Fountain Codes Simulation in NS3

ECS 252 Project

Yanan Bao, Christopher Buckley, Pengfei Hu, Muchen Wu, Chao Xu

1. Introduction

Wireless channels have high packet loss rate due to path loss and channel fading, which makes reliable transmissions complicated and less efficient. When packet loss occurs, the TCP protocol interprets it as packet congestion and reduces the transmission rate. Therefore, if the traditional TCP protocol is directly used in wireless networks without any adaptation, the throughput greatly decreases.

2. Related Work

Fountain codes were first proposed in 1998 by Michael Luby. At that time, only the concept was proposed and no implementation was included [1]. In 2002, Luby proposed the first implementation of fountain codes, which was called LT (Luby Transform) code [2]. Later, Amin Shokronllahi proposed another implementation called Raptor Code that had better performance than LT code [3]. In recent years, fountain codes have attracted more and more attention within the IT industry. Luby started a company called Digital Fountain, which was bought by a telecommunications company called Qualcomm. Using fountain codes, k packets can be encoded into a near infinite set of encoded packets. A receiver only needs to receive N (which is slightly larger than k) of these encoded packets to recover the k original packets. N/k–1 is defined as the decoding overhead rate. In general, the decoding overhead rate is smaller than 5%. Therefore, fountain codes are a very efficient means of information coding.

The advantages of fountain codes are summarized as follows:

1. No feedback is needed. Fountain code is a kind of FEC (Forward Error Correction). Compared to traditional ARQ, it uses less channel resources.
2. Low encoding and decoding complexity. It has linear complexity as the packet number grows.
3. Adapt to the channel variations. It is a kind of rateless code. When the packet loss is high, more encoded packets need to be send by the server, and the server doesn't need to adjust the coding rate.
4. Provide different QoS for receivers with different channel conditions. In this case, the user with a good channel will not be affected by users with bad channels, when all users are served by a multicast or broadcast simultaneously.

5. Receivers can start to receive data after a multicast session has been set up. In a nonsystematic fountain code, every encoded packets are interchangeable for the source data – ANY combination of N packets will work.

6. Infinite encoded packets from a finite number of original packets.

7. Receivers don't need to ask for a certain packet when packet loss occurs. They only need to receive a certain number of encoded packets before decoding.

8. A receiver can receive from multiple servers simultaneously and the servers don't need to coordinate.

Due to the advantages of fountain codes, they have been or are going to be used in wireless multimedia broadcast / multicast, Space communications, distributed data storage, and P2P data download. In wireless broadcast / multicast scenarios, 3GPP standard MBMS (Multimedia Broadcast Multicast Service) [4] and 3GPP2 standard BCMCS (BroadCast and MultiCast Service) [5] have utilized Raptor codes.

3. Fountain codes

LT code is simpler than Raptor code. It is also the first version of fountain codes. Shown in Fig.1, The process of encoding is:

1. Divide the original file into K original packets.
2. Choose an integer d in [1,2,…,K] randomly as the degree for one encoding packet.
3. Randomly choose d original different packets among the K packets.
4. XOR the d packets and get a new encoded packet.
5. Send the new encoded packet to the encoded buffer.
6. Repeating steps 2 through 5, we can get infinite number of encoded packets.

