A Self-Adaptive Protocol Stack for Underwater Wireless Sensor Networks

Valerio Di Valerio,* Francesco Lo Presti,† Chiara Petrioli,*‡ Luigi Picari*‡ and Daniele Spaccini*‡
* Dipartimento di Informatica
Università di Roma "La Sapienza," Rome, Italy
{divalerio, petrioli, picari, spaccini}@di.uniroma1.it
† Dipartimento di Ingegneria Civile e Ingegneria Informatica
Università di Roma, "Tor Vergata," Rome, Italy
lopresti@info.uniroma2.it
‡ WSENSE s.r.l., Rome, Italy

***Abstract*—In this paper we propose the adoption of a self-adaptable cross-layer and modular Software Defined Communication Stack (SDCS) for Underwater Wireless Sensor Networks. The SDCS is a modular stack solution which is capable to run different protocols at each layer of the network stack; a new component, named policy engine, autonomously and adaptively, as the operational conditions vary, selects the protocol of each layer so as to optimize application scenario metrics of interest, e.g., packet delivery ratio, end-to-end packet latency and energy consumption. As a proof of concept, the paper presents the design and performance evaluation of a policy engine to dynamically and autonomously change the MAC protocol adopted in Underwater Wireless Sensor Networks. The best MAC protocol is chosen according to network conditions and application requirements, without any a priori knowledge. We consider three different MAC protocols running in the SDCS: CSMA, T-Lohi and DACAP that represent the class of simple, intermediate and fully negotiated MAC protocols, respectively. The performance of the three protocols are first compared via simulations considering different network conditions, such as traffic load and packet size. Then, we evaluate the ability of our policy engine to dynamically estimate the network changes and then to select accordingly the best MAC protocol without any a priori knowledge. Results show the effectiveness of our solution in that it is always able to quickly find and choose the MAC protocol that optimizes a given metric in a particular scenario by introducing a really limited overhead in the network.**

***Index Terms*—Underwater Sensor Networks, Adaptive protocols, Policy engine, Software Defined Communication Stack, SUNSET.**

I. Introduction

In recent years, Underwater Wireless Sensor Networks (UWSNs) have emerged as the key enabler of a wide range of applications ranging from port security and environmental monitoring, to discovery and protection of marine archaeology [1]. To meet the diverse requirements of an increasing number of applications, many research efforts have focused on designing, implementing and testing novel and increasingly more flexible communication protocols (see [2] for a comprehensive survey). In this context, the challenge is represented by the unique characteristics of the underwater environment, like long propagation delays, low bandwidth, slow decaying signal attenuation, and asymmetric links, among others. Underwater signal propagation conditions depend on the specific scenario, i.e., shallow and deep water, port, sea, ocean, lake, river, etc, and also exhibit significant variability over time [3], [4]. These challenges dictate the need to develop flexible solutions, which dynamically adapt their operation to varying conditions and demand, ensuring to achieve good performance over time.

As confirmed by in field experiments, up to date no existing solution is capable to offer consistent behaviour in the different operational scenarios: solutions which are considered best for some particular scenarios, might perform poorly in others [5], [6]. This is further exacerbated by the inherent non-stationarity of the underwater acoustic channel. To address this issues, recently adaptive protocols have started being proposed in the literature [7], [8], [9], [10]. Nevertheless, their adaptation capability appears to be limited in scope and effectiveness.

To overcome these limitations, in this paper we propose to design the network protocols stack as a self-adaptive system, capable to dynamically adapt to the different and ever changing conditions. To this end, we propose to move away from the monolithic communication stack to an open, cross-layer and modular Software Defined Communication Stack (SDCS) [11] that acts as a self adaptive system[1]. To manage the self-adaptation at each layer of the SDCS, we define a new component, called *policy engine*. The policy engine dynamically adapts the protocol stack configuration - by even possibly modifying the protocol of a given layer at run time - according to dynamic changes in physical parameters, network topology, acoustic noise sources, etc., in order to optimize the performance with respect to the ever changing scenario conditions.

As a proof of concept, in this paper we focus on a policy engine which dynamically and autonomously adapts the node MAC protocol. This is motivated by previous studies like [6], [13], [14], in which the authors showed that different MAC protocols are needed to provide good performance under different network configurations (e.g., packet size), environ-

[1]We have implemented such approach in the SUNSET [12] enabled SDCS to enable reliable, robust underwater assets communication and cooperation within the FP7 SUNRISE project [11]. Patents have been filed on SUNRISE SDCS.

mental conditions and traffic loads. So, for instance, if the channel becomes highly asymmetric, better performance can be expected by switching from handshaking-based solutions, that require symmetric communications, to protocols which do not rely on handshaking. However, we remark that the proposed approach is general and can be used, possibly with minor changes, at any layer of the SDCS.

