time applications but has an inefficient use of the communication channel. A significant improvement is given by the Greedy TDMA (GTDMA) access policy. In this paper, we present a complex simulation platform built on a popular sensor network simulator. We present an accurate analysis over the benefits of the GTDMA policy over the classic TDMA after intense simulations in many scenarios.**

***Index Terms*—TDMA; slot-stealing; real-time; sensor network, simulation**

I. Introduction

The typical sensor network contains a great number of sensor nodes. These are usually set up in dense clusters inside the environment they monitor in order to gather the most accurate data possible. Such networks can either send the data monitored directly to a processing unit or do simple computations locally before sending the data to a central network node. This cumulative processing done by each node represents a key feature of such networks.

Sensor nodes usually use a broadcast communication system. This can create conflict over the transmission medium and data mat be missed or corrupted. In order to solve this issue we can use a medium access control protocol. This logic gives order to the transmission. The data corruptions can be diminished or even eliminated.

These medium access protocols can take many different forms. Based on existing wireless sensor network MAC protocols we can divide them in four categories:

- Asynchronous - In such policies each node decides its own transmission schedule. Without the need for synchronization these kinds of MAC protocols can be very efficient in energy consumption because of low duty cycles. The downside is that a node that wishes to transmit needs to maintain a preamble long enough for all nodes to hear. This causes neighboring nodes to not be able to transmit during that time.
- Synchronous - In these types of protocols, the node listens to the channel for a determined amount of time. If it does not hear any schedule from other nodes, it calculates its next wake up and broadcasts its schedule. If it does receive a schedule from another node it follows the received schedule.
- Frame-slotted - These are protocols that are usually based on Time Division Multiple Access (TDMA) schemes. TDMA can be used in synchronized MAC protocol but if active periods of two clusters overlap collision free transmission can not be guaranteed. Therefore TDMA is usually implemented with a stricter global time synchronization.
- Multichannel - Because WSNs usually have limited bandwidth, it is desirable to use multiple channels to cope with burst transmissions. These types of protocols focus on channel allocation and on the problem of cross channel communication.

Time-division multiple access (TDMA) is a frame-slotted method for networks which consist of several nodes that use the same medium. It allows a number of nodes to share the same frequency channel by dividing the signal into different time slots. Each network node transmits in succession, one after the other, each using its own time slot. Usages of TDMA can vary from satellite communication, cellular systems to small scale networks. Such examples have relatively constant network structures and fixed sensor functions. Therefore, many recently proposed MAC protocols are TDMA-based when using networks with similar properties [1] [2]. In these networks the synchronization procedure can be simplified due to the rigid structure.

One important drawback of the TDMA access policy is the inefficient usage of the communication channel in the situation where most of the network nodes are silent, when other nodes have a large amount of data to be transmitted. A solution to this issue is represented by the GTDMA (Greedy TDMA) access policy [3]. In this paper we present a simulation environment designed to evaluate both TDMA and GTDMA medium access policies. Furthermore, we present our extensive measurements which demonstrate the effectiveness of the GTDMA scheme against the inefficient use of the communication channel by the classic TDMA.

II. TDMA based MAC protocols

In Fig. 1 we can see a visual representation of 2 TDMA frames consisting of 7 slots. Each slot is assigned to a different

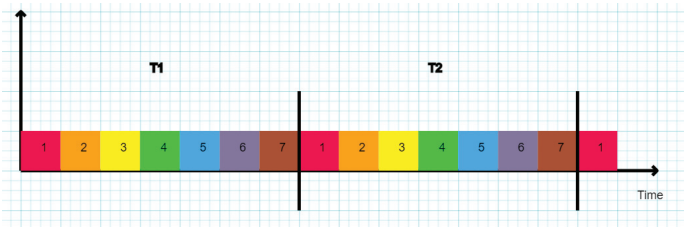


Fig. 1: TDMA slot assignment

node based on color. In this particular image the sequence of slots is unchanged in the two frames.

Gathering the data from [4], [5], [6] we can deduce the following aspects of TDMA based MAC Protocols.