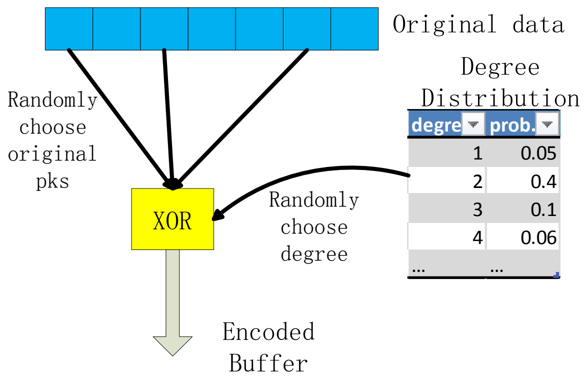
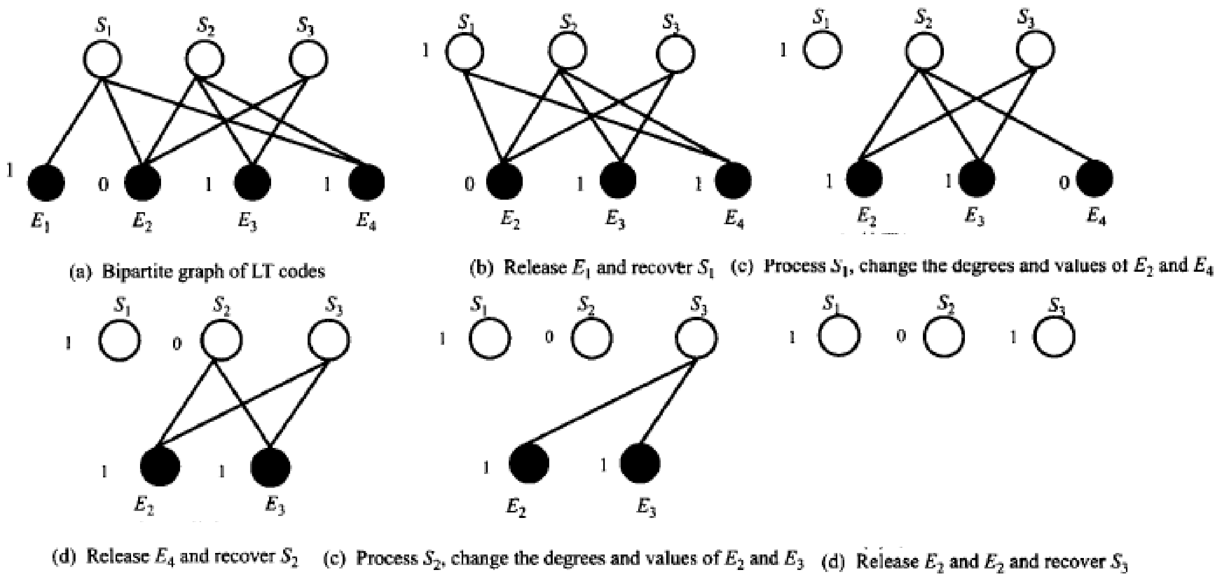


Fig.1 Process of encoding

There are multiple decoding algorithms, such as BP (Belief Propagation) and Gaussian elimination. Here we use the BP algorithm, since it is easy to implement and has low computational complexity. Note that sometimes the encoded data can be decoded by Gaussian elimination, even when it cannot be decoded by the BP algorithm.

Fig.2 shows the process of decoding. It is easy to see that the decoding starts from the node with one degree and the number of nodes with one degree is called decoding ripple. After the node with one degree is decoded, we get a source node. The value of the new source node can be used to eliminate all the other encoded nodes with relation to the source node. The iteration will stop when there is no decoding ripple. This is the drawback of the BP alogorithm, since at this time other decoding algorithms may continue decoding. If all the source nodes are decoded before the decoding ripple disappears, the BP algorithm succeeds.



(a) Bipartite graph of LT codes (b) Release E_1 and recover S_1 (c) Process S_1, change the degrees and values of E_2 and E_4

(d) Release E_4 and recover S_2 (c) Process S_2, change the degrees and values of E_2 and E_3 (d) Release E_2 and E_2 and recover S_3

Fig.2 Process of decoding

Degree Distribution

It is worthwhile to note that the decoding efficiency depends on the degree distribution. Choosing a good degree distribution keeps the size of decoding ripple stable, and thus speeds up the decoding process and reduces the amount of encoded data needed to recover the source file.

