OCControl: A Novel Congestion Protocol for Oceanic Delay Tolerant Networks

LI Bing-xin
Department of Science and Technology
Wuhan Digital Engineering Institute
Wuhan, China

ZHANG Xu-dong
Department of Science and Technology
Wuhan Digital Engineering Institute
Wuhan, China

***Abstract*—Because of oceanic surface's complex environments, such as varied weather conditions, and its easy reflection and refraction of wireless signals, Oceanic Wireless Communication Networks (OWCN) suffer low link quality and intermittent connectivity, which in turn disable the stability of end-to-end communication path. The above characteristics make oceanic wireless communication network a de facto Delay/Disruption Tolerant Network (DTN). Although numerous existing DTN congestion control schemes have been proposed, most of them assumed that link congestion rarely occurs, which is not true for oceanic wireless communication networks, because oceanic wireless communication networks are not typical DTN. For oceanic wireless communication networks, their transmission bottleneck lies in network links instead of network nodes. Therefore, we design a novel congestion control protocol for oceanic wireless communication networks, called OCControl, which borrow the idea of back-pressure routing. The experimental results have shown that OCControl effectively improves data transmission rate and greatly reduce transmission delay, compared with existing TCP/IP protocols.**

Keywords—DTN, congestion control, oceanic wireless networks

I. INTRODUCTION

Recently, along with the rise of the communication demand under various extreme conditions, such as deep-space communication[1,2,3], maritime communication[4,5], underwater communication[6,7], and communication in remote areas[8,9,10], DTN[10], as a new routing model for the network scenarios with intermittent connectivity, has become a hot topic in network community.

The TCP/IP protocols adopted by Internet assume that the underlying physical network has a relatively stable transmission path at any time, through which data can be reliably transferred. However, due to link intermittence, DTN does not exist a stable transmission path, which causes TCP failure at the DTN scenarios. To this end, network community has proposed a large number of transfer and routing protocols especially for DTN. However, those proposed DTN protocols are not compatible with the existing application layer protocols designed and implemented for the TCP/IP protocols.

Due to the wide use of TCP/IP protocol, almost all Internet applications are based on the TCP/IP socket to write.Apparently, the application layer protocol and new DTN network protocol is not compatible, and it is still not realistic which that is asked to rewrite all the existing application layer protocol to transplant with the DTN network scenario.It would require to design a set of solution that is compatible with the existing application layer protocol, and is suitable for the DTN network scenario.

To this end, we specially designed a TCP compatible DTN overlay layer protocol, OCControl, which can still use the TCP/IP protocols to transfer application data in a stable connected network scenario, and can still able to transfer the data at its best effort in a DTN scenario without modifying the existing application-layer protocols.

The basic idea of OCControl is to supplement the TCP/IP protocol to transmit data rather than completely replace them. When a data-transmission request arrives, it first attempts to transfer the data through TCP/IP. When the TCP/IP service is not desirable due to the intermittent connectivity, the data transmission responsibility will be transferred to a DTN protocol. Then, the data will be delivered to the destination node through a custody-transfer mode.

To verify OCControl's performance, we implemented it based on an open source framework, ION[11,12], and compared its performance with that of TCP on a testbed, consisting of over 30 oceanic nodes, each of which has installed the open source network simulation tools, CORE[13,14] and EMANE[15,16,17], under numerous network scenarios. Experimental results have shown that the OCControl can effectively improve the data transmission rate, and greatly reduce the transmission delay, compared to the existing TCP/IP protocols. Therefore, OCControl is potential practical solution to be compatible with existing TCP-based application protocols and adapted to the DTN scenarios with intermittent connectivity.

The rest of this paper is organized as follows. Section II presents the design scheme of TC-DTN. Section III describes TC-DTN's implementation details based on ION. Section IV shows the experimental scheme based on the open-sourced network simulation tools, CORE and EMANE, and analyzes the experiment results. Section V concludes this paper.

