INT J COMPUT COMMUN, ISSN 1841-9836
8(6):800-811, December, 2013.

Adaptive Network Coding Scheme for TCP over Wireless Sensor Networks

Y.-C. Chan, Y.-Y. Hu

Yi-Cheng Chan*, Ya-Yi Hu
Department of Computer Science and Information Engineering,
National Changhua University of Education
No.2, Shi-Da Road, Changhua City 500, Taiwan
ycchan@cc.ncue.edu.tw, m9954016@mail.ncue.edu.tw
*Corresponding author: ycchan@cc.ncue.edu.tw

Abstract: The purpose of this paper is to develop a network coding scheme to enhance TCP performance in wireless sensor networks. It is well known that TCP performs poorly over wireless links which suffer from packet losses mainly due to the bad channel. To address this problem, it is useful to incorporate network coding into TCP, as network coding can offer significant benefits in terms of throughput, reliability, and robustness. However, the encoding and decoding operations of network coding techniques will bring an additional delay that has a negative effect on applications of wireless sensor networks. In this paper, we propose an adaptive network coding (ANC) scheme which contains two major aspects: the adjustment of the redundancy factor R and the adjustment of the coding window size CW. We dynamically adjust these two parameters depending on the measured packet loss rate, so that the proposed ANC can effectively mask packet losses and reduce the decoding delay of network coding. The performance of our scheme is evaluated by simulations using NS-2 simulator. Compared to other schemes, the ANC not only achieves a good throughput but also has the lowest average delay and the lowest maximum delay in all experimental environments.

Keywords: network coding, TCP, delay, wireless sensor networks.

1 Introduction

The Transmission Control Protocol (TCP) is the main transport protocol that provides reliable transmission in the current Internet. It has been developed for many years. Most applications on the Internet depend on TCP to ensure safe delivery of data, such as FTP (File Transfer Protocol), HTTP (HyperText Transfer Protocol), SMTP (Simple Mail Transfer Protocol), and so on. TCP performs well in wired networks where packets losses mainly occur due to congestion. However, the performance of TCP degrades very fast in wireless networks. In a wireless environment, there is not only congestion but also numerous other reasons for packet losses exist. Traditional TCP treat a packet loss event as the indication for network congestion, and then decrease its congestion window size. This may severely impair the throughput when the TCP runs on a wireless channel. Therefore, it is very important to improve the performance of TCP in wireless networks. And this goal can be achieved by combining network coding with TCP.

Network coding is a new transmission paradigm originally proposed by Ahlswede et al [1]. In recent years, it has received much attention and has been generated huge research in communication networks [2]. TCP/NC [3] is the first one that incorporates network coding into TCP with minor changes to the protocol stack. They present a solution which embeds the network coding operation in a separate layer below transport layer and above network layer on the source and receiver side. The idea of network coding is that, instead of transmitting individual packets, the sender takes several packets and combines them together for transmission. Consequently, successful reception of information does not depend on receiving specific packet content but rather on receiving a sufficient number of combinations [4]. Then it can compensate for the packets in the presence of random losses, as long as the sent redundant combinations are enough. This characteristic is very attractive for TCP to improve the robustness and effectiveness of data transmission over lossy wireless networks.

Wireless sensor networks (WSNs) are usually composed of a large number of radio-equipped sensor devices to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion, or pollutants. These devices typically include some strong constraints in terms of energy, memory, computational speed and communication bandwidth. In fact, most applications on wireless sensor networks prefer faster and reliable packet delivery to higher throughput [5]. Generally, these applications are not only loss-sensitive that require successful transmission of all packets or at a certain success ratio but also delay-sensitive that require timely delivery of data [6]. For these applications, packet loss will lead to retransmission and the inevitable consumption of additional battery power. In addition, if the delay time is too long, the monitored data may be outmoded. To overcome these problems it is beneficial to use network coding by transmitting redundant packets to mask packet losses. Thus, the number of retransmissions and timeouts can be reduced.

In this paper, we propose a new network coding scheme to improve TCP performance over wireless sensor networks which is called adaptive network coding (ANC). The concept of ANC is divided into two parts. In the first part, we adjust the redundancy parameter dynamically according to the network situation. As a result, the event of packet loss can be masked from the congestion control algorithm of TCP by sending enough redundant combinations. In the second part, we change the coding window size contingent on the measured packet loss rate to obtain the optimal throughput-delay trade-off. The results of simulation show that our scheme achieves higher throughput and lower delay in difference network scenarios.

The remainder of this paper is organized as follows. In Section 2 we provide an overview of related work. Section 3 we propose our schemes and algorithms focus on the redundancy parameter and the coding window size of network coding. In Section 4 we present an experimental evaluation of the performance, and finally, our conclusions as well as the future work are discussed in Section 5.