1) Ideal Soliton Distribution

$$\rho(d) = \begin{cases} \dfrac{1}{K} \cdots\cdots\cdots\cdots d = 1; \\ \dfrac{1}{d(d-1)} \cdots\cdots d = 2, 3, \cdots K; \end{cases}$$

2) Robust Soliton Distribution

In the Robust Soliton Distribution, δ and c are two parameters.

$$\tau(d)=\begin{cases}\dfrac{s}{K}\dfrac{1}{d}\cdots\cdots\cdots\cdots\cdots\cdots d=1,2,3,\cdots,(K/s)-1;\\ \dfrac{s}{K}\log_e(s/\delta)\cdots\cdots\cdots d=K/s;\\ 0\cdots\cdots\cdots\cdots\cdots\cdots\cdots d>K/s;\end{cases}$$

$$\mu(d)=\frac{\rho(d)+\tau(d)}{Z}$$

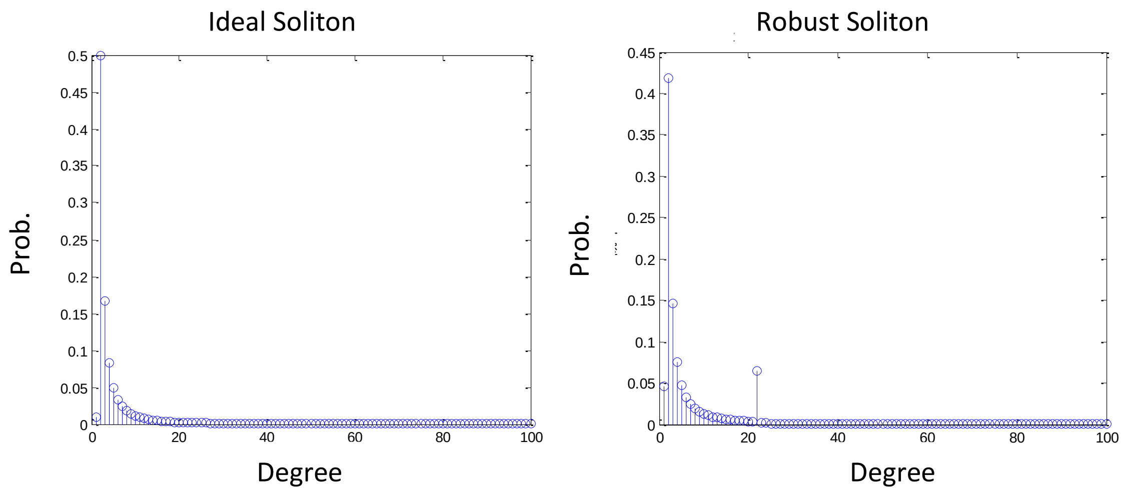


Fig.3 Degree distribution

4. System Model

LT code is the first implementation of fountain codes and it has both excellent performance and low complexity. Fig.4 shows the encoder design and decoder design based on LT code respectively.

