A Comparative Performance Evaluation of MAC Protocols for Underwater Sensor Networks

Chiara Petrioli and Roberto Petroccia
Dipartimento di Informatica
Università di Roma "La Sapienza"
Roma, Italy
E-mail: {petrioli,petroccia}@di.uniroma1.it

Milica Stojanovic
Massachusetts Institute of Technology
Boston, MA, U.S.A.
Email: millitsa@mit.edu

***Abstract*—A propagation-delay-aware MAC protocol, based on carrier sensing multiple access, is proposed. The design aims at maximizing the bandwidth utilization by keeping track of neighboring transmissions to avoid collisions, thus enabling interleaved packet transmission between different pairs of users. The performance is compared to several representative MAC protocols: the standard and slotted ALOHA, and three protocols designed specifically for the underwater acoustic environment, APCAP [1], DACAP [2] and T-Lohi [3]. Simulation results identify network settings (traffic load, node density, single/multi-hop topologies) in which each protocol offers the best performance.**

I. Introduction

Underwater sensor networks (UWSNs) have become an important area of research with potential practical impact on a host of different applications, ranging from monitoring and discovery of the marine environment, remote control of submarine oil extraction, underwater safe CO_2 storage, etc. Low cost, medium and large scale monitoring systems are becoming possible through the deployment of underwater wireless sensor nodes equipped with acoustic modems. Given the challenges posed by the very specific environment, solutions from terrestrial wireless networks cannot always be applied to UWSNs, and new protocol stacks are required for both single-hop and multi-hop communications. Specifically, the acoustic channel is characterized by high propagation delays, low bandwidth, and frequency-dependent attenuation, which affect protocol design on all layers of a network architecture.

The focus of our work is on medium access control (MAC). Over the recent years, a number of MAC solutions that specifically address random-access underwater acoustic networks have been proposed [1]-[3].

Among the solutions based on carrier-sensing (CSMA), Slotted Floor Acquisition Multiple Access (S-FAMA) proposed in [4] combines carrier sensing with a dialog between the source and receiver prior to data transmission. Although time slotting eliminates the need for excessively long control packets thus reducing the overall energy consumption, due to the high propagation delay, the handshaking mechanism may still introduce long delays leading to low system goodput.

A different approach to channel access was proposed in [5]. This solution is strictly tied to a sleeping schedule, which is optimized for minimal energy consumption and does not consider bandwidth utilization or access delay as objectives.

The Distance-Aware Collision Avoidance Protocol (DACAP) [2], like S-FAMA, combines carrier sensing and an exchange of request-to-send / clear-to-send (RTS/CTS) control packets prior to data transmission, but it does not require the nodes to be synchronized to common time slots. Nodes obtain the distance information using the control packet round-trip time, and use this information to improve the channel utilization by reducing the number of collisions. In [1], the fact that different links exhibit different delays is exploited in the design of the so-called adaptive propagation-delay-tolerant collision avoidance protocol (APCAP). Like DACAP, APCAP uses an RTS/CTS scheme to reserve channel.

Another way to improve the overall system utilization is to reduce the overhead, i.e. the amount of control information sent. This is the idea of the Tone-Lohi (T-Lohi) protocol [3] where nodes send short packets (tone packets) to notify the neighbors before sending data. Employing tone packets the reservation is made in a rapid and energy-efficient way. Each node sends its own tone, listens the channel before transmitting data packets, counts how many other nodes do the same, based on the number of tone packets received, and, if necessary, delays its transmission according to it.

In this paper we propose a new CSMA-based MAC solution, termed Propagation Delay Aware Protocol (PDAP). The protocol aims at maximizing the bandwidth utilization to enable interleaved, yet reliable communications between different pairs of nodes. To improve the channel utilization, PDAP keeps track of the neighboring transmission to avoid collisions and retransmissions.

The performance of PDAP is compared via simulation to the performance of representative MAC protocols: ALOHA [6] with and without acknowledgment, slotted ALOHA [7] with and without acknowledgment, APCAP [1], DACAP [2] with and without acknowledgment and T-Lohi [3]. The objective in doing so is to identify the protocols that are particularly suited for a particular type of scenario specified by a single-hop or multi-hop topology with a given node density, data rate, traffic load, maximum propagation delay, etc. Our comparison concerns key metrics such as the percentage of data sent and correctly received at destination, goodput and packet latency. The results show that three protocols, PDAP, DACAP (with no ACKs) and ALOHA (with ACKs), in this order,

978-1-4244-2620-1/08/$25.00 ©2008 IEEE

always perform significantly better than all the other protocols in a single-hop scenario. In multi-hop scenario lightweight protocols such as ALOHA and slotted ALOHA with ACKs are able to scale to higher loads and achieve better performance. The paper is organized as follows. Section II describes the details of the protocols that we have investigated. The performance evaluation is investigated in Section III. Finally, Section IV concludes the paper.

II. Description of the protocols

The following definitions are used in the description of the protocols considered in this paper. With **txRetry** we indicated the current number of retransmissions of a packet. **dataTime** is the time that a node spends transmitting a data packet. **rtsTime** is the time a node spends transmitting an RTS control packet. **ackTime** indicates the time a node spends transmitting an ACK control packet (which is 3 Bytes long). Finally, **maxDelay** is the maximum propagation delay of a packet in the network, computed based on to maximum transmission range and on the speed of sound underwater.

1. ALOHA is the well-known protocol for channel access [6]. We here consider carrier-sensing ALOHA. When a node has a data packet to transmit, it first checks whether the channel is idle or busy. In the first case, it starts the packet transmission. If the channel is busy, the node delays the transmission according to the ALOHA exponential backoff mechanism. We consider two versions of this protocol. The first is the following: A node that transmits a data packet receives no feedback about whether the intended recipient has received it or not. The second adds robustness by having the destination node acknowledging the data reception to its source. If the ACK is not received within a given time (set to 2Delay + ackTime), the data packet is retransmitted either till successful reception, every time choosing the backoff time in an interval twice as long as the previous one, or till the maximum limit of retries has been reached. Here Delay is the transmission delay between source and destination. Its value is initially set to maxDelay and successively set by the nodes to a value computed according to the (estimated) distance between source and destination (which is based on the time difference between data packet transmission and ACK reception). The backoff time is chosen randomly and uniformly in $[0, T]$, where $T = 2^{\text{txRetry}}(2\text{maxDelay} + \text{dataTime})$.

