A Study on Remote Data Retrieval Strategies in Underwater Acoustic Networks

Federico Favaro[‡], Loris Brolo[*], Giovanni Toso[*], Paolo Casari[*‡], Michele Zorzi[*‡]

[*]Department of Information Engineering, University of Padova, Italy
[‡]Consorzio Ferrara Ricerche, Ferrara, Italy
E-mail: {favarofe,brololor,tosogiov,casarip,zorzi}@dei.unipd.it

***Abstract*—In this paper we consider the problem of uploading data from a network of fixed sensors to a mobile sink, incorporated by an Autonomous Underwater Vehicle (AUV). We approach this problem by presenting U-Fetch, a protocol based on two levels of coordinated access. In U-Fetch, some head nodes retrieve data from their neighbors locally, and send all data to the mobile sink in bunches via a contention-free link, as the sink passes by. We compare this approach against two approaches endorsed by several works in the literature, namely multi-hop routing and polling. U-Fetch represents an intermediate approach between these two schemes, and constitutes a valid tradeoff in the presence of a mobile sink. We first substantiate this claim via some simple Monte-Carlo simulations. Finally, to confirm the validity of our approach, we reproduce the detailed behavior of U-Fetch in the event-driven ns2/NS-MIRACLE simulator endowed with the DESERT Underwater libraries, and present a set of simulation results that compare the performance of U-Fetch against multihop routing and polling.**

***Index Terms*—Underwater acoustic networks, AUV, Routing, Polling, MAC, U-Fetch.**

I. Introduction

One of the foremost applications of underwater acoustic networks [1] is the continuous monitoring of the environment or of man-made underwater assets. The sensed data is typically transferred to some remote node (or sink) that acts as a control center or is connected to it. This solution is preferred when the data should be harvested on demand, or should be available within a limited amount of time: in these cases it is not affordable to deploy the sensors, leave them on site to record the data, and retrieve them days or weeks later.

The data retrieval from the nodes can be performed using networking techniques which have been the subject of conspicuous investigation in recent years. However, it remains debatable how to retrieve the data so that some quality metric (e.g., the data delivery delay experienced by the sink) is minimized. A technique endorsed by several works is multihop routing [2], where data packets are forwarded to one or more final destinations (the sinks) thanks to the cooperation of intermediate relays. Several protocols can be employed for this task, including simple flooding-based strategies, source-driven approaches, hop-by-hop relay selection and multipath routing. A second technique [3], partly derived from the data mule approach known from terrestrial wireless networks, involves a mobile sink (e.g., incorporated by an Autonomous Underwater Vehicle, or AUV), that patrols the network area and concentrates the administration of data transmissions via a polling-based mechanism.

Multihop routing and polling have pros and cons. Multihop routing requires some form of signaling to set up routes, and therefore works better in static networks, where route updates are infrequent. However, signaling and data packets are prone to collisions, which may disrupt the packet forwarding process; in addition, data packets are typically funneled to the sink through its immediate neighbors, originating an inherently congested area around the sink. Polling supports mobile sinks and sets up inherently collision-free links between the nodes and a mobile sink (the AUV). This makes communications efficient; however, the data retrieval process is strongly limited by the speed of the mobile sink that surveys the network area.

In order to strike a balance between the parallelism offered by multihop routing, while still preserving the advantages of collision-free transmissions in controlled access mechanisms, we propose U-Fetch. Our protocol conveys data to specific nodes called Head Nodes (HNs) via single-hop transmissions, so that the Head Nodes can perform a bulk data transfer towards the mobile sink, as it passes by.

The rest of the paper is organized as follows. In the next section, we take a look at the related literature. In Section III, we describe the protocols tested in this paper, with special focus on U-Fetch, our hybrid approach. In Section IV we test the feasibility of our idea by discussing preliminary results obtained via Monte-Carlo simulations. In Section V we present simulation results obtained via the ns2/MIRACLE network simulator. Section VI concludes the paper.

II. Related Work

In [4] the authors present a hybrid Routing/MAC approach for delivering data to a sink in two-tier networks. When a node generates packets (e.g., sensor data, or an alarm triggered by a specific event), it informs all neighbors about the packets ready for transmission. This information is forwarded through the network within a fixed number of hops. When a mobile sink queries a specific node, which has the responsibility to forward the data packets to the sink (e.g., because it is located in a predetermined position), this node knows in advance which neighbors have generated data, and asks them to transmit their data packets. The main difference between this work and our proposal is that, in [4], the data-generating nodes proactively

inform their neighbors about the presence of new data packets. Furthermore, the control messages used to communicate the presence of data packets ready to be transmitted are sent in anycast, not in broadcast. This implies that the sensors must be aware of their neighbors and of the network topology in advance, which is not the case for U-Fetch.