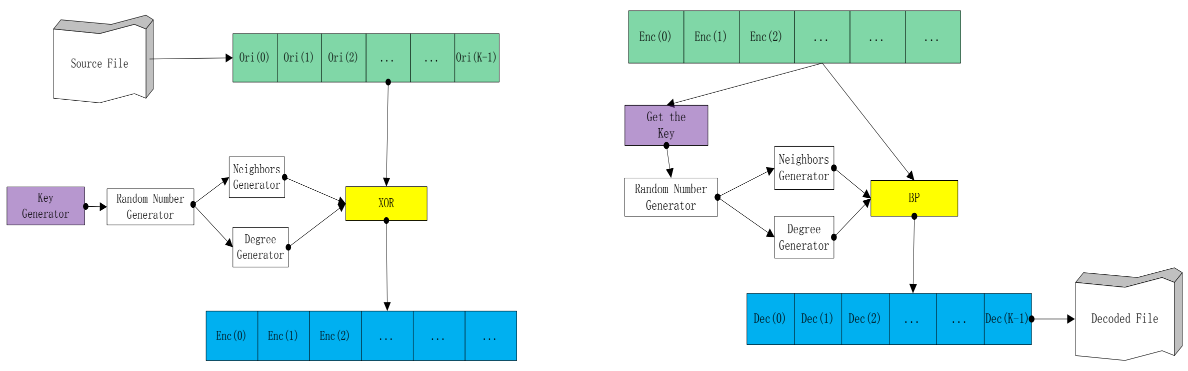


Fig.4 Design of encoder and decoder based on LT code

1) Data Encapsulation

After applying fountain codes in a higher layer, great changes have taken place. If the size of the download file is large, we shall cut it into several blocks before encoding and sending to clients one by one. To encode a part of the data, we first cut it into small packets which are called source packets (SP is short for source packet in Fig.5). Then we encode these packets into encoded packets (EP is short for encoded packet in Fig.5). The server encapsulates the encoded packets, at which point the packets are sent and decoded as described above.

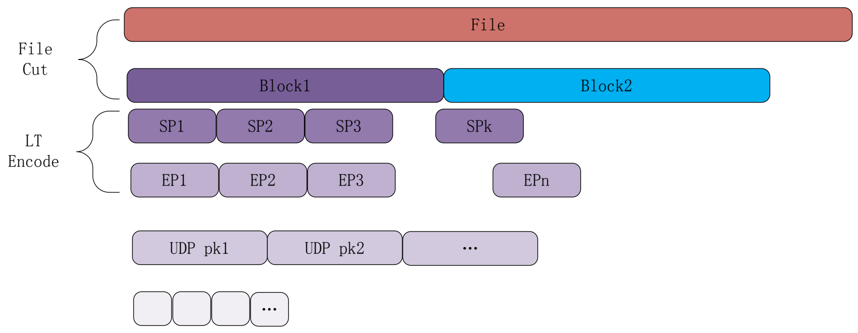


Fig.5 Data encapsulation

As shown in UDP segment format, a checksum is incorporated in the header of the UDP segments. When the calculated checksum is different from the checksum value in the UDP header, it indicates a transmission error occurred and the packet has to be discarded. Therefore, the receiver only forwards correct packets to the application layer. As the receiver gets plenty of encoded packets, decoding these packets with LT code can provide reliability of packet transmission with very high probability.

2) Reliable transmission scheme

   a) User termination scheme

The server encodes and sends the encoded packets to all clients who subscribe to the service. The client who has received enough amount of data will send a feedback to the server which can be a short message. When the server receives the stop message from all the clients, it stops encoding and sending. Fig.6 shows the model of this kind of scheme.

The message and data exchange between the server and clients in the user termination scheme are shown in Fig.6 below.