2. slotted ALOHA. We consider the version of slotted ALOHA described in [7], adding the feature of carrier sensing. Time is divided into slots that start at the same time for all nodes. The duration of a slot depends on a packet maximum propagation delay and on the data transmission time. As in [7] we set the slot duration σ to βmaxDelay + dataTime, where β is the fraction of the maximum propagation delay that a node waits after transmitting the packet. Choosing $\beta = 1$ ensures that no collision occurs at the receiver (unless data packets are transmitted in the same slot and they overlap at the receiver). However, this has a negative impact on the goodput because of the long propagation delays. In [7] the authors show that setting $\beta = 0.5$ allows to improve goodput performance. When a node has a data packet to send and a new slot is starting, the node checks whether the channel is idle or busy. The packet is transmitted in the first case. In the second, instead, the nodes use the exponential backoff mechanism of ALOHA protocols, now counted in number of slots. Differently from [7] we also consider a version of slotted ALOHA with acknowledgments. In this case, the slot duration σ is set to 2βmaxDelay + dataTime + ackTime. If after transmitting a data packet a node does not receive an ACK it tries again after a number of slots as dictated by the exponential backoff mechanism. More precisely, the number of slots that a node waits before trying to retransmit a packet is chosen randomly and uniformly in $[0, T]$, where $T = \sigma 2^{\text{txRetry}}$.

3. APCAP, The Adaptive Propagation-Delay Collision Avoidance Protocol (APCAP), described in [1], uses the RTS/CTS scheme to reserve channel and send data. All nodes are synchronized. When a node has a data packet to transmit, it sends an RTS packet (13Bytes long). Differently from the classic RTS/CTS scheme, the sender does not need the receiver to send a CTS back immediately. Instead, it sets a CTS_window indicating when it is available for receiving the CTS packet. This allows the destination to negotiate with the sender the time when it can transmit the CTS. Similarly, the sender also sets a DATA_window indicating the time it is available to transmit the data packet. The actual transmission time is negotiated with the destination. The use of these windows reduces the chance of the destination failing to transmit the CTS when the source requires it because it is unavailable, and also the likelihood of the destination not receiving the data packet because it is not ready for it. More precisely, the APCAP protocol works as follows. When a node wants to transmit a data packets it checks its own schedule and finds when it is free to send the RTS. It also set its CTS_window and DATA_window. The window values are sent in the RTS. If the sender receives the 7Bytes CTS in the time it is waiting for it, it updates its schedule and transmit the packet at the negotiated time carried in the CTS. Otherwise, it clears the CTS_window and DATA_window for that data packet, and retransmits the RTS after a while. Upon receiving an RTS if the destination is free to send a CTS within the time frame set by the sender, it does so and indicates a good time for data packet reception. The choice of the time complies with both the CTS_windows and DATA_window as set by the sender, and takes into account the distance from source and destination (computed via time stamps in the RTS and CTS control packets). If the destination cannot send a CTS when the source can receive it, it just ignores the RTS. If the source does not receive the CTS, it schedules to resend it as soon as it has free time. According the RTS/CTS mechanism for channel access, when a node receives an RTS and is not the intended destination of that control packet, it sets its Network Allocation Vector (NAV) so that it does not transmit a control packet that will arrive at the sender during the CTS_window. It will also refrain from transmitting a control packet that could reach the sender during its DATA_window. When a potential interferer receives a CTS, it sets its NAV according to the final time

of the data transmission indicated by the CTS. Consequently, it will not transmit a control packet that would arrive at the destination during the reception of the packet from the source. The APCAP mechanism adopts an aggressive approach to channel access, in the sense that a node that has scheduled a given (control or data) transmission does not delay or cancel it if it is made aware of other communications going on in the channel. This is because of the typically long propagation delays in an underwater environment: Chances are that when a node that has scheduled a transmission senses that another transmission is ongoing, that transmission could be finished already and therefore there is no reason to delay its own sending.

4. DACAP stands for Distance Aware Collision Avoidance Protocol. This protocol, defined in [2], uses the RTS/CTS scheme to reserve the channel and for transmitting data packets. When a node has a data packet to send it transmits an RTS (6Bytes). Upon receiving it, the destination replies right away with a CTS (6Bytes). It then waits for the data packet. If during this time it hears a control packet for some other node, it sends a very short (3Bytes) WARNING packet to its sender. Upon receiving a CTS, a sender waits some time before transmitting the data packet (WARNING time). If it hears another control packet or receives a WARNING from the destination during this time, the node aborts the packet transmission. The length of the WARNING time depends on the distance between the source and destination. The sender can compute this distance by measuring the RTS/CTS round-trip time. Potential interferes are blocked as usual in RTS/CTS schemes.

5. T-Lohi. T-Lohi is a protocol for single-hop underwater networks defined in [3]. When a node has a data packet to transmit, before the actual transmission it starts a reservation period (RP). An RP is made up of a certain number of slots called contention rounds (CRs). During a CR, the sender transmits a short 3Bytes control packet (tone packet) to inform other nodes about its need to access the channel. It then listens to the channel to detect if other nodes also have data packets to send. Each node contending for the channel counts how many other nodes do the same based on the number of tone packets received during the CR. If no other tone is heard during the CR, the node seizes the channel. Its RP is over and it transmits the data packet. If contention occurs, the contenders back off for a number of CR chosen randomly and uniformly between 0 and the number of competitors. A node RP continues until successful channel access. We notice that nodes are not synchronized: Each node that has data packets to send starts its own RP for channel access and transmission, independently of other nodes. The duration of a CR is appropriately set so that a node has enough time to detect as many contenders as possible. Of the many flavors of T-Lohi described in [3] we consider the most aggressive, i.e., the one that maximizes the goodput. In the aggressive T-Lohi the CR lasts for the time needed to transmit a tone packet plus the maximum propagation delay.

6. PDAP. The Propagation Delay Aware Protocol (PDAP) uses the RTS/CTS mechanism for channel reservation and transmission. All nodes are synchronized. Similarly to APCAP, when a node has a data packet to transmit, it checks its own schedule to find free times for the whole communication exchange; namely, for sending the RTS, receiving the CTS, and sending the data packet. The time for receiving the CTS must start after the time needed for the RTS to reach the destination plus the time for the CTS to get back. Similarly, the time for sending the data packet is scheduled immediately after the CTS time reception. Before sending the RTS, the sender waits for a random time to avoid synchronization with other potential senders. This waiting time is randomly and uniformly chosen in $[0, T]$, where $T = 2^{\text{txRetry}}(2\text{maxDelay} + \text{rtsTime})$. When it is time to send the RTS a node checks the channel. If it is idle, the node sends the RTS and waits for the CTS. Otherwise, the sender clears its schedule (RTS, CTS and data packet times), increases its retry counter and selects new times for sending the RTS, receiving the CTS and for sending data. Both RTS and CTS control packets contain information about the distance between the source and destination. This distance is computed based on the channel propagation delay and the time stamps on the control packets. Therefore, this information is updated every time nodes communicate. The RTS is 15Bytes long and the CTS is 9Bytes. When a potential interferer receives a control packet from a neighboring node, in case it has scheduled a transmission that could collide with ongoing communication it updates its own schedule and delays its transmission. Reception of RTS and CTS by a possible interferer is dealt with as follows. When the interferer receives an RTS packet, it sets its NAV so that it will not be transmitting control information that would arrive at the sender at the scheduled CTS reception time. It also will not transmit a control packet that would arrive at the destination while the destination is sending the CTS. Similarly, the interferer will not transmit control information that would arrive at the sender during the transmission of the data packet and at the destination while it is receiving the data. When the interfering node receives the CTS, it updates its NAV according to the reception time carried on the CTS. As a consequence, the interferer will not transmit control information that would arrive at the destination during the reception of the data packet, and at the source during the data transmission. Differently from APCAP, PDAP does not have an aggressive policy of channel access. Although trying to schedule the highest possible number of parallel transmissions, nodes always attempt to avoid collisions and that senders are synchronized when transmitting packets.

