A Cache-Aware Congestion Control for Reliable Transport in Wireless Sensor Networks

Melchizedek I. Alipio(✉) and Nestor Michael C. Tiglao

Ubiquitous Computing Laboratory, EEE Institute,
University of the Philippines Diliman, 1101 Quezon City, Philippines
{melchizedek.alipio,nestor}@eee.upd.edu.ph

Abstract. Data caching and congestion control are two strategies that can enhance the transport reliability in constrained Wireless Sensor Networks. However, these two mechanisms are designed independently for most transport protocols developed for WSN. This work developed a new cache-aware congestion control mechanism for reliable transport. RT-CaCC utilizes cache management policies such as cache insertion, cache elimination and cache size to mitigate packet losses in the network while maximizing cache utilization and resource allocation. It uses two cache management policies for packet loss detection: implicit notifications and expiration of timeout. In addition, it utilizes congestion avoidance using cache-aware rate control mechanism employing transmission window limit as a function of cache size. Results showed that the RT-CaCC obtained significant improvement gain in terms of cache utilization, end-to-end delay and throughput performance specifically during high level of packet loss in the network.

Keywords: Cache-aware · Congestion control · Intermediate caching Internet of Things · Wireless Sensor Networks

1 Introduction

A typical Wireless Sensor Network (WSN) is consists of tiny nodes that are equipped with embedded computing devices interfacing with sensors or actuators. These sensor networks are vital component of Internet of Things (IoT) and characterized as constrained networks due to limited memory, computing and energy capability. A sizable set of these nodes is dispersed over a wide geographical area to monitor a physical or environmental event. Therefore, packets generated at source nodes are usually transmitted to the sink through multi-hop communication [1]. In effect, these networks experience high probability of packet drop due to poor link quality, link contentions or buffer overflows. Thus, an effective transport protocol is a must. One mechanism of improving reliability is intermediate caching thru local retransmissions. Another mechanism is to utilize congestion control strategies which can alleviate packet losses due to network congestion. Some transport protocols combined these two mechanism to further enhance the reliability of data transport in WSN.

To the best of our knowledge, this paper makes the following contributions: (1) design a cache-aware congestion control mechanism based on different cache management policies which are utilized to optimize cache utilization of the transport protocol. To achieve this intention, we (2) perform simulations to evaluate and analyze the performance gain improvement of the congestion control mechanism as compared with the baseline protocols.

The rest of the paper is organized as follows: Sect. 2 discusses some of the related works. In Sect. 3, we discuss transport layer caching and the cache-aware congestion control. In Sect. 5, we discuss the simulation environment, results and discussions. Finally, Sect. 6 concludes the paper.

2 Related Work

Existing transport protocols in WSN combined both reliability and congestion control mechanisms to improve the performance of packet transmissions. Increasing the reliability is achieved by using local retransmission at intermediate nodes. While congestion control alleviates the network during high level of packet losses due to poor wireless quality, contending flows or buffer overflow. DTC [2] and TSS [3] modified TCP protocol in order to provide direct TCP/IP - WSN compatibility which is implemented in the intermediate nodes and requires no protocol changing in the end nodes. It uses intermediate caching and provides hop-by-hop reliability. Segments are cached at the intermediate nodes based on the highest sequence number and link layer feedback. However, it uses AIMD rate control mechanism to mitigate packet losses during congestion states. Therefore, it does not classify packet losses due to wireless link error which makes the congestion window oscillates aggressively.

On the other hand, ERCTP [4] implements a modular approach for congestion control and reliability mechanisms. Its source rate adjustment module defines the new transmission rate adjustment for child nodes in order to mitigate congestion. It monitors the instantaneous network statistics which helps sink to explicitly and periodically send the estimated value of rate adjustment to source nodes, which is obtained based on congestion index calculation. The use of explicit notification to infer possible congestion may lead to additional traffic load to the network. Thus affecting the throughput performance of the transport protocol. Moreover, RCRT [5] also uses AIMD while it adapts the total aggregate rate of all the flows as observed by the sink, rather than the rate of a single flow. Whenever RCRT determines the network is congested, it applies the rate decrease step by time-dependent multiplicative decrease factor, computes a new rate allocation for all the flows, and sends the new rate for each flow to the corresponding source.

Intermediate caching and congestion control mechanisms are different techniques that effectively improve reliability of data transport performance in WSN. However, these two mechanisms are designed independently for most transport protocols developed. In previous works, congestion control techniques are not cache-aware. This can possibly result to non-optimal use of intermediate caching, inappropriate congestion window movement and increase in energy consumption of intermediate nodes. To the best of the our knowledge, no study has yet to develop a cache-based transport protocol that has an appropriate congestion control mechanism that can improve cache utilization, network efficiency, and resource allocation.