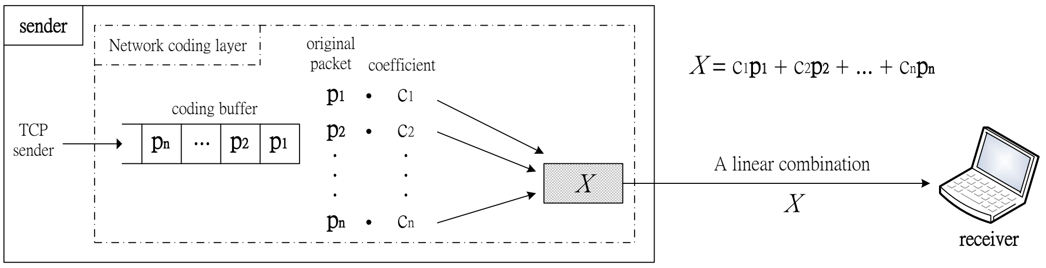


Figure 1: An example of random linear network coding

2 Related Work

Random linear network coding is one of the major techniques in the network coding. The encoding coefficients are randomly chosen from a set of coefficients of a finite field. A linear equation of packets is then performed to generate a coded packet, and the receiver only needs to receive a sufficient number of linear equations in the forms of the coded packets to successfully decode the original packets. For instance (see Figure 1), suppose that the sender buffers the first

n packets $p_1, \dots, p_n$ in its coding buffer, then the sender chooses n coefficients $c_1, \dots, c_n$ from a finite field and mates n coefficients with n packets. Next, the sender encodes these pairs into a linear combination X, $X = c_1 p_1 + c_2 p_2 + \dots + c_n p_n$, then sends it to the receiver. If the receiver receives enough number of these combinations then it can decode the original packets. In recent years, there have been many studies make use of random linear network coding to improve TCP performance.

The main aim of TCP/NC [3] is to mask losses from TCP using random linear coding. The source transmits random linear combinations of packets currently in the congestion window, and the receiver acknowledges every innovative linear combination it receives, even if it cannot decode an original packet immediately. This scheme gives a new interpretation of ACKs, and brings a new concept that "seen packet" which is defined in [7] as an abstraction for the case in which a packet cannot yet be decoded but can be safely removed from the coding buffer at the sender. TCP-Vegas is chosen for the transport layer protocol in TCP/NC, as it is more compatible with their modifications. When losses are effectively masked, TCP-Vegas can infer congestion from increased $RTTs$. Additionally, TCP/NC uses a constant redundancy parameter R to compensate for the loss rate of the channel, for every packet arrives from TCP, R linear combinations are sent to the IP layer on average. However, in a wireless environment, the packet loss rate is very likely to not a constant value. A fixed redundancy parameter R may damage network performance, since it is not always suitable for all network conditions. If the value of parameter R is too small, then the losses are not effectively masked from the TCP layer and will result in timeout. On the other hand, if the value of parameter R is too large, the source may send too many linear combinations that consume network resources. Thus, the value of R should be dynamically adjusted depending on the estimated packet loss rate.

The feedback based network coding (FNC) [8] uses the implicit information behind of the seen scheme to find out the exact number of packets needed by receiver to decode all data. The receiver computes the $DIFF$ value by two variables: the number of seen packets and the largest packet index in the coefficient matrix, and then embeds this value into the ACK header. When the sender receives an ACK, it uses this value to decide how many random linear combinations should be retransmitted and how many original packets should be combined in a linear combination. In this way, the FNC retransmission scheme can reduce the decoding delay and the number of redundant retransmissions. But the FNC retransmission scheme highly relies on feedback, this will greatly impair the throughput when the network with a large round-trip time. Furthermore, if the ACK is lost, the sender cannot repair the loss of linear combinations through instant retransmissions.

The SANC-TCP protocol [9] is designed primarily to optimize the TCP/NC protocol. The redundancy factor R of TCP/NC is constant, while SANC-TCP adjusts the redundancy factor R adaptively based on existing network conditions. In order to implement this approach, SANC-TCP adds some information in the ACK header to indicate the current network state, thus enable the sender to dynamically change the value of R according to the real system. Our scheme is similar to the SANC-TCP that with dynamic adjustment of redundancy factor R, but in different ways. Moreover, we limit the coding window size and dynamically change it depending on the packet loss rate to reduce the delay of network coding.

3 Proposed Method

In this section, we describe our adaptive network coding scheme (ANC) which consists of two parts. First, we adjust the redundancy factor R dynamically by estimating the packet loss rate in current network, so that the value of R can represent the actual network state. Second, in order to reduce the decoding delay of network coding, we limit the coding window size and dynamically adjust it contingent on the measured packet loss rate.

