Scaling up Routing in Nanonetworks with Asynchronous Node Sleeping

Ali Medlej*, Kamal Beydoun†, Eugen Dedu* and Dominique Dhoutaut*
*FEMTO-ST Institute, Univ. Bourgogne Franche-Comté, CNRS
Montbéliard, France
Email: firstName.lastName@univ-fcomte.fr
†L'ARICoD Laboratory, Faculty of Sciences I, Lebanese University
Beirut, Lebanon
Email: kamal.beydoun@ul.edu.lb

***Abstract*—In any network, routing and congestion control protocols play a key role, as they allow packets to reach the intended destination. We are interested in wireless nanonetworks (WNNs), which are networks whose nodes have a nanometric size and can be potentially dense in terms of neighboring nodes. The nodes have limited processing capabilities and power. The objective of this research is to present a fine-grained sleeping mechanism for nodes, whose aim is to reduce node resource usage and thereby increase the network life. We evaluate the sleeping mechanism by presenting the impact of network density on packet reachability and network resources used. Simulations demonstrate the effectiveness of this mechanism and show that nanonode resources (CPU, memory, energy) can be preserved by decreasing the number of forwarded and ignored packets while ensuring the arrival of the data packets to the destination.**

***Index Terms*—Routing, Congestion, Nanonetwork, Duty-cycling, Scalability**

I. Introduction

Wireless sensor networks are built from tiny nodes, equipped with embedded computing, sensing and actuating devices. They usually have small CPU, small memory and low battery. Upcoming nanosensor networks will use electromagnetic waves from the THz band (0.1–10 THz) for their communications [1]. Due to power constraints, they will also have to use multi-hop communications to cover large areas.

In a traditional dense sensor network [2], congestion is the state where the number of packets exceeds the network capacity. Some of them are discarded or lost due to collisions. Contrary to traditional wireless networks, congestion in nanonetworks [3] does not arise from a saturated channel, but from an insufficient capacity of individual nodes on the multi-hop path to process all the incoming concurrent packets.

Routing protocol is responsible to route packets from source to the desired destination. As mentioned before, during packet routing and due to network congestion, packets may not reach their destination, which makes the communication in the network unreliable. To ensure packets delivery, a reliable routing protocol taking congestion into account should be used. SLR (Stateless Linear-path Routing) [4] is the protocol we use in our evaluation. It implements a coordinate-based routing, in which data packets are routed in linear routing path. Nodes are assumed to be placed in a cubic space, distributed

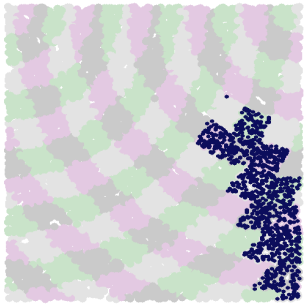

Fig. 1. SLR routing zones.

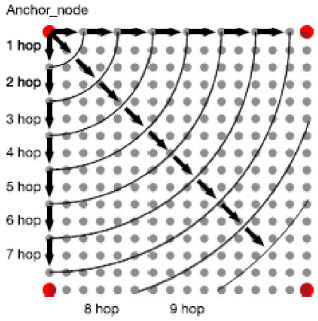

Fig. 2. SLR addressing phase.

in zones (Fig. 1). In the initial SLR phase, during network deployment, a few anchor nodes broadcast a packet (beacon) to the whole network. The hop counter in those beacons is used to define the coordinates of all nodes as a distance to the anchors (Fig. 2). In the second phase, during data packet routing, nodes choose to forward a packet if and only if they are on the path between the source and the destination of the packet (Fig. 1).

SLR protocol uses the TS-OOK (Time Spread On-Off Keying) modulation [5] to share the radio terahertz channel for nanodevices, which is based on femtosecond-long pulses where packets are transmitted as a sequence of pulses interleaved by a given duration, cf. Fig. 3. "1" bits are encoded with a power pulse of duration T_p and "0" bits are encoded as silence. Because sending consecutive pulses needs unavailable hardware and power at such small sizes, consecutive bits are spaced with a duration T_s which is usually much longer than the pulses themselves. Collision occurs when the receiver is receiving a "0" bit (silence), but in the same time a "1" bit arrives, which effectively shadows the "0", cf. Fig. 4. TS-OOK parameters can be summarized by T_p (pulse duration), β (spreading ratio, $\beta = T_s/T_p$) and T_s (time between symbols).