3 Overview of Transport Data Caching and Management Policies

Intermediate caching alone can mitigate packet losses either due to contentions or congestion from local retransmissions up to a certain optimal transmission window size [6]. In addition, the information from cache elimination policies and cache size can be used to adapt the source rate control in improving the performance of a cache-based transport protocol in terms of throughput, end-to-end delay and cache utilization [7]. These ideas provide the underlying support for the new cache-aware congestion control integrated in a transport protocol (RT-CaCC).

The caching mechanism of the RT-CaCC protocol was inspired by the DTSN$^+$ [8]. RT-CaCC uses both positive acknowledgment (ACK) and selective negative acknowledgment (NACK) to be sent from the receiver upon the request of the sender through an Explicit Acknowledgment Request (EAR). The EAR signal is piggybacked on to a data packet. After sending an EAR, the source launches an EAR timer. If the EAR timer expires before an ACK/NACK is received, the source retransmits the EAR packet. Upon the reception of EAR at the receiver node, a NACK, which contains a bitmap of missing packets, is generated and transmitted back to the sender. While relaying such NACKs, intermediate nodes learn about the missing packets and check if those packets are present in their cache. If so, the intermediate nodes retransmit those packets towards the receiver and modify the NACK bitmap accordingly before sending it towards the sender. Likewise, RT-CaCC adapts a NACK repair mechanism whereby intermediate nodes can issue NACK signals to hasten the repair process. In addition to receivers being able to detect lost packet, intermediate nodes detect packet loss and signal the previous hop node through a repair negative acknowledgment (RNACK) control packet that contains the sequence number of lost packet. Upon receiving the RNACK, the previous hop node will retransmit the lost packet towards the destination if a copy is found in the cache. If not, the RNACK will be propagated towards the source. The sending of the RNACK is not timer-driven but triggered as soon as an out-of-sequence packet is detected. This feature further reduces the probability of packet loss from poor wireless link errors.

At intermediate nodes, RT-CaCC used cache insertion policy with a certain probability which can be based on cache partitioning scheme [9]. The probability value must be chosen to maximize cache utilization of data packets requested by the NACK along the reverse path. At each intermediate node, the total cache size CS is divided among the different flows that cross the node. Let $\omega_i^n \geq 0$ be a weight related to the fraction of cache at node n that is assigned to flow i. The actual fraction is given by the normalized weight $\rho_i^n = \frac{\omega_i^n}{\sum_{j=1}^{F_n} \omega_j^n}$. A packet that belongs to flow i can only be cached in the fraction of cache assigned to flow i, whose size is equal to $\rho_i^n \times CS$. Consequently, the caching probability P_{cache} for a packet that belongs to flow i at node n is given by $min\,(1, \rho_i^n \times CS)$. Considering that each node in the network is simultaneously cross by more than one flow wherein all flows have equal hop length, an equitable way for partitioning the cache is to divide it equally between the flows. In this case, a uniform cache partitioning can be implemented where each flow is assigned the same ω_j^n to achieve fairness. However, for complex scenarios with higher number of concurrent flows, heterogeneous link quality and length, a non-uniform cache partitioning can be used [9] and is out of the scope of this study.

RT-CaCC also used cache elimination policy in the form of implicit notifications that the sender nodes receive: ACK ($holes_{ACK}$), NACK without holes ($holes_{NACK} = 0$) and NACK with holes ($holes_{NACK} \neq 0$). Holes represent the number of packets that were not successfully retransmitted by intermediate caching. Therefore, the list of packets in the hole is retransmitted by the sender together with the new batch of packets. From the three notifications, the reception of NACK with holes indicates a high degree of packet loss that the local retransmissions from intermediate caching was not able to handle.

The probability of success in an end-to-end transmission can be evaluated by the number of hops H and probability of packet loss P_{loss} as $(1 - P_{loss})^H$ shown in Fig. 1a [10]. When caching is use at intermediate nodes and assuming that the given CS can store all the packets needed for local retransmission, the expected number of transmissions (ENT) as shown in Fig. 1b is given by $H \cdot \sum_{i=0}^{\infty} P_{loss}^i$.