III. Performance Evaluation

In this section we present the results of the comparative performance evaluation of the protocols described in Section II, namely, ALOHA, slotted ALOHA (both with and without ACKs), APCAP, DACAP (with and without ACKs), T-Lohi and PDAP. In order to assess the performance of the selected protocols in a realistic underwater setting we have extended the network simulator ns2 to include key characteristics of

the submarine environment such as 3D nodal deployment and a physical model for underwater acoustic communications [8] that also takes into account packet collisions and interferences.

A. Simulation scenarios and settings

We consider a shallow water scenario where N underwater static sensor nodes are placed on the sea bottom at a depth of 200m. The nodes are randomly and uniformly scattered on the lower face of a cuboid, which is a square of side L. (The cuboid height is 200m). Packets are transmitted from the nodes to a sink, which is centrally located on the upper face of the cuboid, at sea level.

Our experiments concerns the following two scenarios. **1. Single-hop**: All nodes can communicate to each other directly. **2. Multi-hop**: Communications from a source to the sink may go through a multi-hop path. The path is determined by a shortest path routing protocol. In this case each node also acts as a relay for packets generated by other nodes.

In both scenarios, nodes are equipped with an acoustic modem with a transmission range of 1000m. In the single-hop scenario, N takes values in the set $6, 11, 16$, while L has been set to 700m. This corresponds to a nodal degree of 5, 10 and 15 neighbors, allowing testing of the protocols in sparse and dense networks. In the multi-hop setting, N takes value $35, 65, 100$, which achieves average densities similar to the single-hop case (nodal degrees of 5, 10 and 15), while L has been set to 4000m. The average number of hops traversed varies in the multi-hop cases between 2.3 and 2.8 hops, depending on N.

Traffic is generated according to a Poisson process with rate λ packets per seconds. Once a packet is generated it is associated to a source selected randomly among all nodes. All packets are addressed to the sink. λ has been varied between 0.033 and 2 to create scenarios with low, medium and high traffic. The size of the data packet payload is set to 300B. The total size of the data packet is given by the payload plus the headers of the different layers (physical through network). The physical layer header contains all the information needed by the acoustic modems to correctly receive packets on the channel (preamble, delimiters, etc.). This amounts to 60Bytes.

The MAC header contains the sender and destination IDs and the packet type. Its length has been set to 3Bytes.

To correctly receive each packet the SIR at the receiver must be $\geq$ 15dB. Each node has a buffer of 50 packets, where data coming from the upper layers are stored before transmission. Whenever the buffer is full and a new packet arrives, the oldest packet is dropped from the buffer.

We have considered acoustic modems with three different data rates, namely, 2000bps, 8000bps and 28000bps. (The data rate is the same for all nodes). Given the transmission range, when the data rate is 2000bps the transmission delay is twice the maximum propagation delay. When the data rate is 8000bps the transmission delay is half the maximum propagation delay. The data rate of 28000bps is the highest possible achievable by an acoustic modem using a 3dB bandwidth with a transmission range of 1000m and a bandwidth efficiency of 1b/Hz [8]. In the latter case the transmission delay is $1/6$ of the maximum propagation delay. The maximum number of packet retransmission is set to 7. Every point in the figures has been obtained by averaging over the number of experiments needed to achieve a statistical confidence of 95% with a 5% precision.

B. Metrics of interest

We use the following notation:

- $Pack_g$ is the number of data packets generated in the simulation time.
- $Pack_{tr}$ is the number of data packets actually transmitted.
- $Pack_{ds}$ is the number of data packets discarded by sources.
- $Pack_r$ is the number of data packets received at the sink.

The following metrics have been investigated.

(a) Percentage of data packets sent (P_{sent}). P_{sent} is defined as the ratio between the number of data packets injected in the network and the total number of packets generated by source nodes ($100 - P_{sent}$ is the percentage of packets generated by sources that are automatically discarded due to buffer overflow). $P_{sent} = (Pack_{tr}/Pack_g) * 100$.

The **(b) Percentage of packets received (P_{rec})** is the ratio between the number of data packets correctly received at the destination and the total number of generated packets $P_{rec} = (Pack_r/Pack_g) * 100$.

The **(c) Percentage of lost packets (P_{lost})** indicates the ratio between the number of data packets sent that never made it to the destination (due to too many retransmissions, collisions or packet discarding due to buffer overflow) and the total number of generated packets: $P_{lost} = ((Pack_{tr} - Pack_r)/Pack_g) * 100$.

(d) End-to-end latency. This metric is defined as the average time between the packet generation time and the time of its correct delivery at the sink. (It is computed for the set of packets correctly delivered.)

We define the **(e) Goodput** as the percentage of time when the underwater channel is busy sending data packets.