Our contributions are as follows:

- We propose the adoption of a self-adaptable cross-layer and modular Software Defined Communication Stack (SDCS) [11] for optimizing Underwater Wireless Sensor Networks. We design the SDCS as an autonomous system with self-adaptability capability. Adaptation is enabled by means of a new component, the policy engine. The policy engine is designed according to the so called MAPE-K loop [15], which represents a general architectural paradigm to build self-adaptable systems. Its behaviour is based on a feedback-control loop that *Monitors* the environment, *Analyzes* data to detect changes, *Plans* the reconfiguring actions and *Executes* them.
- We propose the use of reinforcement learning techniques to determine the adaptation strategy - which corresponds to the *Plan* phase of the MAPE-K loop of the policy engine. In particular, we formulate the planning strategy as an N-armed bandit problem, an effective strategy belonging to the Reinforcement Learning family. Since N-armed bandit problems do not require any a priori knowledge about the environment, they are particularly suitable for our setting, in which it is so difficult to explicitly model all the underwater environment features that affect network performance. Using an N-armed bandit problem we are able to learn over time what is the best protocol for each environment, network condition and application requirements.
- We extend the SUNSET framework [12] to support a full Software Defined Communication Stack (SDCS) to provide for flexible choice of protocols and optimization throughout the protocol stack. This innovative component allows modular and cross-layer selection of pre-defined protocols at all levels of the stack through the policy engine module.
- We evaluate our approach for dynamic MAC protocol selection through simulations by using SUNSET connected to the Bellhop ray tracing tool [16] and considering single-hop networks composed by 6 nodes plus the sink node. The SDCS implements three different MAC protocols: the well-known CSMA protocol [17]; the T-Lohi protocol [18], that uses a type of weak negotiation to reserve the channel based on a short control packet (`TONE` packet); the DACAP protocol [19], that uses an `RTS/CTS` handshake to reserve the channel, enhanced with very short `WARNING` packets. We evaluate the ability of our policy engine to dynamically switch and to choose the best MAC layer protocols according to the current operating scenario. Our results show the effectiveness of our solution. They show that the policy engine is always able to learn the optimal protocol and to quickly react to changes in the environment, under various underwater conditions and application requirements.

The rest of this paper is organized as follows. We discuss related work in Section II. An overview of the proposed approach is described in Section III. The policy engine is validated through an extensive performance evaluation presented in Section IV. The conclusions are drawn in Section V.

II. Related Work

In recent years, several MAC protocols have been proposed for UWSNs due to the unique characteristics of the underwater channel. The majority of these protocols are based on a *contention-based* approach, e.g., [7], [8], [9], [17], [18], [19], [20], [21], [22], [23], [24]. Only few solutions, such as [10], [14], [25], [26], [27], [28], explore instead *contention-free* techniques focusing on a Time Division Multiple Access (TDMA) approach. However, TDMA based protocols require a tight synchronization between network nodes that is not easy to achieve in the underwater environment.

A coarse grain classification of contention-based protocols is proposed in [6], in which the authors group existing solutions into *simple* [17], [20], *intermediate* [9], [18], [20] and *fully* [8], [19], [21], [22], [24] negotiated protocols. Simple negotiated protocols does not use any kind of control message before the actual data packet transmission. Conversely, a MAC protocol is defined intermediate negotiated if a node sends a control packet to reserve the channel before sending a data packet. These protocols, though, are affected by the hidden terminal problem. To overcome this problem, fully negotiated solutions perform a full control packets handshaking between the sender and the intended receiver(s), to acquire the channel before each data packet transmission.

The well-known Carrier Sense Multiple Access (CSMA) [17] protocol is an example of a simple negotiated protocol. In [20] the authors propose the Aloha with Collision Avoidance (Aloha-CA) protocol that exploits the typical long propagation delay of underwater network. In particular, every node overhearing data packets can estimate for how long the channel will be busy and therefore schedule, accordingly, the next data packet transmission.

Since the simple negotiated protocols do not reserve the channel, they might suffer from high collisions rate. In [20] authors present an intermediate negotiated protocol, termed Aloha-AN. It is an extension of Aloha-CA where the nodes send a small notification packet (`NTF`) to reserve the channel before the actual data packet transmission. Similarly, in the T-Lohi protocol [18] nodes contend for the channel sending a wakeup tone. The `TONE` packets allows to speed up the reservation phase thus reducing the energy consumption. Moreover, the number of tones received are also used to compute the backoff time before a transmission attempt. Similarly, the Adaptive Energy Reservation MAC protocol (AER-MAC) [9] uses reservation packets to schedule the actual data packet transmissions with neighboring nodes. Additionally, it adapts

the transmission power according to node distance, in order to reduce packet collisions and energy consumption.

Since the use of the intermediate negotiated protocols do not solve the hidden terminal problem, several fully negotiated MAC solutions have been proposed in the past years. In [22] the authors presents MACA-U that provides an adaptation of the terrestrial multiple-access collision avoidance (MACA) protocol to the underwater acoustic channels. In particular, the size of the control packets is increased to compensate for the long propagation delay typical of the underwater environment. More sophisticated handshake-based solutions are the Distance-Aware Collision Avoidance Protocol (DACAP) [19] and the Bidirectional-Concurrent MAC protocol (BiC-MAC) [21]. In particular, DACAP estimates the distance among nearby nodes and uses this information to improve the efficiency of the handshake and the channel utilization. To overcome the spatio-temporal uncertainty [25] problem, DACAP introduces a short `WARNING` control packet. This packet is sent by a node that is waiting for a `DATA` packet when it receives an `RTS` from another node. The node that receives the `WARNING` packet aborts the transmission. MACA-APT [8] is a fully negotiated MAC protocol which exploits an adaptive packet train size transmission after every successful handshake. The packet train size is selected according to the network topology and the channel conditions. Noise-aware MAC (NAMAC) [7] is another example of adaptive MAC protocol. It uses the knowledge of the noise generated by vessels passing close-by the network to choose the best channel to use for communications.

