NC-ODTN：Ocean DTN Network Coding Protocol

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

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

***Abstract*—In the oceanic environments, oceanic wireless communication networks suffer low link quality and intermittent connectivity, which in turn disable the stability of end-to-end communication path, which makes the oceanic wireless communication network a de facto Delay/Disruption Tolerant Network (DTN). Although numerous existing DTN protocols adopted replication-based schemes to deal with the long transmission delay incurred by the intermittent connectivity issue, the replication-based schemes had reached its limit to reduce the transmission delay. Moreover, the replication-based schemes can cause the transmission inefficiency and even congestion problems. Therefore, replication alone cannot meet the service requirement of oceanic wireless communication networks for transmission delay, reliability, and throughput, etc. Therefore, we designed an Oceanic DTN Network Coding Protocol (NC-ODTN) protocol for oceanic wireless networks, considering the characteristics of oceanic wireless communication. To evaluate the performance of NC-ODTN, we implement NC-ODTN on a testbed consisting of over 20 wireless nodes. The experimental results have shown that NC-ODTN effectively improves data transmission rate and greatly reduces transmission delay, compared with existing TCP/IP protocols.**

Keywords—Network Coding, DTN, Oceanic Wireless Communication Networks

I. INTRODUCTION

Along with the requirement of the communication under various extreme network conditions, such as, deep-space communication [1-3], maritime communication [4, 5], underwater communication [6, 7], and communication in remote areas [8-10], DTN [11] gradually get more and more focus as a new network model with intermittent connectivity.

The Internet TCP/IP protocols assume that there are stable transmission paths between end and end at any time, so the data could be sent through a connection-oriented TCP transport protocol. However, in the DTN networks, it cannot guarantee the stable transmission path because of the network environment. For this reason, TCP often fails in DTN scenarios.

With the widespread use of those TCP/IP protocols, almost all network applications are coded based on the TCP/IP sockets. Obviously, these application layer protocols are not compatible with the new DTN network protocols and it is unrealistic to rewrite all existing application layer protocol to port to the DTN network scenario. This requires creating a package of solutions which not only is compatible with existing application layer protocols, but also can apply to solutions DTN network scenarios.

For all this, we have designed a DTN overlay layer protocol, NC-ODTN, which is compatible with existing application layer protocols. So that, in the network environment with a stable and reliable transmission path, the application data could be transferred through TCP/IP protocols and in DTN network environment without modifying existing application layer protocols, the application data could be transferred as much as possible.

In order to verify the performance of NC-ODTN protocol, we implement NC-ODTN protocol based on the open source framework ION[8,9] and test the protocol at multiple network scenarios based on the open source network simulation software CORE[12,13] and EMANE[14-16]. The experimental results show that compared with the existing TCP/IP protocols, NC-ODTN effectively improves the success rate of data transmission, and greatly reduces the latency of data transmission. Therefore, NC-ODTN protocol is a better routing scheme applying in marine wireless communication network.

II. NC-ODTN Design Principles

The NC-ODTN design focuses on unicast since RAR [17] is designed for unicast. However, NC-ODTN should be able to readily extend to multicast and broadcast. Besides unicast and multicast, the NC-ODTN design should consider five factors, including the coding-generation management, the control-signal mechanism, the copy-control mechanism, the transmission-schedule and cache management, and the obsolete-information cleansing strategy. In the following, this paper will describe these five factors, respectively, so as to illustrate the design principle of NC-ODTN.

As for the code-generation management, from the perspective of the source and destination of the data packets that comprise the cod generation, three alternative schemes have been proposed, *i.e.*, data packets from the same source-destination pair, from different sources but to the same destination, and from different source and to different destinations, respectively. For the sake of simplicity, we denote the above three schemes as the single source-destination pair, multi-source but single-destination, and multi-source and multi-destination. Existing studies have demonstrated that the multi-source but single destination scheme performs worse than the single source-destination scheme, while the multi- source and multi-destination scheme is even inferior to the one without coding, because each of those destination nodes has to collect k linear independent coded packets for decoding a single intended packet, if k data packets from different source-destination pairs are coded

together. It is obviously a waste of bandwidth. Thus, we focus on the single source-destination scheme for code-generation management. Besides, the code generation management also involves the determination of the generation size. Existing researches have shown that the benefit of random linear coding decreases along with the increase of the generation size. The underlying reason is that, along with the increase of the generation size, the relative benefit of the non-coding scheme will increase. Therefore, it is appropriate to adopt a relative small generation size.