3.1 Adjustment of the redundancy factor R

Before adjusting the redundancy factor R, we have to find the packet loss rate of current network. In our scheme, we insert two variables into the header of ACK, which are *send_count* and *seen_count*. The variable *send_count* refers to the number of packets that have been sent from the sender and the variable *seen_count* refers to the number of seen packets at the receiver. Then, the sender can use these and other related variables to calculate packet loss rate. For example, when the ACK arrives from receiver, the sender retrieves the variables *send_count* and *seen_count* from ACK header and compares *seen_count* with a threshold T which is the variable used to determine if it is time to adjust the value of R. If *seen_count* is more than or equal to T, the sender starts to calculate the packet loss rate by the following equations:

$$diff\_send = send\_count - send\_old, \tag{1}$$

$$diff\_seen = seen\_count - seen\_old, \tag{2}$$

$$loss\_rate = \frac{diff\_send - diff\_seen}{diff\_send}, \tag{3}$$

where *send_old* and *seen_old* are the previous value of *send_count* and *seen_count* respectively. The initial *send_old* and *seen_old* is set to 0. The $diff\_send$ refers to the number of packets that have been sent during this calculation cycle while $diff\_seen$ refers to the number of seen packets during this calculation cycle. Therefore, the *loss_rate* means the packet loss rate of current network. Through the above equations, we can obtain the accurate packet loss rate to conduct the correct adjustment of redundancy factor R.

After calculating the packet loss rate, the sender adjusts the redundancy factor R accordingly. If current T is equal to *init* means this is the first time for the sender to adjust the value of R. In the simulation, we choose $init = 20$ which is the same as initial coding window size. The equation below shows the calculation of the redundancy factor R for the first time:

$$R_{current} = \frac{1}{1 - loss\_rate}, \tag{4}$$

where $R_{current}$ is current redundancy factor R. On the other hand, if it is time to adjust the value of R after the first calculation, the value of R can be obtained as follow:

$$R_{current} = a * \frac{1}{1 - loss\_rate} + b * R_{old}, 0 \leq a, b \leq 1, a + b = 1. \tag{5}$$

The above equation is calculated using moving average to take into account the previous trends on R_{old}, where R_{old} is the last value of redundancy factor R. The values of variables a and b can be set based on the network situations. If the random loss rate of the network environment tends to change frequently, the value of a should be set greater than b. In contrast, if the random loss rate of the network environment tends to change occasionally, the value of a should be set less than b.

3.2 Adjustment of the coding window size

The coding window size represents that the largest number of original packets can be encoded in a linear combination. To reduce the decoding delay of network coding, we limit the coding window size and adjust it according to the packet loss rate.

After the sender computed the redundancy factor R, the next process for the sender is to adjust the coding window size. The following equation describes the adjustment of coding window size for the first time:

$$CW = 20 + \lfloor loss\_rate * 10 \rfloor, \tag{6}$$

where CW refers to the coding window size. In our scheme, the coding window size is dynamically adapted depending on the packet loss rate. This means that the coding window size increases as the packet loss rate increases, and the coding window size decreases as the packet loss rate decreases. After the first calculation, the value of coding window size can be obtained as follow:

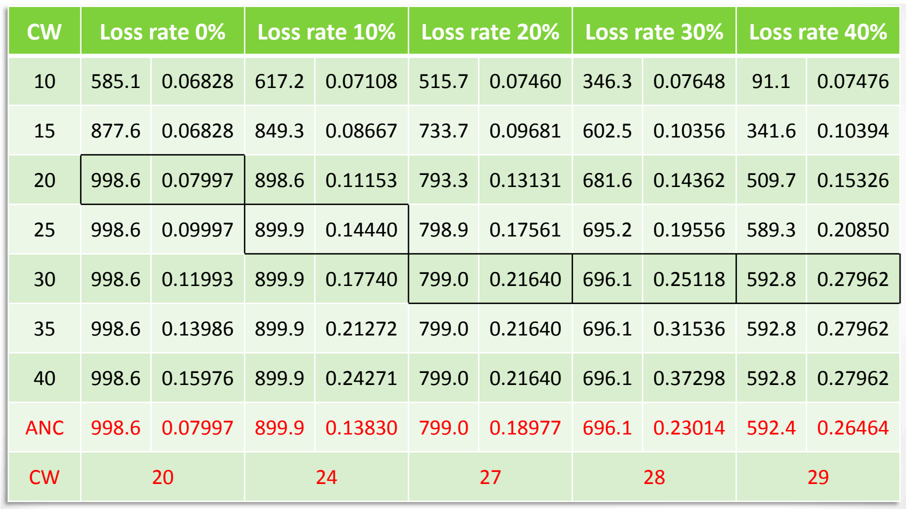
$$CW = 20 - \left\lfloor \frac{1 - R_{current}}{R_{current}} * 10 \right\rfloor . \tag{7}$$

The above equation shows that the packet loss rate estimation for adjusting the coding window is derived from the value of $R_{current}$ rather than the measured packet loss rate at this period. That is because the value of $R_{current}$ is used to predict the packet loss rate in the next period.

