Implementation of an underwater acoustic modem with network capability

T. Husøy, M. Pettersen
Kongsberg Maritime
Horten, Norway

B. Nilsson, T. Öberg
FOI – Swedish Defence Research Agency
Stockholm, Sweden

N. Warakagoda, A. Lie
SINTEF ICT
Trondheim, Norway

Abstract— **This paper introduces the underwater acoustic modem as implemented within the UAN – Underwater Acoustic Network project. The low power modem has implemented turbo equalization algorithms in addition to variable spread rate direct sequence spread spectrum signaling. The network layer implemented on the modem support automatic network discovery, multi hop routing and support for mobile nodes, and when expanded with a single board computer via serial line it supports IP connectivity end-to-end. Experimental results from sea trials are presented.**

Keywords-component; underwater acoustic modem, network, turbo equalization

I. Introduction

Underwater communication and networking is continuously being developed through combined effort from industry and research initiatives. Partners from universities, research institutes and industry has joined in the UAN – Underwater Acoustic Network – project receiving funding by the European Community's 7th Framework Programme. The partners include CINTAL - Centro de Investigação Tecnológica do Algarve (coordinator), SSI - Selex Sistemi Integrati (Italy), SINTEF ICT (Norway), ISME - Interuniversity Centre of Integrated Systems for the Marine Environment (Italy), Swedish Defence Research Agency FOI (Sweden) and Kongsberg Maritime (Norway). UANs objective is to conceive, develop and test at sea an innovative wireless network integrating submerged, terrestrial and aerial sensors for the protection of off-shore and coastline critical infrastructures. The UAN concept is to gather environmental information during the acoustic transmission and use it to predict the acoustic propagation conditions and the optimal obtainable performance at any given time [1].

The final demonstration of the network will be in the Norwegian Sea outside Trondheim, Norway during summer 2011. The three year project period commence in October the same year.

Kongsberg Maritime (KM) is delivering the underwater communication modems for the project, a total of six modems. The modems are based on the Kongsberg Maritime cNode® Mini transponder platform (Figure 1), which is modified for the UAN experiments. FOI has developed the turbo equalization algorithms which is implemented on the modem by KM. SINTEF ICT have been responsible for the network architecture and routing algorithms which is implemented on the modem by KM. In addition SINTEF has interfaced the modem to an external Linux host.

This work was supported within EU FP7 UAN project, grant agreement no.: 225669

The contribution of this paper is to give information on UAN modems protocol layers, which includes underwater IP network connectivity end-to-end when the modem is expanded with a single board computer (Section II and III). Further, the paper gives insight into Turbo equalization for implementation on a low power platform (Section IV), and demonstrates its current performance through sea trial measurements (Section V).

II. Background

A. cNode

The cNode® is a family of different sized transponders with different configurations including sensors or just as a pure modem (Figure 1). In the UAN project the cNode Mini was configured with a transducer having a hemispherical beam pattern with maximum source level SL = 189 dB re 1uPa at 1 meter. The centre frequency is 25.6 kHz with variable bandwidth. Maximum operating depth is 4000 m depth in a cylindrical aluminum housing containing an internal rechargeable NiMH battery.

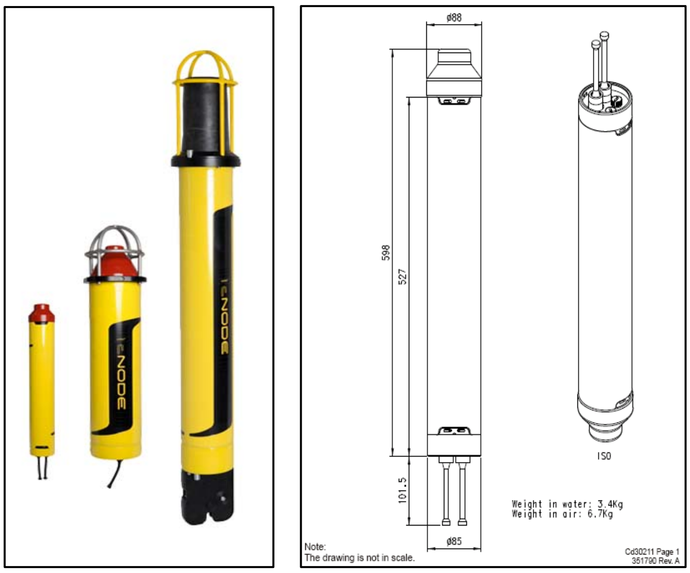


Figure 1. cNode® family (Mini, Midi and Maxi) with physical outline of the Mini to the right

The UTB – Universal Transponder Board – is the main circuit board of the modem (Figure 2). This board has a transmitter and receiver as well as a processing and control part. The Texas Instrument Low Power DSP TMS320C5509 forms the central processing part to which all implementation of PHY, MAC and routing layer in UAN is performed.

Figure 2. Universal Transponder Board

B. *Cymbal®/LTP-Link Transport Protocol*

The Kongsberg Maritime Cymbal® Protocol, based on variable spread factor DSSS signaling, is also available at the UTB board. The link layer protocol LTP supports addressing, error detection, segmentation and augmentation of packets and retransmission of any lost telegrams. The concept of transport formats make different signaling selectable to adjust for transmission in different acoustic propagation conditions; to combat time and Doppler spread and poor signal to noise ratio. Table 1 presents the transport formats used in the UAN project.

Table 1 Transport formats

Transport Format	**Telegram**			**Packet Size**
	Size	***Burst rate***	***Duration***	
1	50 B	200 b/s	2.0 s	1600 B
2	120 B	470 b/s	2.0 s	3000 B
3[a]	231 B	1710 b/s	1.1 s	3000 B

a. Including turbo equalization implemented within the UAN project

III. PROTOCOL STACK

A. *UAN network requirements*

Current underwater acoustic networks consist only of a small number of nodes, mostly limited by equipment cost and availability. Due to the UAN project focus on supervision of submerged infrastructure of critical ocean installations (e.g. related to gas/oil and power-plants), a full sized network could consist of 20–30 nodes equipped with sensors and modems. The maximum foreseen coverage area is in the order of a few km^2. The network protocol should support low to medium traffic load in a low-rate low-energy consumption network, as well as a high-rate uni-directional mode using a dedicated vertical array at the master location for critical events. This paper covers only the former network mode, consisting of a mix of fixed and mobile nodes.