Fully negotiated MAC protocols usually lead to higher packet delivery ratio than other solutions, but at the price of higher delays. Unfortunately the presence of varying asymmetric links, that are typical in the UWSNs [29], can lead fully negotiated MAC protocols to undergo dramatic performance degradation. In general, there is no protocol able to fit all network conditions and application requirements [14]. Even adaptive solutions [7], [8], [9] offer a limited adaptivity, and only to some environmental variables, e.g., propagation delay between neighboring nodes, nodes distance, noise generated by vessels and so on.

In this context, our design choice is radically different. We propose a self-adaptive solution that is able to select the best MAC protocol according to the application requirements and the environmental conditions. While existing adaptive solutions only modify some internal parameters, our solution dynamically select the best MAC protocol among those available in the protocol stack.

III. Policy Engine Overview

We consider a single-hop UWSN in which each node can directly send data to the sink. The communication stack of each node is designed as a Software Defined Communication Stack (SDCS). The SDCS implements several MAC protocols at the datalink layer and can be dynamically configured at runtime, allowing to switch seamlessly from one protocol to another. In particular, we consider an SDCS that is able to self-adapt its configuration at the datalink layer, without any human intervention. The self-adaptation of the datalink layer is managed by a specific software module called *policy engine*. Its goal is to allow the autonomous selection of the most appropriate protocol solution in real-time according to changes in the underwater environment, network topologies, network load, or even to support different application requirements (high packet delivery ratio, short delivery time, etc.).

From a high level perspective, the policy engine works as follows. It receives as input the application requirements, i.e., the performance metric that the MAC protocol has to optimize. For each scenario, the network operator can choose between different metrics to be improved, such as packet delivery ratio (PDR), end-to-end latency, energy consumption, or a proper combination of all of them. Then the policy engine starts to execute a learning algorithm that learns over time which is the MAC protocol that better optimizes the given performance metric according to the environment and network conditions. Since the learning starts without any a priori knowledge about protocols behaviour, the environment conditions are continuously monitored to detect changes that require to switch from a MAC protocol to another.

We design the policy engine as a MAPE-K loop [15], a general architectural paradigm to build self-adaptable systems. It is based on a feedback-control loop that *Monitors* managed system and the surrounding environment, *Analyzes* data to detect changes, *Plans* the reconfiguring actions and *Executes* them. There is also a Knowledge layer that support all the phases (see Figure 1).

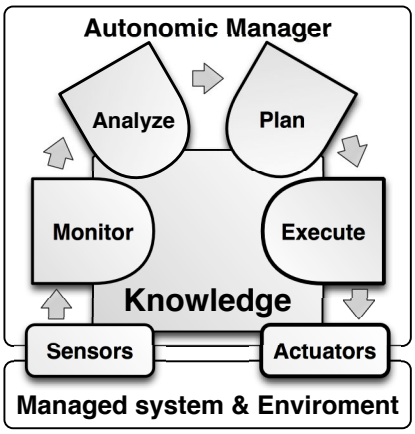

Figure 1: MAPE-K loop.

The MAPE-K loop allows to design the policy engine as an intelligent agent that perceives the system and environment through "sensors" and uses the collected information to compute the actions to be performed on the system itself, to optimize some performance metrics.

In the context of UWSNs, the managed system is made of the set of network nodes, while the environment is made of the acoustic channels interconnecting them. The policy engine is executed on the sink node in a centralized fashion, although other network nodes actively participate to the monitor and

execute phases. Depending on the particular scenario, the policy engine can also run on a master node/cluster head in charge of managing a set of network nodes.

During the network operations, the policy engine iterates through the four different phases:

- *Monitor*: the network is monitored on a continuous time basis. The information required by the policy engine which include network traffic and the links signal to noise ratio is directly added to the packet header by network nodes before the packet transmission and then they are extracted by the sink upon packet reception.
- *Analyze*: the data collected by the monitor phase are taken as input by the analyze phase. This phase is executed periodically every $t_{analyze}$ minutes and performs statistical computation on the raw data collected by the preceding phase. It is based on the online adaptive cumulative sum (Cusum) algorithm [30], an enhanced version of the standard Cusum algorithm [31] specifically designed for non-stationary and high variable scenarios. The main goal of this phase is to determine whether the network or the environment have changed. In such a case, it triggers the plan phase.
- *Plan*: the plan phase computes the optimal MAC protocol to be used in the network according to the performance metrics (and/or their combination) of interest, namely, the end-to-end latency, packet delivery ratio (PDR) and energy per bit. In our policy engine, the *Plan* phase is based on an N-armed bandit problem, a simple but effective variant of the more general Reinforcement Learning framework. It learns online, during network operation, the optimal MAC protocol for each operating scenario, without any a priori knowledge. The plan phase is executed either when it is triggered by the analyze phase, or periodically every t_{plan} minutes. The latter allows the learning algorithm to refine its knowledge about the protocols behavior in each network condition, and it corresponds to a round of the bandit problem.
- *Execute*: the execution phase implements the policy engine plan. In our setting, it is the sink node, by means of suitable control packets broadcast to the network, which command the nodes to change the used MAC protocol. This is implemented as to ensure seamless transition from one protocol to another without network interruption. Given the importance of this phase, the sink node also implements a recovery and alignment procedure to guarantee that all nodes received the control messages to ensure consistent network behaviour by all the network nodes.