II. OCCONTROL DESIGN PRINCIPLE

The congestion-control protocol used in the Internet TCP/IP protocol stack cannot achieve a desirable performance in

OWCN due to the unstable wireless links, which show the intermittent characteristics.The link instability greatly increases the uncertainty of the end-to-end transmission delay, which in turn arouses numerous issues for the TCP protocol.

First of all, the TCP three-handshake protocol, which is used to establish a connection, might fail because of the highly uncertain end-to-end transmission delay. The underlying reason is that the Round-Trip Time (RTT) will be hard to estimate due to this highly uncertain end-to-end delay, while the TCP three-handshake protocol need this RTT to estimate the timeout time, based on which the re-transmission will be issued. The RTT estimated in the case of stable links is usually much smaller than the one estimated in the case of unstable links. Thus, when the link are unstable, the SYN packet, which is used to synchronize the two ends of the TCP protocol during the connection-establishing stage, will experience timeout event, which in turn results in the re-transmission of the SYN packet. Eventually, it will lead to connect failure.

Second, the reliable transmission of TCP is accomplished through the feed-backed ACK packets issued from the destination upon its reception of data packets. To avoid the loss of data and ACK packets, which will result in the failure of the reliable transmission, TCP sets the timeout value for the re-transmission timer through estimating the RTT. Similar to the previous analysis, the RTT estimation based on the unstable links is a big challenge.

Finally, no explicit mechanism exists in TCP, which can directly transfer the transmission-congestion status information. In fact, in TCP, congestion control is achieved through ACK packets. If the TCP source receives three repeated ACK packets or its re-transmission timer expires, it will determine that a congestion has occurred and will reduce transmission rate accordingly. However, the TCP congestion control mechanism might fail to take effect in OWCN, because the congestion status cannot be timely obtained due to the difficulty to receive the ACK packets, which originated from the unstable link and the long delay incurred by the ultra-long transmission range in OWCN. Therefore, in OWCN, it is hard for the TCP source to make accurate and timely response to congestion.

Therefore, the end-to-end congestion-control mechanism in TCP is not suitable for OWCN, which contains unstable links and longer propagation delay. Thus, OWCN should adopt a hop-by-hop congestion-control mechanism, i.e., each node in OWCN dynamically adjusts its transmission rate according to the network-congestion status.

However, it is not appropriate to directly apply those cache-based congestion-control algorithms, which are adopted by DTN, to OWCN. A typical DTN assumes that the cache capacity of a node is limited, while the link stability and bandwidth are usually not a problem. On the contrary, in an OWCN, a node's cache capacity is not the bottleneck, which lies in the link stability and bandwidth. In a typical DTN, since the node's cache capacity is the bottleneck, the major concern for a DTN congestion-control algorithm is to avoid the nodes' cache overflow. Thus, the focus of those congestion-control algorithms is the node-cache management. As for an OWCN, the main concern becomes the link stability and bandwidth.

Considering OWCN's characteristics, an appropriate congestion-control scheme is the hop-by-hop back-pressure based congestion-control scheme. The main idea is as follow. Each node calculates the difference between the length of its outgoing buffer-queue and that of its next-hop node, and estimates the corresponding link/path congestion status accordingly. For each node, the larger the difference, the severer the congestion, and vice versa. In the former case, a node should reduce the transmission rate to avoid or mitigate the congestion, while, in the latter case, it should increase the transmission rate to improve the link/path utilization.

The main advantages of the backpressure-based congestion control scheme lies in the fact that it not only can theoretical prove its optimization in terms of maximizing network throughput, but also can achieve a good effect in real-world scenarios. Moreover, it has been theoretically proved that the backpressure-based method has a good robustness in the time-varying network environment.

III. OCCONTROL DESIGN

OWCN differs significantly from Internet in the network connectivity due to its intermittent links. Therefore, the TCP protocol used for Internet is not suitable for OWCN. Meanwhile, OWCN is also different from a typical DTN, where a link's connectivity depends on the corresponding nodes' movement, in that the link connectivity is independent from the node movement and the link state itself is hard to predict. In addition, in an OWCN, its links does not exist a large delay, and the nodes' energy consumption is not an issue.