One advantage for using such protocols is the possibility for a specific node to have a flexible transmission rate. We also have the possibility to assign several time slots to just one specific node. This is in order to easily have nodes with variable bit rates be part of the same network. Because of the logic of TDMA, where each node has its own time to transmit, we can no longer have the hidden node problem during transmission. Other advantages of TDMA based MAC protocols are the lack of need for a guard band, the slot assignment can be changed frame by frame and it can withstand a variable bit rate during transmission.

The major disadvantage of such MAC protocols is the large amount of data needed to synchronize the transmission. Besides the large amount of data, the synchronization together with the slot allocation logic can become very complex.

There are a lot of TDMA based protocols that have been proposed in research, but only a few have been actually implemented in reality. Of those even fewer have the backing to be part of a standard in an industry. Because there are a lot of existing protocols that derive from TDMA, we will go over a few directions of improvement that already exist. Energy efficient TDMA (E-TDMA) is based on a low-energy adaptive clustering hierarchy and focuses on low energy consumption. In this method the cluster head is selected based on remaining energy and it then attributes the time slots to other nodes [7]. Bit map assisted (BMA) is designed for event driven applications. Each cycle is divided into three parts. Contention period is based on TDMA schedule and each node transmits its contention data. Head then sends the transmission schedule [8]. EA-TDMA is mainly design for monitoring lateral and vertical instability in train rail carts. It is energy efficient and can handle high loads. Each node's transmitter is active only during their allotted time slot. It is not very adaptable at run time [9]. BMA Round-Robin MAC protocol design to increase throughput and reduce energy consumption depending on transmission requests. Vacant time slots are allocated to needy nodes by using the round-robin technique [10]. E-BMA is a more energy efficient BMA. The source node does not transmit data immediately after it becomes available, it waits for the next frame to check whether it has successive packet to send [11]. BS-MAC is design for hierarchical WSN. It utilizes small time slots which are allocated to nodes according to their traffic load. It prioritizes the shortest job first. Reduces node address from 8 to 1 byte to reduce control overhead and reduce energy drain [12]. BEST-MAC is bit map assisted, efficient and scalable. It utilizes small time slots which are allocated to nodes according to their traffic load. It utilizes Knapsack Algorithm [13] for slot scheduling. It reduces packet delay, increases link utilization. It introduces a Contention Access period to provide scalability to the network. It assigns a short address to each node to reduce the overheads.

III. GTDMA

As mentioned for TDMA each node gets assigned a slot in a frame and can only transmit during that time interval. In order to take advantage of the slots where a node does not transmit we let other nodes transmit during that time. This is the main idea behind GTDMA. In order to enable the feature we take each node slot and split it further in smaller slots. Each of these slots get assigned to other nodes. The first small slot is assigned to the node who would normally send during the normal TDMA frame. Each small slot gives the nodes they belong to the opportunity to send outside their normal assigned TDMA slot. Each node corresponding to one of the small slots can start sending when the time reaches its small time slot and only when the nodes corresponding to previous small slots did not start sending. Once a node starts sending, the whole TDMA time slot that is left is used up. In a GTDMA slot, to put it simply, we have a list of nodes in order of priority that sequentially get the opportunity to send if the previous nodes in the list do not use up the opportunity already. In Fig. 2 we see how the GTMDA small slots get assigned to each node compared to normal TDMA. For example for TDMA node 1 (red) we assign to it four GTDMA slots corresponding to nodes 1(red), 7(brown), 6(gray), 5(blue). If node 1 does not have data to transmit then node 7 gets the opportunity to take the slot.

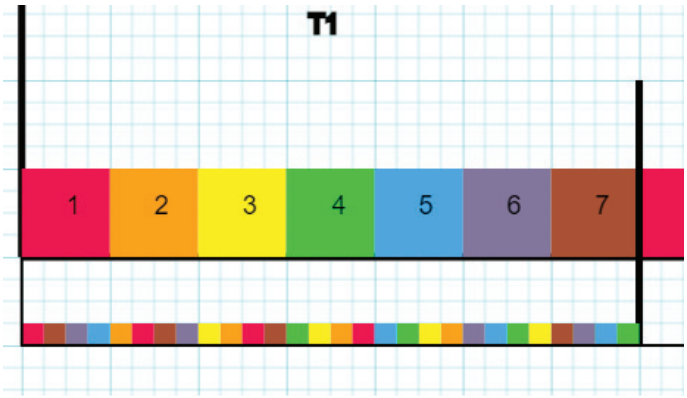


Fig. 2: GTDMA slot assignment

IV. Simulation environment

OMNeT++ is a component based simulation tool primarily used for building network simulations. It is based on C++