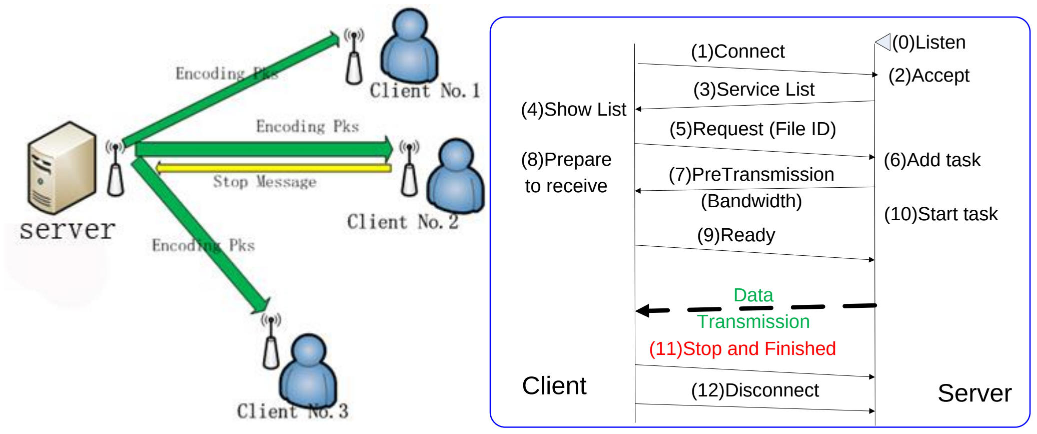


Fig.6 User termination scheme

b) User request scheme

The server encodes and sends the encoded packets once, and then waits for the client's feedback to know how many more encoded packets are needed. After estimation and analysis, the server encodes and sends a fixed number of packets, as Fig.7 shows.

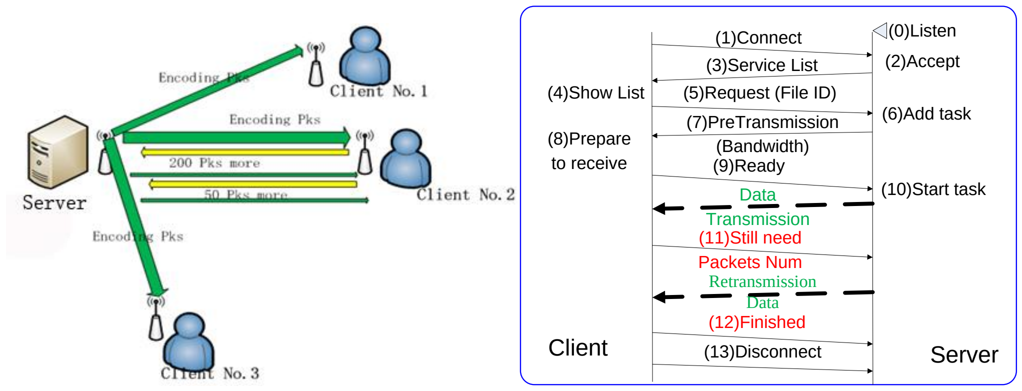


Fig.7 User request scheme

5. Simulation Results and Discussion

1) Simulation without experiencing the channel loss

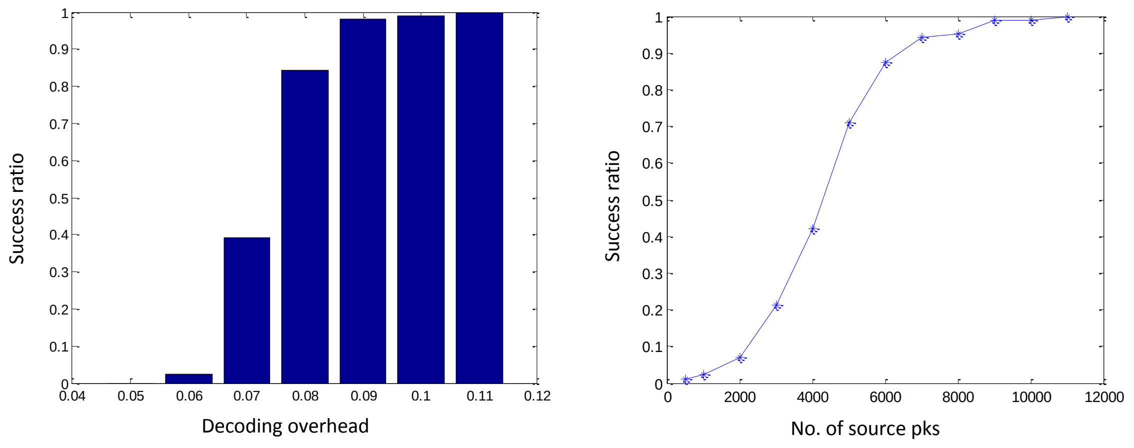


Fig.8 Success ratio varying with decoding overhead and the number of source packets

In order to compare the new system with the original one, we do some simulations. The overhead rate of decoding LT code is 0.1. Fig.9 is the comparison of throughput when the number of users is fixed to 100, but the PER (packet error rate) varies. Fig.10 shows the comparison of throughput when the PER is fixed to 0.001, but the number of users varies.