IV. Simulation Results

We evaluated the proposed police engine through extensive simulation. In this section we report the simulation results where the policy engine can switch between three MAC protocols, one for each class of contention-based MAC protocols: CSMA for the class of simple negotiated MAC protocols, T-Lohi for the intermediate negotiated and DACAP for the class of fully negotiated MAC protocols. Our aim is to show the ability of the policy engine to dynamically adapt the data link layer protocol to the different network conditions. CSMA, DACAP and T-Lohi protocols and the policy engine have been implemented in SUNSET [12], connected to the Bellhop ray tracing tool [16] via the WOSS interface [32]. Bellhop is used to compute acoustic path loss at a given location, as well as the spatially-varying interference induced by node transmissions. The historical environmental data input to Bellhop refer to an area located off the coast of the Palmaria island (La Spezia, Italy). Sound speed profiles (SSP), bathymetry profiles and information on the type of bottom sediments of the selected area are obtained from the World Ocean Database,[2] from the General Bathymetric Chart of the Oceans (GEBCO)[3] and from the National Geophysical Data Center Deck41 data-base,[4] respectively. In the following we first describe the selected scenarios and protocol parameters settings (Section IV-A). We then discuss the metrics that we have investigated (Section IV-B). We finally report the results of our simulation experiments in Section IV-C).

A. Simulation scenarios and settings

We consider a static single-hop UWSNs with 7 nodes (6 nodes plus the sink) randomly and uniformly placed in a region with surface equal to 2Km^2 at different depths, ranging from 10 to 50m. We simulate a scenario considering an environmental monitoring network where the deployed nodes report to sink the sensed value every $\lambda_1 = 0.033$ seconds on average. Once a particular event is detected by the network nodes, i.e., a sensed value is greater than a given threshold, nodes start transmitting Constant Bit Rate (CBR) data at a higher fixed sample rate of $\lambda_2 = 0.05$ packets per second. When the event expires, nodes continue to report data at lower rate λ_1 according to a Poisson process. The data packet payload size (in bytes) varies in the set $\{128, 2000\}$. The total size of a data packet is given by the payload plus the headers added by the different layers. Besides, the size of generated packets is not constant. The payload of each packet depends on the type of information that the packet contains. The physical header overhead changes according to the data rate but is dominated by a 10ms synchronization preamble. The CSMA, DACAP and T-Lohi MAC headers are 3B long and contain the sender, the destination addresses and the packet type. The header of the policy engine is added to each data packet resulting in an additional overhead of 4B. The size of `RTS`, `CTS` and `WARNING` control packets used by DACAP is 6B, 6B and 3B respectively. The size of the `TONE` used by T-Lohi is set to 3B. In our simulations, we assume BPSK modulation. The carrier frequency is 24500kHz for a bandwidth of 4000Hz. The transmission power is set to 3.3W. The reception power consumption is set to 0.5W. Reception and transmission powers are estimated based on the energy consumption of existing acoustic modems.

[2] http://www.nodc.noaa.gov/OC5/WOA05/pr_woa05.html
[3] http://www.gebco.net
[4] http://www.ngdc.noaa.gov/mgg/geology/deck41.html.

Finally, we set the policy engine analyze and plan phases to be triggered every $t_{analyze} = 60s$ and $t_{plan} = 1800s$.

B. Simulation metrics

Effectiveness and costs of delivering data to the sink are assessed through the investigation of the following metrics:

- *Packet delivery ratio* (PDR) at the sink, defined as the ratio between the packets correctly received by the sink and the packets generated by the nodes;
- *End-to-end latency*, defined as the time between the packet generation and the time of its correct delivery at the sink;
- *Energy per bit*, i.e., the energy consumed by the network to correctly deliver a bit of data to the sink.