and has the advantages of being modular and extensible [14]. Because of the focus on simulating messaging systems, it is particularly suited to simulate different network types. It has the possibility of simulating networks with varied properties (wired, wireless, on-chip, etc) in a multitude of domains (sensor networks, internet protocols, cellular networks, etc.). Due to these features it is well suited to simulating the TDMA and GTDMA protocols.

The Network is composed of three distinct modules. These are the Coordinator, Nodes and Sink modules. They are connected in the following manner: Coordinator to all instances of the Node through a bidirectional connection; All node instances to the Sink though a one way connection. All the connections together form the transmission medium. Each connection can be set up with a particular delay. The current simulation is running under ideal scenarios, where there are no delays or collisions.

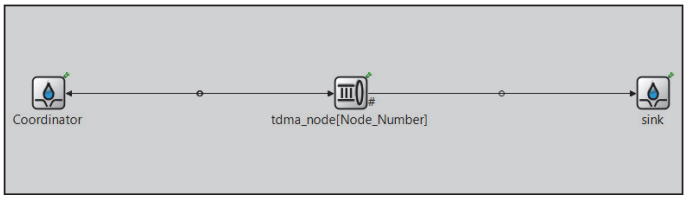


Fig. 3: Parametric network structure

An overview of the network structure is visible in Fig. 3. Here we can see the connections between the elements and also the direction of the communication. In order to better visualize the network, the array of nodes is displayed as a single entity in this case.

a) The Coordinator: is the most important module in this simulation. It has the role of coordinating the moments in which data needs to be transmitted. It also manages the data getting transmitted. It has the responsibility to decide if a node can send data and what data it should send. It also keeps track of whether a node is sending. This is in order to manage all the time slots for GTMDA.

b) The Node: can be instanced multiple times in the network. It receives from the coordinator, with no delay, when it needs to send data and what data to send. It has the role to send the messages to both Sink and Coordinator. Its existence permits the expansion of the simulation to include transmission delays and collisions.

c) The Sink: gets the messages from all nodes and has the duty to report them to statistics.

The network simulation is very configurable and can easily run different scenarios. The following Configurations from Table I need to be set up in order for the simulation to run.

The *Node_Numbe* indicates how many nodes get simulated. The *Time_Slots* parameter sets the number of slots present in a frame. While *Slot_Bytes* sets the size of the slots, *Small_Slots* indicates in how many parts a slot gets split in order to simulate GTDMA. The *Time_Period* sets how much time a frame takes to transmit. While *Data_cycles* indicates

TABLE I: Simulation Parameters

Name	Data type	Min Value	Max Value
Node_Number	int	0	1
Time_Slots	int	1	Node_Number
Slot_Bytes	int	1	0
Small_Slots	int	0	Slot_Bytes
Time_Period	double	0.1	1
Stop_after_cycles	int	1	
Data_cycles	int	1	
TDMA_only	int	0	1
Input_File	int	0	1
Small_Slots_Arrangement	int	0	1
Data_Generation_Probability	int	1	100
Data_High	int	Data_Low	
Data_Low	int	1	Data_High
Data_Transmission_Mode	int	1	5

how many frames data is generated, *Stop_after_cycles* is the number of frames until the simulation is forcefully shut down. The parameter *TDMA_only* indicates what type of network to simulate TDMA or GTDMA. *Data_Generation_Probability* sets the probability of generating data for each node. *Data_High* and *Data_Low* are being used to set the minimum and maximum byte count of the data to be generated. *Data_Transmission_Mode* has the role to select between the 5 modes available for transmitting the data generated. With these parameters, several prebuilt configuration files were built, for ease of use, for some key scenarios that were simulated numerous times.

