Efficient Concurrent Multipath Transfer using Network Coding in Wireless Networks

Zhuofeng Li[1], Changqiao Xu[1,2], Jianfeng Guan[1], Hongke Zhang[1,3] and Gabriel-Miro Muntean[4]
[1]Institute of Networking Technology, Beijing University of Posts and Telecommunications, Beijing, China
[2]Institute of Sensing Technology and Business, Beijing University of Posts and Telecommunications, Wuxi, Jiangsu, China
[3]National Engineering Lab. for Next Generation Internet Interconnect. Devices, Beijing Jiaotong University, Beijing, China
[4]Performance Engineering Lab., School of Electronic Engineering, Dublin City University, Dublin, Ireland

***Abstract*—Concurrent Multipath Transfer (CMT), enabled by Stream Control Transmission Protocol (SCTP), is considered as one preferred data-transport mode due to increased available bandwidth. However, CMT performance degrades seriously in terms of data reordering due to path dissimilarity and frequent packet loss from wireless unreliability. Most relevant solutions follow the packet sequence numbers and thereby focus on strict in-order reception and packet-specific retransmission. Passively adapting to the network conditions, those approaches are not general and well enough responding to the dynamicity of wireless environment. By applying Network Coding (NC) to CMT, this paper proposes a progressive end-to-end solution (CMT-NC) to those problems in heterogeneous wireless networks. CMT-NC is capable of avoiding reordering and compensating lost packets. Further, an innovative group-based transmission management mechanism enhances the robustness and reliability of data transfer. Simulation results show how by using CMT-NC significant improvements in comparison to another state-of-the-art solution are obtained.**

I. Introduction

With the trend towards ubiquitous and heterogeneous mobile internet access support, Concurrent Multipath Transfer (CMT) based on Stream Control Transmission Protocol (SCTP) [1] is rendered as one most-preferred transport-layer mode to enhance data delivery [2], in order to meet the over-growing demand for high-quality, efficient and convenient delivery of various network services. CMT provides benefits of bandwidth aggregation, fault tolerance and load balance that are very attractive for network content distribution.

However, due to highly dissimilar path characteristics (e.g. bandwidth, delay, loss rate), data from different paths arrived out of order. The receiver needs to wait for delayed data on slow paths, while placing the arrived data on fast paths into its buffer, for subsequent reordering and satisfying in-order delivery requirement. Hence, CMT is bound to buffer blocking that leaves the connection idle [3]. Many works [2][4] focused on alleviating this problem, thus had the challenge of data scheduling. But they agreed to the strict in-order data reception and still passively suffered from reordering in the probably blocking receiver buffer.

Another concern is that, due to the unreliability of wireless channels, packet loss and retransmission occur frequently. For the sake of reliable data transfer, sometimes repeated retransmission is needed for the same packet. In this context, it is also important to analyze the loss reason due to either congestion or wireless error [3], so as to take proper actions responding to the loss reason. Some works [5][6] tried to strengthen CMT reliability and save the cost of retransmission. They considered the uniqueness of individual packets and carried packet-specific loss detection. However, this hardly reduced the retransmissions.

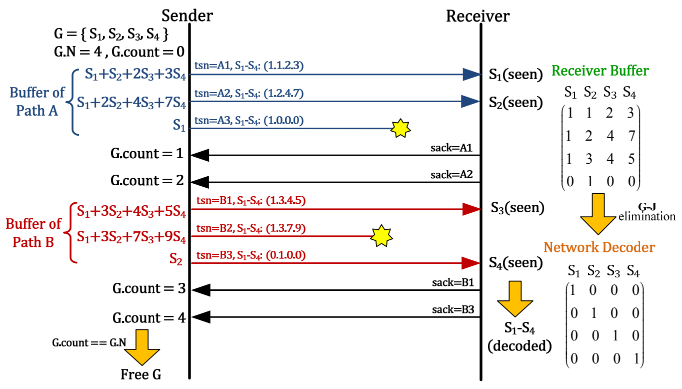
Recently, network coding [7] at transport layer emerges as one elegant strategy to break the strong binding between data packet and its transmission sequence number (TSN), which has been the critical issue of in-order reliable data transfer. Then, the receiver does not care about the actual packet arrival order and there is no need of reordering. Further, the coded packets can be replaced and supplemented with each other if loss happens. Sundararajan *et al.* proposed a special online ACK mechanism [8] to make network coding effective for TCP. Later, Li *et al.* adopted that mechanism to Mutipath TCP (MPTCP) [9] and revealed goodput improvement in multipath scenarios [10]. However, their proposal did not pay attention to optimizing the congestion control and identifying the packet loss due to wireless error.