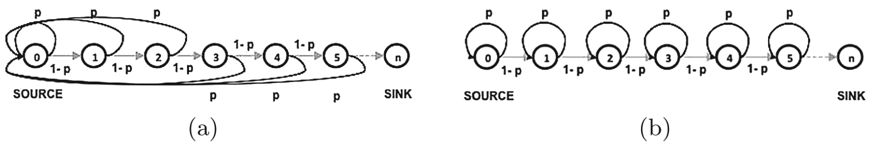


Fig. 1. Data packet transmissions probabilistic model: (a) without intermediate caching and (b) with intermediate caching

However, since intermediate nodes are constrained and CS has limited capacity, the number of packets to be inserted into the cache can be based on a certain probability. This caching probability can be based on cache partitioning scheme P_{cache}. Therefore, the actual expected number of transmissions is given by

$$ENT_{cache} = H + \sum_{i=1}^{\infty} P_{loss}^i \cdot \sum_{h=1}^{H} \left[1 + P_{cache}^{h-1} \cdot ENT(h-1) + \sum_{j=2}^{h-1} P_{cache}^{h-j} \cdot ENT(h-j) \right] \quad (1)$$

where ENT is the expected number of transmission without caching which is given by

$$ENT = H + \left[\frac{\sum_{h=1}^{H-1} P_{loss} \cdot (1 - P_{loss} \cdot h)}{1 - (1 - P_{loss})^H} + 1 \right] \cdot \sum_{i=1}^{\infty} (1 - (1 - P_{loss})^H)^i \tag{2}$$

In this case, retransmission can be performed from any of the nodes in the forward path that are behind the hop link where the packet loss occurred taking into consideration the best possible value in P_{cache}.

4 Cache-Aware Congestion Control

The context of cache-aware is based on different cache management policies which are utilized by the congestion control mechanism to mitigate packet losses while optimizing cache utilization and bandwidth allocation. This approach assumes that the network is uncongested as long as end-to-end losses from transient congestion and poor wireless links are repaired immediately. Furthermore, it permits the source to transmit at a higher rate even if there are occasional end-to-end losses, since these losses can be recovered by intermediate caching.

The RT-CaCC protocol utilizes cache-aware strategies to detect, notify and avoid packet losses due to poor wireless channel or buffer congestion in the network discussed as follows:

4.1 Packet Loss Detection Using Cache Elimination Policy

RT-CaCC utilizes implicit notifications in the form of cache elimination policy to infer the approximate degree of packet losses. NACK notifications can provide the most updated packet loss level in the network as they know the number of packets already recovered through the updated bitmap as they travel along the return path and the number of packets (holes) which are not. This strategy eliminates the need for channel probing while making the most effective use of cache space. Therefore, it will lessen the sensor node's memory and computing requirements, energy consumption and maximize cache utilization. However, in this strategy, there is no way to differentiate the packet losses. In low power and lossy types of networks such as WSN, previous results show that when congestion occurs, the majority of packets are lost due to node level congestion as compared to link level contention [6]. Therefore, RT-CaCC uses a second strategy to mitigate packet losses mainly due to buffer congestion.

4.2 Packet Loss Detection Using EAR Timeout (*ETO*)

The expiration of EAR timer is used to detect node-level congestion which predominates during high level of network traffic in WSN. EAR timeout *ETO* is dynamically set using congestion round-trip time (RTT). In ad hoc wireless networks, RTT is compose of processing, queuing, transmission, propagation and contention delays. To estimate the congestion RTT of a segment, RT-CaCC marks the time when the frame reaches the output queue header at the intermediate node. The estimation only uses the output queue since it is assumed that the input queue only refers to the routing path availability and discovery and is out of the scope of this work. RT-CaCC does not include contention delay in the RTT estimation to ensure that the delay is mainly contributed by buffer overflow. The total time from propagation to transmission delay is designated as one-hop delay of a packet at node n indicated as d_h^n assuming the clock of adjacent nodes is synchronous. For N number of hops in the forward path, the previous d_h^n is added to the current d_h^n until it reaches the sink node. Once the sink node received the EAR notification piggybacked in the last packet, the sum of all d_h^n for M number of packets in a segment is computed as $T_{delay-DATA}$. Afterwhich, an implicit notification either ACK or NACK is propagated in the return path until it reaches the destination and the total time of propagation corresponds to $T_{delay-ACK}$. Therefore, the total congestion RTT is given by $RTT = T_{delay-DATA} + T_{delay-ACK}$.