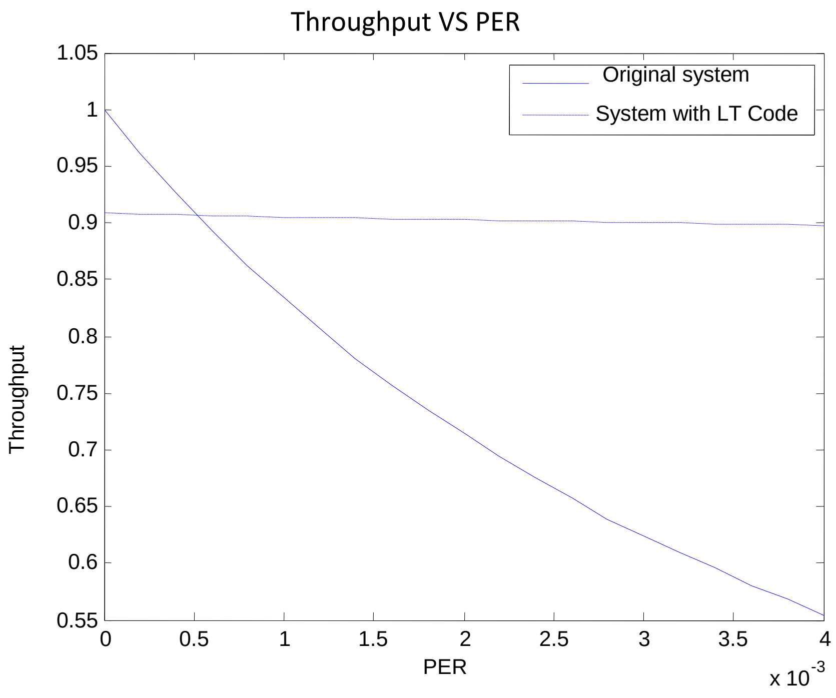


Fig.9 Comparison of Throughput with varying PER values

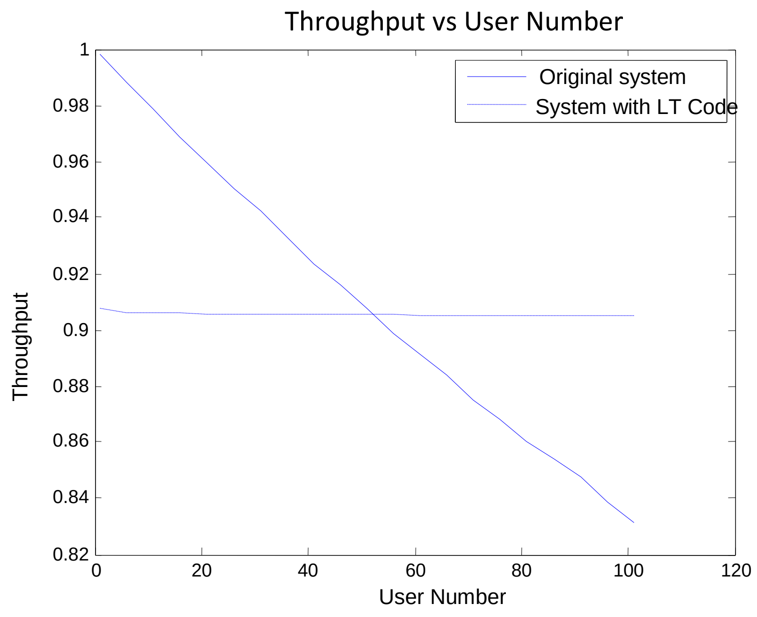


Fig.10 Comparison of Throughput with varying number of users

It’s shown that by applying fountain codes in our file transfer system, the system performance is significantly improved. To summarize, the throughput is improved, and the feedback congestion problem is solved.