This paper proposes a general SCTP CMT Network Coding-based solution (CMT-NC) in order to achieve high data-transport efficiency in heterogeneous wireless networks. Compared with MPTCP guidelines, SCTP is a more sophisticated protocol which has a full set of standards [11]. More importantly, SCTP CMT provides some good features that are very conducive to network coding, such as flexible multiple-chunk packet format, Heartbeat mechanism and Selective ACK (SACK) that we will apply in CMT-NC. As our distinct contributions, CMT-NC employs new outstanding mechanisms for refined path characteristics estimation, fast data distribution and group-based transmission management. To the best of our knowledge, CMT-NC is the first attempt to apply network coding to SCTP and its CMT extension.

II. Applying Network Coding to SCTP CMT

The design principle of network coding is illustrated by Fig. 1. The sender combines several data packets ($S_1 - S_4$) for encoding once (splicing them into one linear combination).

978-1-4799-3083-8/14/$31.00 © 2014 IEEE



$$\begin{array}{cccc} S_1 & S_2 & S_3 & S_4 \\ \end{array} \begin{pmatrix} 1 & 1 & 2 & 3 \\ 1 & 2 & 4 & 7 \\ 1 & 3 & 4 & 5 \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad \begin{array}{cccc} S_1 & S_2 & S_3 & S_4 \\ \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Fig. 1. Example of network coding in CMT-NC

This batch of packets with DATA chunks[1] and continuous TSN that are encoded together is called a *group* (G). Then, regardless of sending original or coded packets, they uniformly have associated coding coefficients. As a result, they are no longer carrying application data as before, but represent the relationship within the original data. Hence, it is not necessary to receive them in strict order, just buffer them and extract the relationship information (coefficients). Additionally, the information can be interchanged such that lost packets can be supplemented by other packets. As long as the receiver collects enough number of packets (4 in this example), it can decode the original data correctly by Gauss-Jordan elimination, and the sender does not get concern about any other lost packets or retransmissions. In summary, by means of network coding, the data packets are not uniquely determined by their TSN, the receiver needs no reordering and the sender reduce retransmissions to a great extent.

A. Network Coding Operations

In our network coding design, the first question is how many packets should be in one group, called *group number* (N). It should be properly set a bit above the number of out-of-order arrivals, trading off between low requirement of buffer space and sufficient disorder compensation. N is updated according to bandwidth, delay and loss rate when a retransmission event happens, which indicates that this group number is hysteretic for the variation of the multipath environment.

The delay factor is referred to as Round-Trip Time (RTT). Since SCTP CMT allows replying SACK messages on different paths from their corresponding DATA packets, the general Smoothed RTT ($SRTT$) measurement may lead to incorrect delay estimation. Fortunately, SCTP has the Heartbeat mechanism for path probing, which requires HEARTBEAT_ACK messages be replied on the same path as HEARTBEAT packets [1]. When it is time to update the group number, two HEARTBEAT packets are sent immediately; until both HEARTBEAT_ACK packets are received, we obtain two RTT values for the probing and take the min as path *Reference RTT* ($RRTT$). If $SRTT$ is close to $RRTT$, meaning large probability that SACK is replied on the same path as DATA, or the forward and backward paths are with similar delay, then we use $SRTT$ as path RTT because there are many samples available thus the estimation is more accurate; otherwise, we use $RRTT$ to calibrate the path RTT. For any active path i, we follow Eq. (1) to determine the path RTT:

$$RTT_i = \begin{cases} SRTT_i & (\text{if } SRTT_i \in [a * RRTT_i, b * RRTT_i]\,) \\ RRTT_i & (\text{otherwise}) \end{cases} \tag{1}$$

where $a < 1$ and $b > 1$ are RTT boundary factors.

The bandwidth factor is offered by bandwidth estimation. As with $SRTT$, all typical methods based on ACK receipt, such as Packet Pair and Westwood+ mentioned in [4], may not be appropriate in this case. We try inversing the viewpoint to the sending moment: the available bandwidth is equal to the ratio between sending data amount and sending duration. When the group number needs update, we collect one bandwidth sample for path i according to Eq. (2):

$$BWsample_i = \frac{sendsize_i}{Tl_i - Te_i} \tag{2}$$

where $sendsize_i$ is the sending data amount since the last sample collection, Tl_i and Te_i are the time of the last packet leaving and the first packet entering the path i's buffer during that time, respectively. In order to eliminate oscillations, the bandwidth samples should be further smoothed to give the estimation result:

$$\begin{aligned} BW_i = & \left[1 - \exp\left(\frac{-(Tl_i - Te_i)}{T_0}\right)\right] * BWsample_i \\ & + \exp\left(\frac{-(Tl_i - Te_i)}{T_0}\right) * BWprevious_i \end{aligned} \tag{3}$$

where T_0 is the bandwidth smooth factor and $BWprevious_i$ is the last-time estimation result.