Even if the channel capacity is very high, nodes have very low capabilities and cannot completely use it individually. Due to the TS-OOK's ability to interleave packets over the channel, nodes can receive multiple packets concurrently. However, the number of packets that could be received concurrently is

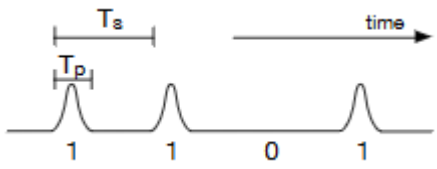

Fig. 3. TS-OOK modulation.

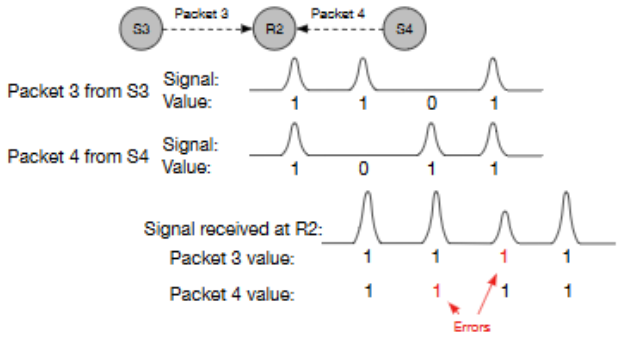

Fig. 4. TS-OOK error bits reception.

bound to be limited, due to technical constraints. We call this limitation emanating from CPU, memory or other hardware *maximum concurrent receptions*. In the following evaluations, we will consider this limitation to be equal to 5. This means that while 5 packets are already being received, new incoming ones will be silently ignored. This is different from collisions, as the packets being received are not affected. This mechanism nevertheless means that nodes can very easily be saturated in a given area, effectively reducing the available network capacity and producing a new form of congestion.

Keeping a nanonode awake for the whole T_s duration makes it easy to saturate. Our idea is to make it inactive for a fraction of T_s, effectively implementing a very fast and sub-packet level duty cycle. Thus, the node will not be able to receive *all* incoming packets anymore. But we compensate this by having the other nodes asynchronously apply the same mechanism. By doing so, incoming packets are handled and routed only by nodes that are awake.

In this paper (1) we introduce a novel fine-grained sleeping mechanism dedicated to nanoscale networks and integrate it with SLR protocol. (2) We prove the efficiency of this combination in reducing congestion in nanonetworks, allowing packets to reach their destination and preserving nanonode resources (CPU, memory, energy). Also, (3) we investigate the impact of the network density and how it reflects on the optimal awake duration.

The rest of this paper is organized as follows. Section II introduces the aspect of congestion detection and recovery between macro-scale and nanoscale. Section III presents the sleeping mechanism. Evaluation via simulations takes place in Section IV. Finally, conclusion and future work are drawn in Section V.

II. Related Work

Congestion occurs when numerous packets have to be exchanged but nanonodes are not able to process them because of the limited resources. Congestion leads to packets loss and interference, which affect the network efficiency and lifetime [6].

Congestion can have multiple causes, but the most common ones are either a low link bandwidth or the lack of network resources - especially the buffer size in the nanonodes [7]. The traditional solution was either to decrease the rate of packets injected at the source, or to drop packets when nanonodes are unable to receive and store them due to buffer saturation.

The mechanisms used to avoid congestion are listed under the term of congestion control. In macro-scale internet, TCP (Transmission Control Protocol) is one of the well known protocols that have a network congestion avoidance algorithm [8]. It includes various aspects of an additive increase / multiplicative decrease (AIMD) scheme [3], along with other schemes such as slow start and congestion window. Over time, researchers have expanded the TCP protocol and produced several versions such as TCP New Reno, TCP Vegas, TCP Fast [9], [10], [11], focused on congestion avoidance techniques to solve the packet loss problem.

The approach is different when moving to nanoscale networks based on tiny nodes, equipped with embedded computing devices interfacing with sensors/actuators. They are used in indoor and outdoor applications to monitor a physical or environmental event. Depending on application, the upstream traffic delivery can be [3]:

- *Event-based*: network load is light but becomes active in response to a detected event.
- *Continuous sensing*: some applications require continuous sending of sensing values, ex: nuclear monitoring.
- *Query-driven*: sink node invokes and queries sensing nodes to answer.
- *Hybrid*: bulk data is generated in addition to the constantly sensing data.