C. Simulation results

First, in a preliminary set of experiments, we evaluate the performance of CSMA, T-Lohi and DACAP considering different network conditions. Table I shows simulation results according to the different packet sizes, 128B and 2000B and traffic loads, $\lambda_1 = 0.033$ and $\lambda_2 = 0.05$. As we can see, none of the protocol fits all the scenarios in that their performance vary according to the considered metric, traffic load and packet size. In particular, when the considered traffic load is λ_1 and the packet size is 128B long, CSMA obtains the lowest end-to-end latency and the lowest energy per bit consumption, but at the toll of the lowest PDR. This is because CSMA does not use control packets to access and reserve the channel thus reducing the latency and the energy spent to deliver a packet and, at the same time, increasing the probability of packet collisions. Both T-Lohi and DACAP instead reserve the channel through the use of short control packets that allow to obtain the highest packet delivery ratio but at the cost of higher latency and energy consumed with respect to CSMA. In case of (higher) CBR traffic ($\lambda_2 = 0.05$), all the three protocols have the same PDR (100%) since the traffic load is now constant rather than Poissonian, and the packet generation rate is lower than the capacity. CSMA is the protocol that shows the best performance by saving more energy and by delivering faster the packets than T-Lohi and DACAP since it does not reserve the channel before the actual data packet transmission. When considering 2000B as packet size, Poisson packet generation, and $\lambda_1 = 0.033$ as traffic load, the probability of packet collisions become higher since the transmission time is very high. Therefore DACAP shows the best PDR by properly accessing the channel through the use of `CTS/RTS` control packets. However, the high introduced overhead results in high end-to-end latency and energy consumption. T-Lohi instead shows the best trade-off between delivered data packets and overhead introduced in the network, thus allowing to obtain the lowest energy consumption per bit and a quite high PDR. On the other hand, CSMA delivers a lower number of data packets but with lowest latency with respect to T-Lohi and DACAP. As the traffic switches to CBR with rate $\lambda_2 = 0.05$, DACAP delivers less packets than the other protocols due to the high

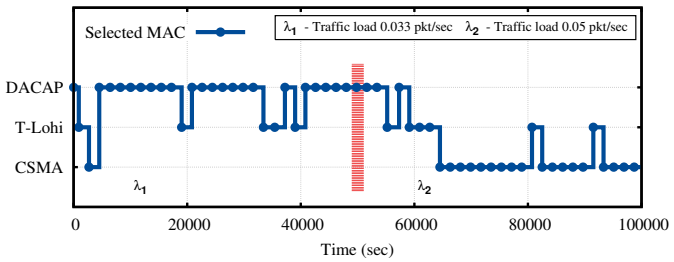

Figure 2: Selected protocols when the policy engine optimizes PDR according to two different traffic loads and fixed packet size.

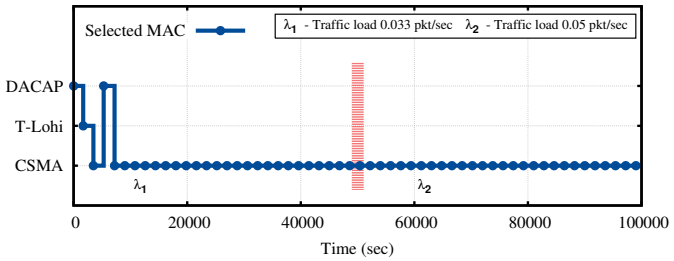

Figure 3: Selected protocols when the policy engine optimizes end-to-end latency according to two different traffic loads and fixed packet size.

packet exchange which also induces high latencies. On the other hand, CSMA shows the lowest latencies and CSMA and T-Lohi deliver the highest number of packets. Even if the use of the `TONE` packets in T-Lohi leads to a higher overhead than CSMA, energy consumption of the two protocols is the same due the small size of the `TONE` packets if compared to the packet data size.

We now turn our attention to the policy engine performance. In the first set of experiments, we vary at runtime the traffic load after 50000 seconds of simulation, switching from $\lambda_1 = 0.033$ to $\lambda_2 = 0.05$ pkt/sec while keeping the packet size constant to 2000B. The experiment is repeated three times, optimizing each time a different performance metric, i.e., PDR, end-to-end latency and energy per bit. Figures 2, 3 and 4 show the protocols selected by the policy engine over time (the vertical bar represents the time instant in which the traffic load changes). As we can see, the policy engine is always able to find the optimal protocol and to adapt its behavior when traffic load changes.

Figures 2, 3 and 4 give us also some useful insights about how the learning algorithm works. At the beginning of each simulation (or after the vertical bar) the policy engine focuses mainly on exploration, to learn how each protocol behaves in the current operating scenario. On the long run, conversely, it exploits the acquired knowledge and focuses on the best performing MAC protocol. Sometimes it also selects suboptimal protocols to refine and improve its knowledge about all available protocols. Note that only few protocol evaluations are needed to identify the optimal protocol. In

Metric	128 Bytes						2000 Bytes					
	$\lambda_1 = 0.033$			$\lambda_2 = 0.05$			$\lambda_1 = 0.033$			$\lambda_2 = 0.05$		
	CSMA	T-Lohi	DACAP	CSMA	T-Lohi	DACAP	CSMA	T-Lohi	DACAP	CSMA	T-Lohi	DACAP
Packet delivery ratio	0.97	**0.99**	**0.99**	**1**	**1**	**1**	0.89	0.94	**0.98**	**1**	**1**	0.9
End-to-end latency (sec.)	**1.9**	4.5	6.5	**1.9**	3.9	5.5	**29.1**	36.1	42.9	**16.8**	18.9	112.9
Energy per bit ($J/b \cdot 10^{-3}$)	**3.4**	3.5	3.8	**3.4**	3.5	3.8	3.5	**3.3**	3.4	**3.3**	**3.3**	3.6

Table I: Simulation results with standalone protocols - 128 and 2000 Bytes.