At the beginning of the simulation the parameters are taken by the Coordinator from the network parameters set up by the initialization file. Default variables, like the current time slot, are also initialized. If GTMDA is simulated, the small time slots are assigned to particular nodes based on the input parameter *Small_Slots_Arrangement*. They are set either in a circular manner or random, depending on the parameter value.

If an input file is given, each node receives the data that it needs to transmit from the file. The option to have an input file as the source for the data is indicated by the parameter *Input_File*. The file is an .xml and contains a string of pairs of probability to transmit and the number of bytes to transmit. This data is Generated externally to have better control and visibility. The structure of the file can be seen in Fig. 4.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<heatSupplyLoadProfile>
<param>
  <data> 57 </data>
  <probability> 88 </probability>
</param>
```

Fig. 4: XML file format

If there is no input file the probability of a node is taken from the *Data_Generation_Probability* parameter. For each TDMA slot where we want to generate data we randomize a number between 1 and 100. If the generated number is smaller or equal to the probability parameter we also generate data. This data is generated depending on the *Data_High* and

Data_Low parameters. OMNet++ allows us to randomizes the same values in the same order when running the same scenario. This is enabled by using the same random seed each time we generate data. In order to generate different values we need to change the random seed.

After this setup, the Coordinator creates a master self message representing the first TDMA slot time. When receiving this message it always schedules the next one with the same time delay. By sending this message with a constant delay we can recreate the TDMA slots. Each moment the self message returns, it represents another slot. The Coordinator also keeps track of the current node that needs to transmit and if it has data available it sends a message to the current node that is allowed to transmit.

In case it runs the GTDMA protocol it also generates self messages for each extra time slot. The normal TDMA self messages remain the same. When such a message is received, depending on the number of small slots, we generate self messages to indicate the GTDMA slots. The GTDMA slot messages are sent with a fixed delay between them and all get received before the next TDMA slot self message. When receiving one of these GTDMA self messages, the Coordinator checks if a node has already used up the slot. If the node that corresponds to this time has something to send, the Coordinator sends a message to it to inform the node that it is allowed to send and how much data it can send. Because some of the TDMA slot was already used up finding a node that wants to transmit, the node that gets the opportunity will only have limited bytes to send traffic.

The data generated or input through a file can be interpreted in different manners based on the parameter *Data_Transmission_Mode*. Current simulation can be configured in the following manners:

- Mode 1 - It works per frame. For each node of the frame, it randomizes if a node wants to transmit based on the probability given. The data for that particular node is transmitted only in this frame and discarded for the rest of the frames that follow.
- Mode 2 - It checks the data that a particular node wants to transmit for the whole simulation and sums the number of bytes. The data to be sent is again randomized, as in Mode 1. Each node then sends the data when a possibility appears for it to transmit.
- Mode 3 - Data to be sent is being processed exactly the same as for Mode 2 . The difference is that a node can transmit only if the data for that particular frame passed the random probability check.
- Mode 4 - Similar to Mode 3. It transmits only when the probability check passes. This mode takes inter packet gaps into account. Data from 2 different packets can not be in the same frame.
- Mode 5 - Each frame, the node has a probability to generate data. Data generated is added to a waiting queue. Data is not dropped and it is sent at the next opportunity for the node.

The time of transmission, node index and byte count are the most important values that get extracted out of the simulation. They are extracted for each data sent by each node. Furthermore, the total number of bytes of data and total bytes transmitted get gathered for each node. The format of the data in the simulator is vectorial but can be easily exported in several file formats. The one used in this analysis of the CSV format because of the ease of iteration in different scripting tools.

V. Simulation results

One way to measure the efficiency of utilizing the transmission medium is to count the total number of frames taken by each node to transmit the data it generated completely. In order to more easily process and visualize the data we only look at the last frame in which a node transmits data. This way we might miss on slight improvement of the transmit speed, on each individual packet in GTDMA compared to TDMA. These small improvements will eventually propagate to the end of transmission but will be averaged over all transmissions. For packets considered large in comparison to each node slot, we can completely disregard these small improvements, since each node is expected to send almost each transmission frame due to data back pressure.

If the data generated by the nodes is smaller than the TDMA slot size assigned to it, there is no difference in efficiency between TDMA and GTDMA. In the following simulations several scenarios were run under similar network parameters. Number of sensor nodes used in the simulation is 10. Number of cycles used to generate data is also 10. The simulation runs as long as there is still data to be sent. For this scenario the data generation percentage is set to 20. Normal TDMA slot size for each node is 64 bytes. For GTDMA the slots are split in 16 and assigned in a circular manner to the other nodes.

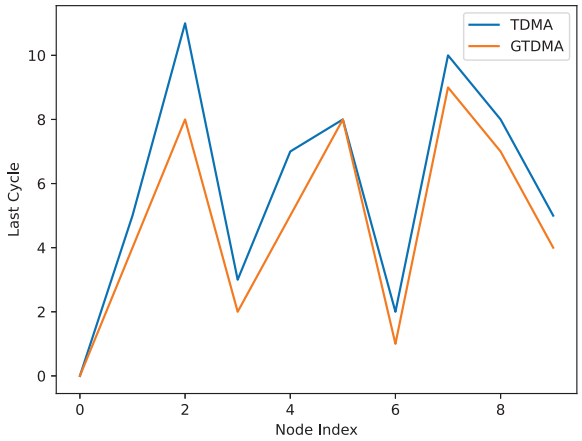
For this Network we simulated scenarios with packet sizes in several ranges

- Packet size between 64 and 96 bytes
- Packet size between 96 and 128 bytes