In [5] the authors present *MDC/PEQ*, a multi-hop routing protocol that dynamically modifies routing tables based on a BEACON message sent periodically by the sink. The authors deals with several interesting issues and provide insight on the possible problems that may arise. In particular, the *energy-hole* problem, where the nodes at one hop from the mobile sink consume more energy because they have to forward the packets from the other nodes, besides transmitting their own packets.

In [6] the authors present *ProFlex*, a protocol to deliver data packets to one or more mobile sinks in a terrestrial wireless radio network. This protocol is based on a multi-hierarchical network structure, where every node can store packets and deliver them to the sink. ProFlex prescribes three phases: *Tree construction*, *Importance factor delivery* and *data delivery*. Furthermore, a form of redundancy is implemented, in order to avoid packet losses when delivering data to the sink. Although ProFlex is well structured, our approach is lighter and easier to implement on a real modem. In addition, it is more feasible in an underwater scenario, where the available bandwidth is very limited, and should be devoted mostly to data transmissions rather than control signaling.

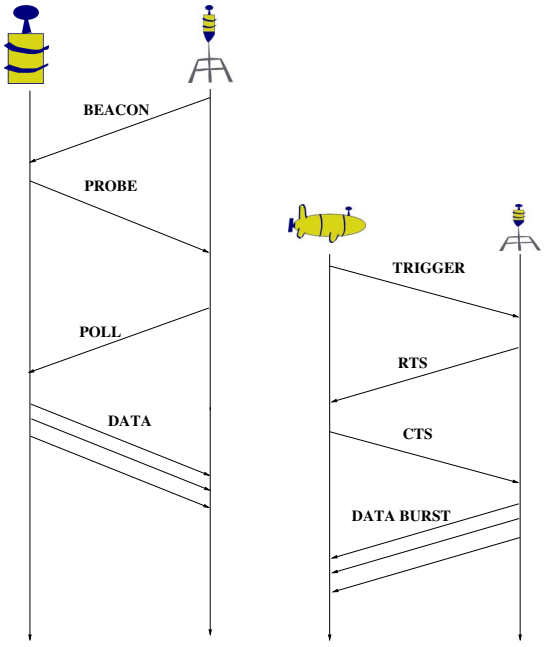
III. Protocol Descriptions

In this section we describe U-Fetch, the two-level coordinated access protocol we propose in this paper. In addition, we recall the description of Uw-Polling [3] and MSUN, a multi-hop routing protocol based on source routing.

A. U-Fetch

U-Fetch organizes the nodes of an underwater network into three roles: the *Mobile sink* (typically an Autonomous Underwater Vehicle, or AUV), which patrols the network to retrieve data packets; the *Sensor nodes*, which sense environmental quantities or monitor man-made assets; the *Head-Nodes* (HNs), special sensor nodes endowed with the responsibility to retrieve the data packets from their neighbors, and forward them to the mobile sink as it passes by. We start the description of U-Fetch by referring to Fig. 1, which shows the control message pattern (a) between the HN and a sensor node and (b) between the HN and the mobile sink.

Every HN sends a BEACON message to inform all neighbors about its presence. Every node that correctly receives the BEACON stores the MAC address of the HN, and answers with a PROBE message after a random backoff period. A node that receives several BEACONs coming from multiple HNs can choose one HN, following a predetermined metric (e.g., the BEACON received with highest SNR) and accordingly answers with a PROBE to the chosen HN. The transmission of data packets from the sensors to the HN is performed following a



(a) Sensor – HN (b) AUV – HN

Figure 1. Control traffic pattern between sensors, HN and AUV

polling scheme. The HN sends a POLL packet to its children in a round-robin fashion, in order to allow each node to transmit their packets. Every node has a fixed amount of time to perform the transmission of the packets.

Despite the use of random backoff times to avoid collisions, some PROBE messages can collide at the receiver. To handle this possibility, we proceed as follows. After the first beacon cycle, the HN sends a cBEACON message (a special type of BEACON) intended only for the nodes that have not transmitted any data packet in the previous beacon cycle. In this manner, we can give a possibility to send data packets also to the nodes whose PROBE packets collided (we recall that such nodes were not polled during the first cycle). The nodes that transmitted data packets in the first beacon phase discard the cBEACON, whereas those that have not transmitted data packets send a PROBE and wait for the POLL message. The transmission of the cBEACON is iterated until nothing[1] is received, or until the maximum allowed number of cBEACONs is reached.

The mobile sink (i.e., the AUV) periodically sends TRIGGER packets to inform possible nearby HNs of its own presence. Every HN that correctly receives this packet answers with a RTS packet including information such as the number of packets it wants to transmit to the AUV, and the time needed to transmit these packets. The latter is based on the estimation of the Round-Trip Time (RTT) between the HN and the AUV, that is performed using the protocol handshake timing and the information contained in the TRIGGER packet (which includes the timestamps of the transmission and reception times). The AUV will choose one node following a predetermined metric

[1]Neither erroneous PROBE packets coming from the neighbors of the present HN, nor other types of packets coming from other nodes.