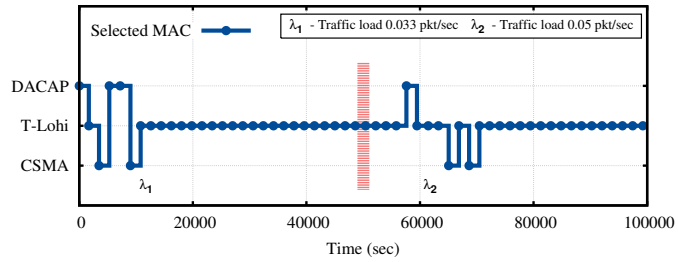

Figure 4: Selected protocols when the policy engine optimizes energy per bit according to two different traffic loads and fixed packet size.

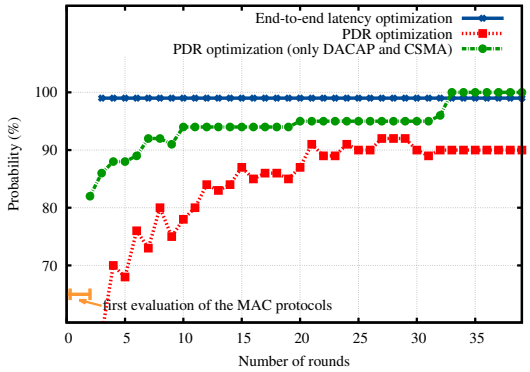

Figure 5: Probability that the policy engine has learnt the optimal protocol as function of the number of rounds of the N-armed bandit problem.

general, the performance of the learning algorithm heavily depends on the relative performance of the available MAC protocols. Basically, greater is this difference, easier is for the policy engine to learn which is the best protocol for a given scenario. This is somehow expected and perfectly shown by our experimental results in Figure 5. In this figure we plot the probability that the policy engine has learnt the optimal MAC protocol, as function of the number of rounds of the N-armed bandit problem (each curve starts only when all protocols have been evaluated at least once). The blue line with cross points corresponds to the optimization of the end-to-end latency, the red dotted line with square points and the green dotted line with round point correspond to the optimization of the PDR. In the latter scenario, the SDCS does not implement the T-Lohi protocol and the PDR is therefore optimized considering only CSMA and DACAP. As we can see, very few rounds are needed to reach an accuracy greater than 80% in all cases, but if we focus on latency optimization a single evaluation of each protocol is sufficient to achieve an accuracy close to 100%. If we compare these results with those in Table I, we can easily see the reason is that latencies are quite different among protocols. The green dotted line with round point confirms this trend: if we consider only DACAP and CSMA protocols and we optimize the PDR, we can significantly improve the learning performances.

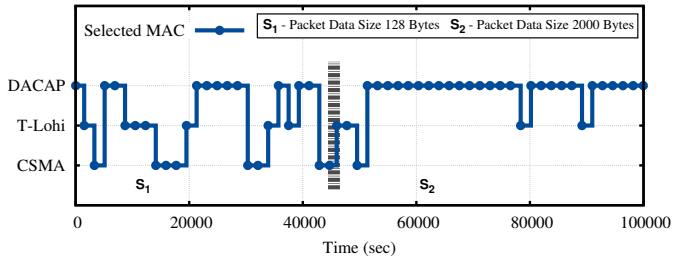

Figure 6: Selected protocols when the policy engine optimizes PDR according to two different packet sizes and fixed traffic load.

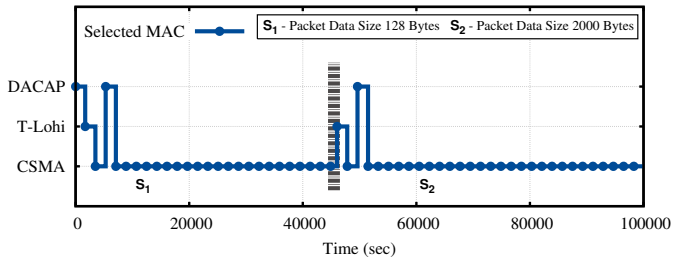

Figure 7: Selected protocols when the policy engine optimizes end-to-end latency according to two different packet sizes and fixed traffic load.

In the second set of experiments we vary at runtime the packet size, switching from 128B to 2000B, and we keep constant the traffic load to $\lambda = 0.033$. Again, the experiment is repeated three times optimizing each time a different performance metric. Results are shown in Figures 6, 7 and 8 (the vertical bar represents the time instant in which the packet size changes). As we can see, the policy engine is always able to find the optimal protocol. Note that when the optimization focuses on PDR and the packet size is 128B long (first part of Figure 6), the policy engine uses almost uniformly all MAC

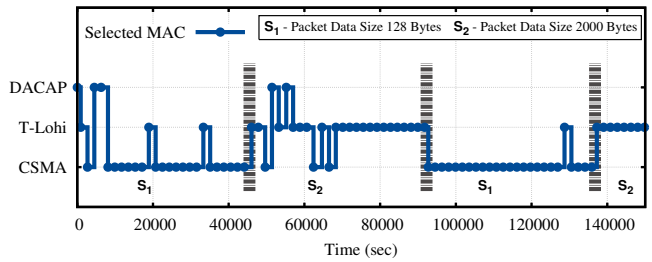

Figure 8: Selected protocols when the policy engine optimizes energy per bit according to two different packet sizes and fixed traffic load.

protocols. This is expected, since in this operating scenario all the considered protocols obtain the same performance. However, when the packet size switches to 2000B, the policy engine quickly reacts and reconfigures the MAC layer with the optimal DACAP protocol. It is worth to be noted that if the packet size switches again to 128B there is no need to restart learning from scratch. Conversely, the policy engine can leverage on the knowledge already acquired during the first 45000 seconds of network operation. This is perfectly shown in Figure 8, in which we change packet size multiple times. Once a change in the network is detected, the policy engine is able to directly switch to the optimal MAC protocol.