B. *MAC*

The MAC protocol should be capable of providing random access to the channel, without any time synchronization between the modems. Thus, a CSMA/CA based scheme was selected, similar to that one found in 802.11 WiFi. In WiFi, the backoff interval is calculated as an integer number of *slot times*, where the slot interval is in the order of the measured round-trip time (RTT). Due to the pronounced propagation delay in acoustic communication, this was modified to become a *fraction* of the RTT [2]. To avoid packet collisions when submitting long packets, these can be preceded by explicit RTS/CTS packets so that channel access can be granted.

The MAC implementation is modified and built on the code implemented in a previous “NNN-UTS” project [3], which is also true for the network discovery and routing layer.

C. *Network discovery and routing*

The underwater modems have no a priori knowledge of the network configuration at power-up, except the knowledge on being the master or a slave. The UAN network discovery protocol is based on a flooding principle, in where the master sends a flood broadcast, upon the slaves receiving this broadcast joins in and broadcasts their own “I’m here too, anyone listening” type of flooding. Each slave records all the other nodes it can hear, and eventually sends a *flood report* unicast packet towards the master. An example is shown in Figure 3 where one master (node 1.0) and two slaves (node 2.0 and 3.0) are setting up a network, and where one of the slaves can not hear the master directly. As a result, the master will obtain a complete overview of how to reach all slaves with a minimum of transmission power (using the Dijkstra’s algorithm). The downlink routes are given in the packet header, while each of the slaves can do with the knowledge of the best next hop against the master only. The UAN routing discipline is thus centralized.

D. *Mobile node support*

In UAN some of the nodes are mobile nodes (using Folaga vehicles), motivated from either the ability to perform sensing at different locations, and/or move position in order to relay network connections due to changing channel conditions. If the Folaga is given precise coordinates, it will surface and move to the correct lat-long position using GPS, before diving to the given depth. During this period it might be wise to signal to the network that it is leaving, to avoid relaying packets into broken links, and minimizing the signalling overhead. A dedicated flood delete message unicasted to the master will initiate a recalculated routing table, which results are unicasted to all remaining slaves in the network. When the Folaga has reached its new location it will submit a flood add *broadcast* message, in where all receiving slaves (and master, if possible) will respond back with an acknowledge packet. The Folaga will respond back after the last acknowledge with a report unicasted to the master, which in turn recalculates the routing table, and sends its results to the slave in a similar manner as when the node left the network.

In other situations it might be valuable that the moving node is always present in the network while moving. One example can be intruder pursuing. In this case, the Folaga will never submit the flood delete message, but the *flood add* broadcast is submitted periodically. The length of this period can be based on vehicle speed, to ensure that the moving node will always have a valid and optimal route within the network.

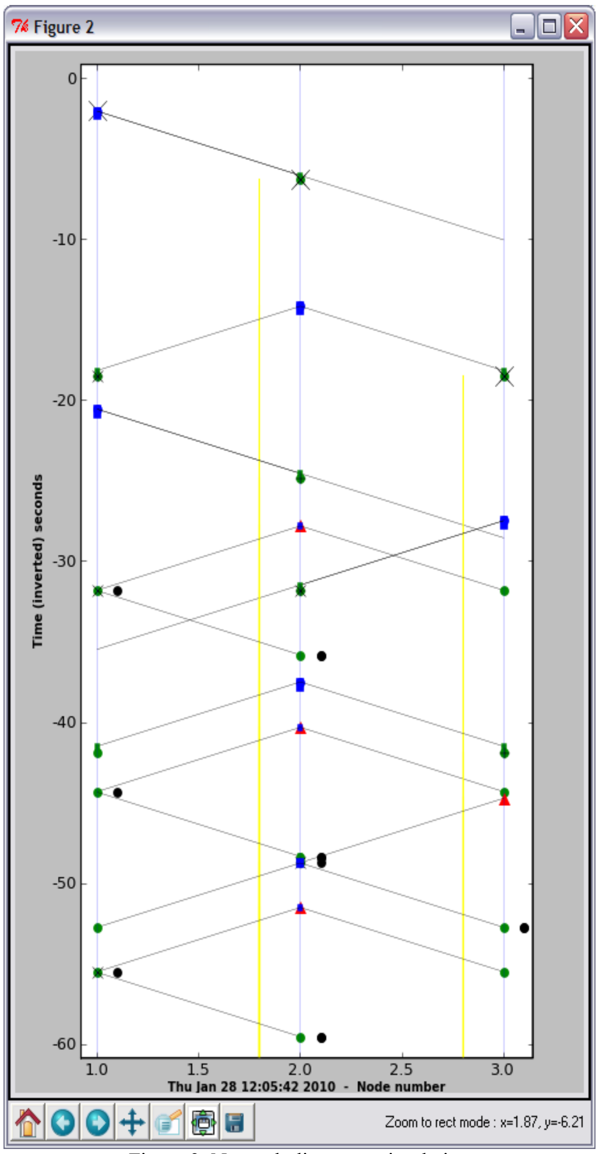


Figure 3. Network discovery simulation

E. *PPP as IP interface*

The information exchange between upper layers of the UAN network is designed to use the open source publish/subscribe middleware MOOS. This software is using TCP/IP for communication between entities, and modified by UAN to also support UDP/IP. A MOOS database (MOOS DB) is run at a single surfaced computer, while each underwater network node is running a MOOS client on the slave Linux host. The Linux host is connected to its local KM modem (KMM) using a RS-232 serial line. The UAN project decided to implement IP connectivity end-to-end because of the native MOOS IP interface. The IP protocol overhead can be easily minimized using well known compression techniques [4][5]. Since layer 1 (PHY) and 2 (FEC, MAC, ARQ and routing) is located in the KMM, the PPP protocol [6] was a natural and efficient solution since it also includes the header compression algorithms. The master will have one PPP interface per subsea slave, while the slaves will have one single PPP interface towards the master. Activity below the PPP layer such as rerouting and ARQ is completely transparent. The PPP is a connection oriented protocol that demands bidirectional LCP (link control protocol) request and respond messages, as well as IPCP (IP control protocol) request and respond messages, negotiating for IP addresses and header compression. When both master and slaves are requesting the same parameters this setup is completed in 4 messages transmitted downlink and correspondingly uplink. There need not be any additional signaling.