(e.g., by accepting the first RTS it receives and discarding the others) and will answer with a CTS packet. Using the RTS/CTS handshake, we can also inform all neighboring sensors that the AUV has queried one of the HNs. This way, every sensor that receives a CTS intended for its HN or, similarly, that receives an RTS intended for the AUV but coming from its HN, understands that the AUV is querying the HN and stops the transmission of the data packets. In this version of the protocol, the HNs are fixed and predetermined, and one-hop communications between them and the sensors takes place, since every sensor is within coverage of one HN, and we assume that communications to different HNs do not interfere with one another.

B. Protocols considered for comparison

Uw-Polling [3] is a MAC protocol that adopts a polling-based controlled access scheme. An AUV that patrols the network to retrieve the data packets generated by the nodes sends a TRIGGER message to start a neighbor discovery process. Every sensor that correctly receives this message and has data packets to transmit answers with a PROBE message. The PROBE contains useful information, such as the number of data packets that the node wants to transmit. The AUV receives the PROBEs for a fixed amount of time, and then cyclically polls all the PROBE senders via sequential POLL messages. Once a node is polled, it has a fixed amount of time to transmit its packets.

The Multi-sink Source routing protocol for Underwater Networks (*MSUN*) is a routing protocol partly inspired to Dynamic Source Routing (DSR), to which it adds several new features that improve the performance of source routing in underwater scenarios, especially in the presence of multiple sinks. MSUN has been designed to be scenario-independent, i.e., it works in any connected topology and does not need side information (e.g., location, depth, or channel state) in order to operate correctly. MSUN is a reactive source routing protocol, in that the source nodes choose a complete route towards the sink for each of their packets, and embed a full specification of this route in the packet header. In turn, this avoids a hop-by-hop relay election process. A reactive behavior is a valid approach because the bandwidth available for transmissions underwater is typically very narrow, and it is preferable not to employ a too large portion of it for proactive route discovery.

IV. Network topology and preliminary results

In this section we present the network topology considered in this paper and we show preliminary results obtained via a simple Monte-Carlo simulation with Matlab.

Some assumptions, described below, are made to simplify the problem. The scenario we consider is depicted in Fig. 2. A network of 36 nodes is arranged into a square grid topology within a 5 km $\times$ 5 km area, with nearest neighbors 1 km apart. To simulate the traffic generation process, a number of packets is randomly generated and distributed among the nodes, who need to deliver their packets to an AUV which represents the mobile sink. This effectively configures our test as a "queue emptying" simulation, making it possible to consider a simplified model for the behavior of the nodes, instead of simulating the full state machine of all protocols. (The latter, more detailed approach is employed in Section V.) We first assume that one AUV is present in the network to act as the sink. We assume a disk connectivity model, where each node can transmit (receive) with no errors to (from) nodes located within a range of 1.5 km. We also assume that the communications between any two nodes take place at a bit rate of 2 kbps, except those involving the AUV, which employs a higher bit rate of 10 kbps. In addition, we consider the following simplifying assumptions for the three protocols under study:

Uw-Polling—this is the simplest case: the AUV follows a lawnmower trajectory (represented by the black line in Fig. 2). Every time a node is within the coverage range of the AUV, it uploads either all its data packets, or those that fit within the contact period, until the AUV gets too far to communicate. When the contact is interrupted, another node is chosen at random among those presently within the coverage range of the AUV, and uploads its data for the duration of the contact, or until its queue is empty. This process goes on until all nodes have an empty queue.

MSUN—In this case, the AUV is located at fixed position on the right side of the network, so that its neighbors are the two central nodes of the last column of the grid in Fig. 2. The network collaborates to forward data to the AUV. For simplicity, assume that a spanning tree is known to all nodes, so that each node knows its next hop towards the AUV. We approximate the routing process (and the fact that multiple nodes can concurrently forward their packets in non interfering portions of the network) in the following way. Consider two nodes, say A and C, and their respective next hops, say B and D. We call A$\rightarrow$B and C$\rightarrow$D non-interfering pairs whenever A and B are located outside both C and D's coverage range (and vice-versa). We start by selecting a node n at random. This also identifies its next hop, say m. We then identify at random other pairs, non-interfering with $n\rightarrow m$, until no more nodes can be enabled to transmit without causing interference to the pairs thus identified. In all pairs, the source nodes transfer all their data packets to their respective next hops. After that, we start over with new pairs, and go on until all packets have been funneled towards the AUV.