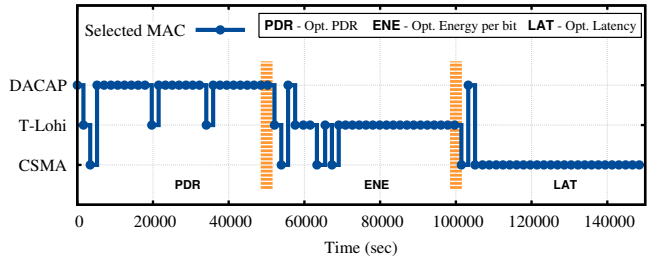

Figure 9: Selected protocols when the application scenario requires the optimization of different metrics at runtime according to a fixed packet size and traffic load.

Finally, we investigate the policy engine behavior when the application scenario requires the optimization of different metrics at runtime. Therefore, we keep constant the traffic load and the packet size to $\lambda = 0.033$ and 2000B and we change the optimization metric every 50000 seconds. Results are shown in Figure 9. This experiment confirms the effectiveness of the proposed learning strategy. The policy engine is always able to quickly react to changes and to select the optimal protocol according to the considered metric.

V. Conclusion

In this paper we considered an UWSN in which the network stack is implemented as a modular Software Defined Communication Stack (SDCS). We proposed the design and evaluation of a novel component of the SDCS, termed policy engine, to dynamically and autonomously adapt each layer of the network stack selecting the optimal protocol according to network condition and application requirements. As a proof of concept, we focused on a policy engine that adapts the datalink layer selecting the optimal MAC protocol. Our work is motivated by several studies that showed that different MAC protocols are needed to face different network configurations, environmental conditions and traffic loads. The policy engine we proposed is based on a feedback control loop that monitors the environment, analyzes data to detect changes, computes the new MAC protocol and reconfigure the datalink layer. The core of the policy engine is a learning algorithm that is able to learn, without any a priori knowledge, which is the best protocol for each operating scenario. Experimental results validate our solution and show that the policy engine is always able to learn the optimal MAC protocol and to quickly react to changes in the network environment.

References

[1] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: Applications, advances and challenges," *Philosophical Transactions of the Royal Society A*, vol. 370, pp. 158–175, August 2 2012.

[2] T. Melodia, H. Kulhandjian, L.-C. Kuo, and E. Demirors, "Advances in underwater acoustic networking," in *Mobile Ad Hoc Networking: Cutting Edge Directions*, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, Eds. Hoboken, NJ: John Wiley & Sons, Inc., March 5 2013, ch. 23, pp. 804–852.

[3] S. Basagni, C. Petrioli, R. Petroccia, and D. Spaccini, "CARP: A channel-aware routing protocol for underwater acoustic wireless networks," *Ad Hoc Networks*, vol. 34, pp. 92–94, 2015.

[4] B. Tomasi, G. Toso, P. Casari, and M. Zorzi, "Impact of time-varying underwater acoustic channels on the performance of routing protocols," *IEEE Journal of Oceanic Engineering*, vol. 38, no. 4, pp. 772–784, October 2013.

[5] P. Casari, D. Spaccini, G. Toso, B. Tomasi, R. Petroccia, C. Petrioli, and M. Zorzi, "A study on channel dynamics representation and its effects on the performance of routing in underwater networks," in *Proceedings of the IEEE Asilomar Conference on Signals, Systems and Computers 2012*, Pacific Grove, CA, USA, November 4–7 2012, pp. 1536–1540.

[6] C. Petrioli, R. Petroccia, and J. Potter, "Performance evaluation of underwater mac protocols: From simulation to at-sea testing," in *Proceedings of IEEE OCEANS 2011*, Santander, Spain, June, 6–9 2011, pp. 1–10.

[7] L. Pescosolido, C. Petrioli, and L. Picari, "A multi-band noise-aware MAC protocol for underwater acoustic sensor networks," in *Proceedings of IEEE WiMob 2013*, Lyon, France, October 7–9 2013, pp. 530–537.

[8] S. Azad, P. Casari, K. T. Hasan, and M. Zorzi, "MACA-APT: A MACA-based Adaptive Packet Train Transmission Protocol for Underwater Acoustic Networks," in *Proceedings of the International Conference on Underwater Networks & Systems*, ser. WUWNET '14. New York, NY, USA: ACM, 2014, pp. 18:1–18:5.

[9] T. H. Nguyen, S.-Y. Shin, and S.-H. Park, "Adaptive Energy Reservation MAC Protocol for Underwater Acoustic Sensor Networks," in *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, vol. 2, Dec 2008, pp. 670–675.

[10] X. Guo, M. Frater, and M. Ryan, "An adaptive propagation-delay-tolerant MAC protocol for underwater acoustic sensor networks," *IEEE Journal of Oceanic Engineering*, vol. 34, no. 2, pp. 170–180, April 2009.