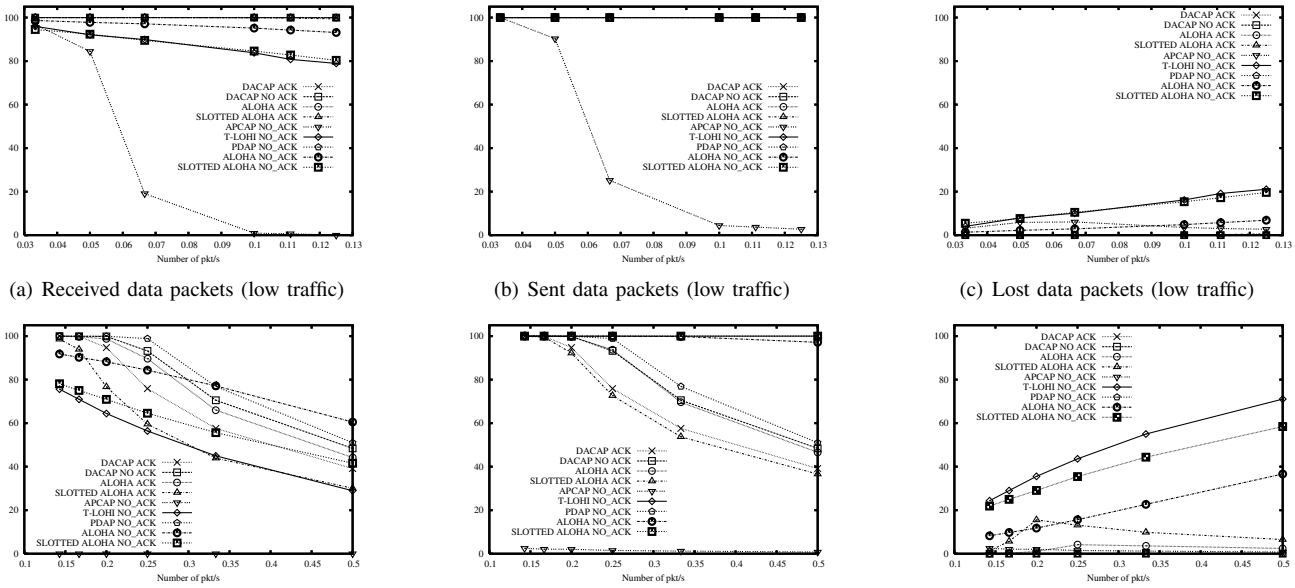
C. Experiments

We present our simulation results starting from the single-hop scenario. The multi-hop scenario follows.

1) **Single-hop networks:** We start by discussing the lower data rate case (2000bps).

We present here results concerning the highest density scenario (network with 16 nodes). The other nodal densities show similar trends.

Fig. 1 shows the percentage of data packets received (figures 1(a) and 1(d)), sent (figures 1(b) and 1(e)), and lost (figures 1(c) and 1(f)) for different traffic loads. At low/medium traffic (top three figures) the best performing protocols are PDAP, DACAP and the two versions of ALOHA with ACKs. For these protocols the packet delivery ratio is 100%. Protocols without a reliable mechanism for channel access and with no ACKs suffer instead performance degradation. This is clear from the performance of ALOHA and slotted ALOHA, both



(a) Received data packets (low traffic)

(b) Sent data packets (low traffic)

(c) Lost data packets (low traffic)

(d) Received data packets (high traffic)

(e) Sent data packets (high traffic)

(f) Lost data packets (high traffic)

Figure 1. Single-hop networks: Received, sent and lost data packets (2000bps)

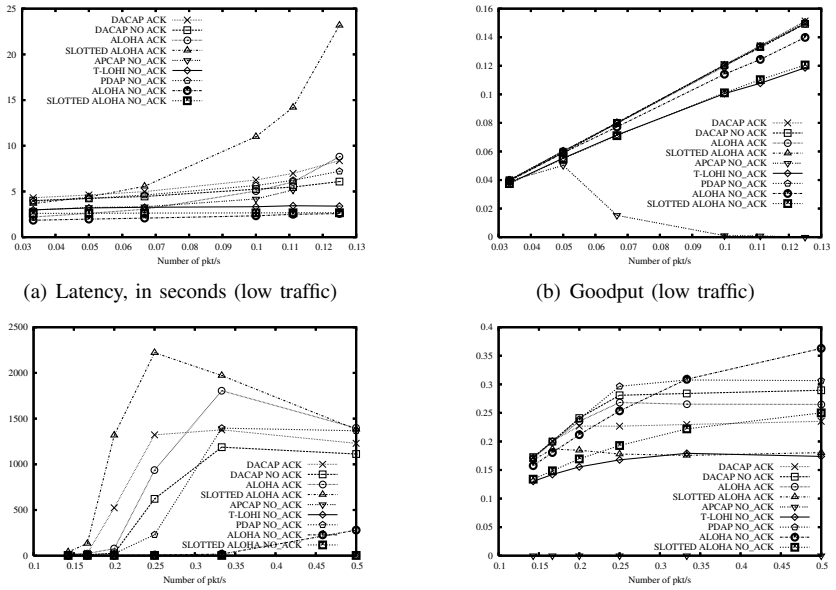
without ACKs. When $\lambda = 0.13$ slotted ALOHA with no ACKs is able to correctly deliver only 80% of the generated data packets, while the unslotted version obtains a 93% packet delivery ratio. APCAP and T-Lohi also perform poorly. Despite its capability of enabling multiple simultaneous transmission and the freedom given to sender and destination to negotiate transmission times, we observe that APCAP performs poorly due to its aggressive approach to channel access which is not suitable in scenarios where data transmissions are much longer than the propagation delay. In this case a high number of control packet collisions occur, and the actual goodput is detrimentally affected. Furthermore, in scenarios where all nodes can communicate to each other, all nodes can hear the same communications. As a consequence, when there is a collision and both senders do not receive the expected CTS packet they try to resend the RTS according to their schedule. This implies choosing to retransmit as soon as possible, which is highly likely to produce another transmission overlap. In other words, the exponential backoff mechanism of the ALOHA schemes proves to be crucial for better performance in this setting. Not having it, as in APCAP, creates a very high number of control packet collisions and retransmissions. APCAP nodal buffers therefore start to overflow soon, resulting in a quickly increasing number of packets that cannot be sent even at moderate traffic. This justified the observed delivery ratio of APCAP that falls to zero for values of λ higher than 0.1.

The reasons for T-Lohi bad performance are different. The usage of a tone packet is effective if the packet is really short. However, when we add the needed headers to the short payload, the tone packet size grows, resulting in long transmission delays at low data rates. In fact, we observe a remarkable number of collisions of tone packets, which makes the correct count of the contenders problematic. It therefore may happen that a node considers the channel free and starts transmitting data even if there are multiple transmission attempts. This explains why 20% of the generated packets are not correctly delivered to the sink. when $\lambda = 0.13$ (Fig. 1(c)).

Things change when the traffic increases (three bottom figures of Fig. 1). In this case, PDAP, DACAP without ACKs and ALOHA with ACKs (in this order) are still the three protocols with the best performance. PDAP delivers all generated packets for $\lambda \leq 0.25$, and more than 90% of them for $\lambda \leq 0.27$. These values decrease to 0.2 and around 0.25 for DACAP without ACKs and ALOHA with ACKs (with the former performing slightly better than the latter). DACAP with ACKs performs the worst, being able to deliver all packets only for $\lambda \leq 0.17$. The reason is that it generates a higher amount of control traffic than the other three protocols. The performance of the remaining protocols degrades significantly, reaching unacceptable values of lost packets when $\lambda > 0.13$. Except for ALOHA and slotted ALOHA with no ACKs and T-Lohi, the large majority of packets that are not delivered to the sink are discarded directly at the source, i.e., even before any attempt to transmit them.