U-Fetch—We assume that the head nodes are fixed (they are represented using a different icon in Fig. 2) and that they sequentially ping all their neighbors to retrieve their data. Simultaneously, the AUV travels along the trajectory depicted in Fig. 2 as a dashed grey line. Notice that the AUV trajectory can be shorter with respect to the Uw-Polling case, since data packets are concentrated by HNs before they are actually uploaded to the sink. In turn, this allows to save the time required by the AUV to patrol all nodes. As it enters the coverage range of a HN, the AUV connects to the HN, which uploads its packets to the AUV until the AUV gets outside its coverage range, or until there are no more packets to transmit. The process goes on until all nodes have uploaded their data

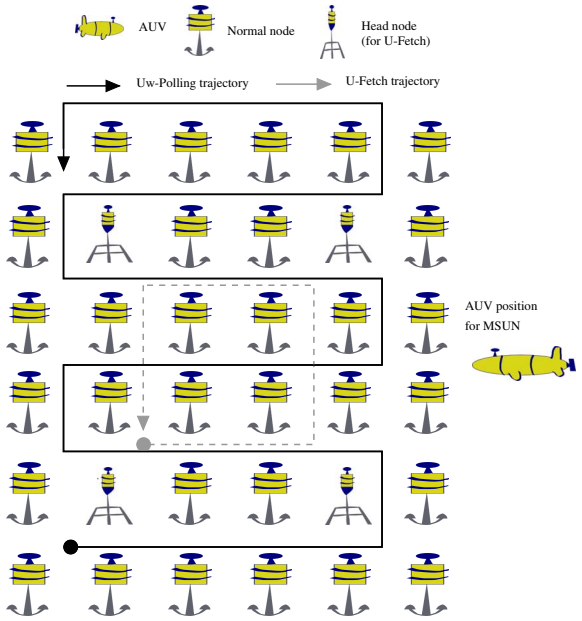


Figure 2. The scenario considered in our evaluation. For U-Fetch, the head nodes are represented as modems on a tripod.

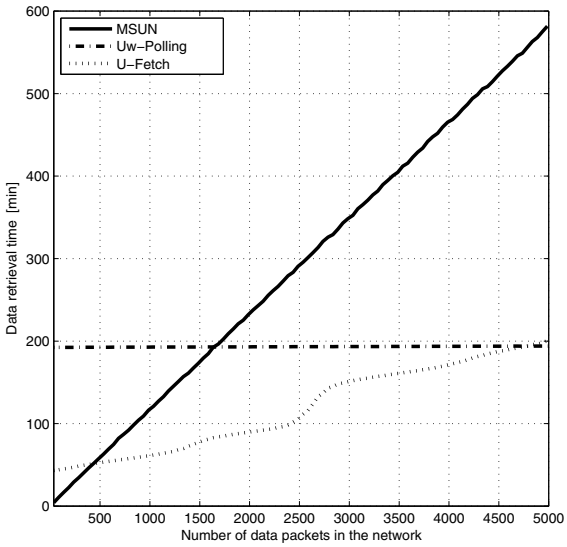


Figure 3. Data retrieval time vs. number of data packets generated in the network.

to their respective HNs, and the AUV has been able to retrieve them all.

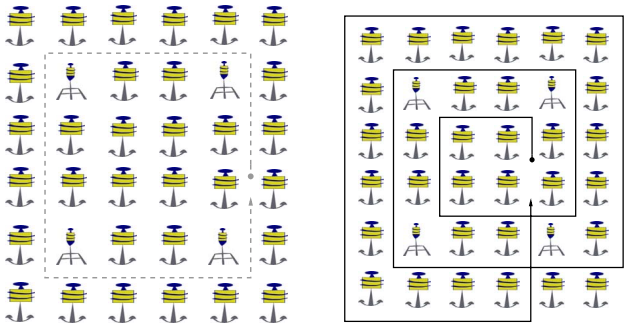
The average data retrieval time in minutes (taken over several realizations of the packet generation process) is shown in Fig. 3 for all protocols, as a function of the total number of packets generated in the network. As expected, the retrieval time of MSUN increases linearly as the initial number of packets in the network increases, due to the funneling effect. The retrieval time experienced by Uw-Polling is almost constant and equal to the time required to complete one trajectory. Initially, this time is higher than MSUN's due to the need to complete almost a full trajectory in order to make contact with all network nodes. However, for increasing traffic, Uw-Polling takes less than MSUN, because the AUV can leverage on its higher bit rate link for communicating with all nodes, whereas the communications among ordinary nodes take place at a lower bit rate in MSUN.

For a low number of packets generated in the network, U-Fetch performs worse than multihop routing because of the limited AUV speed, but it becomes soon more convenient than both MSUN and Uw-Polling. This corroborates our intuition that an approach to data retrieval based on two levels of coordinated access can considerably improve the data retrieval times. However, this improvement also depends on the bit rate of the communications with the AUV and on the navigation speed of the AUV.