From the perspective of the control-signal mechanism, there are three alternatives: the no-signal control, probe-signal control, and complete-signal control schemes. The complete-signal control scheme requires nodes to exchange their coding matrix with their neighbors so at to determine whether their neighbors have novel data, the cost of which is usually much higher than the simple probe-signal scheme. Moreover, even if nodes have successfully exchanged their coding-matrix information, they also have to do a lot computation to determine the information novelty. Existing works have identified that the whole-signal control scheme does not perform much better than the probe-signal control scheme. Therefore, the probe-signal control scheme is applied in our NC-ODTN protocol.

At the same time, the transmission scheduling and cache management should be considered. It has to be noticed that due to the distinguished characteristics, RAR is particularly designed for oceanic DTN. RAR actually forms the upstream-downstream relationship among selected network nodes. Thus, this reduces the transmission-direction uncertainty embedded in opportunity networks to some extent. Since NC-ODTN is based on RAR, it is necessary to consider this prominent feature of NC-ODTN to improve the coding efficiency. Due to the relative reduction of the uncertainty, NC-ODTN does not adopt the random-selection or round-robin transmission strategy, which is usually adopted in opportunistic networks, but integrate ACK, timeout re-transmission, and transmission-factor control to design the transmission-schedule and cache-management scheme.

More specifically, each time after the source node linearly combines n data packets and transmits the coded data packets; it will wait for the ACK from the destination node. The destination node will send ACK immediately after it receives n linearly independent data packets without waiting for encoding. If the ACK is received within a predefined time period, the source node will fetch the next-generation packets to be transmitted. Otherwise, it will re-code the data packets within the current generation and transmit them again. Every intermediate node will maintain a transmitting counter. Each time when it receive a novel data packet, *i.e.*, a packet that is linear independent with the previously received packets, the counter will increase by *f*, where *f* is the transmission factor and the counter will reduce by 1 after successfully transmitting a coded packet. This can make those intermediate node forward received packets in the ratio of 1/f. Once an intermediate node receives a coded packet from the next generation, it will realize that the coded packet within its cache is out-dated and will abandon all of them. Moreover, the intermediate node will also maintain a time-out timer to manage cache. Each time when it receives a novel coded packet, it will update the timer's time-out time from the time-stamp embedded in the packet header. If the time elapsed since the last received novel packet surpasses the time-out value, it will cleanse the cache. This strategy is to avoid the case, when the source re-transmits packets, the intermediate node may reject forwarding them due to the lack of novel packet issue. In the end, since the bottleneck of the oceanic DTN lies in the link instead of the node, NC-ODTN does not have to consider the cache overflow issue.

Finally, the obsolete-information cleansing strategy adopted in NC-ODTN has been discussed previously. Actually, due to the characteristics of oceanic DTN and RAR, the obsolete-information cleansing strategy adopted in NC-ODTN does not adopt the vaccine mechanism, but adopts the easier ACK mechanism instead.

III. NC-ODTN DESIGN

The entire process of NC-ODTN includes three components: coding at the source node and the intermediate node, and the decoding at the destination node. In the following, we will elaborate the three components within NC-ODTN, respectively.

A. The Coding Algorithm at the Source Node

The main coding task at the source node is to randomly linearly encode data packets A_1, A_2, ... A_n, with randomly chosen coding coefficient, where *n* is the number of data packets in a data block.

To encode data packets A_1, A_2, ... A_n, to a coded packet, two necessary steps are required: 1) represent packets as a series of linear equations, from which the matrix reflecting the set of linear equations will be derived readily; 2) apply linear-superposition operations to the matrix corresponding to the linear-equation set, which will create a coded packet in the form of a linear equation. The pseudo code of this algorithm is described in Algorithm 1, where the input is a set of data packets denoted as A_1, ... , A_n, and the output is a coded packet, which is a linear combination of all the *n* input packets.