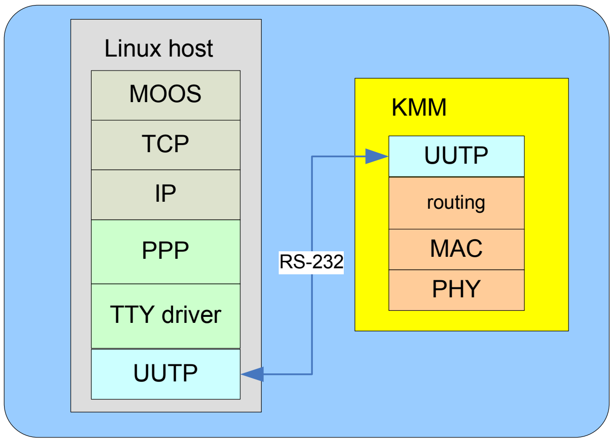


Figure 4. The UAN protocol stack. UUTP is a proprietary protocol.

Figure 4 shows the upper and lower parts of the communication protocol. Note that the KMM in the case of relaying will not submit such messages on the serial line: relaying is handled locally on the KMM. The UUTP is a local protocol used over the serial line only. Software Xon/Xoff messages are sent to the PPP driver to control the KMM sending buffer backlog (not shown in the figure).

F. *Implementation of software*

The software implemented in the project includes the layers of TTY driver, UUTP encoding/decoding and RS-232. Standard Linux functionality was used above the TTY driver. The software modules are identical on the master and the slaves, even though their behaviors are a bit different.

The implementation consists of two main software modules

1. A Linux TTY driver
2. A User domain program

TTY driver is Linux kernel domain program which communicates with higher layers of the Linux kernel as well as with the user domain program. The TTY driver communicates with the user domain program using the *netlink* protocol. TTY driver implements all the essential APIs required to be implemented by any standard Linux tele-typewriter device. It is implemented as a loadable kernel module, and when loaded it creates 16 Linux TTY devices. In order to create a PPP link between two nodes, the well known Linux command *pppd* is run using a TTY device created by this driver as an argument.

The user domain program consists of three main sub modules: netlink controller, socket controller and the serial port controller. Netlink controller is responsible for communication with the TTY driver, whereas the serial port controller takes care of reading from and writing to the hardware serial port to which the KM modem is connected.

The user domain program's main function is to convert between PPP and UUTP packets.

In addition to the main task described above, the user domain program performs the task of transmitting admin packets to and from the KM modems. These packets do not have anything to do with the PPP; they are just used for administrative purposes for the KM modems. In the case of master node, the KM modem administration is handled by an above water module called APOS, whereas on the slaves this function is performed by an on-board controlling module. The socket controller sub-module is responsible for communication with the APOS or on-board controlling module, which is connected to the socket controller module via an UDP based IP link. Admin packets are forwarded from the APOS or on-board controlling module to the serial port or in the opposite direction without performing any form of repackaging.

It was also investigated whether a protocol adaptor between PPP and UUTP is possible without having a kernel domain program (i.e. TTY device). An alternative set of software modules was developed to study this possibility. Essentially this version has the same functionality as the above described implementation, but instead of a TTY driver another user domain program was used as the "glue" between the *pppd* command (using its *–pty* option) and the protocol converter module. The main advantage of such an approach is the possibility of easier debugging, because there is no kernel domain program. The disadvantage is however lesser control over the internal buffering, which is an important issue when the underlying network is of extremely low bandwidth.

G. Network simulation

The network discovery and MAC protocol is simulated using a proprietary simulator based on Python, SimPy and MatPlotLib. Among selected topics of interest are network discovery time as function of network size, and signaling overhead in keeping updated network routing tables while nodes are moving.

Three different network sizes, and three different node densities, were used as scaling range for investigating network discovery time evolution. The network topology form factor is assumed squared, with either 2 x 2, 3 x 3, or 4 x 4 nodes. The distance between each node (along x and y axis, node depth z is assumed identical) was selected as 500, 2000, and 5000 km. Master node is always located at (0,0), i.e. in the corner. Figure 5 reveals the results using SL < 189 dB re 1µPa at 1 meter, carrier frequency of 25.6 kHz, distance path loss and absorption loss using empirical formulas according to [7], and receiver sensitivity (including noise floor) at –70 dBmW. As long as the distance between the nodes is large, the increase in network size has little impact on network discovery time per node, since the number of neighbors disturbed is limited. At more densely spaced networks, a network size increase will eventually create more colliding traffic than the CSMA/CA algorithm is designed to cope with (500 and 2000 m spaced nodes in a 16 node network), and the time it takes to complete the network discovery grows substantially. Even such networks can be created, their throughput will be limited, and thus more advanced MAC protocols should be considered.

Table 2. Simulation of traffic during mobility

Packet type	Packets	Bytes/packet
Initial Network Discovery:		
Flood broadcast	227	20
Flood reports	37	20
Mobility signalling:		
Flood delete	3	20
Flood best	25	20
Flood add	6	20
Flood add ack	58	20
Flood ack report	5	20
Payload traffic	1009	150[1]
Discarded packets	12	
Signalling during mobility	97	20
Signalling overhead	9.6%	1.3%

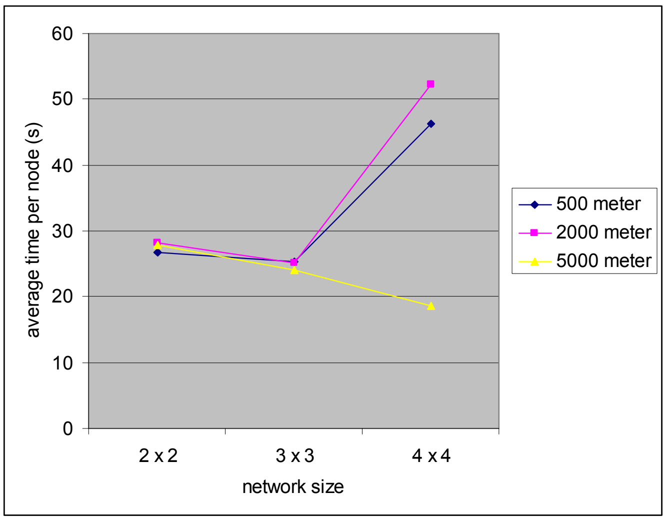


Figure 5. Network discovery time scaling evolution

The next scenario simulated is a 3 x 3 node network with 2000 m spacing, and where the network discovery was finalized after approx. 600 seconds. The middle node (node 5) is moved three times during the simulation, all of which required full network departure and rerouting requests. Payload traffic starts at 1500 s and submits packets at uniformly distributed inter-departure times in the range 30–120 seconds. The simulation ended at 10000 s (approx 2.7 hours), and the modems transmit with 1600 bit/s. Table 2 shows the distribution of the different packet types, including both network discovery, mobility, and payload traffic.