Fig. 2 shows the end-to-end packet latency and the network goodput performance.

At low traffic all protocols apart from slotted ALOHA with ACKs are able to deliver packets to the sink within 10s. As expected, protocols with ACKs or those that require the exchange of control packets for reliable channel acquisition experience a slightly higher latency. The reason for the worse delay performance of slotted ALOHA can be understood recalling that if the transmission delay is twice the maximum propagation delay, no advantage comes from the different propagation delays between nodes: Every time two nodes select the same slot to send data to the sink their messages



(a) Latency, in seconds (low traffic)

(b) Goodput (low traffic)

(c) Latency, in seconds (high traffic)

(d) Goodput (high traffic)

Figure 2. Single-hop networks: Latency and goodput (2000bps)

will collide. Increasing the number of packets considerably increases the number of collisions and retransmissions (and therefore the delay).

The protocols that perform best in terms of packet delivery ratio and that scale to high loads (PDAP, DACAP without ACKs and ALOHA with ACKs) deliver packets with low latencies till they can deliver all generated data, i.e., until there is no buffer overflow. After that, the high number of collisions significantly increases latency. This phenomenon is particularly evident in case of ALOHA with NO ACKs, as it does not implement a reliable scheme to acquire the channel, resulting in a higher number of collisions. When the traffic increases above $\lambda = 0.2$, latency first significantly increases and then it starts decreasing. The lower latency at very high traffic load depends on the queuing policy: When nodes are overloaded the oldest packets are discarded, and therefore newest packets that have been sitting in the queue for a shorter time are transmitted.

Fig. 2(b) and Fig. 2(d) confirm what we have already seen in Fig. 1 for the percentage of data packets correctly received. Protocols using a reliable mechanism for channel access and that use a reduced number of control packets experience less collisions and achieve high goodput. However, Fig. 2(b) and Fig. 2(d) also show the limits of the goodput and throughput metrics. Protocols such as ALOHA and slotted ALOHA with no ACKs suffer severe performance degradation in terms of packet delivery ratio but show high goodput performance. It is therefore important to combine the goodput metric with end-to-end performance metrics in order to have a correct understanding of the different protocol behaviors.

We now focus on the 28000bps data rate scenario. Since the packet transmissions are shorter (the data packet transmission delay is now six time lower than the maximum propagation delay), collisions decrease and the number of packets correctly delivered increases. This is clearly shown in Fig 3(a): All protocols are able to correctly deliver almost all packets for $\lambda \leq 0.17$. When comparing the different protocols performance for varying traffic loads ALOHA and slotted ALOHA without ACKs experience the worst performance (as before and for the same reasons), falling below a 90% delivery ratio for moderately high traffic loads ($\lambda = 0.2$ and 0.3). T-Lohi performs better, being able to deliver almost all generated packets for values of $\lambda \leq 0.35$. When the traffic further increases, however, T-Lohi shows a fast degradation of the packet delivery ratio due to an increasing number of collisions of tone packets. The two best performing protocols are still PDAP and ALOHA with ACKs, which are able to correctly deliver all generated packets even for very high loads. The long propagation delays combined with the short transmission delays allow PDAP to schedule several parallel communications at a time. In a high data rate scenario the probability of collisions decreases and a lightweight protocol such as ALOHA with ACKs makes the best use of it. The difference with respect to the low data rate case is that DACAP falls behind. DACAP with no ACKs is still the third best performing protocol (being able to deliver all generated packets for $\lambda \leq 0.5$). However, PDAP and ALOHA with ACKs are able to provide the same level of service with twice the traffic load! The reason is to be found in the DACAP operations, which do not allow interleaved transmissions and force nodes to delay transmissions for long times (such delays depend on the propagation delay) whenever they detect ongoing communications. The use of ACKs adds to the control information exchange and to the delays in handling new traffic, justifying the fast performance degradation of DACAP with ACKs with respect to the no ACKs case. The protocol that

significantly improves its performance with respect to the low data rate scenario is APCAP. When the ratio between the transmission and propagation delays decreases the aggressive mechanism used to access the channel pays off, allowing APCAP to deliver over 90% of the packets even at high traffic loads ($\lambda = 0.5$).

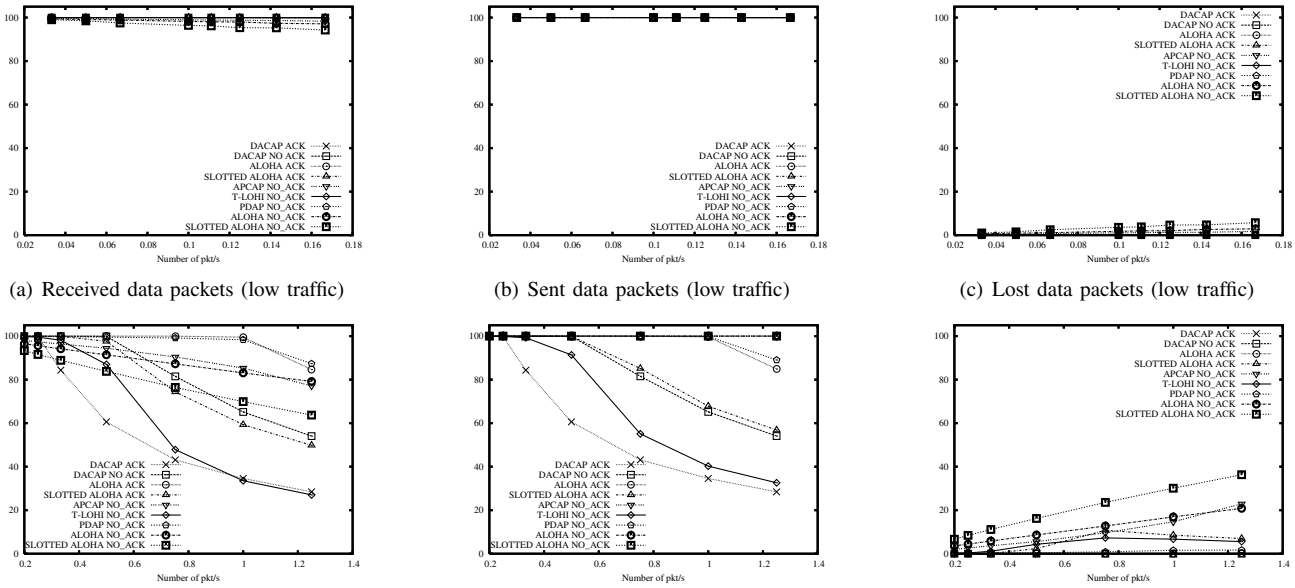
Fig. 4 shows packet latency and network goodput for the high data rate scenario. Fig. 4(a) shows that all protocols deliver packets within a few tens seconds unless congestion builds up (and a high percentage of packets are discarded). At low traffic either exchanging minimum amount of control information or adopting aggressive mechanisms to access the channel pays off in terms of latency: These protocols experience end-to-end latencies below 1s. When the traffic load increases the best performing protocols are those able to scale with traffic. In particular, out of the two best performing protocols, PDAP experiences a 30% to 50% decrease with respect to the latency of ALOHA with ACKs at high traffic. The improved latency and packet delivery ratio performance come at the price of a lower channel utilization (Fig. 4(b) and Fig. 4(d)). Despite all protocols scale better with traffic, the observed increase is lower than the increase in the channel data rate: Long propagation delays (with respect to transmission delays) translate into inefficiencies in the way the channel is acquired and in the way it is used to transmit packets.