V. Simulation Parameters and Results

In order to provide better insight on the feasibility of U-Fetch, we set up a simulation campaign using DESERT Underwater [7], a set of C++ libraries developed for the ns2/NS-MIRACLE [8] network simulator. We used the same topology depicted in Fig. 2. The results are averaged over 50 network realizations, where, in each realization, the position of each node is drawn at random within a circle of radius 125 m around its nominal position as shown in Fig. 2. The depth of the nodes, instead, is fixed at 50 m. The AUV patrols the network at a speed of 4 knots at a fixed depth of 10 m, following the solid black and grey trajectories depicted in Fig. 2 for Uw-Polling and U-Fetch, respectively. The transmission power is set to 160 dB re μPa^2, which yields a transmission range of about 1500 m. The carrier frequency and the bandwidth of the PHY layer are respectively 26 kHz and 16 kHz. These parameters are chosen in order to mimic the operational bandwidth of the EvoLogics S2C modems [9]. The power consumption during transmission, reception and idling is also chosen accordingly (100 W for transmission, 0.8 W for reception, and 8 mW for idling). Via a specific DESERT module, we reproduce the possibility to use different bit rates depending on the nodes that take part in a communication. We set this module so that every communication among the nodes is carried out at a bit rate of 2 kbps, whereas those involving any node and the AUV take place at a rate 10 kbps. This way, we can simulate the transmission bit rate employed in EvoLogics S2C modems for the Instant Message or Burst Data modes, respectively, as defined by the D-MAC protocol [10]. The speed of the AUV along its trajectory is fixed to 4 knots.

We start our study with some considerations on the trajectory of the AUV employed using U-Fetch and Uw-Polling (we recall that the AUV is fixed at a given position in MSUN simulations), and on their impact on the performance of the protocols. We start from U-Fetch, where we recall that the AUV must circulate near the HNs in order to retrieve their data packets and those of their respective neighborhoods. Observe that, for a fixed AUV speed, there exists a tradeoff between the duration of the contact between the AUV and the HNs (which makes it possible to retrieve more data packets), and



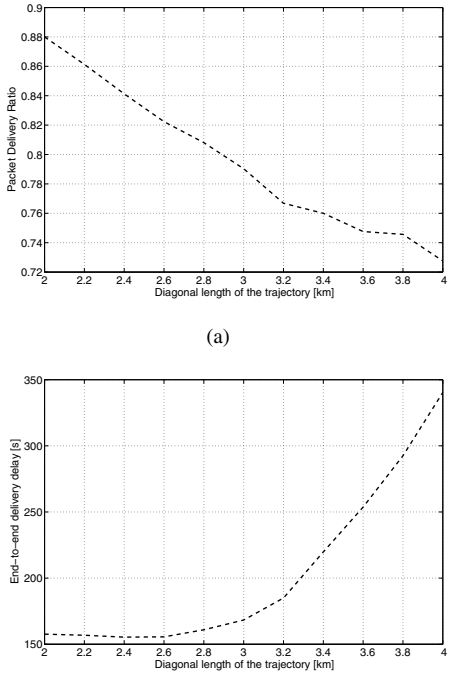
(a) (b)

Figure 4. New trajectories of the AUV in the case of (a) U-Fetch and (b) Uw-Polling.

the average distance of the AUV from the HN during the contact (which decreases the quality of the AUV-HN link). This tradeoff can be managed by increasing the trajectory length (while keeping its shape square for simplicity) from that of the dashed grey line in Fig. 2 to that of Fig. 4(a). In particular, we make a test by increasing the length of the diagonal of each trajectory in steps of 200 m from the former, innermost trajectory to the latter, outermost one.

Fig. 5 shows (a) the Packet Delivery Ratio (PDR) defined as the number of correctly delivered packets over the total number of generated packets and (b) the end-to-end delivery delay of a packet, both as a function of the diagonal length of the tested trajectory (which we recall has a square shape). The traffic generation rate is set to $\lambda = 0.16$ packets per minute per node. We observe that the PDR decreases and the delay increases as the trajectory becomes longer, because the worst–case SNR of the link between the HNs and the AUV decreases. In addition, we observe that while the end-to-end delay is essentially stable until a diagonal length of 2.8 km, it increases after that. This is due to the fact that longer trajectories increase the contact time between the AUV and the HNs, but at the same time they force HNs currently not in contact with the AUV to wait longer for their turn.

We investigate this issue in more detail with the help of Fig. 6, where we show the end-to-end delay as a function of the packet generation rate per node, only for the innermost (shortest) and outermost (longest) trajectories. Consider the outermost trajectory first. When the traffic generation rate is low, the probability that a node has a packet ready for the AUV when the latter is not within its coverage range is high, hence the end-to-end delivery delay is not minimum. When the traffic increases, the delay remains constant and then quickly increases again as the traffic reaches the maximum value considered. In this regime, the protocol cannot effectively deliver packets to the AUV and the average delivery delay increases to about one hour. Interestingly, the delay at high traffic is much higher for the innermost than for the outermost trajectory. This is due to the fact that in the latter case the AUV remains within the coverage area of one HN for a longer time. Hence, even if the PDR is worse on average, the correctly received packets experience lower delay in the presence of



(a)

(b)

Figure 5. (a) Packet Delivery Ratio and (b) end-to-end delay as a function of the trajectory followed by the AUV in U-Fetch, from the innermost to the outermost. $\lambda = 0.16$.