Finally, we must update several related variables, such as R_{old} to $R_{current}$, $send\_old$ to $send\_count$ and $seen\_old$ to $seen\_count$. The value of T also should be reset by the following equation:

$$T = CW + seen\_count. \tag{8}$$

Table 1: Coding window size adjustment in difference loss rates



CW	Loss rate 0%		Loss rate 10%		Loss rate 20%		Loss rate 30%		Loss rate 40%	
10	585.1	0.06828	617.2	0.07108	515.7	0.07460	346.3	0.07648	91.1	0.07476
15	877.6	0.06828	849.3	0.08667	733.7	0.09681	602.5	0.10356	341.6	0.10394
20	998.6	0.07997	898.6	0.11153	793.3	0.13131	681.6	0.14362	509.7	0.15326
25	998.6	0.09997	899.9	0.14440	798.9	0.17561	695.2	0.19556	589.3	0.20850
30	998.6	0.11993	899.9	0.17740	799.0	0.21640	696.1	0.25118	592.8	0.27962
35	998.6	0.13986	899.9	0.21272	799.0	0.21640	696.1	0.31536	592.8	0.27962
40	998.6	0.15976	899.9	0.24271	799.0	0.21640	696.1	0.37298	592.8	0.27962
ANC	998.6	0.07997	899.9	0.13830	799.0	0.18977	696.1	0.23014	592.4	0.26464
CW	20		24		27		28		29	

3.3 Analysis of the coding window size

The initial value of coding window size in our scheme is acquired by the experimental results presented in Table 1. For this experiment, the topology and experimental parameters are the same as that in the subsection 4.1.1. Table 1 shows the data of throughput (TP) and delay which are obtained by adjust the coding window size in difference loss rates. The data that have an outline border is the optimal trade-off value when compared with others data in the same loss rate. The last two rows in Table 1 are the experimental results and the coding window adjustments of our scheme ANC. For the initial setting, we assume the packet loss rate at beginning of network is zero, so we set the initial coding window size to 20 that is the optimal value in Table 1 when the loss rate is 0%. After this, the sender adjusts the coding window size in accordance with the measured packet loss rate. This experimental result (see Table 1) indicates that the throughput of our scheme at last row can reach the optimal value in most cases while has a shorter delay, that is, the adjustment of the coding window size in our scheme is suitable.

As mentioned before, in order to reduce the decoding delay of network coding, we limit the coding window size in our scheme. When the coding window size is limited, the coding coefficient matrix can become smaller and the complexity of decoding can become lower. Hence, the time of packets remain in the decoding buffer is decreased, that is, packets can be discarded earlier from the decoding buffer. For these reasons, our scheme can effectively reduce the decoding delay of network coding. Nevertheless, a smaller coding window size does not mean to be better. If the size of coding window is too small, the coding window will restrict the use of available network resources which could cause a poor throughput. As a consequence, we must take into account the trade-off between throughput and delay to get higher throughput and lower delay.

The algorithm of our scheme is specified using pseudo-code that shown in Tables 2 and 3.

Table 2: The operations of network coding layer at the sender

Event	**Pseudo-code**
Initialization:	1)Set NUM,$send\_count$, $send\_old$ and $seen\_old$ to 0. 2)Set $init$ to 20. Let $T = init$.
When the packets arrive from TCP sender:	1)If the packet is a control packet used for connection management, deliver it to the IP layer then doing nothing; else, move to state 2). 2)If the packet not already in the coding buffer, add it to the coding buffer. 3)Set $NUM = NUM + R$. (R = redundancy factor) 4)Repeat the following $\lfloor NUM \rfloor$ times: 1.Generate a random linear combination of the packets in the coding window. 2.Count the $send\_count = send\_count + 1$. 3.Add the network coding header specifying the set of packets in the coding window and the coefficients used for the random linear combination. 4.Add the variable $send\_count$ to the network coding header. 5.Deliver the packet to the IP layer. 5)Set NUM = fractional part of NUM.
When the ACK arrives from receiver:	1)Pick up the variables $seen\_count$ and $send\_count$ from ACK header. 2)If $seen\_count \geq T$ start to adjust the values of R and CW; else, move to state 3). 1.Compute the $diff\_send = send\_count - send\_old$ and the $diff\_seen = seen\_count - seen\_old$. 2.Compute the $loss\_rate = (diff\_send - diff\_seen)/diff\_send$. 3.If $T = init$: a)$R_{current} = 1/(1 - loss\_rate)$. b)$CW = 20 + \lfloor loss\_rate * 10 \rfloor$. c)Move to state 5. 4.If $T > init$: a)$R_{current} = a * 1/(1 - loss\_rate) + b * R_{old}$. b)$CW = 20 - \lfloor (R_{current} - 1)/R_{current} * 10 \rfloor$. 5.Update R_{old} to $R_{current}$, $send\_old$ to $send\_count$ and $seen\_old$ to $seen\_count$. 6.Reset $T = CW + seen\_count$. 3)Remove the ACKed packet from the coding buffer and hand over the ACK to the TCP sender.