In an OWCN, the bottleneck of its network throughput mostly lies in the link availability. When a link is close to saturation, if a node still sends data through the link at a higher rate, the data will be accumulated at the node rapidly because those data have not been acknowledged. Although an oceanic node has a larger cache capacity, the rapidly accumulated data will incur a longer transmission delay, re-transmission, or even transmission failure. Moreover, a node's storage capacity is limited after all. When a node's outgoing queue becomes too crowded to accept new data, it will be unable to receive data from upstream nodes, which will cause re-transmission or even transmission failure of upstream nodes.

Considering the above discussion and the OWCN's characteristics, we can use the length of a node's outgoing queue as an indicator for congestion. On one hand, if the total cache size is known, the length of the outgoing queue can explicitly reflect a node's remaining cache capacity, i.e., its capacity to receive new data. The shorter the queue length, the more data a node can accept. On the other hand, the queue length can implicitly reflect a node's ability to forwarding data. The short outgoing-queue size might be a result from a node's ability to quickly forwarding data. In that case, if a lot of data from its upstream nodes needs to be sent, they should increase their transmission rates to transfer more data to the node with strong forwarding ability. On the contrary, the long queue length means that a congestion will occur or even have already happened. Thus, injecting more data will exacerbate the congestion. Therefore, even though the upstream nodes have a lot data to be sent, they should still reduce the transmission rate

to avoid worsen the downstream node's congestion.

Therefore, the queue length can reflect a node's abilities to transmit and accommodate data. Hence, based on the downstream nodes' queue-length information, congestion can be avoided or mitigated through controlling upstream nodes' transmission rates. For example, suppose that a message passes through three nodes, A, B, and C, in turn. When node B forwards the message to node C, it needs to adjust its transmission rate first according to node C's queue length. If node C's queue length is larger than a given threshold, it implies that a large number of packets to be forwarded have been accumulated at node C. In that case, if node B still forwards data to node C at the same rate, node C's congestion will be aggravated, which may incur unnecessary re-transmission and reduce channel utilization.

Therefore, similar to TCP, it is better for an upstream node, such as node B, to multiplicative reduce transmission rate in the case of the occurrence of congestion at downstream nodes, and linearly increase the rate otherwise. Moreover, the decrease of the transmission rate at node B will gradually accumulate packets at node B's outgoing queue, which will eventually make its queue length large than the threshold. Again, node B's upstream nodes, such as node A, may receive this congestion indicator, and reduce their transmission rate accordingly. In this way, the congestion information will be feed-backed to upstream nodes hop-by-hop until it reach the source node.

Based on the above analysis, we can formally propose our congestion-control algorithm for OWCN, OCControl. In this algorithm, each node maintains a message-forwarding queue. When it receives a message, if it is the destination node, the message will be transferred to the application layer; otherwise, the message will be put into the queue, and will be forwarded to the next-hop node in the FIFO manner.

To calculate the queue-length difference between upstream and downstream nodes, each node also records its neighbor nodes' queue-length information. When a node forwards a message, it will attach its queue-length information. Through this, when other nodes receive this message, they will be able to extract their neighbor nodes' queue-length information.

Since the routing path required by OCControl is provided by the underlying routing protocol, each node knows it's the next-hop nodes in advance. Let Q_i denotes node *i*'s transmission queue. Whenever node *i* receives or transmits a message, Q_i will be updated. Before node *i* transmits a message, it will calculate its difference from its next-hop node *j* in term of queue length, as shown in Equation (1).

$$QD_{ij} = |Q_i| - |Q_j| \quad (1)$$

In Equation (1), $|Q_i|$ represents the length of node i's transmission queue, and QD_{ij} denotes the queue-length difference between nodes *i* and *j*, which roughly reflects the corresponding link's congested status. A smaller value of QD_{ij} reflects that node *j* is more congested than node *i*. Keep transmitting data to node j will exacerbate its congestion status. Thus, node *i* should reduce its transmission rate properly. On the contrary, a larger value of QD_{ij} means that node *j* is less congested than node *i*. Hence, node *i* should increase its transmission rate accordingly, so that its own queue length can be reduced to avoid false alarm about congestion to its upstream nodes.