[11] "SUNRISE: Sensing, monitoring and actuating on the UNderwater world through a federated Research InfraStructure Extending the Future Internet," http://fp7-sunrise.eu.

[12] C. Petrioli, R. Petroccia, J. R. Potter, and D. Spaccini, "The SUNSET framework for simulation, emulation and at-sea testing of underwater wireless sensor networks," *Ad Hoc Networks*, vol. 34, pp. 224–238, 2015.

[13] S. Basagni, C. Petrioli, R. Petroccia, and M. Stojanovic, "Optimized packet size selection in underwater WSN communications," *IEEE Journal of Oceanic Engineering*, vol. 37, no. 3, pp. 321–337, July 2012.

[14] C. Petrioli, R. Petroccia, and M. Stojanovic, "A comparative performance evaluation of MAC protocols for underwater sensor networks." in *Proceedings of MTS/IEEE OCEANS 2008*, Quebec City, Quebec, Canada, September 15–18 2008, pp. 1–10.

[15] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[16] M. B. Porter, "The BELLHOP manual and user's guide: Preliminary draft," La Jolla, CA, 2011, heat, Light, and Sound Research, Inc.

[17] A. S. Tanenbaum and D. Wetherall, *Computer Networks*, 5th ed. Upper Saddle River, NJ: Prentice Hall PTR, 2010.

[18] A. Syed, W. Ye, and J. Heidemann, "Comparison and evaluation of the T-Lohi MAC for underwater acoustic sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 9, pp. 1731–1743, December 2008.

[19] B. Peleato and M. Stojanovic, "Distance aware collision avoidance protocol for ad-hoc underwater acoustic sensor networks." *IEEE Communications Letters*, vol. 11, no. 12, pp. 1025–1027, December 2007.

[20] N. Chirdchoo, W.-S. Soh, and K. C. Chua, "ALOHA-based MAC protocols with collision avoidance for underwater acoustic networks," in *Proceedings of the* 26*th IEEE International Conference on Computer Communications (IEEE INFOCOM 2007)*, Anchorage , Alaska , USA, May 6–12 2007, pp. 2271–2275.

[21] H.-H. Ng, W.-S. Soh, and M. Motani, "A bidirectional-concurrent mac protocol with packet bursting for underwater acoustic networks," *IEEE Journal of Oceanic Engineering*, vol. 38, no. 3, pp. 547–565, July 2013.

[22] ——, "MACA-U: A Media Access Protocol for Underwater Acoustic Networks," in *Proceedings of the Global Telecommunications Conference (IEEE GLOBECOM 2008)*, November 2008, pp. 1–5.

[23] W. Zhang, Z. Qin, J. Xin, L. Wang, M. Zhu, L. Sun, and L. Shu, "UPMAC: A localized load-adaptive MAC protocol for underwater acoustic networks," in *Proceedings of the* 23*rd International Conference on Computer Communications and Networks, ICCCN 2014*, August 2014, pp. 1–8.

[24] N. Chirdchoo, W.-S. Soh, and K.-C. Chua, "RIPT: A receiver-initiated reservation-based protocol for underwater acoustic networks," *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Underwater Wireless Communications and Networks*, vol. 26, no. 9, pp. 1744–1753, December 2008.

[25] A. Syed, W. Ye, and J. Heidemann, "Understanding spatio-temporal uncertainty in medium access with ALOHA protocols," in *Proceedings of ACM WUWNet 2007*, Montréal, Quebec, Canada, September 14 2007, pp. 41–48.

[26] M. Molins and M. Stojanovic, "Slotted FAMA: A MAC protocol for underwater acoustic networks," in *Proceedings of MTS/IEEE OCEANS 2006*, Singapore, 2006, pp. 1–7.

[27] S. Han, Y. Noh, U. Lee, and M. Gerla, "M-FAMA: A Multi-session MAC Protocol for Reliable Underwater Acoustic Streams," in *Proceedings of the* 32*th IEEE International Conference on Computer Communications (IEEE INFOCOM 2013)*, April 14–19 2013, pp. 655–673.

[28] Y. Noh, U. Lee, S. Han, P. Wang, D. Torres, J. Kim, and M. Gerla, "Dots: A propagation delay-aware opportunistic mac protocol for mobile underwater networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 4, pp. 766–782, April 2014.

[29] S. Basagni, C. Petrioli, R. Petroccia, and D. Spaccini, "Channel replay-based performance evaluation of protocols for underwater routing," in *Proceedings of MTS/IEEE OCEANS 2014*, St. John's, Canada, September, 14–19 2014.

[30] S. Casolari, S. Tosi, and F. Lo Presti, "An adaptive model for online detection of relevant state changes in internet-based systems," *Perform. Eval.*, vol. 69, no. 5, pp. 206–226, May 2012.

[31] D. Montgomery, *Introduction to Statistical Quality Control*. Wiley, 2008.

[32] F. Guerra, P. Casari, and M. Zorzi, "World ocean simulation system (WOSS): A simulation tool for underwater networks with realistic propagation modeling," in *Proceedings of ACM WUWNet 2009*, Berkeley, CA, 3 November 2009, pp. 1–8.