4 Performance Evaluation

Our adaptive network coding scheme is evaluated by means of the Network Simulator (ns-2) under various conditions such as different link bandwidths, packet sizes, and random loss rates. Evaluation metrics in performance test are the throughput, the average delay, and the maximum

Table 3: The operations of network coding layer at the receiver

Event	**Pseudo-code**
Initialization:	1)Set *seen_count* to 0.
When a packet arrives from sender:	1)Count the $seen\_count = seen\_count + 1$. 2)Remove the network coding header, then retrieve the coding vector and the variable *send_count*. 3)Add the coding vector as a new row to the existing coding coefficient matrix, and perform Gauss-Jordan elimination to update the set of seen packets. 4)Add the payload to the decoding buffer. Perform the operations corresponding to the Gauss-Jordan elimination, on the buffer contents. If any packet gets decoded in the process, deliver it to the TCP sink and remove it from the buffer. 5)Generate a new ACK with sequence number equals to that of the oldest unseen packets and add two variables *send_count* and *seen_count* to the ACK header.
When an ACK arrives from the TCP sink:	1)If the ACK is a control packet for connection management, deliver it to the IP layer; else, ignore the ACK.

delay. We compare our scheme ANC with SANC-TCP and TCP-Vegas under both fixed and unfixed loss rate. We use TCP-Vegas as the transport layer protocol, and set the parameters of TCP-Vegas to $\alpha = 28$, $\beta = 30$, $\gamma = 2$. This setting is identical with TCP/NC [3]. The values of a and b in our scheme are set to $a = 0.2$ and $b = 0.8$ individually.

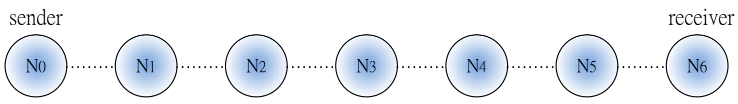


Figure 2: A tandem network consisting of 6 hops

4.1 Results for fixed random loss rates

The topology in this simulation is a tandem network consisting of 6 hops, as shown in Figure 2. The sender and the receiver are at opposite sides of the chain.

In contrast to traditional wireless networks, wireless sensor networks generally have a lower bandwidth for data transmission. We experiment with two different bandwidth settings: (a) bandwidth = 1 Mbps and (b) bandwidth = 250 kbps which is the channel bandwidth of IEEE 802.15.4. In the following experiments we assume that the source always has data to send.

Set the link bandwidth to 1 Mbps

In this experiment, each link has a bandwidth of 1 Mbps, and a propagation delay of 10 ms. The buffer size on the links is set to 200 packets. The packet size is 1000 bytes, and the random loss rate is varied from 0% to 20% on each link. The simulation time is 1000 seconds.

The throughput obtained corresponding to different random loss rates is plotted in Figure 3. Our scheme ANC and SANC achieve a similar throughput, but the throughput of Vegas decrease rapidly when the random loss rate increases. The adjustment of the redundancy factor R that we proposed in subsection 3.1 makes the sender to send out the proper number of redundant linear combinations so that the random loss rate can be masked effectively. As a result, we can get a fairly well throughput.

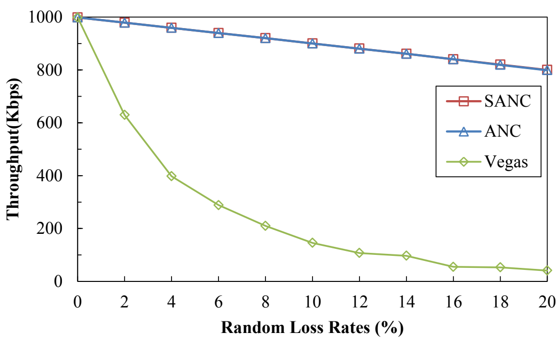


Figure 3: Throughput

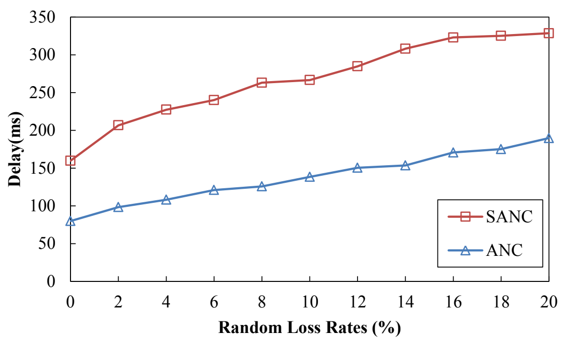


Figure 4: Average delay

Figure 4 shows the average delay of ANC and SANC. Compared with the SANC, our scheme have lower average delay in every case, moreover, the average delay of our scheme is almost half of the average delay of SANC. This is because we limit the coding window size which would decrease the complexity of decoding and reduce the time that packets remain in the decoding buffer. Thus, we can have a lower delay time than that of SANC.