Algorithm 1

```
result = 0;
generate coding vector C_i for each packet A_i
for i = 1 to n
    randomly generate coeff;
    temp = vector_mul (C_i, coeff)
    result = vector_add (result, temp);
```

In Algorithm 1, the coding component iteratively generates a coefficient vector, called coeff, each element of which is a randomly generated integer within a predefined Galois field. The randomly generated coefficient vector will multiply with coefficient vector of the *i*-th data packet, the result of which will be further added to the previously calculated vector summary, the initial value of which is a vector of zero.

B. The Coding Algorithm at the Intermediate Node

Compared with the coding operations at the source node, the coding operations at intermediate nodes coding are relatively easier. Because a data packet received by an intermediate node usually has already been encoded. To generate a new encoded packet, it is simple enough to just linearly combine the existing coded packets that are linearly independent. For each received coded packet, it is necessary to determine whether this packet is linearly independent with the previously received coded packets. If it is linearly independent, it will be put into the coding buffer; otherwise, it will be simply discarded. The coding buffer will be used to generate coded packets through the algorithm similar to Algorithm 1. For the intermediate nodes, the most complicated process is to determine the linear correlation. A particular algorithm for linear correlation determination is described in Algorithm 2, where the inputs are a coded packet just received, denoted as P and k previously received linearly independent packets, A_1, ... , A_k, which have been stored in the coding buffer and reduced to a lower triangular matrix, the output is a boolean value with true denoting linear indepedence and false otherwise.

Algorithm 2
isCorrelated(P, A_1, ... , A_k)
```
while true
    identify P_m, the first non-zero coefficient of P
    if P_m does not exist
        return false
    else if m ≥ k
        return true
    else
        P = P - A_m × P_m
        k = k -1
```

Algorithm 2 first identifies the first non-zero coefficient P_m in the coefficients of P. If such *m* does not exist, it means that P is linear dependent with the packets in the coding buffer. Thus, the algorithm returns false in this case. Otherwise, if *m* is larger than the number of packets in the coding buffer, it means that P is linear independent with those packets. Thus, the algorithm return true in this case. If *m* is no larger than the number of packets in the coding buffer, the coding-coefficient vector of the *m*-th packets will be multiplied with the *m*-th coefficient of the coding-coefficient vector of P, the result of which will be added with the coding-coefficient vector of P. This process will repeat until the correlation relationship between P and the packets in the coding buffer can be identified.

With the help from Algorithm 2, the coding operation at intermediate nodes can be readily described, as shown in Algorithm 3, where G(P) is used to denote the coding buffer of the code generation, to which P belongs to, and C(P) denotes the coding vector of P.

Algorithm 3
```
if G(P) does not exist
   Create G(P) and put P in G(P)
else
  correlated = isCorrelated(P, , A_1, ... , A_k)
  if correlated is true
     return
  else
     identify m, the first non-zero coefficient of P
     normalize P_m
     put C(P) into the m-th row of the coding matrix
     generate a coded packet from the packets in the
coding buffer
```

C. Decoding Algorithm at the Destination

The decoding process at the destination node is similar to the process to solve a set of linear equations. However, this decoding process is discrete, instead of a centralized one. Actually, this process integrates the the operations to determine linear correlation and the operation of decoding. For each received coded packet, when its correlation relationship is determined, the decoding operations will also be carried out. This design allows the decoding process to be distributed to the receiving process, which can greatly reduce the decoding delay and improve performance.

The decoding process is similar to the encoding process at intermediate nodes. The only difference is the last step, where in the encoding process at intermediate nodes, it is required to generate a new encoded packet, while in the decoding process at the destination node, the correlation reduction operation, as shown in Algorithm 4, will execute on the last received coded packet. Algorithm 5 describes the decoding process executed on the destination node.

Algorithm 4
reduceCorrelation(P, A_1, ... , A_k)
i=1
```
while i < k
    if P_i != 0
      P = P - A_i × P_i
    i++
```

Algorithm 5
```
if G(P) does not exist
   Create G(P) and put P in G(P)
else
  correlated = isCorrelated(P, , A_1, ... , A_k)
  if correlated is true
     return
  else
     identify m, the first non-zero coefficient of P
     normalize P_m
     put C(P) into the m-th row of the coding matrix
     reduceCorrelation(P , A_1, ... , A_k)
```

IV. System Implementation

NC-ODTN is implemented based on an open-sourced DTN framework, ION. The framework is a particular implementation of the DTN architecture specified in RFC 4838, including a complete implementation of the Bundle and LTP layer in the DTN architecture.