Congestion detection strategies are numerous [12]. The most used are *packet loss* (can be measured at the sender if ACK is used, also it can be measured at the receiver if sequence numbers are used), *queue length* (buffer usage in each node can serve as an indication of congestion, when the buffer exceeds a fix threshold then the congestion is signaled), *packet service time* (service time is used to continuously adjust the rate at which children send their packets), the *ratio between packet service and packet inter-arrival time* (scheduler between network layer and MAC layer will quantify the number of packet scheduled per time unit, this ratio indicates node level and link level congestion, *delay* (quantifies the necessary time starting the packet generation at the sender until its successfully reception at the next hop receiver).

For congestion control, many algorithms for wireless sensor networks are designed across transport layer and MAC layer (even the network layer) for efficient congestion detection and control.

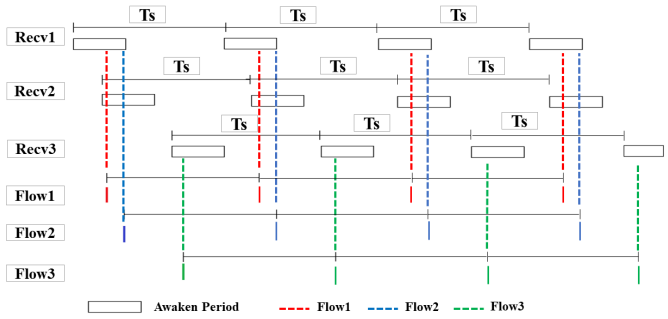

Fig. 5. Sleeping mechanism with three nodes and three flows.

Using sleeping mechanism in sensor network is also considered in the literature. Its goal is to preserve nodes resources (CPU, memory, energy), therefore extending network life time. S-MAC protocol [13] is one of many protocols that use sleeping mechanism. The nodes are awaken for a period of time and asleep in the remaining time. This scheme requires periodic synchronization among neighboring nodes to repair their clock drift. Sleeping time schedule must be created by each node and exchanged between neighbors.

III. Sleeping Mechanism

A common technique to preserve node resources (especially energy consumption) is to use duty cycling (sleeping) techniques, where nodes wake up from time to time to receive packets sent to them. Our proposed sleeping mechanism differs from those used in macro-scale network on two main aspects:

- Fine granularity: A nanonode does not stay awake for the duration of one or several packets, but for a much shorter duration, a fraction of the T_s value. By doing so, it is able to receive only one bit of a given packet at a time, and potentially a few bits that belong to other packets, due to packet interleaving.
- Asynchronism, decentralization: There is no agreement among nodes on awake periods. A node simply receives the bits that arrive when it is awake, whether they are intended for itself or not, and misses all the other bits.

In our proposed sleeping mechanism, all the nodes have the same awake-sleep cycle, equal to T_s. Inside the cycle, all the nodes have the same awake *duration* (or percentage of T_s), but the *beginning* of the awake interval is different for each node, and is randomly determined. For this to work, all the flows must have the same β.

This is illustrated in Fig. 5, where receiver nodes Recv1, Recv2 and Recv3 wake up at different times, but for the same duration. Recv1 and Recv2 are able to pick the bits from flows 1 and 2, as they arrive when they are awake. Recv3 is able to pick bits only from flow 3.

The mechanism ensures that if a node is able to pick the first bit of a packet, it will be able to pick all the followings. By adjusting the awake duration to the local density of the network, one can also statistically assure that a packet will be received by at least one or a few nodes (without knowing which ones in advance).

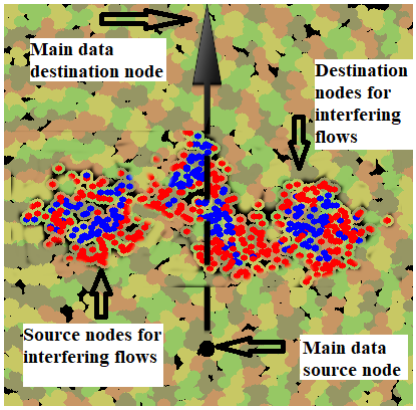

Fig. 6. VisualTracer output for the evaluated network.

During simulations, we vary the network density (changing the number of nanonodes in the network) to observe how packet transmission behaves. Intuitively, the denser the network, the more collisions and ignored packets we should observe. By using the sleep mechanism, we intend to reduce the load on each individual node and thus improve the performance of the network.