The loss rate factor relies on the feedback of group-based transmission management that will be discussed later. Finally, the group number is updated as:

$$N = \left\lceil \sum_{RTT_j > RTT_i} \frac{BW_j\,(1 - pe_j)}{MTU} * \frac{RTT_j - RTT_i}{2} \right\rceil \tag{4}$$

where pe_j is loss rate of path j, MTU is the maximum transmission unit, typically assumed identical across paths.

After identifying the group number, the sender fetches N packets with continuous TSN $\mathrm{S}_1 - \mathrm{S}_N$ (where $\mathrm{S}_N = \mathrm{S}_1 + N$) and hands them on to the **Network Coder**. The **Network Coder** will randomly choose some packets for network coding, such that the original and coded packets are scrambled in path buffers. If **Network Coder** decides to encode one packet, it chooses the coefficients $a_1, a_2, ..., a_N$ randomly from the finite field $GF(2^8)$ then generates the coded packet T as a linear combination of $\mathrm{S}_1 - \mathrm{S}_N$. As each $a_i (1 < i < N)$ can be expressed as one byte and each packet is a set of bytes, the generation is actually byte-to-byte addition and multiplication [12], treating $\mathrm{S}_1 - \mathrm{S}_N$ as byte vectors:

$$\mathbf{T} = a_1\mathbf{S_1} + a_2\mathbf{S_2} + ... + a_N\mathbf{S_N} \tag{5}$$

Further, as shown in Fig. 2, for all DATA packets (original and coded), we insert a new NC chunk with coding information in front of the DATA chunk, including the range of the

[1]Packets with other types of chunks (i.e. control) will not be attributed to any group for network coding, and just sent out as usual.

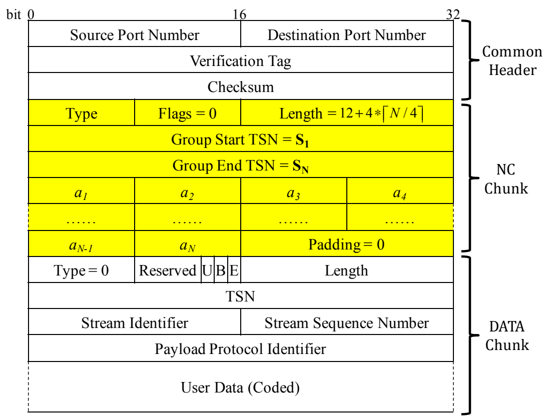


Fig. 2. Data Packet format with the new NC chunk

group and coding coefficients. This NC chunk is to tell the receiver how to handle the DATA packet. In this situation, the payload size for "User Data" should be reduced $12+4*\lceil N/4\rceil$ bytes for all packets in this group.

B. Hybrid and Fast Data Distribution

Since network coding avoids reordering in the group, no scheduling mechanism is needed. All we have to do is to distribute maximum data amount to path buffers to send it out in fast manner. An agile method is to stuff the *effective window*, which determines the allowable sending quantity at any one time. This effective window of path i is defined as:

$$effwnd_i = \min\{cwnd_i, arwnd\} \tag{6}$$

where $cwnd_i$ is the congestion window of path i, $arwnd$ is the advertised receiver window declared in the newest SACK. Next the *space* of path i for the packets to fill in is calculated by Eq. (7):

$$space_i = (effwnd_i - outstanding_i)(1 - pe_i) \tag{7}$$

where $outstanding_i$ is the outstanding data amount in path i. Finally, we iteratively utilize Algorithm 1 to assign packets to the path with most space.

For correct and fast decoding, the sender should insert some redundant packets. Although the redundancy would increase the sending data amount, it is valuable to guarantee enough packets reaching the receiver and largely reduce retransmissions. After distributing N packets in one group, the sender appends some redundant packets to the tail of each path buffer. These redundant packets must be coded as generated by the same method as above, since they are containing information of the whole group in favor of decoding. They would have TSN equaling to the packet ahead of it in the same path, and can be acknowledged by the "Duplicated TSN" fields [1] in SACK.