A. The Principles Of Network Coding In ION

Implement network coding itself is not so difficult. The real challenge is to integrate it into the ION framework. ION, as an implementation for a special type of DTN in the scenario of the deep-space communication, it did not consider network coding at the first place. Moreover, the overall ION framework is a relative large system with over millions of lines of codes. For the above reasons, a general principle to integrate network coding into ION is to modify the original ION structure as little as possible. In order to clarify the implementation of network coding in ION, it is necessary to describe the ION transmission and reception processes as follows.

1) ION process analysis

In ION, after receiving an Application Data Unit (ADU) request, the following steps are required to transmit a data bundle, as shown in Fig.1, where an ADU from an upper-layer application will be packed as a data bundle, which will go through the ForWarding Queue (FWQ), the outbound pipeline, called outduct, in turn, until it reach the underlying LTP layer protocol.

Fig. 1. The data transmission process in ION

The ION data reception process is illustrated in Fig. 2, where data segments received from the *ltpcli* process within the LTP layer will be recovered to a bundle, which will be dispatched to the application layer or re-forwarded according to the targeted destination.

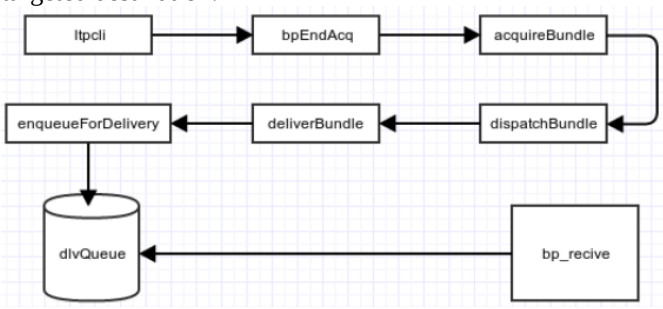

Fig. 2. The data reception process in ION

2) Challenge analysis for network coding in ION

The most difficult part of the network coding in ION is that ION is not particularly designed for network coding. The basic unit of data processing is bundle, through which data transmit and receive. While, in network coding, it is necessary to group data into a generation, and an encoded packet originates from the same generation. The data transmission and reception are also organized from the perspective of generations, in that unless the current generation has been acknowledged, the next generation will not be issued. Thus, the basic unit of data processing is a group of data, instead of a single bundle.

The second challenge for network coding in ION is that the data to encode packet for network coding should have the same length, but the lengths of ADUs, which are provided by the upper-layer applications usually differ in their length. Therefore, it is necessary to unify the data length.

In ION, two types of data storage exist: PSM and SDR. Similar to the memory, PSM is used to store various control information in the ION processing flow. Similar to the file system, SDR can provide reliable and nonvolatile storage through the reliable transaction mechanism. Other than PSM, SDR is used to store data information in the ION processing flow. In SDR, to reduce the number of data-copy operations among different components, Zero Copy Object (ZCO) is introduced to transfer data address instead of data among different layers within the network stack. This mechanism greatly improves ION efficiency. However, it brings a great challenge for network-coding design in ION, because ION stores data in ION files, i.e., SDR, where data is accessed through ZCO, while a coding operation has to be executed in memory, i.e., PSM in ION. This gap makes it hard to implement network coding in ION.

Moreover, as a DTN implementation, ION implements two major protocols in DTN's protocol stack: the BP and LTP protocols. However, these two protocols did not take network encoding into account. Therefore, it is necessary to extend these two protocols to integrate network coding. A further challenge is to determine whether to implement network coding at the LTP layer or the BP layer. In the LTP layer, data bundles will be grouped into a data block, which will be separated into as data segments. This process is similar to the sequential process of network coding, namely grouping, coding, and transmitting one by one. However, in DTN architecture, the routing component is within the BP layer. Therefore, if the coding operation is within the LTP layer, the BP header must be part of the coded data. Hence, an intermediate node has to decode the received coded data to extract routing information. Since the LTP protocol is a link-layer protocol, it means that the decoding operations have to be executed at each intermediate node, which loses the whole meaning of network coding.