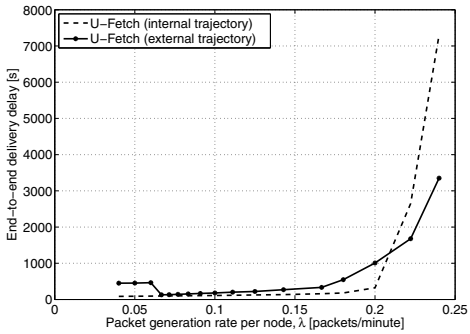
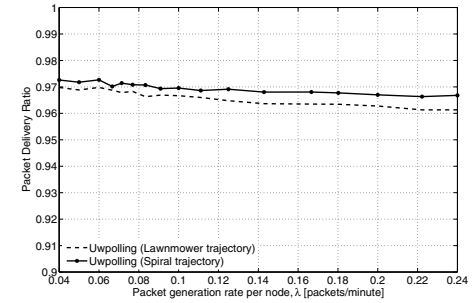


Figure 6. End-to-end delivery delay as a function of the data generation rate for the two trajectories considered for U-Fetch.

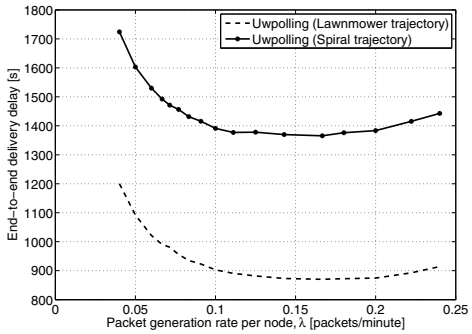
high traffic.

We consider now Uw-Polling, and report in Fig. 7 the difference between the lawnmower trajectory of Fig. 2 and the spiral trajectory of Fig. 4(b) in terms of (a) PDR and (b) end-to-end delivery delay achieved by the protocol as a function of the packet generation rate per node. While the PDR is between 0.96 and 0.97 for both trajectories, we observe that Uw-Polling achieves lower end-to-end delay with the lawnmower course than with the spiral course. In fact, while both trajectories give an equal opportunity to all nodes for uploading their data packets, the spiral trajectory is longer, and leads to longer starvation periods for all nodes, thereby increasing the impact of the queuing time on the end-to-end delay.

We finally note that the delay of Uw-Polling (Fig. 7(b))



(a)



(b)

Figure 7. Comparison between the two trajectories followed by the AUV (lawnmower and spiral) using Uw-Polling in terms of (a) Packet Delivery Ratio and (b) end-to-end delivery delay

decreases for increasing traffic, reaches a minimum and then increases again. This suggests that for low traffic generation rates, packets are generated when the AUV is not in coverage of the source incur high delays. The opposite is true for the highest values of λ: in this case, the probability that the nodes transmit all data in their queue to the AUV before it exits their coverage range is very low. Hence, many packets may remain in the queues of the nodes for more than one AUV lap.

Based on the previous results, we will use the internal trajectory for the U-Fetch protocol in the following. This allows U-Fetch to achieve low delay and higher PDR for most values of the traffic generation rates considered here. According to the same rationale, for Uw-Polling we choose the lawnmower trajectory.

Fig. 8 shows the PDR of the three considered protocols as a function of the packet generation rate per node. We consider also two versions of the MSUN protocol, one with no retransmission of not acknowledged packets and a second one employing up to 4 retransmissions. MSUN with 4 retransmission exhibits the best PDR in the case of low traffic in the network. MSUN, taking advantage of the possibility to retransmit erroneous packets, can achieve a PDR very close to 1. As the traffic increases, Uw-Polling improves its performance by leveraging on the contention-free communications set up between the AUV and the sensors. This allows it to retrieve almost all the packets generated by the nodes. As the traffic increases, the PDR of MSUN decreases steadily both in the case with 4 retransmission and in the case with no

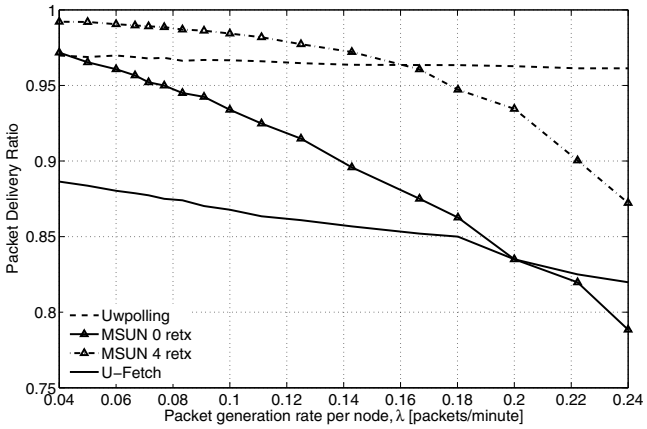


Figure 8. Packet Delivery Ratio as a function of the packet generation rate λ, in packets per minute per node.

retransmission. This is a direct consequence of the funneling of traffic at the sink, and affects in particular the packets that need to travel more hops towards the AUV.