2) **Multi-hop networks***:* In general, multi-hop communications enable a certain degree of parallelism for packet transmission. In the underwater setting, however, the degree of parallelism is more limited than for terrestrial communication. This depends on the very nature of the underwater channel. With a maximum transmission range of 1000m and a SINR of 15dB we can estimate the probability that two transmissions overlapping in time interfere at one of the destinations [8]. Let us consider the case where there are two transmitting nodes A and C with destinations B and D, respectively. Let us also assume that while node B is receiving a packet from node A, it is reached by the signal transmitted by C. If A and B are 50m from each other the overlapping of the two signals results in faulty reception of the packet from A if the distance between C and B is lower than 376m. If nodes C and B are farther than 376m the packet will instead be correctly received. If the distance between A and B is 250m, all transmissions coming to B from nodes closer than 1150m generate a collision (note that this value is higher than the maximum transmission range). If the distance between A and B is 500m all transmissions coming to B from nodes closer than 1750m generate a collision. Finally, if the distance between A and B is as much as the transmission range (1000m), all transmissions coming to B from nodes closer than 2765m (almost three times the maximum transmission range) result in a collision. Our discussion so far considers only two interfering packets. At higher traffic this kind of interference increases, therefore reducing the number of packets received correctly.

We start by presenting the results for the lower data rate case (2000bps). Our results concern the highest density scenarios where the average nodal degree is 15 (which corresponds to $N = 100$). Communications from the sensors to the sink is via shortest path multi-hop routes. Accordingly, nodes are at most 4 hops away from the sink. Our performance comparison does not include T-Lohi which has been designed for single-hop networks [3]. Considering 100 nodes and considering a data rate of 2000bps (i.e., the data transmission time is twice the max propagation delay), the probability of overlapping packets is remarkably high and the number of collisions increases quickly. We observe that many packets have to be discarded because of the overlapping reception of transmissions coming from nodes far away in the network. Fig. 5 shows that only lightweight protocols that exchange very limited amounts of control information are able to deliver all packets to the sink. This is true even at very low traffic. The best performing protocols in this case are ALOHA and slotted ALOHA with ACKs, which are able to deliver all generated packets for $\lambda \leq 0.05$. Despite being lightweight ALOHA and slotted ALOHA without ACKs experience poor performance. This it because they do not have any way to understand if a packet is correctly received or not. Since packets have to traverse multiple links in their way to the sink, packet loss due to collisions or interference is likely to occur, and mechanisms to ensure reliable channel acquisition or to identify and cope with them are needed. PDAP, APCAP and DACAP are able to deliver less than 90% of the generated traffic, even at low load. The reasons are multifold. Each protocol class keeps suffering from the problems discussed for the single-hop case. Transmitting packets through relay nodes increases the time needed to complete their delivery as well as the number of packets in the network, showing such problems for lower λs. Also, interference from nodes far away in the network (a few hops away) may compromise successful packet reception. As expected, the more control packets exchanged, the higher the probability of collision and, hence, more degradation in performance. Nodes usually have a partial view of what is going on in the network and of the possible interferers, and this compromises a protocol's capability to avoid or limit collisions.

We have seen that PDAP is the best performing solution in the single-hop case. However, this protocol is not able to replicate that good performance in the multi-hop scenario because of the way in which parallel communications are scheduled. In a situation where collisions may occur due to interference with transmissions initiated by nodes out of the transmission range, PDAP nodes may not have a complete view of the relevant on-going communications. If information on the possible interferers is not complete, trying to send on the channel as many packets as possible increases the number of collisions and interferences.

One may wonder whether increasing the data rate (here to 28000bps) can solve or mitigate the problems observed at lower rates or not. Increasing the data rate reduces the interference problem since nodes spend less time transmitting and receiving packets. This is evident in Fig. 7(a). For high data rate ALOHA and slotted ALOHA with ACKs are



(a) Received data packets (low traffic)

(b) Sent data packets (low traffic)

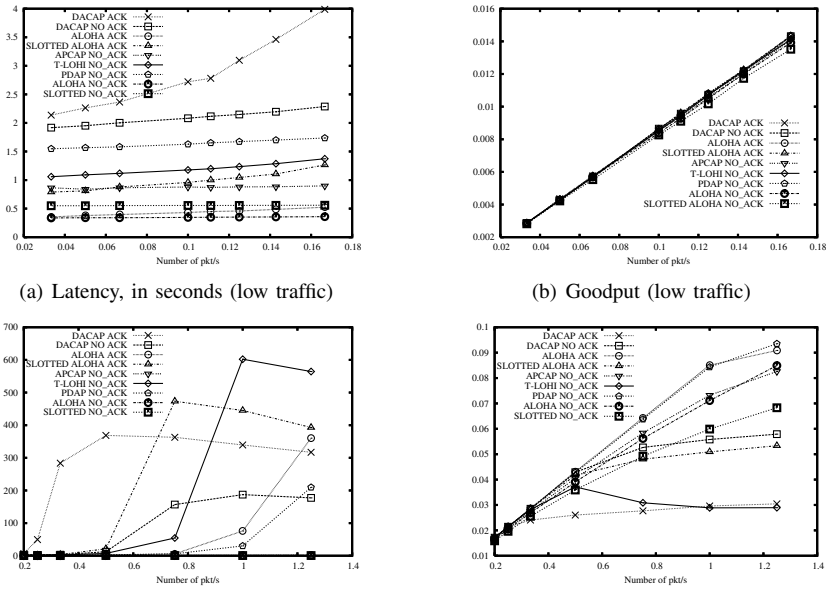
(c) Lost data packets (low traffic)