B. The Network Coding Design within ION

1) Encoding process design

Based on the above discussion on the challenges of the network-coding implementation in ION, the encoding process can be designed as follows. Firstly, network coding should lies in the BP layer instead of the LTP layer to avoid decoding at intermediate nodes. Secondly, to address the issue of different process unit between ION and network coding, when the BP layer receives a transmission request from an application, ADUs will be put into a coding buffer, instead of grouping into a bundle. Once the buffer becomes full, coded ADUs will be generated based on the ADUs in the buffer. Then, the encoded ADUs and the corresponding coding coefficients will be grouped into a bundle. Thereafter, the processing for the

coded data will follow the original ION processing flow. This design solves the conflict between the bundle processing of ION and the block processing of network coding, and implements network coding on the top of the BP layer. Therefore, it simplifies the modification on ION's original interfaces, the modified ION data transmission process with network coding embedded is illustrated in Fig. 3.

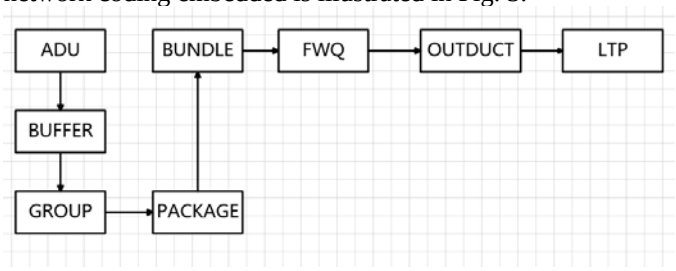

Fig. 3. The data transmission process with network coding embedded.

2) Decoding process design:

From Fig.2, it can be observed that received data will be put into the delivery queue eventually, which is accessed by the bp_recive function to obtain the original data. Since our design principle is to minimize the modification to the original ION code, to make the encoding and decoding processes completely transparent to the upper-layered applications, it is better to decode before data being put into the delivery queue. Moreover, a decoder buffer is required to group coded packets for decoding. The detailed decoding process is illustrated in Fig. 4.

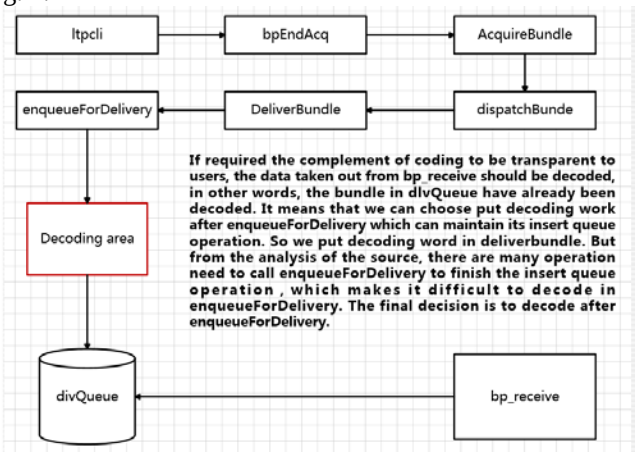

Fig. 4. The design for decoding process

V. SIMULATION EXPERIMENT

To evaluate the performance of network coding, we configured a number of heterogeneous network scenarios with three communication approaches, namely satellite, short wave, ultra-short wave. We also build up an emulation network consists of 31 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.

To evaluate network coding's performance, we construct 2 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 transmission rate and expected transmission delay of network coding are evaluated in each of the scenarios, and are compared with those of without network coding. 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%

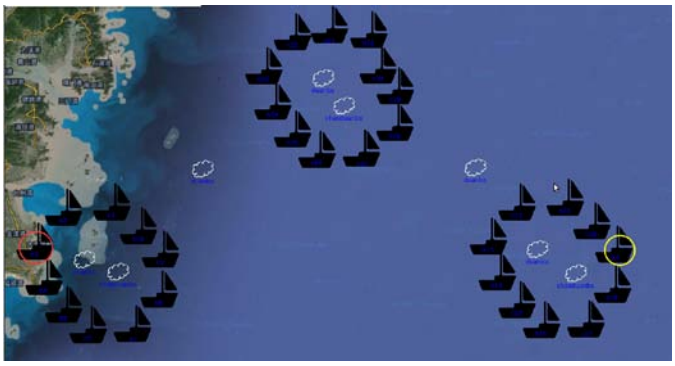

Fig. 5. Network topology for the static scenario.