[1] Uniformly distributed packet sizes in range 100–200 bytes was used, making the average 150 bytes/packet, in emulating MOOS traffic.

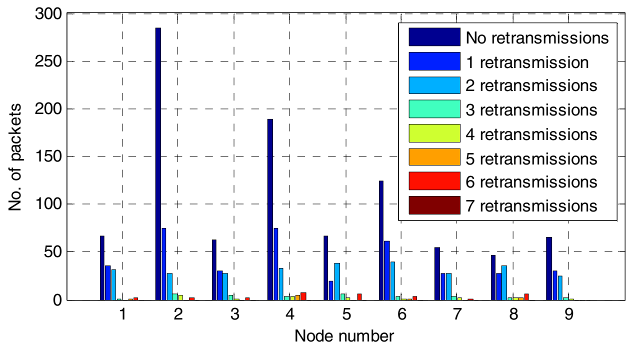


Figure 6. Number of retransmissions due missing ACK packets

Figure 6 shows the performance of the CSMA/CA scheme, in that the number of needed retransmissions is shown (node 1 is the master). The reason for these is missing ACK packets, as a result of packet collisions. Packets are discarded if the 7th retransmission was unsuccessful or that the sending buffer is full (packets for relaying has preference before own generated packets).

H. *Experimental results*

The underwater modems and underwater network was tested autumn 2010 and spring 2011 with five cNode Mini modems and dedicated Linux hosts. The test locations were the Pianosa Island (master node at 42° 35.532'N, 10° 6.514'E), Italy, and Faro (master node at 37° 0.263'N, 7° 59.275'W), Portugal, respectively. Both locations are characterized by shallow water, especially at Faro. Two of the modems/hosts were located in the mobile Folagas, while the rest were static. One of the static nodes was assigned as master. Due to hardware problems two of the modems failed in the Pianosa tests. Thus, this test was limited to setting up a connection with two slaves only towards the master, one of these mounted on a mobile Folaga. It was verified that the modem routing could be changed from a direct path to master, to a topology where one of the slaves acted as relay for the other. This re-routing could take place during a PPP session, because it is totally transparent to this layer. The Folaga was moved to cause variable physical link conditions. It was verified that the PPP link was still functioning once the physical link conditions were re-established. One ping session results are shown in Figure 7 where the Folaga was used as relay for the fixed node (one ping packet of 10 bytes payload each 20 second, 0 RTT denotes lost packet). In the Faro tests it was verified that all four slaves could be connected simultaneously, and that the slaves could connect and eventually reconnect independently.

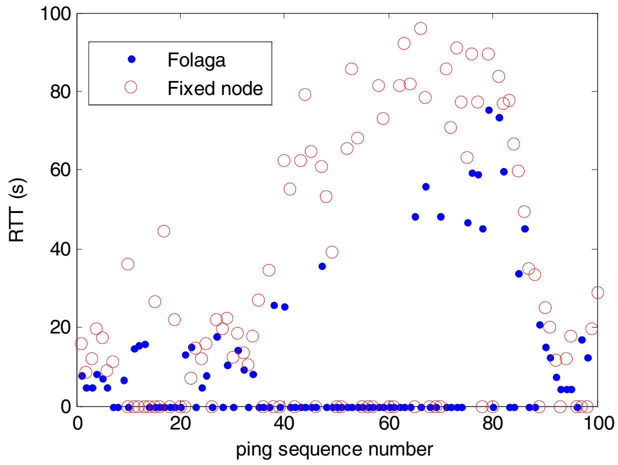


Figure 7. Pianosa ping results

IV. Turbo Equalization

A. *Single Carrier or OFDM*

Implementation of an uw-modem starts with choosing a communication method that has a large potential. To motivate our choice we have to take a glimpse into the information theory. A commonly used modulation format in rich multipath environments is Orthogonal Frequency Division Multiplex, OFDM. The reason is that long symbols reduce the influence of Inter Symbol Interference, ISI. However, the subchannels will be flat Rayleigh fading channels. The multipath arrivals can not be resolved in this case, so we have only a single path channel. This means that it is necessary to use error correcting coding over time or frequency in order to correct for the errors caused by fading dips. This is in contrast to Single Carrier, SC, where the multipath arrivals can be resolved and a matched filter will weight and phase-align all arrivals. This operation is usually considered as matched filtering and equalization in series. But we consider them jointly, as a single matched filter. This can be done when the equalizer is chosen using the MMSE criterion. To study the potential of each system, respectively, we can investigate the channel capacity as introduced by Shannon. To do that we need the signal to noise ratio, *SNR*. Given a multipath channel with *L* arrivals, the impulse response vector **h**, a matched filter at the receiver, and perfect elimination of ISI, the *SNR* for SC is:

$$SNR_{SC} = \frac{\sigma_s^2 \left|\mathbf{h}\mathbf{h}^H\right|^2}{\sigma_n^2 \mathbf{h}\mathbf{h}^H} = \frac{\sigma_s^2}{\sigma_n^2}\sum_{l=1}^{L}\left|h_l\right|^2 \qquad (1)$$

where σ_n^2 and σ_s^2 are the noise- and signal variances, respectively. Note that in this case, the optimal filter only weights and sums the delayed signal contributions the same way as is done in the maximal ratio combiner and rake receiver. The optimal filter gives optimal combination of arriving signals for largest SNR. Thus, from a signal to noise view the multipath contributes to increase the received signal energy. Assume that the impulse response has been measured and thus can be considered given. No randomness is therefore in the channel parameters. The capacity per symbol for SC is:

$$C_{SC} = \log_2\left(1+\frac{\sigma_s^2}{\sigma_n^2}\sum_{l=1}^{L}|h_l|^2\right) \tag{2}$$

Since a comparison with an OFDM system with M sub channels will be done, zero pad the impulse response vector, $\mathbf{h}$, to containing M elements, which we will call $\widetilde{\mathbf{h}}$. The energy conservation implies that the number of samples is insignificant, and thus:

$$\sum_{l=1}^{L}|h_l|^2 = \sum_{m=1}^{M}\left|\widetilde{h}_m\right|^2 \tag{3}$$

The DFT of the vector $\widetilde{\mathbf{h}}$ is denoted $\widetilde{\mathbf{H}}$. Using Parseval's theorem we get:

$$\sum_{m=1}^{M}\left|\widetilde{h}_m\right|^2 = \frac{1}{M}\sum_{m=1}^{M}\left|\widetilde{H}_m\right|^2 \tag{4}$$

C_{SC} in equ. (2) can then be rewritten:

$$C_{SC} = \log_2\left(\frac{1}{M}\sum_{m=1}^{M}\left(1+\frac{\sigma_s^2}{\sigma_n^2}\left|\widetilde{H}_m\right|^2\right)\right) \tag{5}$$

This gives the maximal channel capacity when single carrier is used. How close we get to this limit depends on the implementation at hand, for example how well the channel is estimated.