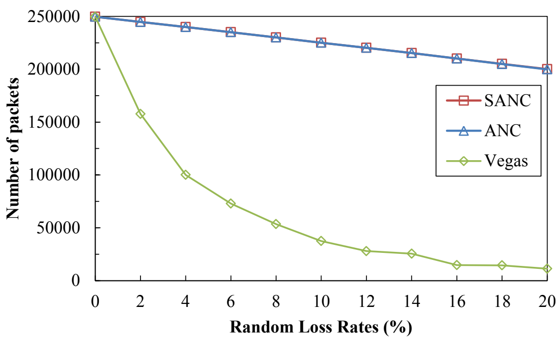


Figure 5: The total number of packets

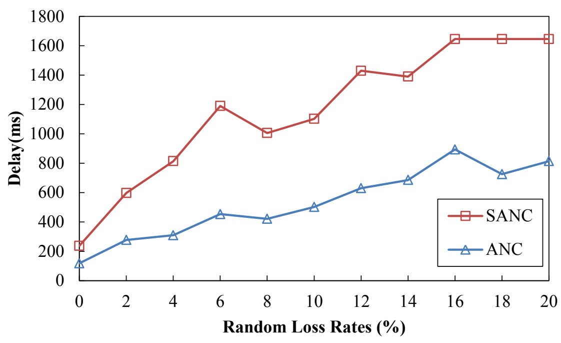


Figure 6: Maximum delay

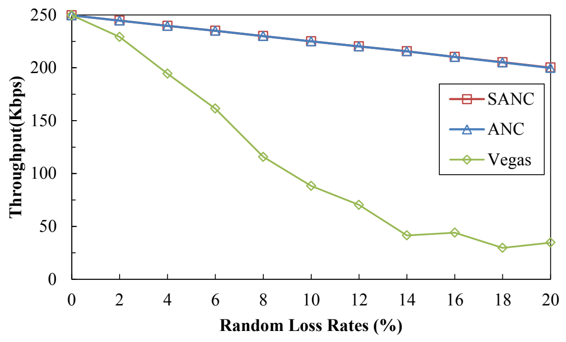
Figure 5 shows the total number of packets that have been send for SANC, ANC, and Vegas. This figure is essentially identical to Figure 3 except the Y-axis represents the total number of sent packets. As shown in Figure 5, the metric of total number of sent packets for ANC and SANC outperform Vegas as the loss rate increases.

With regard to the simulation of delay, we also retrieve the maximum delay at various loss rates. As illustrated in Figure 6, the maximum delay of SANC rises significantly with loss rate, while our scheme rise slightly. Furthermore, the maximum delay of SANC is at least two times more than ANC in most cases.

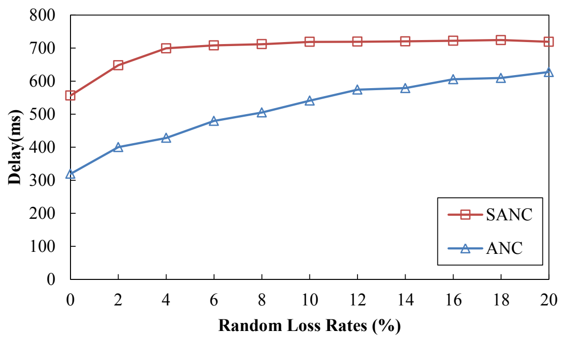
Set the link bandwidth to 250 kbps

The channel bandwidth of IEEE 802.15.4 is 250 kbps [10]. Many WSNs adopt IEEE 802.15.4 for communicating among nodes. The characteristics of IEEE 802.15.4 technology includes low rate, low transit distance, low power, low cost, simple architecture, small size, etc. All these characteristics are applied to the applications of WSN.

The experimental parameters in this simulation are identical to the setting used in subsection 4.1.1, except that the bandwidth of each link is 250 kbps, and the buffer size on the links is set to 50 packets. Besides, we simulate the effect of two different packet sizes on the performance. The packet sizes are set to 1000 bytes and 200 bytes.

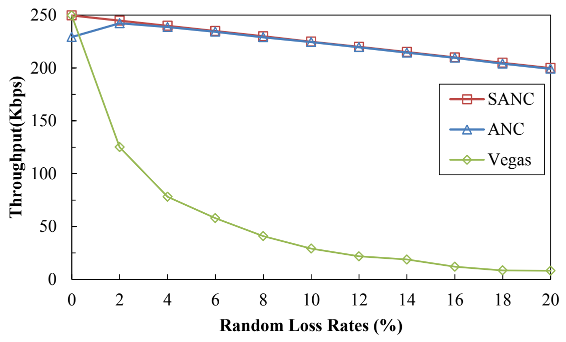


(a) Throughput

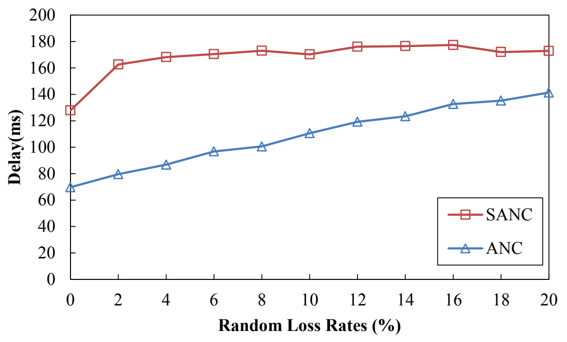


(b) Average delay

Figure 7: The throughput and delay when packet size is 1000 bytes



(a) Throughput



(b) Average delay

Figure 8: The throughput and delay when packet size is 200 bytes

As the link capacity and the buffer size become smaller, our scheme ANC still has high throughput and low delay, as shown in Figure 7. However, it must take a longer delay to transmit a large packet (1000 bytes) if the link capacity becomes smaller. This is demonstrated through the simulation results, when the loss rate is 20%, ANC's average delay is 189.77 ms in Figure 4 where the bandwidth is 1 Mbps, while ANC's average delay is 627.58 ms in Figure 7(b) where the bandwidth is 250 kbps.