When the source node transmits the EAR notification, an EAR timeout is set automatically. In order to dynamically set the EAR timeout, total congestion RTT is used taking into consideration the additional delay caused by local retransmissions from intermediate caching. In WSN, the local retransmissions performed by intermediate caching at wireless link can cause the end-to-end RTT to increase significantly in a short time due to the long processing delay. In effect, huge time gaps in receiving ACK and NACK at the sender nodes is experienced. When there are no packet loss and an $holes_{ACK}$ is received at the sender, the estimated delay variation is minimal and EAR timer is set accordingly. On the other hand, if there are frequent packet loss occur, a $holes_{NACK} \neq 0$ is received at the sender and local retransmissions are triggered, the estimated EAR timer should be adjusted in order to handle delay variation caused by local retransmissions from intermediate caching. Therefore, an additional delay interval δ_{delay} is computed as a function of estimated congestion RTT, probability of packet loss and caching probability in (1) leading to $\delta_{delay} = RTT \times ENT_{cache}$. It can be noticed that when P_{loss} and P_{cache} approaches to 1, most packets are being lost and are need to be recovered. Therefore, end-to-end delay should be increased by at least another RTT in order for these lost packets to be recovered. To attain this, ETO is computed at the sender as

$$ETO = RTT + \delta_{delay} \tag{3}$$

The expiration of ETO before an ACK or NACK is received at the sender node infers a buffer congestion. The sender node will act by retransmitting an EAR packet towards the sink node and reduces its transmission rate accordingly. This approach could be seen as an extension of the classical TCP algorithm. However, instead of constants that are used to take into account history of the current state, dynamically changing parameter is used.

4.3 Congestion Avoidance at the Source Node

At the sender node, a cache-aware rate control strategy was used by RT-CaCC based on packet loss detections described in the previous sections. It also used a bounded congestion window size based on bandwidth-delay product (BDP) and CS. The idea is to limit the upper transmission window size to the average BDP of the network which serves as the congestion window limit (CWL). BDP is an important indicator of network capacity which refers to the maximum number of bits a connection can accommodate. Therefore, the total number of outstanding data packets, like in-flight or unacknowledged ones, cannot exceed this upper bound. For constrained networks like WSNs, it is important that a transmission window limit must be used and tuned to an appropriate value to ensure sufficient pipelining and to avoid the risk of overloading the network. In addition, RT-CaCC used CS value as the minimum transmission window size to optimize cache utilization since moving the window below CS can lead to sub-optimal cache performance. Assuming that CS is always less than the total buffer size B, cache-based transport protocol starts to obtain optimum cache utilization when the transmission window is equal to CS. On the other hand, RT-CaCC can achieve the optimum throughput at CWL as a function of B.

RT-CaCC used the average BDP (BDP_{ave}) as the upper bound of CWL as a function of congestion RTT discussed previously. RT-CaCC only considers BDP determined by the time that data packets flow continuously. In this case, contention delay is not included since it is the period where packets are temporarily blocked by the node when contending to access the wireless channel to send the data and is not an indicator of available bandwidth capacity. Therefore, RT-CaCC only uses the output buffer in the determination of BDP_{ave}. However, to ensure optimum cache utilization, BDP_{ave} is only used if $BDP_{ave} > W_{CS}$ wherein W_{CS} is the window size equal to CS. Since packets spend longer time passing through a bottleneck link, RT-CaCC measures the available bandwidth based on its share at this link. Since the BDP carried out by each packet is continuously changing, RT-CaCC computes the BDP_{ave} at the sender node using $BDP_{ave} = \frac{\sum(\frac{S}{d_{max}}) \times RTT}{M}$ where S is the packet size, M is the total number of observed packets and d_{max} is $max(d_h^n), 1 \leq n \leq N$. In order to achieved this, RT-CaCC protocol used additional packet header fields.

Let $R(t)$ denote the rate allocated for current congestion window (W_t) which is calculated at the sender node once implicit notifications are received. The rate control mechanism used an AIMD on $R(t)$ which is counteractive with the traditional TCP-AIMD. When the network experience packet losses either due to poor channel quality or link contentions, intermediate caching will act primarily to perform local retransmissions. The rate control mechanism of RT-CaCC protocol is summarized in Algorithm 1.

If the sender node receives an ACK or NACK without holes, $R(t)$ additively increases as a function of α. If the source node receives NACK with holes, $R(t)$ multiplicatively decreases by β. The idea is to dynamically adjust the transmission rate according to packet loss level. This will continue until W_t reaches BDP_{ave}. On the other hand, the expiration of ETO will decrease W_t equal W_{CS}.