For OFDM, each channel will have its own SNR, which is:

$$SNR_m = \frac{\sigma_s^2\left|\widetilde{H}_m\right|^2}{\sigma_n^2} \tag{6}$$

The total capacity per symbol for the OFDM signal is the mean value of the capacity of all subcarriers:

$$C_{OFDM} = \frac{1}{M}\sum_{m=1}^{M}\log_2\left(1+\frac{\sigma_s^2\left|\widetilde{H}_m\right|^2}{\sigma_n^2}\right) \tag{7}$$

Using the transformation 2^C, on equ. (5) and equ. (7) gives:

$$\left(\prod_{m=1}^{M}\left(1+\frac{\sigma_s^2\left|\widetilde{H}_m\right|^2}{\sigma_n^2}\right)\right)^{1/M} \le \frac{1}{M}\sum_{m=1}^{M}\left(1+\frac{\sigma_s^2}{\sigma_n^2}\left|\widetilde{H}_m\right|^2\right) \tag{8}$$

To the right we have an arithmetic mean, which is always larger or equal to the geometric mean, on the left side. This means that the single carrier has always better capacity as long as we have the same transmitter power in each OFDM channel. However, of course it is possible to achieve the Shannon bound with OFDM also. The means to do this is water filling, which means that we invest power where it pays off the best. But to do that the receiver has to feed back M channel estimates. In the uw-channel with its rapid variations in combination with the long propagation delay, these estimates are outdated before they reach the receiver. This should be compared to a single parameter for SC, which is also much more constant because of the averaging effect from the multipath.

When choosing a communication method a lot of other issues have to be considered. Generally one can say that SC gives a simple transmitter and a complicated receiver. The opposite is true for OFDM.

B. Iterative equalization

For channel equalization we use the low-complexity soft-input/soft-output (SISO) MMSE equalizer by Tüchler et. al. [9]. More specifically it is a single-input/multiple-output (SIMO) implementation that was published by Dejonghe et.al. [10] and Wautelet et.al. [11]. Turbo equalization has been described in several papers so we only give an overview here. Given a received sample vector $\mathbf{r}_k$, linear estimates $\hat{s}_k$ of the emitted symbols s_k are formed according to:

$$\hat{s}_k = \mathbf{w}_k^H\left(\mathbf{r}_k + \mathbf{H}_k\bar{\mathbf{s}}_k\right) \tag{9}$$

where $\mathbf{w}_k$ is the equalizing filter and $\mathbf{H}_k$ is an $L\times 2L-1$ band matrix with the impulse response vectors $\mathbf{h}_{k-L1},\ldots,\mathbf{h}_{k+L2}$, in the band and $L1+L2+1=L$ is the length of the channel impulse response. The vector $\bar{\mathbf{s}}_k$ is the expected values of the estimated symbols within the span of the impulse response.

In the SISO equalizer the estimate of symbol k, $\hat{s}_k$, will depend on the *a priori* information of the emitted symbols in the form of symbol means and variances, $\bar{s}_k$ and v_k. However, when estimating symbol k, in a recursive equalizer it is important that the *a priori* information about symbol k is cancelled by setting $\bar{s}_k = 0$ and $v_k = \sigma_s^2$, where σ_s^2 is the symbol variance of the constellation. In this way only extrinsic information gained from surrounding symbols is used, which leads to more stable convergence.

Extrinsic information obtained by the equalizer about the transmitted bits is sent in the form of log-likelihood (LLR) ratios to the decoder. The latter uses the BCJR algorithm [12] to decode the parallel concatenated code that was used here. The output from the decoder is new soft information about the infobits and the codebits. When calculating bit extrinsic information the same strategy is used, that is, only information gained from bits "surrounding" the one under inspection is used. A hard decision on the soft infobit LLR:s is used to test the CRC (Cyclic Redundancy Check) that is part of the dataframe. If the CRC is met, the iteration is terminated and decoded infobits are delivered to their final destination. If on the other hand the CRC is not met, the soft information gained by the decoder is fed back as new a priori information about the emitted bits for another iteration.

Concerning computational complexity, the most intense part of the receiver programming code is estimating the filter parameters. This involves solving a system of linear equations. The size of the system is largely determined by the impulse response length (expressed in symbols or fractions thereof) and it is performed for each symbol of each frame. By using the fact that: going from one symbol to the next entails only a small change of the impulse response matrix, the inverse can instead be maintained through low-rank updates. This will make the update procedure O(L^2) instead of O(L^3), i.e. potentially a very large saving [9].

C. *Channel estimation*

The foundation for reliable underwater communication is a good channel estimate. The uw-channel is inherently unstable and sometimes even approaching overspread. The usual way to estimate the channel in an iterative receiver is to find the least squares solution. Given a received signal vector $\mathbf{r}_k$, the estimated channel state at time k is, ref. [8]:

$$\hat{\mathbf{h}}_k = \arg\min_{\mathbf{h}_k} E\left\{\left|\mathbf{r}_k - \mathbf{H}_k \bar{\mathbf{s}}_k\right|^2\right\} \tag{10}$$