- Packet size between 128 and 256 bytes
- Packet size between 256 and 512 bytes

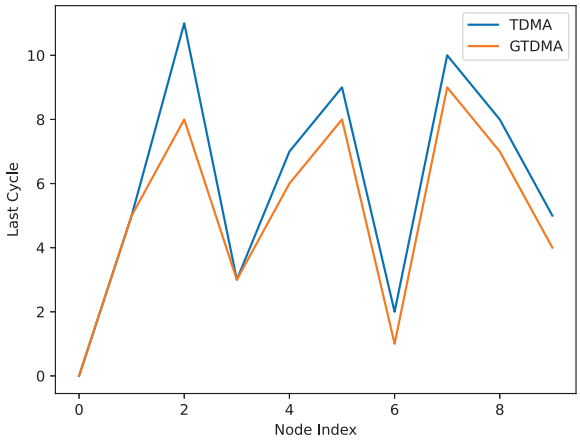
The results of these scenarios, seen in Fig. 5, show the number of frames used up by each node in both TDMA and GTDMA. We can notice that as the size of the packets increases, the difference in frames used up increases as well.

For the same network configuration described before we simulated the same packet size for an increased probability to transmit data. In Fig. 6 we see that with an increase in probability from 20% to 50% the difference of frames used increases.

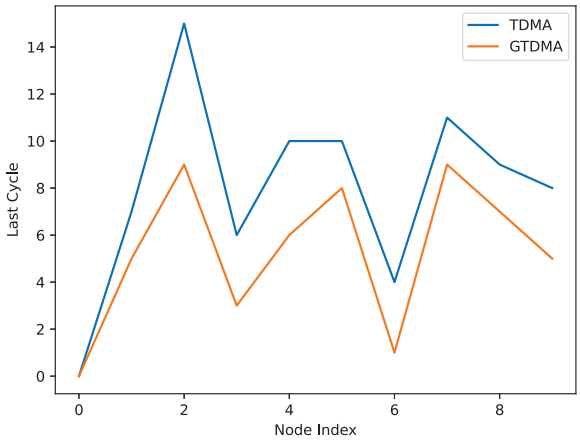
We see that as the probability of transmitting traffic increases, the advantages of GTDMA over TDMA will increase as well. We expect this to happen up to a point. From that moment on we expect to see that the difference becomes less. When there are a lot of nodes generating traffic at the same time, there are fewer and fewer slots that are left unused. This decreases the chances of other nodes using up the unused



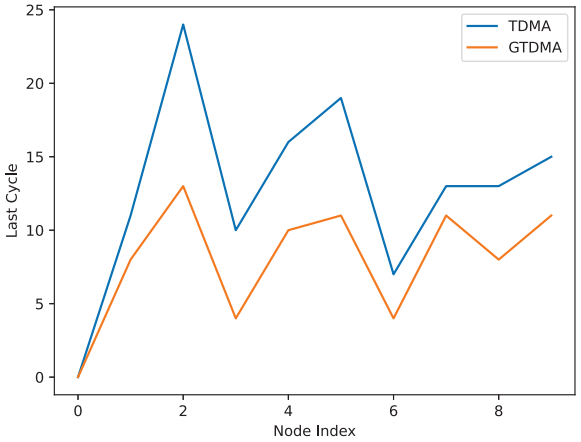
(a) Packet size between 64 and 96 bytes



(b) Packet size between 96 and 128 bytes



(c) Packet size between 128 and 256 bytes



(d) Packet size between 256 and 512 bytes

Fig. 5: Last frame used in TDMA versus GTDMA for 20% probability

slots, thus makes it so that the efficiency of GTDMA over TDMA diminishes as well. In Fig. 7 we see the difference in frames used up by the TDMA compared to GTMDA. We see that the best results are for 50% transmit chance. While 20% and 75% come close to each others performance, the 90% transmit chance barely shows any difference between TDMA and GTDMA.

Fig. 8 presents the average frame usage improvement per node of a GTMDA over a TDMA network. Each node of this network has a change of 50% to generate data in a particular frame. The data generated has an average size of 6 times the slot size. The scenarios simulated were with 10, 100, 200, 500 and 1000 nodes. In all cases, the TDMA time slots were set to size 64 and were split in 16 equal parts for the GTDMA simulation. As we see in Fig. 8, the advantage of GTDMA remains constant between 5 to 6 frames efficiency on average for each node. This indicates that the benefit of using GTDMA over TDMA can scale to larger networks.

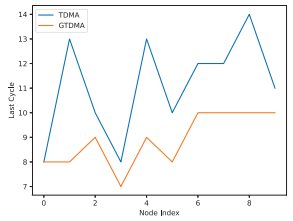
VI. Conclusions

As the priority of speed increases over energy efficiency in TDMA based MAC protocols, GTDMA represents a clear method of improvement. We see that in a burst environment, typical for sensor networks, the transmission medium usage can be improved by using GTDMA over TDMA.

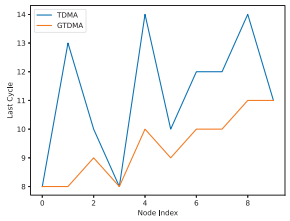
The improvements of GTMDA over TDMA do not affect all networks. Those networks, where nodes generate data very frequent, do not get any benefit. The networks, where nodes generate data seldom, will get a great improvement for each individual packet, but over a longer period of time the improvements are not as clear. For the rest of the networks, where improvement is seen, it is considerable and remains constant over changes in the scale of the network.

References

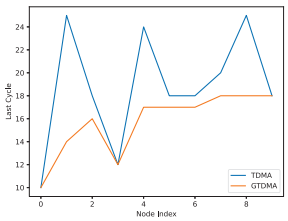