Suppose M_i packets out of the N-packet group are already allocated to path i, then the number of redundant packets for path i should be:

$$R_i = \left\lceil M_i * \frac{pe_i}{1-pe_i} \right\rceil \text{, where } N = \sum_i M_i \tag{8}$$

Since the transmission success rate is $1 - pe_i$ for path i, this redundancy guarantees M_i packets reaching the receiver

Algorithm 1 Fast Data Distribution

```
/* When there is one DATA packet to send */
$max\_space = 0$;  $j$ =**null**;  //initial traverse variables
**for** ($\forall$ active Path $i$)
  Calculate $i.space$ according to Eq.(7);
  **if** ($i.space > max\_space$)
    $j = i$;  $max\_space = i.space$;  //find the path with largest space
  **else if** ($i.space == max\_space$ **&&** $i.BW > j.BW$
            **&&** $max\_space > 0$)
    $j = i$;  //if space is equal, select the path with larger bandwidth
  **end if**
**end for**
**if** ($j$ ==**null**)  //no path is selected
  Wait any Path $p$ to free its $p.effwnd$ and distribute the packet to $p$;
**else**
  Distribute the packet to Path $j$;
**end if**
```

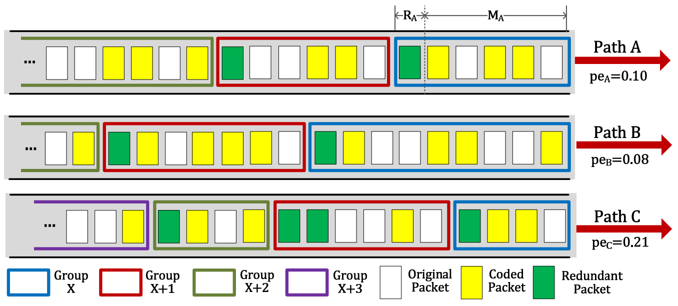


Fig. 3. Path buffer utilization in CMT-NC

side over this individual path, such that there can be totally N packets over multiple paths. Fig. 3 shows the utilization of path buffers. We can see that there is only one redundant packet in each path for one group in most cases, because Algorithm 1 will distribute more packets to the path with lower loss rate, such that the redundancy is balanced to minimum.

C. Group-Based Transmission Management

CMT-NC is creative to perform transmission management in units of group instead of individual packet, including congestion control, compulsory retransmission and path loss rate recording. All are detailed by Algorithm 2. We define *Group Time-Out* (GTO) as the Retransmission Time-Out (RTO) of the first packet (S_1) in this group. Two group pointers GC and GP are maintained in the management.

GC is the current group not yet confirmed to complete decoding by the receiver, whose packets are in the head of path buffers and waiting for acknowledgements. GC has similar usage for congestion control as outstanding packets in standard SCTP. That is, if SACK reports one new TSN in GC, execute slow start or congestion avoidance algorithm to update the path $cwnd$; if GC's missing report exceeds three, then carry out fast retransmission; if GTO is expired, then carry out GC's timeout retransmission. What is distinguishing, since the congestion control is based on overall group rather than individual packet, the update of $cwnd$ is reinforced, as well as the tolerance for missing report or SACK timeout is enhanced greatly. Therefore, $cwnd$ can rapidly grow to sufficiency and the retransmission is largely cut down.

Moreover, in case fast retransmission happens, the sender chooses one path p with most outstanding packets in GC

Algorithm 2 Group-based Transmission Management

```
if (SACK is received in one $GTO$)
  $minGroup =$ the latest group sent; //track the earliest group
  $maxGroup =$ the current group $GC$; //track the latest group
  /* Handling of the SACK */
  for ($\forall$ new or duplicated TSN $x$ in the SACK)
    Search the group $G$ that $x$ belongs to;
    $G.count$++ and free $x$ in $G$; $G.missing\_report = 0$;
    if ($G$ is before $minGroup$) $minGroup = G$; end if
    if ($G$ is after $maxGroup$) $maxGroup = G$; end if
    if ($G == GC$) //current group ack
      Find the Path $p$ that $x$ was sent over and then update $p.cwnd$;
      if ($GC.count \geq GC.N$) //grateful completion of GC decoding
        Free all original and coded packets in $GC$;
        Update the loss rate of all paths based on the past group $GP$;
        $GP = GC$ and $GC$ advances to the next group;
      end if
    end if
  end for
  /* Handling of fast retransmission */
  for ($\forall$ existing group $G$ between $minGroup$ and $maxGroup$)
    if (there is no TSN in the SACK belong to $G$) //group gap ack
      $G.missing\_report$++;
      if ($G.missing\_report > 3$)
        Select the Path $p$ with most outstanding packets in $G$;
        Adjust $p.ssthresh$ and $p.cwnd$ according to Eq.(9) and (10);
        Retransmit one different coded DATA packet in $G$;
      end if
    end if
  end for
else //$GTO$ is time-out
  /* Handling of time-out retransmission */
  for ($\forall$ Path $i$ with $RTO$ time-out TSN in $GC$)
    $i.ssthresh = \max(i.cwnd/2, 4 * MTU)$; $i.cwnd = MTU$;
  end for
  Retransmit $GC.N - GC.count$ coded DATA packets in $GC$;
end if
```