Here $\bar{s}_k$ is not nulled in the $\bar{\mathbf{s}}_k$ vector. However in the first iterations the decoder is insecure of its estimation and the expected values are rather small, i.e. amplitudes are close to zero. This will lead to the effect that the values of $\hat{\mathbf{h}}_k$ will be inflated. Given a constant envelope modulation, in this case 4-QAM, we know the amplitude of the symbols. The amplitude of each symbol is $|\bar{s}_k| = 1$. Therefore we can say that the symbols are located on a ring with radius one. Then the uncertainty of the received symbol will only be reflected in the angle of the estimated symbol. Let us choose the most probable symbol at that ring as our estimate, i.e. make a hard decision. Denote this estimate $\breve{s}_k$. Let us also use the good channel estimates we have during the training sequences. Each tap in the channel response can be expressed in the polar domain by $h_k = a_k \exp(j \cdot \theta_k)$. Denote $a_{k,b}$ and $a_{k,a}$ the amplitudes measured during the training sequence before and after the data frame, respectively. Denote the phases $\theta_{k,b}$ and $\theta_{k,a}$ in the same way. Then let us make a simple model of the evaluation of the channel taps during the data frame by:

$$\hat{h}_k^{\xi} = \left(a_{k,b}\left(1 - \frac{k}{K}\right) + a_{k,a}\frac{k}{K}\right) \cdot \exp\left(j\left(\theta_{k,b}\left(1 - \frac{k}{K}\right) + \theta_{k,a}\frac{k}{K}\right)\right) \tag{11}$$

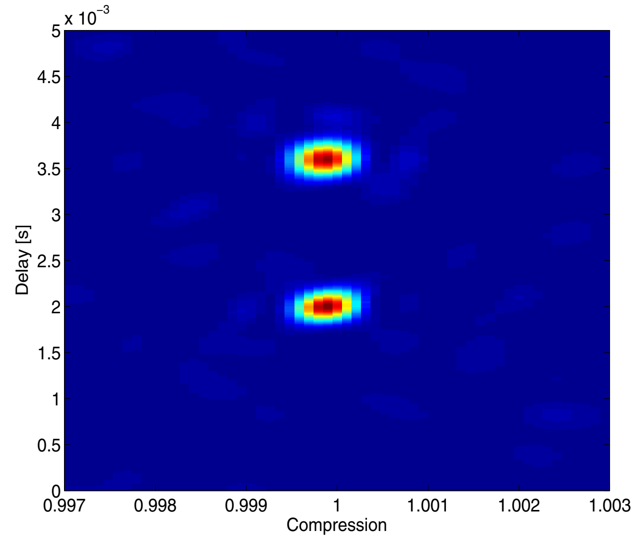
where K is the number of symbols in a data frame. The least squares solution is approximated by an LMS solution. However, if the reception is bad the received signal and our estimated symbol will be completely uncorrelated. The channel update will be zero. By using our channel model, in eq. (11), as the mean value of the channel estimate, the estimate will follow what is predicted by the training sequences before and after the data frame. The LMS update will be:

$$\hat{\mathbf{h}}_k^{\Delta} = \hat{\mathbf{h}}_{k-1}^{\Delta} - \mu\left(r_k - \left(\hat{\mathbf{h}}_{k-1}^{\Delta} + \hat{\mathbf{h}}_k^{\xi}\right)^H \breve{\mathbf{s}}_k\right)\breve{\mathbf{s}}_k^H \tag{12}$$

where μ is the LMS step size and $\hat{\mathbf{h}}_k^{\Delta}$ is the incremental channel estimation and the total estimate will then be:

$$\hat{\mathbf{h}}_k = \hat{\mathbf{h}}_{k-1}^{\Delta} + \hat{\mathbf{h}}_k^{\xi} \tag{13}$$

This is used as the channel estimate in the decoding and will give a channel estimate that has better properties during the start of the iterations, which is crucial for good results. When the uncertainties about the decoded symbols are reduced, the estimate will coincide with the standard least square estimate.



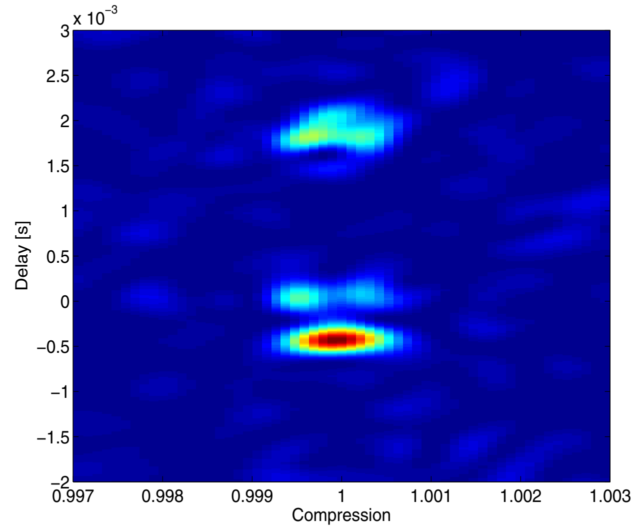


Figure 8. Compression ambiguity function before (top) and after (bottom) a data frame. The training sequences before and after the data frame are used for calculating the ambiguity. The data set, including the data frame and the training sequences, is approximately 370 ms long.

D. *Results of channel measurements*

To see what kind of channel we encounter, we now show some analysis results of the data obtained during one seatrial in the Oslofjord during winter 2011. The received signals were obtained from one of the bottom nodes. In particular we studied the channel impulse response, which varied significantly over the duration of a frame. This of course violates the common assumption that the impulse response is constant over a single data frame. With the channel tracking, which was explained in the previous section, time variation can be allowed without interruption of the communication. A transmission consists of a data frame with a training sequence before and after. The training sequences consist of 255 pilots each. The data frame consists of 960 symbols and the complete data set is approximately 370 ms long.

The compression ambiguity function, which is used for the channel estimation and compensation of the doppler effect, is calculated using the training sequences before and after one of the data frames. An example is shown in Figure 8. Thus, the interpolation in eq. (11) is done between the values of the ambiguity function at compression 1, taken from the training sequences surrounding the data frame.

We see in the upper figure that there is a slight compression of the training sequence before the data frame, and that there are two distinct contributions to the impulse response. The lower figure shows three signal arrivals with a larger separation in time. Two of the arrivals also show a split in the doppler direction.

The initial doppler compensation is estimated using only the training sequence before the data frame. This could be a problem when we have split contributions after the frame. However, the mean value of the doppler of all taps is approximately the same as the one before the data frame, and the frame also decodes without problems.