U-Fetch exhibits the worst PDR among the three protocols in this scenario. The two phases in which the protocol divides the delivery of the packets to the sink (sensor→HN and HN→AUV), require a fine tuning of the packet exchange timing. Even so, some packets transmitted by an HN to the AUV may harmfully interfere with the communication between a nearby HN with a sensor, even if the RTS and CTS packets used between HN and AUV to establish the connection are useful to inform the nodes that a transmission between a HN and an AUV will occur. Moreover, some packets may be transmitted from the HN to the AUV when the AUV is not within the coverage range of the HN any more. However, as the traffic increases, U-Fetch's PDR becomes slightly better than MSUN's with no retransmissions. This is because, in high traffic conditions, the links between the AUV and the nodes one-hop away become busy and several packets may be lost due to interference.

We now turn to the analysis of the total energy spent by a node during a simulation time of ≈29 h using each of the three protocols considered in this study. Fig. 9 shows that Uw-Polling exhibits the lowest energy consumption, because the sensors transmit (for a limited amount of time) only if the AUV is within this coverage range and triggers them. MSUN, instead, exhibits the highest power consumption. Every node transmits for a large amount of time during the simulations, both to exchange control messages for path establishment and to deliver the packets to the sink AUV through multiple hops. With 4 retransmissions, the energy spent by the protocol is even higher. Especially when the data generation rate is high, the energy spent by the nodes is considerable. U-Fetch achieves an intermediate power consumption, higher than Uw-Polling's (because U-Fetch prescribes the exchange of more control messages to establish the sensor→HN and the HN→AUV connections, and because most packets reach

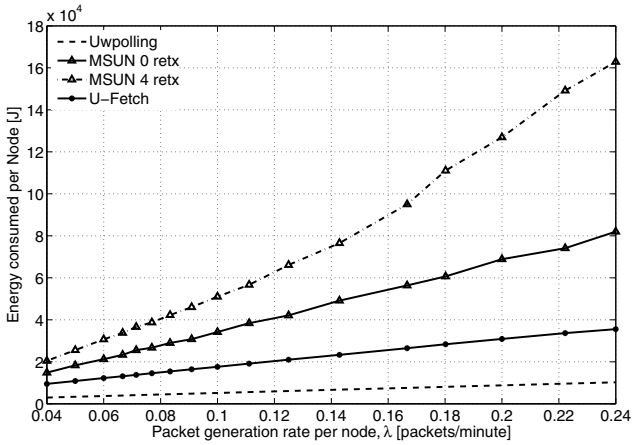


Figure 9. Total energy consumed by the nodes (HN and sensors) during the simulation time (≈29 h) as a function of the packet generation rate λ, in packets per minute per node.

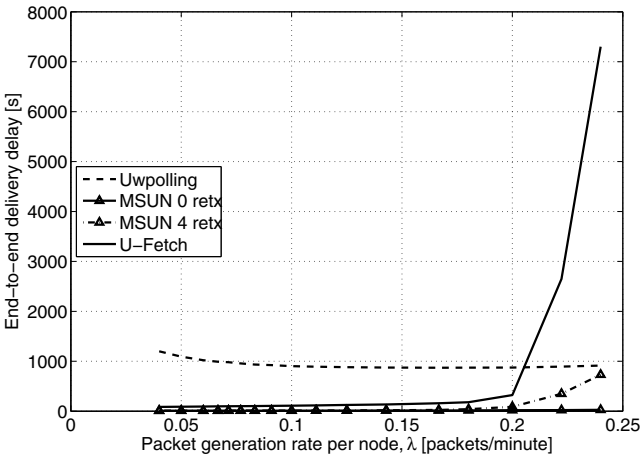


Figure 10. End-to-end delivery delay as a function of the packet generation rate λ, in packets per minute per node.

the sink only via an additional hop through one HN) but lower than MSUN's (due to the control messages that regulate the sensors→HN transmissions and to the RTS/CTS paradigm implemented to build the communication among the AUV and the HN). In this perspective, U-Fetch can be a good alternative between a source routing protocol and a polling protocol.