for the adjustment. We exploit the relation between BW_p and $cwnd_p/RTT_p$ as the discrimination of loss reasons: if $cwnd_p/RTT_p < BW_p$ holds, meaning the path is not congested, consider the loss is due to wireless error and then do not adjust $cwnd_p$; otherwise, attribute the loss to congestion and reduce $cwnd_p$ size.

$$ssthresh_p = \max(BW_p * RTT_p, cwnd_p/2, 4 * MTU) \quad (9)$$

$$cwnd_p = \begin{cases} cwnd_p & (\text{if } cwnd_p/RTT_p < BW_i) \\ \min(cwnd_p, ssthresh_p) & (\text{otherwise}) \end{cases} \quad (10)$$

where $ssthresh_p$ is the slow-start threshold of path p. It should be adjusted by one more factor $BW_p * RTT_p$ aligning to our bandwidth estimation. In case timeout retransmission unhappily happens, we just enforce the adjustment similar to standard SCTP. All retransmitted packets should be coded with TSN equaling to the last packet (S_N) in the group, which can also be detected by the "Duplicated TSN" fields in SACK. Finally, those packets should be retransmitted on another path q with maximum $BW_q(1 - pe_q)$ as soon as possible.

GP is the past group one before GC, whose packets are already released but remembering the reception state. GP is used for computing the path loss rate (pe). As SACK is applied, GP can record how many packets (including redundant and retransmitted packets) are sent over each path and know exactly which packets are successfully received. When GC is completely decoded, the sender collects the packet unacknowledged percentage per path from GP, regards it as one sample and then uses Confidence Interval [2] for statistical update of the path loss rate. Finally, it feedbacks the new set of path loss rate to the group number calculation. Both GC and GP advance to the next group for Algorithm 2 iterations.

TABLE I
PARAMETER SETTINGS FOR TWO PATH SCENARIO

Parameters	**Path A**	**Path B**
Access bandwidth	6Mbps	6Mbps
Queue length (model)	50 (Droptail)	50 (Droptail)
Default delay	50ms	55ms
Delay variation	fixed	5-105ms
Default loss rate (model)	0.05 (Uniform)	0.05 (Uniform)
Loss rate variation	fixed	0-0.1ms

III. PERFORMANCE EVALUATION

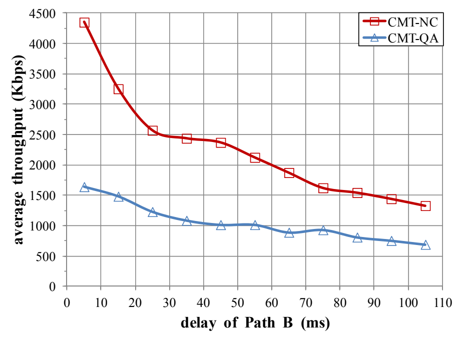
In this section, we evaluate the performance of CMT-NC on bulk data transmissions. As we have demonstrated that CMT-QA is an outsanding data-schceduling based scheme and outperforms classic CMT with round-robin scheduler [2], this section compares CMT-NC with the state-of-the-art CMT-QA to reveal the advantages in terms of throughput, delay and retransmission times. Beforehand, a series of dedicated tests have found out that in the group number calculation, RTT boundary factors $a = 0.9$, $b = 1.1$ and bandwidth smooth factor $T_0 = 1$s work the best for CMT-NC.