Therefore, node *i* can adjust its transmission rate with Additive Increase and Multiplicative Decrease (AIMD), similar to TCP, as shown in Algorithm 1.

Algorithm 1 Transmission Rate Control(*i*)

1. $QD_{ij} = |Q_i| - |Q_j|$
3. if QD_{ij} > QUEUE THRESH
4. rate = rate/β ;
5. else
6. rate = rate + α ;

Once node *i* receives application data, it will attach the OCControl protocol header and push them into the transmission queue, as shown in Algorithm 2.

Algorithm 2

On receiving packet P from local application

1. Encapsulate P with OCControl header;
2. Enqueue P into Q_i(F);

When node *i* receives data from its neighbor node *j*, if it identifies itself as the destination node, it will remove OCControl header and deliver the remaining data to the upper-layered application. Otherwise, if node *j* is the next-hop node, node *i* will update QD_{ij} accordingly as shown in Algorithm 3.

Alogrithm 3

On reception of packet P from node *j*

1. **if** *i* is the destination
2. De-capsulate OCControl Header;
3. Deliver data to the application;
4. **else**
5. **if** node *j* is the next-hop
6. $QD_{ij} \leftarrow |Q_i| - |Q_j|$;
7. **else**
8. Enqueue P into Q_i;

OCControl applies the principle of back-pressure mechanism to transfer the congestion pressure on transmission queue to the transmission-rate adjustment, so as to relieve the congestion. Suppose a flow *f* flows through X, Y, and Z in turn. When node Z's queue length decreases,

it can transmit data to its next hop faster, which will increase QD_{YZ}. According to the back-pressure principle, Y will increase its transmission rate to Z. Along with the increment of Y's transmission rate, its queue length will also be reduced. Once Y's queue length reduces to a certain value, the subsequent increase of QD_{XY} will cause X to increase its transmission rate. In the end, this forwarding pressure will feed-backed to the source. On the contrary, when node Z's next-hop becomes congested, Z's transmission queue will increase, which in turn will reduce QD_{YZ}. Eventually, this reverse pressure will lead to the source queue increases, which in turn reduce the transmission rate at the source node.

IV. SYSTEM IMPLEMENTATION AND EXPERIMENTAL ANALYSIS

A. OCControl Implementation

We implement OCControl based on an open source DTN framework, ION. The architecture of OCControl is shown in Fig.1, where rectangles denote network protocols, while rounded rectangles represent daemon processes.

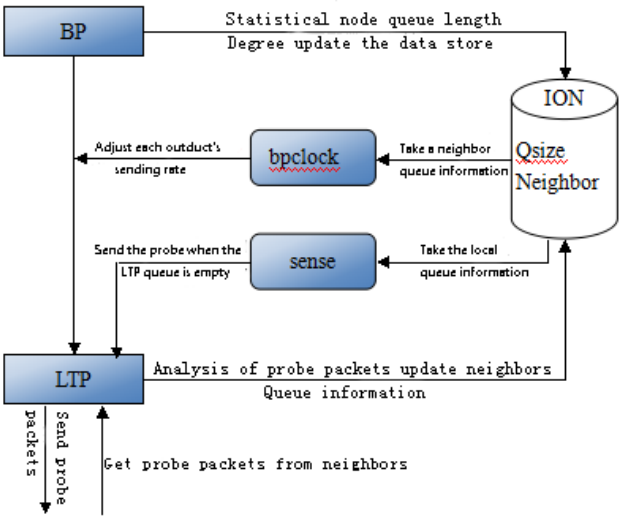

Fig. 1. OCControl architecture.

OCControl can be divided into three modules: 1) a bpclock-based rate-control module; 2) a neighbor-queue-length maintain module, based on the sense daemon process; 3) a queue-length update module, based on ION database. In the following, each module's design will be described, respectively.

1) bpclock-based rate control

Since the congestion-control module in ION is implemented in the BP layer, to re-use the existing components that have been repeatedly verified, we will also implement OCControl's congestion-control module in the BP layer. The bpclock daemon process within the BP layer controls the transmission rate for each LTP out-duct, and acquires the queue length information of its neighbor nodes from the ION database.