In Fig. 10, we now examine the average end-to-end delivery delay for the three protocols. Uw-Polling exhibits the highest delay, one order of magnitude higher than U-Fetch's, that takes advantage of the presence of the HNs to perform shorter trajectories for retrieving the data packets. On the contrary, in Uw-Polling, a node has to wait until the AUV is within its own coverage range to deliver the packets. Only in the high traffic regime does U-Fetch become less effective in routing packets to the AUV; in this case, the sensor nodes have a limited amount of time to transmit their own data packets to the HNs, and HNs cannot transmit all the data packets coming from their associated sensors nodes plus their own packets to the AUV during a single contact. Hence, it becomes likely that many packets have to remain in the queues of the HN or of the sensor for a long time before the AUV performs another lap. This leads to a very rapid increase of the delay in the high traffic regime. For most of the traffic generation rates considered in this paper, however, the end-to-end delivery delay of U-Fetch is quite stable and comparable to that of MSUN, which exhibits the best delay, on the order of tens of seconds per packet. This delay is mostly due to the establishment of the paths; once the paths have been created, however, they ensure a continuous flow of packets towards the sink. Note that this behavior is in agreement with the preliminary results shown in Fig. 3 for a low number of packets in the network, which corresponds to a stable, low traffic region in Fig. 3. For this reason, MSUN performs better than the other protocols in terms of delay. Even with four retransmission, MSUN's end-to-end delivery delay remains the best among the three protocols considered. In the presence of high traffic, however, the delay increases and becomes quite similar to Uw-Polling's. With high traffic generation rates, retransmissions become more likely as the probability that a packet is lost due to interference is higher.

VI. Conclusions

In this paper we presented U-Fetch, a data retrieval protocol for hybrid static/mobile underwater acoustic networks where a mobile AUV needs to retrieve data packets from several sensor nodes. U-Fetch is based on two levels of controlled access, and thereby aggregates traffic at Head Nodes, before uploading it to the sink.

It has been shown that U-Fetch represents a viable tradeoff between the high PDR and high delay of Uw-Polling (a polling-based MAC protocol relying on an AUV to administer the data retrieval) and the intermediate PDR and low delay of MSUN (a source routing protocol operated in the absence of sink mobility). Finally, a preliminary study has been conducted for the optimization of the AUV trajectory in U-Fetch and Uw-Polling. For the former, it has been shown that the most detrimental effect is the lower worst-case PDR. In turn, this requires to design the trajectory so that the contacts between the AUV and the HNs are long enough to deliver all packets in the queue, but at the same time the AUV never gets too far from the HNs. For Uw-Polling, it has been shown that the shortest trajectory leads to the best results in terms of both PDR and delivery delay.

Acknowledgments

This work was supported in part by the Italian Institute of Technology within the Project SEED framework (NAUTILUS project), by the European Commission under the 7th Framework Programme (Grant Agreement 258359 – CLAM), and by the US National Science Foundation under Grant no. CPS-1035828.

REFERENCES

[1] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking," in *Proc. IEEE WCNC*, Las Vegas, NV, Apr. 2006.

[2] L. Guangzhong and L. Zhibin, "Depth-based multi-hop routing protocol for underwater sensor network," in *Proc. ICIMA*, Wuhan, China, May 2010.

[3] F. Favaro, P. Casari, F. Guerra, and M. Zorzi, "Data upload from a static underwater network to an AUV: Polling or random access?" in *Proc. MTS/IEEE Oceans*, Yeosu, Korea, May 2012.

[4] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "TTDD: Two-tier data dissemination in large-scale wireless sensor networks," *Wireless Networks*, vol. 11, pp. 161–175, Nov. 2005.

[5] R. W. N. Pazzi and A. Boukerche, "Mobile data collector strategy for delay-sensitive applications over wireless sensor networks," *Elsevier Comput. Commun.*, vol. 31, no. 5, pp. 1028–1039, Mar. 2008.

[6] G. Maia, D. L. Guidoni, A. C. Viana, A. L. Aquino, R. A. Mini, and A. A. Loureiro, "A distributed data storage protocol for heterogeneous wireless sensor networks with mobile sinks," *Elsevier Ad Hoc Networks*, vol. 11, no. 5, May 2013.

[7] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "DESERT Underwater: An NS–Miracle-based framework to design, simulate, emulate and realize test-beds for underwater network protocols," in *Proc. MTS/IEEE Oceans*, Yeosu, Korea, May 2012.

[8] N. Baldo, M. Miozzo, F. Guerra, M. Rossi, and M. Zorzi, "MIRACLE: The Multi-Interface Cross-Layer Extension of ns2," *EURASIP Journal on Wireless Communications and Networking*, Jan. 2010. [Online]. Available: http://www.hindawi.com/journals/wcn/2010/761792/cta/

[9] P. Casari and M. Zorzi, "Protocol design issues in underwater acoustic networks," *Elsevier Computer Communications*, vol. 34, no. 17, pp. 2013–2025, Nov. 2011.

[10] O. Kebkal, M. Komar, and K. Kebkal, "D-MAC: Hybrid media access control for underwater acoustic sensor networks," in *Proc. IEEE ICC*, Cape Town, South Africa, 2010.