V. Modem Range Test

In the following are test results presented from a sea trial in the Breidangen area in the Oslofjord in February 2011. The objective of the test was to test the range capability of the UAN modem using transport format 3 including the turbo equalizer. The Breidangen area is characterized by a bathymetry of approximately 200 meter over the most of the area, having a soft seabed with mostly mud-like sediments. Throughout the day was the wind speed measured to an average speed of 25 knots from the North. Figure 9 presents the measured sound velocity profile with a corresponding ray trace plot from Kongsberg Maritimes APOS – Acoustic Positioning Operator Station. APOS is the operator station from which the modem is controlled.

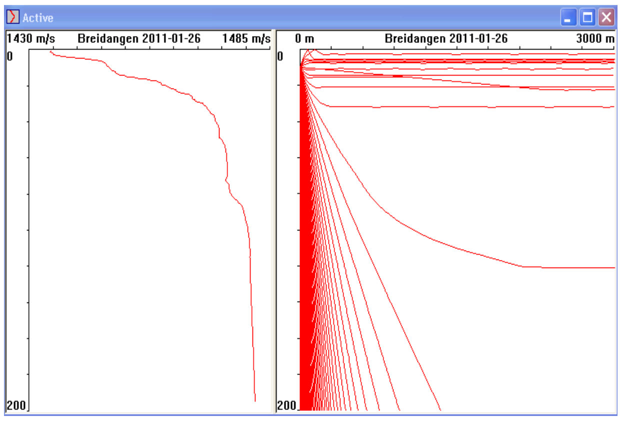


Figure 9. Typical sound velocity profile (left) and ray trace plot (right) for the period

A. *Measurement setup*

The modems were tested according to the setup in Figure 10 and Figure 11 were two[2] modems were placed in basket at the seabed (200 m depth) and another two modems suspended in a rope from the buoy moored to the modems at the seabed. Identification of the modems is done by their serial numbers. Modem 10039 and 10103 was placed with a distance of approximately 3 meters at an average depth of 10 meter. On board the boat was the master modem with the transducer also at approximately 10 m depth. Most of the test was performed with the boat drifting away from the modems due to the rather strong wind.

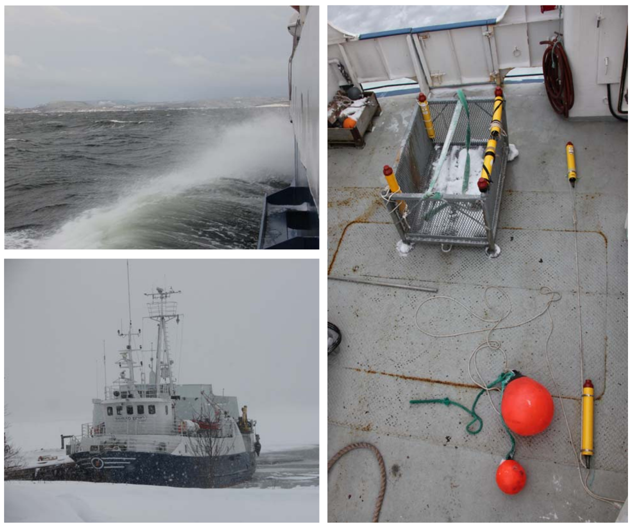


Figure 10. Modem range test scenario and equipment including all 6 UAN modems

Due to the computational complexity the length of the equalizing filter in equation (9) need to be kept short. The maximum length of the filter was configured to be 15 to meet the near real time requirements. The real time requirement was set to a real time factor of 5, meaning that the one second telegram could maximum take 5 seconds to decode. Accordingly, the maximum number of iterations was set to 3 since the processing time is predominantly dominated by the number of iterations needed for successful decoding of the telegram. Utilization of the limited length equalizing filter on the actual impulse response was discovered by configuring the modems with different windowing with reference to (9). Table 3 gives the overview of the windowing in addition to the threshold level for which arrivals are discarded and not included in the filter.

[2] Only two of the modems in the steel basket shown in Figure 10 was used in the actual range test

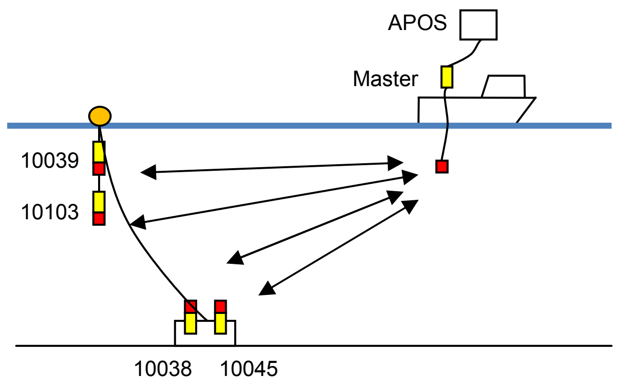


Figure 11. Measurement setup

B. *Results*

The test was performed by sending 80 Byte payload telegrams from the master to a modem and waiting for a reply with the copy of the same payload. The transmission was noted as success if a reply was received and successfully decoded within a predetermined time out. Modems were automatically tested in sequence as fast as propagation time and processing time allowed. Total test time from starting transmissions until no contact was approximately 2 hours. From time to time was the test intermittently stopped to log signals on the master for post analyzes of transmissions that was not decoded correctly at the master side.

Table 3 Turbo equalizer parameters

Modem	L1 max	L2 max	Threshold
10039	2	12	-6 dB
10103	6	8	-10 dB
10038	6	8	-10 dB
10045	2	12	-6 dB
Master	6	8	-10 dB

Figure 12 show the success rate of the transmissions for the different modems as a function of distance. Successful transmissions are plotted as a blue marker with value 1, while unsuccessful transmissions are plotted as a blue marker with value 0. The solid line shows the moving average over 11 transmissions. The results show a higher success rate for bottom modems compared to the surface modems. Another general trend is that coverage is not gracefully degraded as a function of distance, but varies substantially within some range intervals. Modem 10045 demonstrates successful transmissions up to 1800 m before contact is lost and gives the most stable connection of the modems. The longest range of 2500 m is demonstrated by modem 10038.

Previous experience from the measurement site says that these results are good. Post analysis of signals logged for not successful transmissions showed similar performance when run in a Matlab model of the algorithms with the same filter lengths and parameters as implemented in the modem.

C. *Further work*

The turbo equalizer has been demonstrated by many others to be successful. Further tuning of the implementation on the modem for more efficient computation should be possible. This will allow longer equalizing filter lengths and further improved performance is expected.