IV. Evaluation

This section evaluates the sleeping mechanism in improving network performance. As a detailed analytic study is not possible, we instead evaluate the protocol through simulations. Technical details and information about full reproducibility of our results are provided on a separate website[1].

A. Simulation platform

We use BitSimulator [14] to evaluate our proposed ideas. BitSimulator has been designed to allow simulation of applications and routing protocols while keeping a relatively detailed model for the MAC and physical levels. As such, it enables exploration and understanding of the effects of low level coding and channel access contention. It uses the TS-OOK modulation. It comes with a visualization program which displays graphically the simulation events, cf. Fig. 6.

B. Network scenarios

The simulation parameters are shown in Table I. The network is a 2D area. The main flow has the source at the bottom of the network, and the destination at the top. The source sends 100 unique packets to the destination. Several interfering flows (more precisely, 92, with specific parameters β = 1000, packet size = 1000 bits, interval time between packets is 0), if any, cross the network from left to right of the network, as shown in Fig. 6.

The communication range is chosen so that several hops are necessary to cross the network. Indeed, the mono-hop communication surface is a disc $\pi 0.35^2 \approx 0.38\,\mathrm{mm}^2$, so the

[1] http://eugen.dedu.free.fr/bitsimulator

TABLE I
SIMULATION PARAMETERS.

Size of simulated area	6 mm * 6 mm
Number of nodes	3000 to 15 000
Communication radius	350 μm
β (TS-OOK time spreading ratio)	1000
T_p	100 fs
Packet size	1000 bit

simulated area contains $6*6/0.38 = 94$ disjoint discs. Routing is done through the SLR protocol [4], an improved version of CORONA protocol. Fig. 6 shows a map where each colored zone corresponds to a different distance expressed in hop count from the SLR anchors. The node density is high, i.e. from $3000/94 \approx 32$ to $15000/94 \approx 160$ neighbors.

In the following analysis, all the nodes use the sleeping mechanism. We vary the *number* of randomly positioned nodes and the *number* of competing data flows that may interfere with the main flow. To avoid random effects, each point in the following figures is the average of 15 simulations using different RNG seeds, used for the position of all the nodes except source and destination ones, all the other parameters being kept identical.

C. Comparison metrics

The sleeping mechanism uses less node resources (CPU, memory, energy) but can prevent packets to arrive to the destination. Therefore, the metrics we use are the following:

- *Arrived packets*: the number of packets that have reached the destination node; only *unique* packets from the 100 original ones reaching the destination are counted, otherwise said multiples copies of the same packet are counted as one.
- *Main data flow forwards and receptions*: the number of times the packets of the main data flow are sent or received by a node in the network (at "MAC" level).
- *Forwards and receptions*: the total number of forwards and receptions, i.e. for the main flow plus the interfering flows; this metric is a good indication on the energy used.
- *Collided and ignored packets*: a collision occurs when packets arrive at the same time and hence prevent correct decoding by the receiving node. Ignored packets are discarded due to insufficient capacity to process all the incoming concurrent packets (buffer overflow or other physical limitations). Nodes are aware of collisions, but may be unaware of ignored packets.

D. Results without interfering flows

In SLR, nodes receiving a packet check if they are on the path from its source to its destination, and forward the packet if this is the case. But multiple nodes may share the same coordinates in SLR and they will all take the same decision to forward. In very dense networks, as many neighbouring nodes try to forward at the same time, this can cause collisions and ignored packets.

Fig. 7. Packets sent without interruption or sent at a fixed interval.

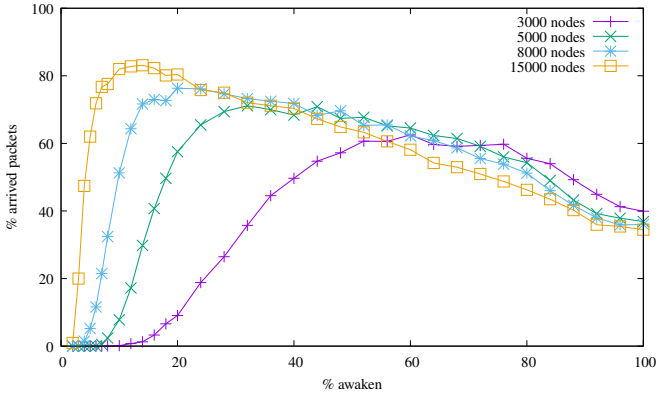

Fig. 8. Percentage of arrived packets depending on awake time and network density, for one flow and no interval between packets.