2) neighbor-queue-length maintain

The major tasks of the LTP layer is to fetch data from the BP layer, and then repackage those data into data segments with the same size to be delivered to the underlying network layer. To embed the transmission-rate control into the LTP layer, a daemon process, called sense, is introduced. This daemon process is responsible for monitoring the LTP queue. If the LTP queue becomes empty, i.e., no data to be transmitted at present, a probing segment will be put into the LTP queue to be transmitted. Even if the LTP queue is not empty, the probing information will attach to the data segment to be piggybacked. Through this probing information, nodes can learn the queue-length information of their neighbors, so as to determine the occurrence of congestion.

3) queue-length update

On one hand, a node's local queue length can be obtained through the Qsize variable defined in the ION database, which is updated through the data statistics at the BP layer. On the other hand, neighbor nodes' queue-length information can be updated through probing or data segments, which will also be used to the neighbor-node information within the ION database, received from those neighbor nodes.

B. Simulation experiment

To evaluate the performance of OCControl, we configured two OWCN scenarios with three communication approaches, namely satellite, short wave, ultra-short wave. We also build up an emulation network consists of 30 computers. Each computer is installed with the network emulation software CORE, which can be used to simulate one or more oceanic nodes. The main reason to adopt CORE is that the virtual network constructed by CORE is an image of the real network. Although the underlying physical links in the emulated OWCN is software-simulated based on an underlying LAN wired links, the network protocols implemented on CORE can readily execute in a real network without modification.

Besides, since CORE is just used to emulate the network lay, transport layer, and application layer, an additional emulation tool, EMANE, is introduced to emulate the heterogeneous links in OWCN.

We construct two different network scenarios, including a static scenario with 30 nodes, and a dynamic scenario with the number of nodes ranges from 10 to 30. The delivery ratio and expected transmission delay of OCControl are evaluated in each scenario with varied transmission rate, and are compared with those of TCP. The communication parameters of the satellite, short wave, and ultra-short wave communication are listed in Table I.

TABLE I. LINK PARAMETER SETTING

	delay	bandwidth	Packet loss rate
Satellite	4s±1s	5Mbps	80%±10%
Short wave	2s±0.8s	9500bps	60%±20%
Ultra-short wave	1s±0.5s	1Mbps	40%±20%

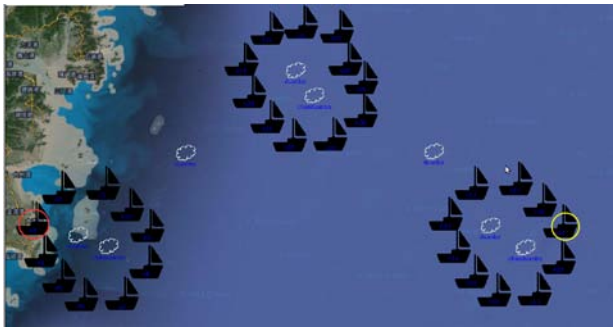

Fig. 2. Network topology for the static scenario.

The network topology of the static scenario is illustrated in Fig. 2, where three groups of oceanic nodes exist. The nodes within the same group can communicate with each other via short wave and ultra-short wave, while the nodes from different groups have to communicate through the gateway nodes, which can communicate with each other via short wave. The node surrounded with a red circle at left-bottom corner in Fig. 2 is the source node, which will send data a destination node in the other group, while the node surrounded with a yellow circle at the right-bottom corner in Fig.2 is the destination node, which will receive data from the source node. The experiment results about the delivery rate and the expected delay is enumerated in Fig.3.

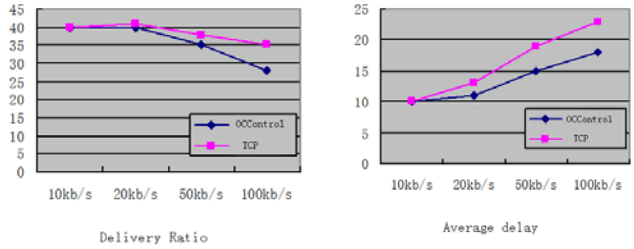

Fig. 3.Experiment result for the static scenario.