A. Parameter-Controlled Two Path Scenario

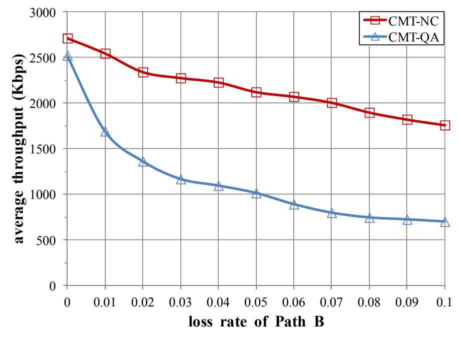
This simulation has been carried out on NS-2.35 that includes lastest SCTP (CMT) module developed by the University of Delaware. CMT-NC and CMT-QA are implemented based on this module. We arrange two mobile terminals directly connecting with each other via two independent paths (denoted Path A and B) using WiFi (IEEE 802.11a). The parameter setting of both paths is listed in Table I. As they are very similar in default, we change one specific parameter of Path B, while fixing Path A, to produce variation and dissimilarity. The receiver buffer is set to default 64KB, whereas the sender buffer is set to large enough. Other parameters use the SCTP suggested values. All results presented are the average of 100 trials with confident level 95% to guarantee the accuracy of the measurement.

Firstly, we change the delay parameter of Path B from 5ms to 105ms and obtain the throughput comparison in Fig. 4(a). CMT-NC has significant throughput improvement with respect to CMT-QA. This is because the strategy of network coding frees data reception from strict in-order delivery requirements, eliminates packet reordering and mitigates buffer blocking. So CMT-NC has powerful ability countering path dissimilarity. Additionally, our fast data distribution design has resulted in high utilization of path bandwidth resources.

Next, we modify the loss rate parameter of Path B from 0 to 0.1. Fig. 4(b) also presents throughput improvement of CMT-NC. The throughput variations demonstrate that the throughput of CMT-NC decreases linearly but CMT-QA suffers higher-order decrement with the loss rate of Path B growing, although the two schemes has similar throughput for zero loss rate. That proves CMT-NC is more robust and reliable for data transport



(a) delay variation



(b) loss rate variation

Fig. 4. Throughput comparison with variations in Path B

in lossy wireless environment. The reason lies in the group-based transmission management which protects the window growing and reduces retransmissions.

B. Simulated Vehicular Network Scenario

This simulation is to apply CMT-NC into practical network scenario and observe how it accommodates the dynamic and unreliable wireless environment. As shown in Fig. 5, we design a heterogonous (vehicular) network topology for the experiments based on our previous work [2][13]. The vehicle is downloading multimedia data (FTP, HTTP, video streaming and etc.) from the server via three independent paths (denoted Path A, B and C) with configurations listed in Table II.

In real deployment, the data rate is mainly constrained by the narrow bandwidth of the wireless access links. So the wired links bandwidth is set to 100Mbps in order not to limit the transmission in the core network (but to introduce delay). To simulate wireless error, we attach to each wireless links the Uniform loss model to represent distributed loss due to random contention, wireless interference or link handoff, plus the Two-State Markov loss model to represent infrequent continuous loss due to signal fading, transient failure or stream burst. Moreover, cross traffic is injected through the network to simulate the Internet background as [2] did. The aggregate cross traffic on each path varies randomly between $0-50\%$ of the access link bandwidth. All the wired and wireless links are set to 50 packets queue limit with Droptail queuing model, such that any saturated link would drop new incoming packets to simulate congestion loss.

In the simulation, the receiver buffer is varied to observe the performance of CMT-NC and CMT-QA, while the sender buffer is set to large enough. Other parameters use the SCTP suggested values. Each scheme starts at 5.0s, whereas the background traffic always starts at 0.0s. All results presented are the average of 100 trials with confident level 95%, which ensure the results not influenced by any stochastic factors.