2) Simulation of successful transmission time with packet drop channel

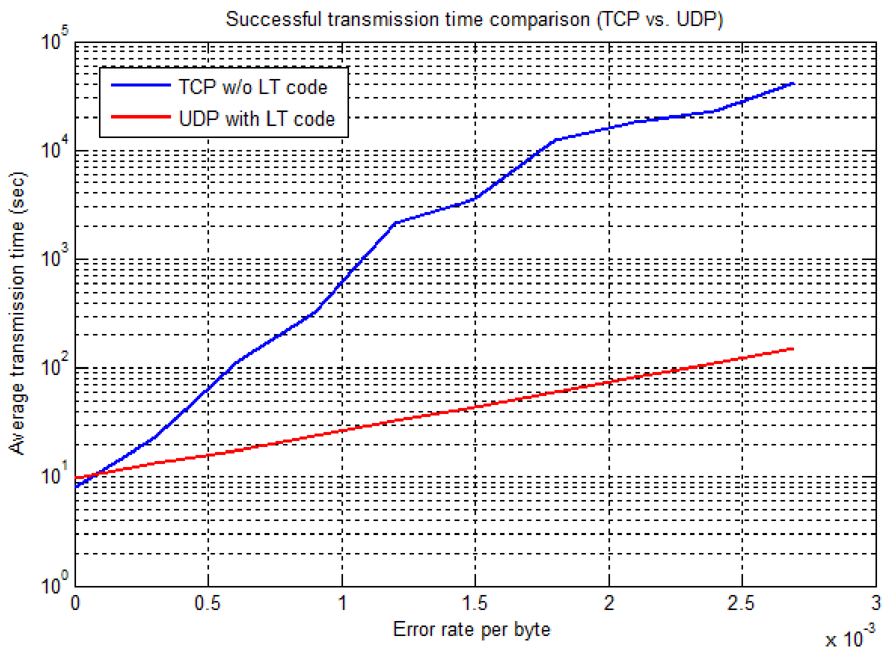


Fig. 11 Transmission time comparison between TCP and UDP with different error rate

We simulate the transmission time under TCP and UDP protocols, respectively. When the server sends our test file (a 4.9MB video clip) using the standard TCP protocol, the successful transmission time increases significantly along with the error rate. The payload of one packet is 1000bytes, which means that in total we have about 5000 packets being encoded and decoded for the 4.9 Mbytes file. Since the server attributes the transmission failure to congestion, the transmission rate of the sender decreases according to the TCP protocol’s congestion control mechanism. When the sender sends the test file using our modified UDP, (i.e., UDP protocol with LT code), the rising slope is much smaller than the one with standard TCP protocol. As illustrated in Fig.11, using the UDP protocol with LT code has the superior average transmission time compared to using the standard TCP protocol.

6. Conclusions

LT code is suitable for wireless networks, especially under conditions with high packet loss rate. It efficiently reduces the overwhelming retransmission caused by the congestion control in TCP protocol. Therefore, LT code can make a great contribution to the unstable and unreliable wireless environment.

7. Reference

[1]J Byers, M Luby,and M Mitzenmacher. A Digital Fountain Approach to asynchronous Reliable Multicast. IEEE Journal on Selected Area in Communication, 2002, 20(8):1528-1540

[2]M Luby. LT Codes. In Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science(FOCS), 2002:271-280

[3]A Shokrollahi. Raptor Codes. IEEE Transaction on Information Theory, 2006, 52(6):2551-2567

[4] 3GPP TS 23.246, Multimedia Broadcast/ Multicast Service (MBMS); Architecture and functional description.

[5] 3GPP2 X S022-A v1.0, Broadcast and multicast service in cdma2000 wireless IP network, revision A.