The experiment results shown in Fig.3 illustrate that in the case of low transmission rate, both TCP and OCControl achieve a relative good performance in terms of delivery ratio and transmission delay. However, along with the increase of transmission rate at the source node, network congestion become severer, and TCP's performance decrease rapidly, while OCControl maintain a relatively better performance.

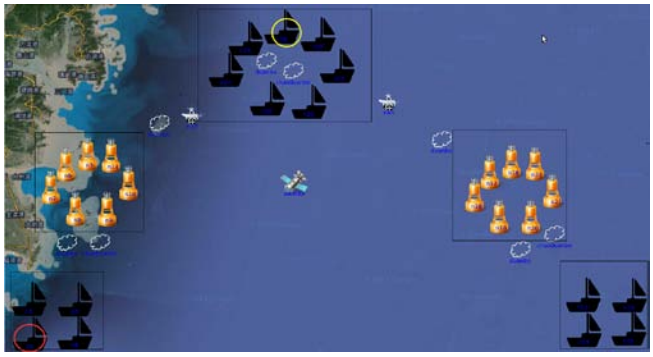

Fig. 4. Network topology for the dynamic scenario.

The network topology of the dynamic scenario is illustrated in Fig.4, where 30 nodes divided into five groups, each of which is surrounded by a rectangle. The nodes within the same group are of the same type. Besides, there exists three airplanes, each of which is an independent node. A plane can communicate with a ship or a buoy via short ware, when it is close to them. Both the submarine and plane groups move randomly.

The nodes within the same group can communicate with each other via satellite, short wave and ultra-short wave, while the gateway nodes can communicate with each other via satellite and short wave only. The node surrounded by a red circle at the left-bottom corner in Fig.4 is the source, while surrounded by a yellow circle at the top-center in Fig. 4 is the destination. The experiment results on the deliver rate and the expected delay is shown in Fig.5.

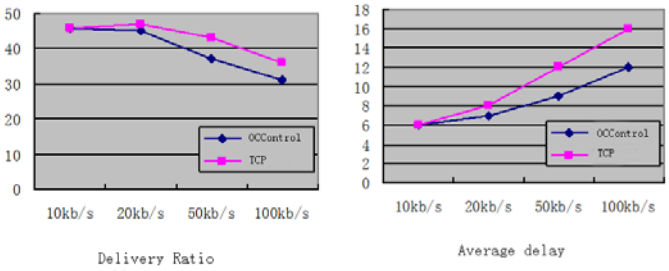

Fig. 5. Experiment result of the dynamic scenario

In the dynamic scenario, similar to the static scenario, along with the increase of transmission rate at the source node, the performance gap between TCP and OCControl expands rapidly. The fundamental reason is that the network becomes congested along with the increase of the transmission rate at the source node, but TCP fails to feed-backed the congestion information to the source node in a timely manner, due to intermittent connectivity. On the contrary, OCControl can rapidly feed-back the congestion information through a hop-by-hop transmission-queue-length feedback scheme, thus OCControl can mitigate network congestion.

V. RELATED WORKS

Generally there are two forms of communication network congestion, link congestion and node-cache congestion. It is generally assume that link congestion rarely occurs in DTN[12].

However, OWCN is not a typical DTN network, because the bottleneck of an OWCN is network link instead of network node. Therefore, the existing method of congestion control for node cache management[13-15] is not applicable to DTN.

As for link congestion, there exist also two types of congestion control schemes, the end-to-end congestion-control scheme[16] and the hop-by-hop congestion-control scheme. However, the former does not apply to OWCN, because the end-to-end connection between two nodes often fails due to the intermittent links in OWCN. As a result, a better OWCN congestion control scheme is the hop-by-hop congestion control scheme.