In this scenario, the source sends 100 packets (1) as fast as possible (one after the other), or (2) with an interval between packets equal to 5 times the duration of a packet (see Fig. 7). In such a multi-hop scenario, sending as fast as possible is quite demanding, as copies forwarded by neighbours tend to interfere with new incoming packets. Slowing the transmission speed even if data are available at the source is a good strategy, especially as nodes are not expected to send long bursts of data.

When inter-packet waiting period is null (Fig. 8), only around 40% of unique packets reach the destination for always awake nodes (100% on horizontal axis). The more the nodes sleep, the higher the reachability, up to the point where not enough nodes are awake and the transmission becomes impossible. Finally, denser networks benefit even more from the sleeping mechanism.

In case of an interval between packets, the intensity of the

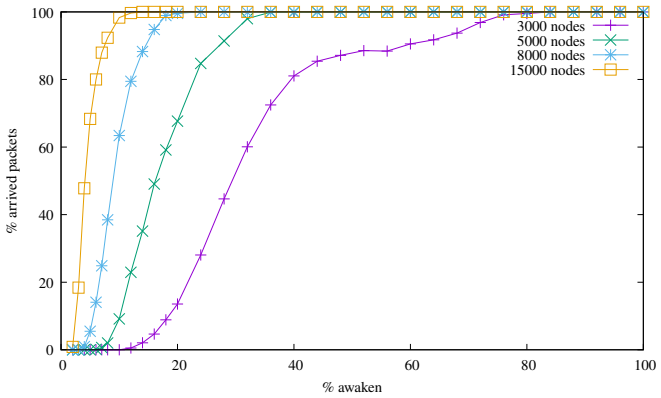

Fig. 9. Percentage of arrived packets depending on awake time and network density, for one flow and with interval between packets.

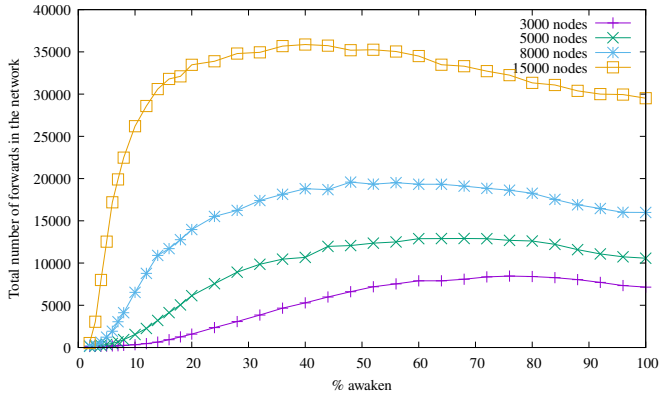

Fig. 10. Total number of forwards, for one flow and with interval between packets.

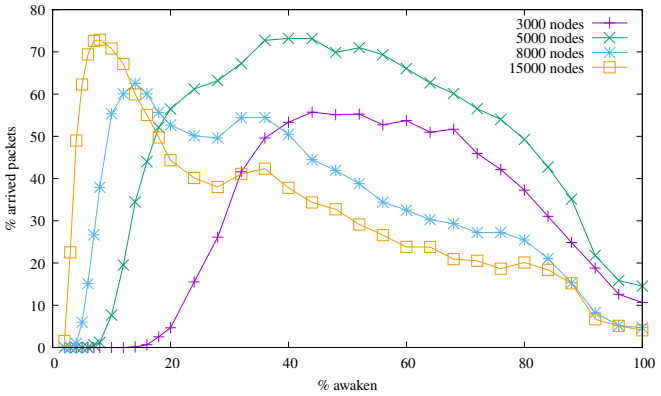

Fig. 12. Percentage of arrived packets depending on awake time and network density, for 20 concurrent flows and with interval between packets.