[1] Z. Chen, C. Hu, J. Liao, and S. Liu, "Protocol architecture for wireless body area network based on nrf24l01," in *2008 IEEE International Conference on Automation and Logistics*, 2008, pp. 3050–3054.



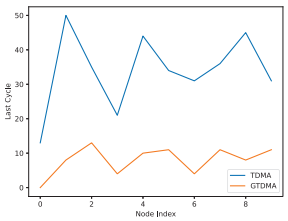
(a) Packet size between 64 and 96 bytes



(b) Packet size between 96 and 128 bytes



(c) Packet size between 128 and 256 bytes



(d) Packet size between 256 and 512 bytes

Fig. 6: Last frame used in TDMA versus GTDMA for 50% probability

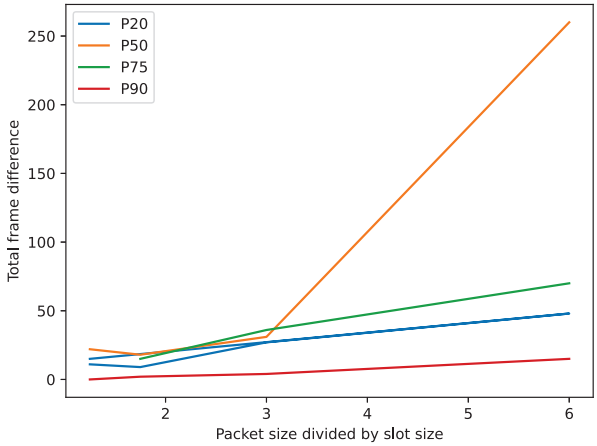


Fig. 7: Frame Difference Improvement

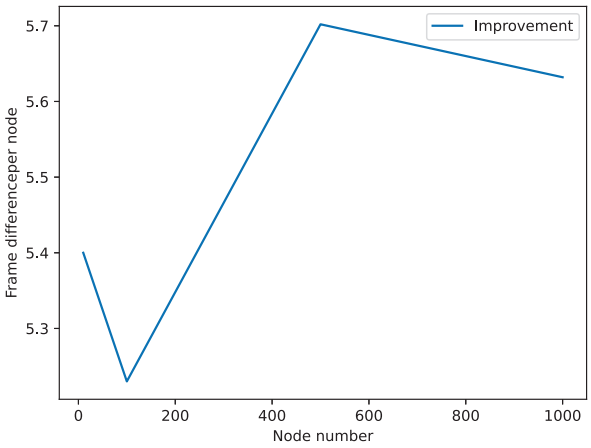


Fig. 8: Improvement Scaling

[2] O. Omeni, A. C. W. Wong, A. J. Burdett, and C. Toumazou, "Energy efficient medium access protocol for wireless medical body area sensor networks," *IEEE Transactions on biomedical circuits and systems*, vol. 2, no. 4, pp. 251–259, 2008.

[3] V. Stangaciu, M. V. Micea, V. I. Cretu, and V. Groza, "General slot stealing tdma scheme to improve the low channel utilization factor," in *2015 IEEE 9th International Symposium on Intelligent Signal Processing (WISP) Proceedings*, 2015, pp. 1–4.

[4] A. Kakria and T. C. Aseri, "Survey of synchronous mac protocols for wireless sensor networks," in *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*, 2014, pp. 1–4.

[5] V. Sevani, B. Raman, and P. Joshi, "Implementation-based evaluation of a full-fledged multihop tdma-mac for wifi mesh networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 2, pp. 392–406, 2014.

[6] T. Kaur and D. Kumar, "Tdma-based mac protocols for wireless sensor networks: A survey and comparative analysis," in *2016 5th International Conference on Wireless Networks and Embedded Systems (WECON)*, 2016, pp. 1–6.

[7] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[8] J. Li and G. Lazarou, "A bit-map-assisted energy-efficient mac scheme for wireless sensor networks," in *Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004*, 2004, pp. 55–60.

[9] G. M. Shafiullah, A. Thompson, P. J. Wolfs, and S. Ali, "Energy-efficient tdma mac protocol for wireless sensor networks applications," in *2008 11th International Conference on Computer and Information Technology*, 2008, pp. 85–90.

[10] T.-H. Hsu and P.-Y. Yen, "Adaptive time division multiple access-based medium access control protocol for energy conserving and data transmission in wireless sensor networks," *IET communications*, vol. 5, no. 18, pp. 2662–2672, 2011.

[11] G. Shafiullah, S. A. Azad, and A. S. Ali, "Energy-efficient wireless mac protocols for railway monitoring applications," *IEEE transactions on intelligent transportation systems*, vol. 14, no. 2, pp. 649–659, 2012.

[12] A. N. Alvi, S. H. Bouk, S. H. Ahmed, M. A. Yaqub, N. Javaid, and D. Kim, "Enhanced tdma based mac protocol for adaptive data control in wireless sensor networks," *Journal of Communications and Networks*, vol. 17, no. 3, pp. 247–255, 2015.

[13] A. N. Alvi, S. H. Bouk, S. H. Ahmed, M. A. Yaqub, M. Sarkar, and H. Song, "Best-mac: Bitmap-assisted efficient and scalable tdma-based wsn mac protocol for smart cities," *IEEE Access*, vol. 4, pp. 312–322, 2016.

[14] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2010.