Then, we change the packet size to 200 bytes. Figure 8(a) demonstrates that when the loss rate is 0%, the throughput of our scheme ANC is lower than SANC and Vegas. This is because that the coding window size we adjust according to subsection 3.2 is not big enough in such a network environment, thus the available network resources is restricted by coding window and result in lower throughput. That is to say, when the packet size is very small, we must have a sufficient coding window size to take full advantage of network resources. In Figure 8(b), the average delay of our scheme is lower than SANC in every case. In addition, the average delay in Figure 8(b) is lower when compared with the Figure 7(b) because the smaller packet size has the lower transmission time.

4.2 Results for unfixed random loss rates

All previous simulations focus on the behavior of ANC under the fixed loss rate. Now, we evaluate its performance in an unknown environment with unfixed loss rate. We use the same topology and parameters as 4.1.1. The background loss rate in this scenario is 2%. The loss rate is changed to 6% from 200 to 400 second, and changed to 10% from 600 to 800 second. Total simulation time is 1000 seconds.

In Figure 9, the X-axis represents the simulation time, and the Y-axis represents the average throughput. It clearly shows that ANC and SANC can quickly and effectively handle the sudden bursty losses by adjusting the redundancy factor R dynamically, but Vegas is seriously affected by bursty losses. In Figure 10, we show the comparison of delay between ANC and SANC. The delay of SANC increases rapidly when the loss rate is suddenly changed at 200 second and 600 second. In contrast to SANC, ANC can still maintain quite low latency under the changed loss rate. This is because our scheme ANC can constantly measure the packet loss rate of network and further adjust the coding window size to get low delay. Based on the above experimental results, we can demonstrate that even under the unknown environment with bursty losses, our scheme ANC still can send enough linear combinations to compensate the lost packets. Thus, ANC can reach high throughput and keep low delay through the adjustment of R and CW.

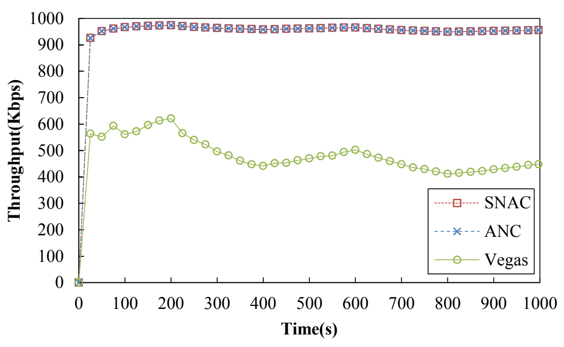


Figure 9: The average throughput under bursty losses

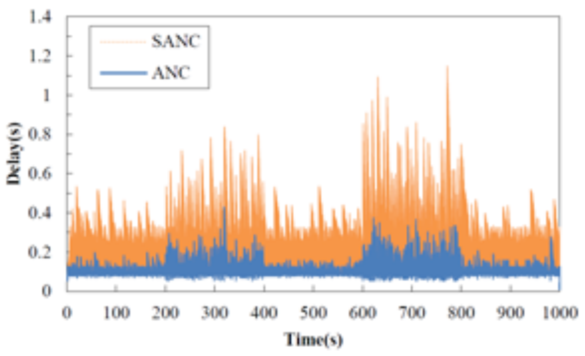


Figure 10: The delay evolution under bursty losses

4.3 Results in a more realistic environment

In this subsection, we compare our scheme with SANC-TCP and TCP-Vegas in a wireless topology shown in Figure 11. There are five nodes (four hops) in this topology. The distance between each node is 30 meters. Table 4 is a parameter table which shows the parameter settings that used in the performance test. The simulation time is 100 seconds. The FTP flow stars at 1 second. In this experimental environment, some packets can be lost for reasons other than congestion.

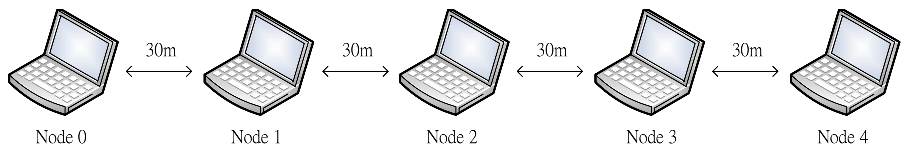


Figure 11: Wireless topology

The experimental results show that the throughput of ANC and SANC are 272.0 kbps and 271.9 kbps respectively. Both of them are outperform Vegas whose throughput is 260.5 kbps. The comparison of RTT between ANC and SANC is shown in Figure 12. At the beginning of this experiment, we can see the RTT for both schemes are rising rapidly, then the RTT of ANC is maintained between 9 to 14 milliseconds that is lower than the RTT of SANC which is maintained between 15 to 20 milliseconds. Figure 13 presents the average RTT and maximum

Table 4: Parameter table

Parameter	**Value**
Packet Size	200 bytes
Buffer Size	50 packets
Buffer Management Scheme	DropTail
Link Bandwidth	2 Mbps
Transmission Range	40 meters
Carrier Sensing Range	90 meters
MAC Protocol	CSMA/CA
Routing Algorithm	DSR

RTT of ANC and SANC. It is clear ANC has lower average RTT and maximum RTT than SANC.