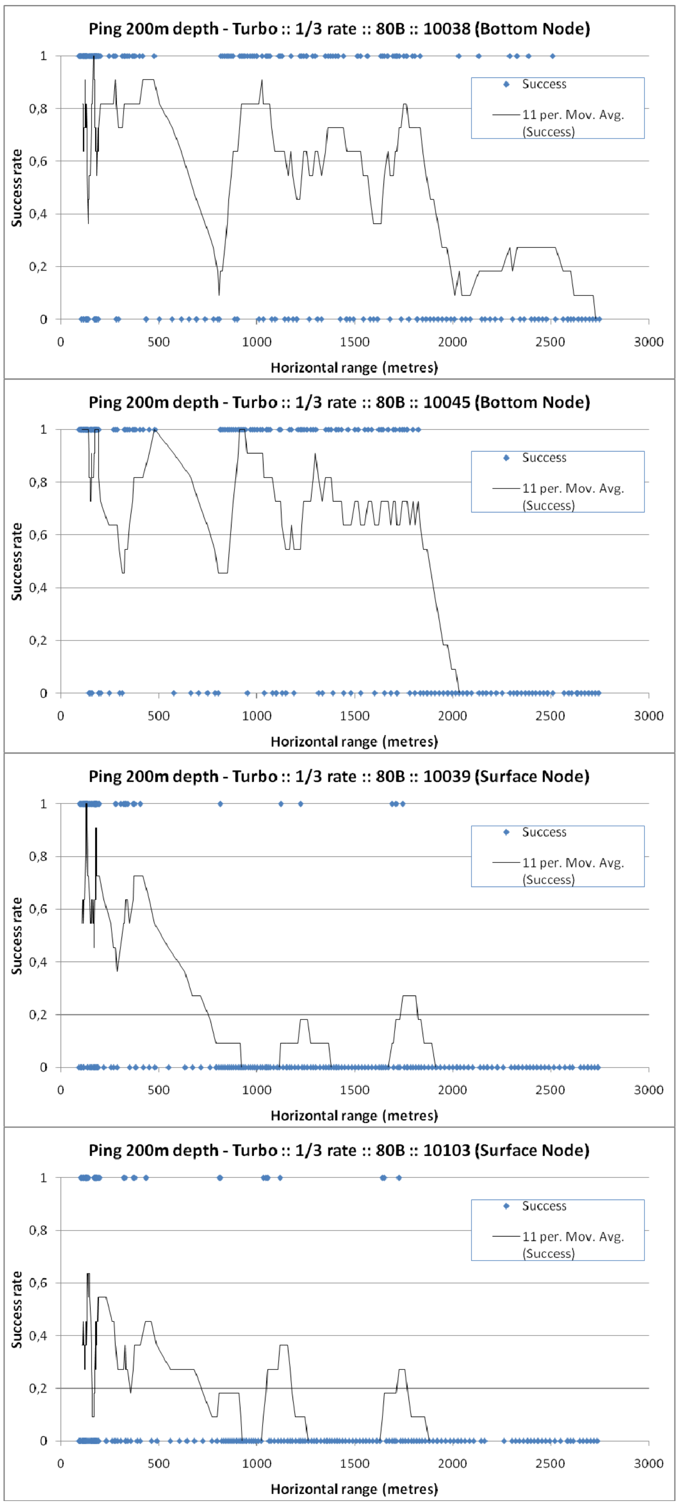


Figure 12 Telegram success rate as function of range

VI. Conclusion

The network layer of the modem has been refined and upgraded with support for moving nodes. The network discovery function FLOOD and multi hop routing has been verified in sea trials were also IP end to end communication has been demonstrated. This is made possible through a modem interface utilizing UDP/IP header compression through the PPP protocol on a Linux host. The physical layer of the modem has been upgraded with the turbo equalization algorithm. Demonstration in sea trials shows horizontal communication at 1710 bps in a 200 m deep channel up to 1800 meter range. These are promising results and further improvement should be possible through tuning of the algorithms and its implementation for a low power DSP.

Acknowledgment

The authors would like to thank the colleagues in the UAN project for contributing to the engineering tests at Pianosa and in Faro allowing full integration tests of the modem. Thank you also to Knut Grythe, SINTEF for contributing to physical layer issues related to the modem.

References

[1] Underwater Acoustic Network Project, http://www.ua-net.eu/ [accessed; Mar 22, 2011]

[2] H. Rustad, “A Lightweight Protocol Suite for Underwater Communication,” in International Conference on Advanced Information Networking and Applications Workshops, Bradford UK, May 2009.

[3] Faugstadmo, J. E., Pettersen, M., Hovem, J. M., Lie, A., Reinen, T. A. 2010. Underwater Wireless Sensor Network. Presetned at the Fourth International Conference on Sensor Technologies and Applications, 2010

[4] V. Jacobson, “Compressing TCP/IP headers for low-speed serial links,” IETF RFC 1144, Feb. 1990.

[5] M. Engan et al., “IP Header Compression over PPP,” IETF RFC 2509, Feb. 1999.

[6] W. Simpson et al., “The Point-to-Point Protocol (PPP),” IETF RFC 1661, July 1994.

[7] J. M. Hovem et al., “Modeling Underwater Communication Links,” in IEEE 2nd International Conference on Sensor Technologies and Applications, France, Aug. 2008, pp. 679-686.

[8] R. Otnes and M. Tuchler, "Iterative channel estimation for turbo equalization of time-varying frequency-selective channels," *Wireless Communications, IEEE Transactions on,* vol. 3, pp. 1918-1923, 2004.

[9] M. Tüchler, A.C. Singer, R. Koetter, “Minimum Mean Squared Error Equalization Using A Priori Information”, IEEE Transactions on Signal Processing, vol 50, no 3, 2002.

[10] A. Dejonghe, L. Vandendorpe, “Turbo-equalization for multilevel modulation: an efficient low-complexity scheme”, IEEE International Conference on Communications ICC-2002, vol 3, pp. 1863 – 1867, 2002.

[11] X. Wautelet, A. Dejonghe, L. Vandendorpe, “MMSE-Based Fractional Turbo Receiver for Space-Time BICM Over Frequency-Selective MIMO Fading Channels”, IEEE Transactions on Signal Processing, vol 52, no 6, 2004.

[12] L.R. Bahl, J. Cocke, F. Jelinek, J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate”, IEEE Transactions on Information Theory, vol. 20, pp. 284-287, March 1974.