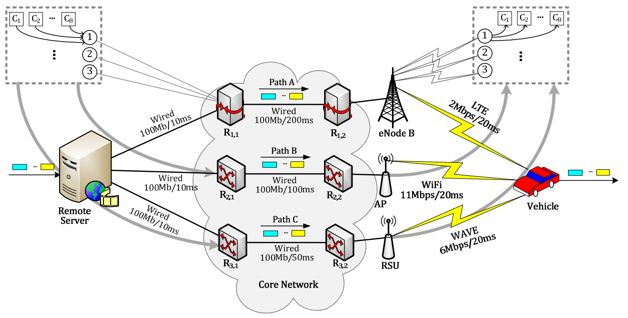


Fig. 5. Vehicular network topology in the simulation

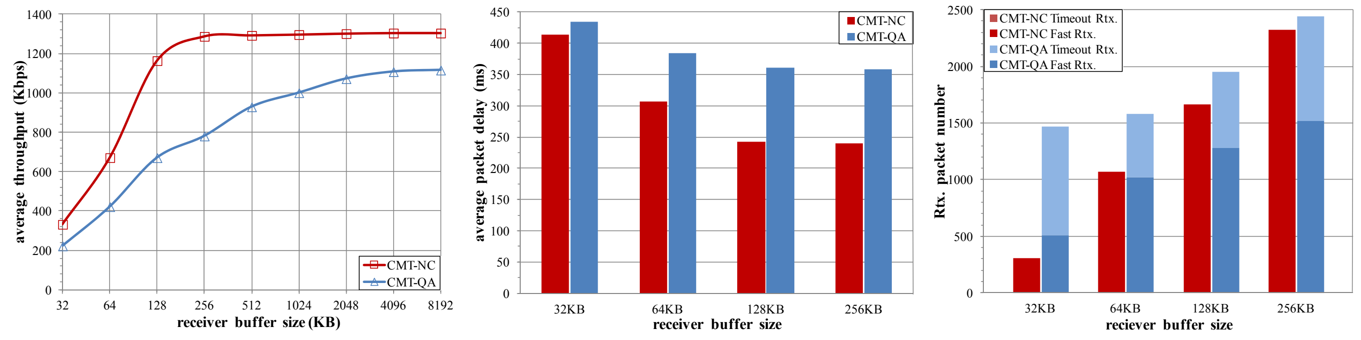
TABLE II
PARAMETER SETTINGS IN THE VEHICULAR NETWORK SCENARIO

Parameters	Path A	Path B	Path C
Wireless technology	LTE release 8	IEEE 802.11b	IEEE 802.11p
Access bandwidth	2Mbps	11Mbps	6Mbps
Access link delay	20ms	20ms	20ms
Core network delay	200ms	100ms	50ms
Uniform loss rate	0.01-0.02	0.01-0.1	0.1-0.2
Markov loss rate	0.01	0.01	0.01

Fig. 6(a) illustrates the average throughput of CMT-NC and CMT-QA. As expected, the two schemes have higher throughput with larger buffer size, since the tolerance for path dissimilarity is mainly determined by the receiver buffer capacity. CMT-NC outperforms CMT-QA all along, having significant 49.4%, 59.2%, 73.6% and 64.9% improvements in 32KB, 64KB, 128KB and 256KB buffer sizes, respectively. The reason is the same as Fig. 4(a) before.

Further with larger buffer size than 256KB, we observe that CMT-NC does not have an evident rise in throughput, whereas CMT-QA approaches CMT-NC performance. In this sense, CMT-NC requires 256KB receiver buffer to achieve stable throughput (a similar situation happens in [10]). On the other hand, CMT-QA still utilizes receiver buffer to store out-of-order packets for reordering. And larger buffer size means more powerful scheduling and reordering capabilities. Eventually the stable throughput is reached when using 4MB buffer size. However, even in the case of the 8MB buffer, CMT-NC still has 17.7% higher (stable) throughput than CMT-QA. This reflects the essential advantage of network coding in terms of throughput.

Fig. 6(b) reveals that CMT-NC has lower delivery delay than CMT-QA. As network coding has substituted its decoding operation for classic reordering in the receiver buffer, these results demonstrate that this kind of substitution is able to reduce the time spent in the receiver buffer and accelerate data delivery. In addition, the arrival of redundant packets can speed up the decoding, also helping reduce this delay. Meanwhile, the fast data distribution collaborates well with the properties of network coding and distributes more data over the faster path, which has reduced the transmission delay. When the buffer reaches some size, the delay also becomes stable and no longer decreases, as shown in the 128KB to 256KB buffer



(a) (b) (c)

Fig. 6. Performance comparison between CMT-NC and CMT-QA with different receiver buffer sizes

size tests. Again, CMT-NC's stable delay is better than that of CMT-QA, which reflects the essential benefits provided in terms of delivery delay by CMT-NC.

Fig. 6(c) shows the retransmission times when the two schemes separately transport one 150MB file. Each histogram differentiates two types of retransmission (fast and time-out). Larger buffer size will not only give rise to transmission rate, but also the probability of congestion and wireless error that lead to more retransmissions. Therefore, it is not a surprise that retransmission number increases as the buffer size grows for the two schemes. Obviously, CMT-NC has much less retransmission than CMT-QA for all buffer sizes, especially for 32KB and 64KB, even though it has higher throughput. In addition, CMT-NC has no timeout retransmissions, while CMT-QA has fair number of them.