In the hop-by-hop congestion control scheme, the backpressure based congestion-control scheme[17] is relatively better, because it can theoretically prove its optimality. Thus, our OWCN oriented OCControl congestion-control scheme adopts the idea of backpressure, while remove the binding between congestion controls and routing within the original backpressure routing method so that it can also apply to other routing scheme. Despite this modification makes it hard to prove the optimality, the experimental study has shown that OCControl the backpressure based OCControl congestion-control scheme has a better performance.

VII. CONCLUSIONS

As a special type of DTN network, OWCN is shortage of proper congestion-control scheme. The currently adopted TCP congestion-control protocol fails to function properly due to the intermittent links. To this end, we designed a backpressure based congestion-control scheme, called OCControl and implemented OCControl based on an open source framework, ION. In order to evaluate the performance of OCControl, we verify the superior performance of OCControl in an open-sourced network-protocol simulation platform, which combined both CORE and EMAINE.

REFERENCES

[1] InterPlaNetary Internet Project[EB/OL]. http://www.ipnsig.org/.

[2] Interplanetary Overlay Network (ION) Design and Operation, Jet Propulsion Laboratory, California Institute of Technology. JPL D-48259

[3] S. Burleigh, A. Hooke, and L. Torgerson, "Delay-tolerant networking: an approach to interplanetary internet", IEEE Communication Magzine, 2003, 41(6): 128-136.

[4] H. Lin, Y. Ge, A. Pang, and J. S. Pathmasuntharam, "Performance Study on Delay Tolerant Networks in Maritime Communication Environments", in Proceedings of IEEE OCEANS 2010.

[5] C. Rigano, K. Scott, J. Bush, R. Edell, S. Parikh, and R. Wade, "Mitigating naval network instabilities with disruption tolerant networking", in Proceedings of IEEE MILCOM 2010.

[6] D. Merani, A. Berni, J. Potter, and R. Martins, "An underwater convergence layer for disruption tolerant networking", in Proceedings of Baltic Congress on Future Internet Communications (BCFIC Riga), 2011.

[7] Z. Guo, G.Colombo, B. Wang, J. Cui, D. Maggiorini, and G.P. Rossi, "Adaptive routing in underwater delay/disruption tolerant sensor networks", in Proceedings of the 5th Annual Conference on Wireless on Demand Network Systems and Services (WONS), 2008.

[8] P. Juang , H. Oki, and Y. Wang, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet", ACM Sigplan Notices, 2002, 37(10): 96-107.

[9] A. Doria, M. Uden, and D. P. Pandey, "Providing connectivity to the saami nomadic community", Generations, 2002, 1(2) : 3.

[10] A. Pentland, R. Fletcher, A. Hasson, "DakNet: rethingking connectivity in developing nations", Computer, 2004, 37(1): 78-83.

[11] A. Balasubramanian, B. N. Levine, and A. enkataramani.
"DTN routing as a resource allocation problem", in Proceeding of ACM Sigcomm'07 .

[12] B. Soelistijanto and M. P. Howarth, "Transfer Reliability and Congestion Control Strategies in Opportunistic Networks: A Survey", IEEE Communications Surveys and Tutorials, 2014, 16(1): 538-555.

[13] S. Burleigh, E. Jennings, "Autonomous Congestion Control in Delay Tolerant Networks", Technical Report, Jet Propulsion Lab., available online: http://hdl.handle.net/2014/40636.

[14] T. Kathiravelu, N. Ranasinghe, A. Pears, "An Enhanced Congestion Aware Adaptive Routing Protocol for Opportunistic Networks", in Proceedings of the 6th Intl. Conf. on Industrial and Information Systems, Sri Lanka, 2011.

[15] M. Radenkovic, A. Grundy, "Congestion Aware Forwarding in Delay Tolerant and Social Opportunistic Networks", Proc. 8th Intl. Conf. on Wireless On-Demand Network Systems and Services, Bardonecchia, 2011.

[16] V. Jacobson, "Congestion avoidance and control", in Proceedings of ACM SIGCOMM, 1988, pp314 – 329.

[17] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "DiffQ: Practical Differential Backlog Congestion Control for Wireless Networks", in Proceedings of IEEE INFOCOM 2009..