(d) Received data packets (high traffic)

(e) Sent data packets (high traffic)

(f) Lost data packets (high traffic)

Figure 3. Single-hop networks: Received, sent and lost data packets (28000bps)



(a) Latency, in seconds (low traffic)

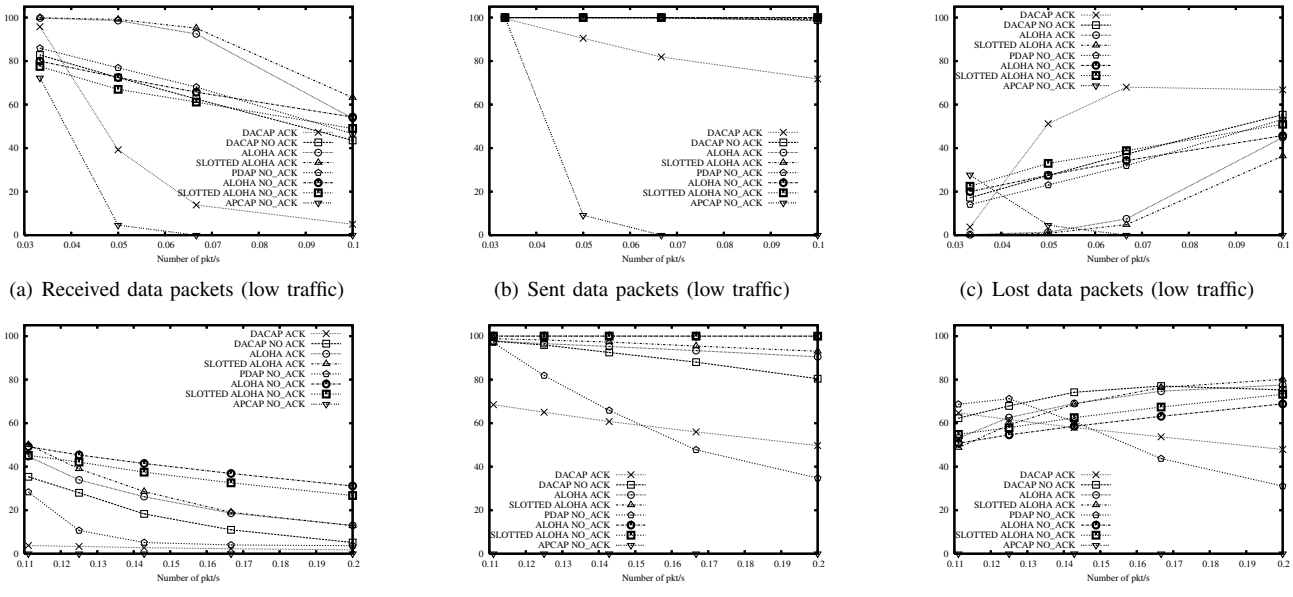
(b) Goodput (low traffic)

(c) Latency, in seconds (high traffic)

(d) Goodput (high traffic)

Figure 4. Single-hop networks: Latency and goodput (28000bps)

effective solutions, being able to deliver all generated packets for all value of $\lambda \leq 0.34$. All the other protocols perform well with low traffic: They are all able to deliver at least 90% of the generated packets for $\lambda \leq 0.11$. However, when the traffic further increases they start loosing high percentage of packets. Among DACAP, APCAP, PDAP and the ALOHAs with no ACKs, the protocols that suffer the most are the DACAP-based ones. This is because DACAP requires higher times to complete communications, it does not allow nodes to interleave transmissions and requires a high amount of control packets. The performance of APCAP is slightly better than that of DACAP. At high data rate, the APCAP aggressive mechanism for channel access generates collisions only at medium-high traffic (when indeed the APCAP performance degrades). As for the low data rate case, PDAP is not able to estimate all the possible interfering transmissions that may result in a collisions (missing information from nodes more than one hop away). This affects the protocol performance especially at medium/high traffic. ALOHA and slotted ALOHA with no ACKs benefit from the fact that they are lightweight protocols: At high data rate the probability of a collision occurring during a transmission to a relay is limited, so that the overall



(a) Received data packets (low traffic)

(b) Sent data packets (low traffic)

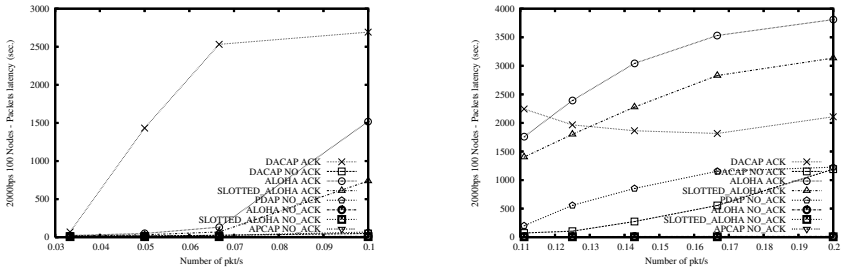
(c) Lost data packets (low traffic)

(d) Received data packets (high traffic)

(e) Sent data packets (high traffic)

(f) Lost data packets (high traffic)

Figure 5. Multi-hop networks: Received, sent and lost data packets (2000bps)



(a) Latency, in seconds (low traffic)

(b) Latency, in seconds (high traffic)

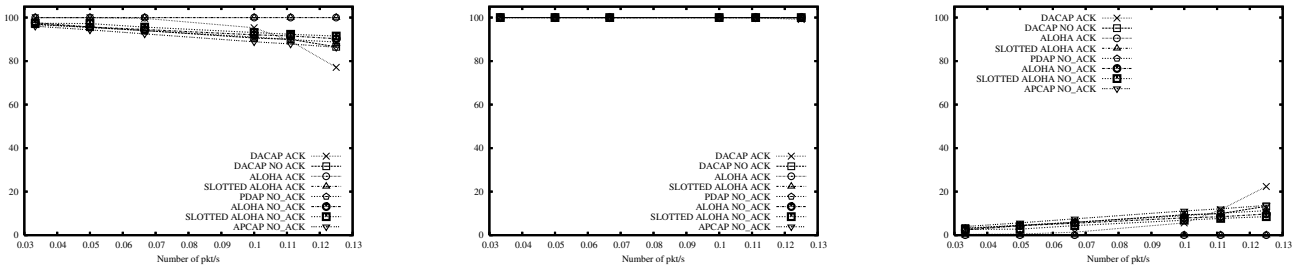
Figure 6. Multi-hop networks: Latency (2000bps)

percentage of packets which are not delivered to the sink is quite limited (between 3 and 20%).