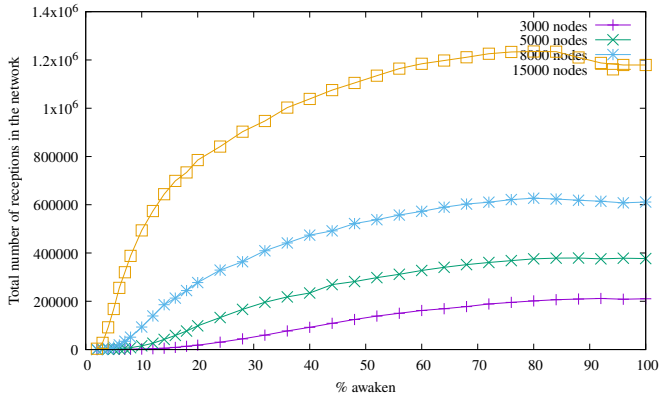

Fig. 11. Total number of receptions, for one flow and with interval between packets.

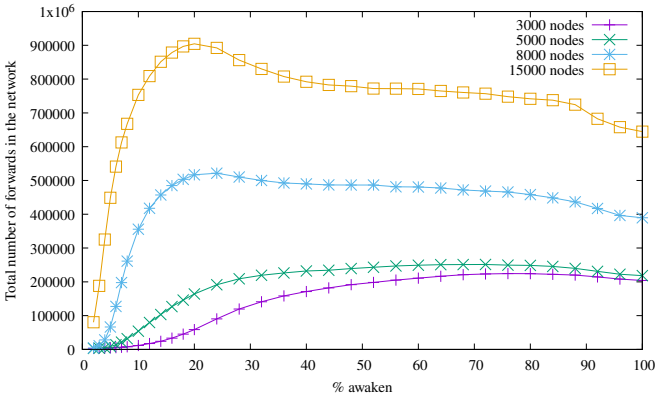

Fig. 13. Total number of forwards in the network, for 20 concurrent flows and with interval between packets.

data burst increases, and the reachability improves greatly. In Fig. 9, at 100% of time awake, *all* the packets are received. Also, decreasing the awake time does not immediately reduce the number of arrived packets, giving space for improving the network usage. This is especially true for very dense networks. For example in Fig. 9, in the densest network (15 000 nodes), 10% of awake time is sufficient to allow all the packets to arrive at the destination.

Fig. 10 and 11 highlight an unexpected behavior of our sleep mechanism. As expected, there is a collapse of forwards and receptions when the awake time is very low and the data packets cannot be forwarded anymore. But for larger percentages of awake time, curves are mostly stable. This is contrary to what Fig. 9 showed, i.e. the reachability improves with the decreasing of the awake time.

A stable or even an increase in transmission has a twofold explanation: First and foremost, the sleep mechanism dispatches the load of forwarding packets among neighboring nodes. As not all of them receive at the same time, they are not saturated (ignoring packets) at the same time. The same number of nodes becomes able to forward *more* data packets. This effectively increases the capacity of the network. This is an important result of this evaluation. As a second reason, we can also note that as packets are not lost (cf. Fig. 8 and 9), they can be repeated along the whole path. This effectively increases the number of times they are counted as forwarded packets.

E. Results with interfering flows

When adding interfering (background) flows in the network, we expect that packets from the main flow have a greater probability of collision, and a greater probability to get ignored by forwarding nodes given that their resources are used for other flows too.

However, Fig. 12 shows that the sleep mechanism can improve very significantly the reception rate, e.g. many more packets arrive for 50% than for 100% of awaken nodes.

This ability to improve the usable capacity of the channel is further illustrated in Fig. 13. It shows that adding more flows in the network translates into much more forwards (it can be compared to Fig. 10), while the reachability stays good.

F. Sleeping and destination node considerations

Sleeping improves network behavior by limiting the amount of traffic an individual node can see. Traffic is statistically dispatched over all nodes, thus sharing the load. This mechanism

is especially effective while routing packets. But provisions have to be made to ensure the destination node is not sleeping and missing some data packets.

A first way to deal with this problem is to ensure the destination node is not sleeping at all. This is the case for networks where the destination node is in a relatively calm area. As not much competing traffic comes near the destination, it does not saturate and can stay awake all the time.

In case of a destination being in a very loaded area, a better strategy is to not target a specific node, but instead "any node providing the required service" in the destination zone. This way, it becomes possible to have multiple nodes implementing the service and sharing the same SLR coordinates. Those nodes will sleep asynchronously, with an awake time percentage depending on how many they are. At any time, some of them will be awake and able to pick any incoming packet.

Another consideration is that usually the best awake percentage is the one that has the lowest number of forwards, receptions, ignored packets, while maintaining a good probability of packets arrived at the destination.

