MDOTS: A Multi-session Delay-aware Opportunistic MAC Protocol for Underwater Sensor Networks

Seongwon Han, Youngtae Noh, Max Ohlendorf and Mario Gerla
Department of Computer Science, UCLA
{swhan, ytnoh, maxo, gerla}@cs.ucla.edu

I. PROBLEM DESCRIPTION

Underwater acoustic networking is a developing technology with the goal of monitoring and exploring the Earth's oceans. While terrestrial wireless networks use radio waves, underwater networks use acoustic waves as their means of communication. Underwater Acoustic Sensor Network (UW-ASN) protocols must work around the long propagation delays and high bit error rates which differentiate acoustic from terrestrial network links. To date, many underwater Media Access Control (MAC) protocols have been proposed to alleviate limitations caused by severely limited bandwidth and long propagation latency of acoustic waves. Most proposed underwater MAC protocols are based on either carrier sense multiple access (CSMA) or code division multiple access (CDMA) because time division multiple access (TDMA) requires strict time synchronization amongst all deployed nodes in the entire network. Since CSMA-based protocols can easily support network dynamics such as node mobility, failure, join, and leave, CSMA-based protocol design has gained much interest recently. It is a known fact that multiple concurrent transmissions of CSMA-based protocols can boost channel utilization. To increase chances of concurrent transmissions, *spatial reuse* involves two or more pairs of nodes participating in transmission sessions at the same point in time. *Temporal reuse* involves a single node participating in multiple transmission sessions with one or more other nodes at the same point in time. Therefore, many proposed CSMA-based protocols are mainly focused on spatial or temporal reuse to increase throughput while trying to avoid collisions [2][3].

II. AN OVERVIEW OF MULTI-SESSION DOTS

An UW-ASN typically consists of a large number of underwater mobile/anchored sensor nodes, equipped with a low bandwidth acoustic modem and various sensors such as pressure and seismic gauges, thermometer, etc. In particular, we consider a SEA Swarm (Sensor Equipped Aquatic Swarm) architecture [4], illustrated in Fig. 1, which involves a large number of acoustic mobile sensor nodes are deployed to the region of interest to compose a swarm that drifts as a group with the ocean current. A SEA Swarm architecture is configured with 3D geographic routing in which all sensor nodes float regardless of the depth of the water. Each sensor node (e.g., Drogues [1]) monitors local underwater phenomena and reports time critical information to the sonobuoys using acoustic multi-hopping. Delay-aware Opportunistic Transmission Scheduling (DOTS [5]) is a recently proposed MAC protocol for UW-ASNs. It has been recently shown that DOTS achieves better throughput by harnessing

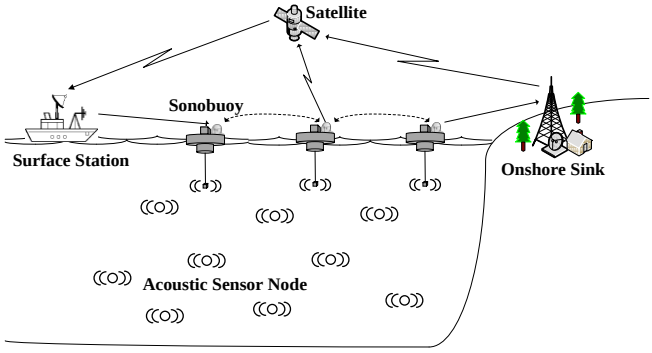

Fig. 1 SEA Swarm Architecture

spatial reuse which relies on maintaining a delay map by passively observing transmissions from all neighbors.

In this paper, we propose a new underwater MAC protocol, Multi-session DOTS (MDOTS), which redesigns DOTS to enable temporal reuse. Although DOTS achieves a fair amount of channel utilization, it is unable to capitalize on temporal reuse because of its single session nature. Rather than each node keeping track of the state of a single session, in MDOTS each node maintains a list of concurrent transmission sessions. Each node has a list of its current sessions, and each session independently keeps track of its own state information.

State information for each session includes whether the partner node is a client or a server in the session, the current status of the partner node with respect to the session (statuses include transmitting RTS/CTS/DATA/ACK, receiving RTS/CTS/DATA/ACK, waiting for CTS/DATA/ACK, and backoff), node IDs for both of the participating nodes (one will always be the parent node's ID), and timer information for the session. Because all the information related to a particular session is maintained independently in its own state within a node, nodes can handle multiple concurrent transmission sessions. Whenever a node creates a session or overhears an RTS or CTS transmission, the node calculates timing information for all impending transmissions and receptions of each message in the transmission. The node creates entries in its delay map for each of the calculated times.

The maximum number of simultaneous sessions each node can have is a configurable value, and the performance of the protocol is heavily dependent on this value. Unless a node's current number of client session does not reach at capacity, a new client session is created whenever the network layer has a packet for the node to send. When the client session is created, the node creates delay map entries associated with the new

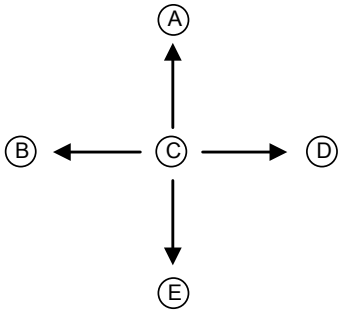

Fig. 2: Simulation Topology 4 Sinks 1 Sender

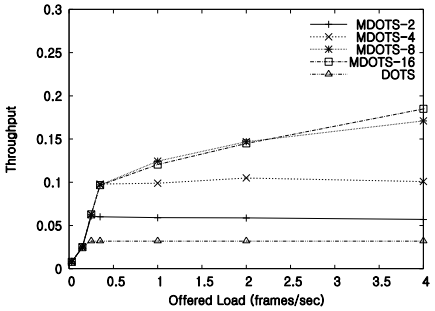

Fig. 3: Small data packet size (512*bytes*), short transmission range (750*m*)