As mentioned before, this is because we limit the coding window size, and then the complexity of decoding can be decreased. Thus, packets will be discarded earlier from the decoding buffer. According to the experimental results, our scheme ANC can also have a good performance in such an environment.

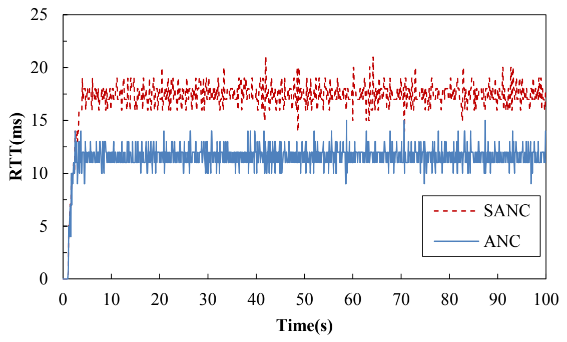


Figure 12: RTT of ANC and SANC

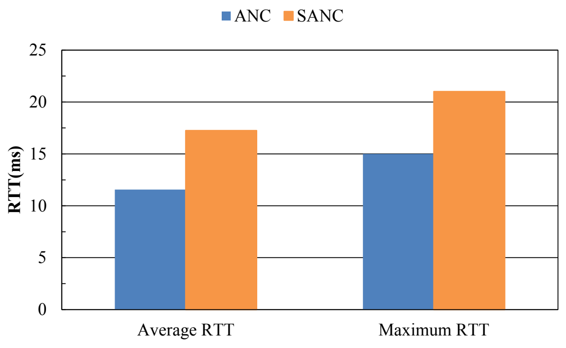


Figure 13: Average RTT and Maximum RTT of ANC and SANC

5 Conclusions

In this paper, we introduced a network coding scheme called adaptive network coding (ANC) that can be applied on wireless sensor networks. The objectives of our study are to mask packet losses by transmitting redundant linear combinations and to reduce the decoding delay of network coding by limiting the coding window size. We compare our scheme with SANC-TCP and TCP-Vegas in difference random loss rates, different link bandwidths, and difference packet sizes. From the experimental results obtained, our scheme has a lower delay than SANC-TCP in all experimental environments without sacrificing throughput. Besides, both the throughput of our scheme and SANC-TCP are significantly better than that of TCP-Vegas in most cases. It is important to note that the adjustment of coding window size can be investigated in more detail so as to improve the throughput when TCP runs on a harsh network environment.

A possible direction for further study is to allow intermediate nodes to perform the encoding and decoding whenever they receive packets. That means encoding and decoding operations are done in a hop-by-hop manner. In this way, the network coding can be used for a wide variety of topologies of wireless sensor networks.

Bibliography

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, Network information flow. *IEEE Trans. on Information Theory*, 46(4):1204-1216, Jul. 2000.

[2] D. Silva and F. R. Kschischan, Universal Secure Network Coding via Rank-Metric Codes, *IEEE Trans. on Information Theory*, 52(2): 1124-1135, Feb. 2011.

[3] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, Network Coding Meets TCP, *2009 Proceedings of IEEE INFOCOM*, 280-288, Apr. 2009.

[4] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, Network coding: An instant primer, *ACM SIGCOMM Computer Communication Review*, 36(1): 63-68, Jan. 2006.

[5] Yao-Nan Lien, Hop-by-Hop TCP for Sensor Networks, *International Journal of Computer Networks & Communications*, 1(1):1-16, Apr. 2009.

[6] C. Wang, K. Sohraby, B. Li, M. Daneshmand, and Y. Hu, A survey of transport protocols for wireless sensor networks. *IEEE Network Magazine*, 20(3): 34-40, Jun. 2006.

[7] J. K. Sundararajan, D. Shah, and M. Medard, ARQ for network coding, *in Proc. of IEEE International Symposium on Info. Theory (ISIT)*, 1651-1655, Jul. 2008.

[8] J. Chan, L. Liu, X.Hu, and W. Tan, Effective retransmission in network coding for TCP. *Int J Comput Commun*, ISSN 1841-9836, 6(1):53-62, 2011.

[9] S. Song, H. Li, K. Pan, J. Liu, and Shuo-Yen Robert Li, Self-adaptive TCP protocol combined with network coding scheme, *In International Conference on Systems and Networks Communications (ICSNC)*, 20-25, Oct. 2011.

[10] F. Xia, A. Vinel, R. Gao, L. Wang, and T. Qiu, Evaluating ieee 802.15.4 for cyber-physical systems, *EURASIP J. Wireless Commun. and Networking*, 1-15, Feb. 2011.