However, there are applications which do not need a 100% reliability (reachability to the destination). For example, in agriculture applications, nanonodes can be used to measure plant life indicators (such as humidity, soil moisture, sun light) and transmit this information to a central server for processing. Whereas such applications need a high network life time, the reliability of arrived packets is not critical, since any lost information will be resent anyway. Our sleeping mechanism helps such applications by improving the network life time using a small awaken period. As for now, experimental validation is not yet possible, because nanonodes are not yet manufactured due to technical constraints.

V. Conclusion and Future Work

In this paper, we discussed the integration of the proposed *sleeping* mechanism into the SLR routing protocol and in the context of very dense nanonetworks. Evaluations were conducted by using a detailed simulation tool. Evaluations first illustrated a very peculiar behavior of nanowireless networks. While the capacity of the THz channel is very high, it may not be directly available. Nanonodes are not able to individually process the full throughput of this channel. Especially in dense to very dense environments, when exposed to broadcasting schemes, all nodes in an area tend to reach saturation and start missing (ignoring) new incoming packets. By allowing an asynchronous sleep mechanism, we are able to improve significantly the amount of data that can be successfully transferred in the network in a given amount of time (we increase the usable capacity). Moreover, as not all nodes participate in all local transmissions anymore (remember that they individually *receive* much less packets), the load is dispatched. As individual nodes see less activity, they also use less resources (energy, CPU, memory, ...) Evaluations clearly show that the percentage of time awake needs tuning by considering the local network density and even the number of locally competing data flows. We have shown that an optimal value can be found, that optimizes the available capacity. The number of packets successfully transmitted increases while the number of ignored packets and collisions does not change much or even decreases. Each application/deployment can benefit very significantly from this tuning. Future work includes the integration of sleeping mechanism in the backoff flooding, by taking into account the flows interference. It also includes an automatic tuning of the awaken duration based on the neighborhood density, using the DEDeN density estimation protocol.

Acknowledgments

Ali Medlej has a grant from Islamic Center Association for Guidance and Higher Education. This work has also been funded by Pays de Montbéliard Agglomération.

References

[1] X.-W. Yao, W. Huang *et al.*, "Routing techniques in wireless nanonetworks: A survey," *Nano Communication Networks*, p. 100250, 2019.

[2] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *1st international conference on Embedded networked sensor systems*, 2003, pp. 1–13.

[3] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion control protocols in wireless sensor networks: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 3, pp. 1369–1390, 2014.

[4] A. Tsioliaridou, C. Liaskos, E. Dedu, and S. Ioannidis, "Packet routing in 3D nanonetworks: A lightweight, linear-path scheme," *Nano Communication Networks*, vol. 12, pp. 63–71, Jun. 2017.

[5] J. M. Jornet and I. F. Akyildiz, "Femtosecond-long pulse-based modulation for terahertz band communication in nanonetworks," *IEEE Transactions on Communications*, vol. 62, no. 5, pp. 1742–1753, May 2014.

[6] J. F. Kurose, *Computer networking: A top-down approach featuring the internet, 3/E*. Pearson Education India, 2005.

[7] J. M. Jornet, J. C. Pujol, and J. S. Pareta, "PHLAME: A physical layer aware MAC protocol for electromagnetic nanonetworks in the terahertz band," *Nano Communication Networks*, vol. 3, no. 1, pp. 74–81, 2012.

[8] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP congestion control with a misbehaving receiver," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 71–78, 1999.

[9] N. Parvez, A. Mahanti, and C. Williamson, "An analytic throughput model for TCP NewReno," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 448–461, 2009.

[10] L. A. Grieco and S. Mascolo, "Performance evaluation and comparison of westwood+, New Reno, and Vegas TCP congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 25–38, 2004.

[11] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *IEEE INFOCOM 2004*, vol. 4. IEEE, 2004, pp. 2490–2501.

[12] C. Sergiou, P. Antoniou, and V. Vassiliou, "A comprehensive survey of congestion control protocols in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1839–1859, 2014.

[13] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *21st Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3. IEEE, 2002, pp. 1567–1576.

[14] D. Dhoutaut, T. Arrabal, and E. Dedu, "BitSimulator, an electromagnetic nanonetworks simulator," in *5th ACM/IEEE International Conference on Nanoscale Computing and Communication (NanoCom)*. Reykjavik, Iceland: ACM/IEEE, Sep. 2018, pp. 1–6.