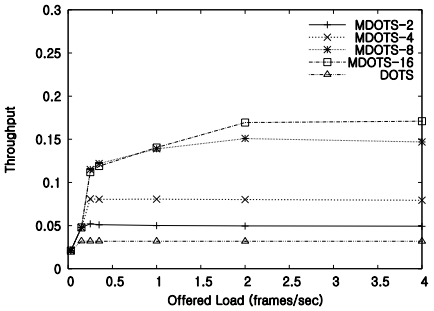

Fig. 4: Large data packet size (1024*bytes*), long transmission range (1500*m*)

session. Then it checks to see if these new entries are compatible with the node's existing delay map entries. If a collision is detected, the state's timing information is reset, and it will reattempt the transmission session after a back-off period. Otherwise, the node will send the RTS packet for this session and wait to receive a CTS packet. When the CTS packet arrives, the node sends a DATA packet and waits for an ACK packet to arrive. When the ACK packet arrives, the session is complete and the state can be taken down.

A new server session is created whenever a node receives a RTS. When the server session is created, the node creates delay map entries associated with the new session. It checks to see if these new entries are compatible with the node's existing delay map entries. If a collision is detected, the state is taken down and the node waits for the client to resend its RTS in order to try again with another server state. Otherwise if no collision is detected, the node will send the CTS packet for this session and wait to receive the DATA packet from the client. When the DATA packet arrives, the node retrieves the message from the packet and sends it up to the network layer. Then the node sends an ACK packet. After the ACK packet is sent, the session is complete the state is taken down. The cases are summarized as follows:

- When a node creates a client session, it calculates its own RTS and DATA transmission times and its own CTS and ACK reception times. It also calculates the server node's CTS and ACK transmission times and its RTS and DATA reception times.
- When a node creates a server session, it calculates its own CTS and ACK transmission times and its own DATA reception time. It also calculates the client node's DATA transmission time and CTS and ACK reception times.
- When a node overhears a RTS or CTS, it calculates the client node's CTS and ACK reception times and the server node's RTS and DATA reception times.

III. Evaluation

Fig. 2 shows the simulation topology for 4 sinks 1 sender. Fig. 3 shows the results when data packet size is 512*bytes* and transmission range is 750*m*. Fig. 4 shows results when data packet size is 1024*bytes* and transmission range is 1500*m*. In both cases, MDOTS-8 and MDOTS-16 dominates over all other flavors. For loads of 4 *packets/sec*, MDOTS-8 and MDOTS-16 outperform DOTS protocol by 200-500% in both cases. The 4 sinks 1 sender topology is similar in layout to the 4 senders 1 sink topology, but with the clients/servers switched. Instead of four outer nodes all trying to start sessions with a central node, 4 sinks 1 sender has the central node try to start sessions with the 4 outer nodes.

MDOTS protocols with high session capacities can take better advantage of temporal reuse in a 4 sinks 1 sender topology. More sessions means the central node has more opportunities to utilize temporal reuse and schedule in new transmission sessions. Because of the central node's full knowledge of the session of the channel, any successfully scheduled transmission session is guaranteed to not have any collisions. Higher session capacities leads to more aggressive scheduling of sessions, and the aggressive scheduling leads to more throughput with no increased risk of collisions.

VI. Conclusions

In this paper, we have presented Multi-session DOTS (MDOTS), an efficient MAC protocol for underwater acoustic networks. MDOTS utilizes temporal/spatial reuse to achieve better channel utilization. Simulation results show that MDOTS significantly outperforms DOTS and other current underwater MAC protocols.

There are many areas of possible future research on MDOTS. Our simulation results show that the channel utilization of MDOTS depends on the maximum number of client sessions. Simulations with more congestion (large packet size and short distance between nodes) perform better with smaller numbers of client sessions. Simulations with lower congestion (small packet size and long distance between nodes) perform better with larger numbers of client sessions. Research on how the optimal number of client sessions changes with varying data sizes will be our future work.

References

[1] J. Jaffe and C. Schurgers. "Sensor networks of freely drifting autonomous underwater explorers". In WUWNet '06, New York, NY, USA, 2006.

[2] N. Chirdchoo, W. seng Soh, and K. Chua, "RIPT: A Receiver-Initiated Reservation-Based Protocol for Underwater Acoustic Networks," Selected Areas in Communications, IEEE Journal on, vol. 26, no. 9, pp. 1744–1753, December 2008.

[3] P. Xie and J. H. Cui, "R-MAC An Energy-Efficient MAC Protocol for Underwater Sensor Networks," in Proc. of the IEEE WASA'07, August 2007, pp. 187 – 198.

[4] U. Lee, J. Kong, J. S. Park, E. Magistretti, and M. Gerla, "Time-Critical Underwater Sensor Diffusion with No Proactive Exchanges and Negligible Reactive Floods," in Proceedings of ISCC'06, June 2006.

[5] Y. Noh, P. Wang, U. Lee, D. Torres, and M. Gerla, "DOTS: Delay-aware Opportunistic Transmission Scheduling Algorithm", submitted to The 18th IEEE International Conference on Network Protocols (ICNP'10), April2010