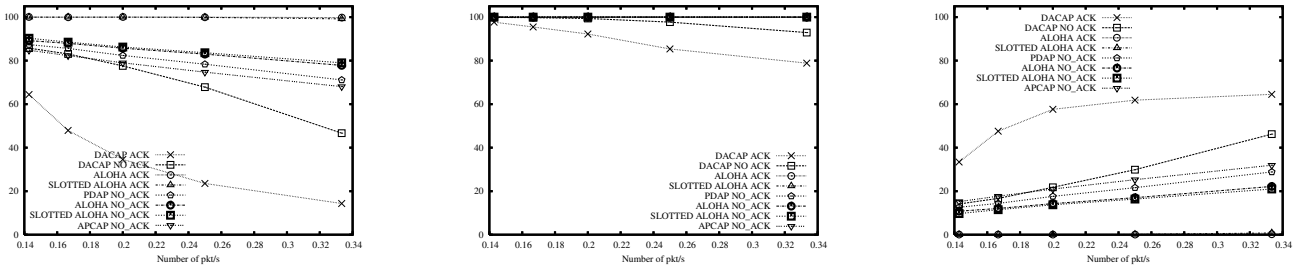
Fig. 8 shows that DACAP also imposes high latencies, because of the longer times nodes wait before transmission, heavy control packet exchange, and collisions resulting in several packet retransmissions. Among the other protocols ALOHA and slotted ALOHA with ACKs experience the highest latencies. These protocols do not have mechanisms for reliable channel access and therefore collisions often occur and packets need to be retransmitted multiple times. ALOHA and slotted ALOHA without ACKs experience the lowest latencies since they do not add delays to data packet transmissions. Furthermore, they do not require control information exchange for ensuring reliable channel access, and they do not require packet retransmissions. The price to pay in this case, however, is a noticeable decrease in the reliability of packet delivery over their versions with ACKs. APCAP and PDAP fall in the middle in terms of latency, with APCAP imposing lower latencies because of its aggressive behavior.

IV. Conclusions

This paper presented a comparative performance evaluation of five protocols for channel access in underwater wireless sensor networks. We have investigated several scenarios that are typical of the current underwater channel access research (single-hop topologies, low data rates) as well as upcoming and future scenarios (multi-hop networks, higher data rates). Our research points out that no protocol fits all scenarios. Depending on the specific setting different classes of protocol perform better than other, and as such the choice of the protocol to deploy is heavily scenario-based. For instance, in single-hop networks PDAP, DACAP with no ACKs and ALOHA with ACKs, in this order, achieve the best tradeoff between packet delivery ratio and latency. Both PDAP and DACAP are effective in reliably seizing the channel, given their knowledge of all communications in the network (readily available in single-hop topologies). In the multi-hop scenario transmissions from interferers far away in the network (even several hops away) may compromise successful packet reception. Not having information of all on-going communications in remote

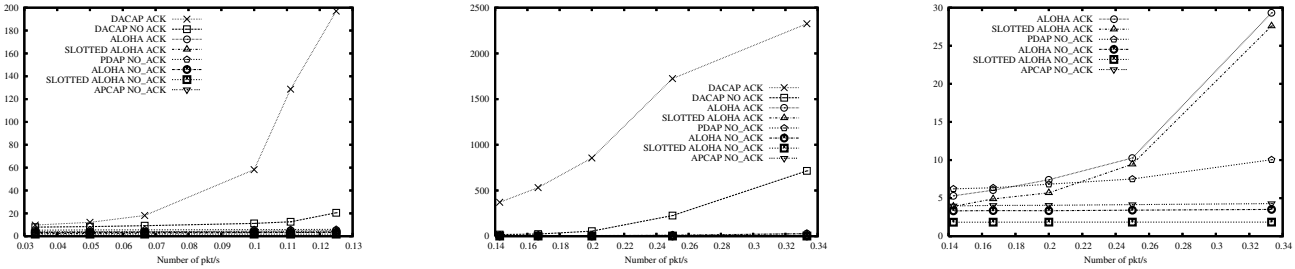


(a) Received data packets (low traffic) (b) Sent data packets (low traffic) (c) Lost data packets (low traffic)



(d) Received data packets (high traffic) (e) Sent data packets (high traffic) (f) Lost data packets (high traffic)

Figure 7. Multi-hop networks: Received, sent and lost data packets (28000bps)



(a) Latency, in seconds (low traffic) (b) Latency, in seconds (high traffic) (c) Latency, in seconds (high traffic). Details

Figure 8. Multi-hop networks: Latency (28000bps)

parts of the network, PDAP and DACAP nodes cannot reliably acquire the channel. Therefore, more lightweight protocols, such as ALOHA and slotted ALOHA with ACKs, lead to significantly improved performance.

ACKNOWLEDGMENTS

The authors wish to thank Stefano Basagni for helpful comments on the writing of this work. Milica Stojanovic was supported in part by the NSF grant 0708420.

REFERENCES

[1] G. Xiaoxing, R. Frater, and J. Ryan. An adaptive propagation-delay-tolerant MAC protocol for underwater acoustic sensor networks. In *Proceedings of MTS/IEEE OCEANS 2007*, Aberdeen, Scotland, June 18–21 2007.

[2] B. Peleato and M. Stojanovic. Distance aware collision avoidance protocol for ad-hoc underwater acoustic sensor networks. *Communications Letters, IEEE*, 11(12):1025–1027, December 2007.

[3] A. Syed, W. Ye, and J. Heidemann. T-Lohi: A new class of MAC protocols for underwater acoustic sensor networks. *Technical report, USC Information Sciences Institute*, July 2007.

[4] M. Molins and M. Stojanovic. Slotted FAMA: A MAC protocol for underwater acoustic networks. In *Proceedings of MTS/IEEE OCEANS 2006*, Boston, MA, USA, September 18–26 2006.

[5] V. Rodoplu and M. Park. An energy-efficient MAC protocol for underwater wireless acoustic networks. In *Proceedings of MTS/IEEE OCEANS 2005*, Washington, D.C., USA, September 18–23 2005.

[6] A. S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, Upper Saddle River, NJ, fourth edition, 2003.

[7] A. Syed, W. Ye, and J. Heidemann. Understanding spatio-temporal uncertainty in medium access with ALOHA protocols. In *Proceedings of the Second ACM International Workshop on UnderWater Networks (WUWNet), September 2007*, pages 41–48, Montréal, Quebec, Canada, September 14 2007.

[8] M. Stojanovic. On the relationship between capacity and distance in an underwater acoustic channel. In *Proceedings of the First ACM International Workshop on UnderWater Networks, WuWNet 2006*, Los Angeles, CA, September 25 2006.