The network topology of the static scenario is illustrated in Fig. 5, 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. 5 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.5 is the destination node, which will receive data from the source node. The experiment results about the delivery rate and the average delay is enumerated in Table II.

TABLE II. EXPERIMENT RESULT FOR STATIC SCENARIO

	Delivery rate	*average delay*
TCP	40%	10s
NC-ODTN	70%	14s

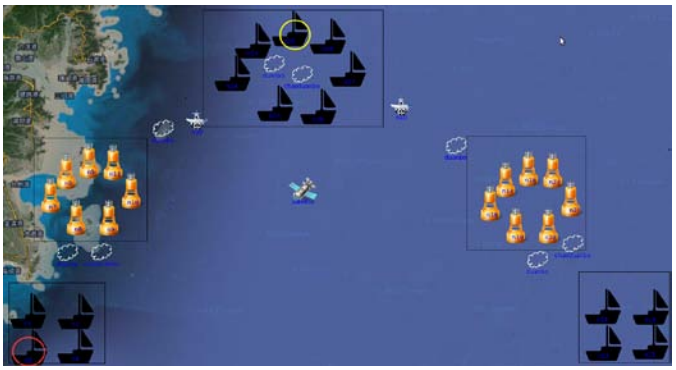

Fig. 6. Network topology for the dynamic scenario.

The network topology of the dynamic scenario is illustrated in Fig. 6, 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. 6 is the source, while surrounded by a yellow circle at the top-center in Fig. 6 is the destination. The experiment results on the deliver rate and the average delay is shown in Table III.

TABLE III. EXPERIMENT RESULT FOR DYNAMIC SCENARIO

	Delivery rate	***average delay***
TCP	46%	6s
NC-ODTN	65%	13s

From the experimental results, both node static network environment and node random motion network environment, after using the network coding is equal to provide some reliable data transmission mode, which can improve success rate of data packet transmission.

To avoid the chaos incurred by coding a large number of data packets, network coding transmits data by batch. Until the last coded batch has been delivered and acknowledged, the next batch will not transmit. This obviously increases the average delay of data transmission. In addition, re-transmission may also increase the average delay. However, compared with the performance improvement, these delay costs are still acceptable.

To avoid the chaos incurred by coding a large number of data packets, network coding transmits data by batch. Until the last coded batch has been delivered and acknowledged, the next batch will not transmit. This obviously increases the average delay of data transmission. In addition, re-transmission may also increase the average delay. However, compared with the performance improvement, these delay costs are still acceptable.

VI. CONCLUSION

This paper analyzed the relationship between OWCN and DTN, and OWCN's unique features, based on which the design space of the network-coding scheme in OWCN has been discussed. Based on the possible design options for network coding in OWCN, a network-coding scheme for OWCN, named NC-ODTN has been proposed, including the coding and decoding process at the source node, intermediate nodes, and destination node, respectively. Finally, based on an open-sourced framework for interstellar communication network, called ION, the designed network coding scheme is implemented. In order to verify NC-ODTN's performance, NC-ODTN was deployed on a testbed with over 30 nodes, in which NC-ODTN's performance has been verified. The experimental results have shown that compared with the existing TCP/IP protocol, NC-ODTN effectively improves the delivery ratio, and greatly reduces the delay of data transmission.

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] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: A real time network emulator", in Proceeding of IEEE MILCOM 2008.
[13] CORE Documentation. Release 4.6. available online:
http://downloads.pf.itd.nrl.navy.mil/docs/core/core_manual.pdf.
[14] S. M. Galgano, K. B. Patel, and E. Schreiber, EMANE User Manual (version 0.8.1), available online:http://downloads.pf.itd.nrl.navy.mil/docs/emane/emane.pdf.
[15] S. M. Galgano, K. B. Patel, and E. Schreiber, EMANE Developer Manual (version 0.7.3)., available online:
http://downloads.pf.itd.nrl.navy.mil/docs/emane/emane-dev.pdf.
[16] J. Ahrenholz, T. Goff, and B. Adamson. "Integration of the CORE and EMANE Network Emulators". in Proceeding of IEEE MILCOM 2011.
[17] XX, "RAR:Replication Adaptive Routing in Oceanic Wireless Networks". In Proceeding of IEEE