There are three reasons for this: the first is the injected redundancy that can make up for the lost packets; the second is the principle of network coding that each packet can be supplemented by other packets in the same group, including redundant packets; the final and the most important is our design of group-based transmission management mechanism. It basically prevents timeout retransmission as long as the network is reachable and allows SACK replying. It also greatly suppresses fast retransmission by group-based control. The composite effect is to enhance the robustness and reliability of data transport in heterogeneous and lossy wireless networks.

IV. Conclusion and Future Work

This paper proposes an outstanding SCTP-based CMT Network Coding solution (CMT-NC) for efficient data delivery in heterogeneous wireless networks. CMT-NC takes advantage of network coding to free data delivery from strict in-order transmission requirements and packet-specific retransmissions. CMT-NC includes a hybrid fast data distribution mechanism which increases path resource utilization and reduces coding redundancy. The CMT-NC group-based transmission management enhances the robustness and reliability of data transport. Simulation results reveal how CMT-NC is able to improve the throughput, reduce the delivery delay, cut down retransmission times and lower buffer requirements in comparison to another state-of-the-art solution CMT-QA. Considering these excellent improvements, our future work targets fitting CMT-NC to multimedia delivery with specific quality requirements and mobility management in the future mobile Internet.

Acknowledgement

This work was supported in part by the National Key Basic Research Program of China (973 Program) under Grant No. 2013CB329102; the National Science Foundation of China (NSFC) under Grants No. 61372112 and 61232017; the Beijing Natural Science Foundation of China under Grant No. 4142037; the Natural Science Foundation of Jiangsu Province under Grant No. BK2011171.

References

[1] R. Stewart, "Stream Control Transmission Protocol," *RFC 4960*, IETF, Sept. 2007.

[2] C. Xu, T. Liu, J. Guan, H. Zhang and G. Muntean, "CMT-QA: Quality-aware Adaptive Concurrent Multipath Data Transfer in Heterogeneous Wireless Networks," *IEEE Trans. on Mobile Computing*, vol.12, no.11, pp.2193-2205, Sept. 2013.

[3] T.D. Wallace and A. Shami, "A Review of Multihoming Issues Using the Stream Control Transmission Protocol," *IEEE Communications Surveys & Tutorials*, vol.14, no.2, pp.565-578, June 2011.

[4] F. Perotto, C. Casetti and G. Galante, "SCTP-based Transport Protocols for Concurrent Multipath Transfer," in *Proc. IEEE WCNC*, Mar. 2007.

[5] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton and P. Hurtig, "Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)," *RFC 5827*, IETF, Apr. 2010.

[6] C.-M. Huang and M. Lin, "Fast Retransmission for Concurrent Multipath Transfer (CMT) over Vehicular Networks," *IEEE Communications Letters*, vol.15, no.4, pp.386-388, Apr. 2011.

[7] R. Ahlswede, N. Cai, S. Li and R. Yeung, "Network information flow," *IEEE Trans. on Info. Theory*, vol.46, no.4, pp.1204-1216, July 2000.

[8] J. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher and J. Barros, "Network Coding Meets TCP: Theory and Implementation," *Proceedings of the IEEE*, vol.99, no.3, pp.490-512, Mar. 2011.

[9] A. Ford, C. Raiciu, M. Handley, S. Barre and J. Iyengar, "Architectural Guidelines for Multipath TCP Development," *RFC 6182*, IETF, Mar. 2011.

[10] M. Li, A. Lukyanenko and Y. Cui, "Network Coding Based Multipath TCP," in *Proc. IEEE INFOCOM Workshop*, Mar. 2012.

[11] T. Dreibholz, E. Rathgeb, I. Rungeler, R. Seggelmann, M. Tuxen and R. Stewart, "Stream Control Transmission Protocol: Past, Current and Future Standardization Activities," *IEEE Communications Magazine*, vol.49, no.4, pp.82-88, Apr. 2011.

[12] N. Wagner (2003), The Laws of Cryptography With Java Code. [Online]. Available: http://www.cs.utsa.edu/~wagner/lawsbookcolor/laws.pdf

[13] C. Xu, F. Zhao, J. Guan, H. Zhang and G. Muntean, "QoE-driven User-centric VoD Services in Urban Multi-homed P2P-based Vehicular Networks," *IEEE Trans. on Vehicular Technology*, vol.62, no.5, pp.2273-2289, June 2013.