Algorithm 1. Rate Control Algorithm at the Source Node

```
1: procedure PKT_RECV(PKT)
2:     %compute the value of $RTT$ and $\delta_{delay}$
3:     %compute the value of $ETO$ from previous segment
4:     %set $ETO$ after transmission of last packet
5:     if ($ETO$ expires) then
6:         decrease current window to minimum limit $W_{CS}$
7:         %retransmit EAR notification to sink node
8:     else
9:         if (packet header received is $holes_{NACK} \neq 0$) then
10:            if (check current window $W_t > W_{CS}$) then
11:                $R(t) = [(W_t - W_{CS}) \times \beta] + W_{CS}$
12:            else
13:                set current window to minimum limit $W_{CS}$
14:            end if
15:        else
16:            %compute the value of $BDP_{ave}$
17:            %use $BDP_{ave}$ if $BDP_{ave} > CS$; otherwise $CWL = CS$
18:            if (check current window $W_t < BDP_{ave}$) then
19:                $R(t) = W_t + (\alpha \times \frac{S}{RTT})$
20:            else
21:                set current window to maximum limit $BDP_{ave}$
22:            end if
23:        end if
24:    end if
25: end procedure
```

The idea is to move W between BDP_{ave} and W_{CS} to prevent network overload while optimizing cache utilization. The values of α and β use increase-by-one and decrease-to-half strategy, respectively, to lessen the effect of aggressive window oscillation that can lead to low network resource utilization.

5 Simulations and Results

Simulations in NS-2 were carried out to evaluate the performance of the RT-CaCC protocol. For the network scenario, it is assumed that all intermediate nodes have CS equal to 10 packets while buffer size is set to 50 packets (default in NS-2). Packet size is set to 500 bytes and each topology used fixed routing (FIXRT) with Binary Symmetric Channel (BSC). Since 802.11b set to 1 Mbps MAC protocol was used, RTS/CTS are disabled and the number of retries was set to 4 in order to make it as similar as possible to 802.15.4 MAC protocol standard for WSN. To evaluate the performance of the RT-CaCC, end-to-end delay, throughput, transmission cost and cache hit are used as primary metrics. *Cache hit* refers to the number of times the data are requested from the intermediate nodes cache memory while *transmission cost* is the total number of data packets, transport layer ACK and NACK packets and MAC layer ACKs per total number of packets that need to be delivered end-to-end.

5.1 Varying P_{cache}

The number of received NACK at the sender node was evaluated while varying the caching probability in a 10-node linear topology. From Fig. 2, for 100%P_{cache}, fewer number of retransmissions is required than with $0\% P_{cache}$. It can be observed that using 50% probability is still efficient in reducing the number of packet retransmissions. However, lower P_{cache} can lead to more retransmissions thus requiring more energy consumption from the congestion control. Appropriate value of P_{cache} should be taken into account due to its effect in RTT and ETO variations. Larger P_{cache} supports more local retransmission from intermediate. Thus, it requires longer RTT and ETO expiration time.

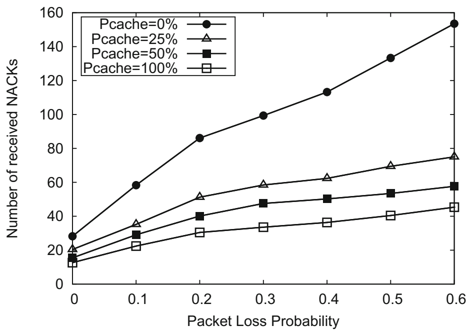


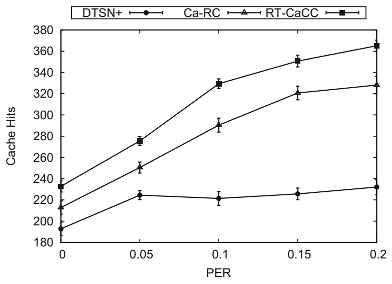
Fig. 2. Number of received NACKs at the sender node

5.2 Varying Packet Error Rates

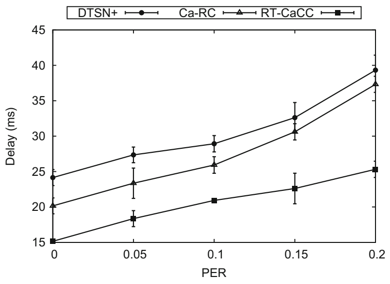
To evaluate the improvement gain of the congestion control mechanism, the RT-CaCC was simulated against the original DTSN$^+$ with fixed transmission window and the modified DTSN$^+$ with dynamic cache-aware rate control mechanism designated as Ca-RC [7]. $50\% P_{cache}$ is uniformly allocated to all intermediate nodes. A linear topology composed of 10 nodes wherein 8 nodes served as intermediate caching nodes with 20% packet error rate was used. RT-CaCC gained 48.09% and 30.88% throughput improvement gain against DTSN$^+$ and Ca-RC, respectively. It also achieved 35.61% and 26.15% end-to-end delay improvement gain as compared with DTSN$^+$ and Ca-RC, respectively. Finally, 36.43% and 10.13% cache hits improvement gain were obtained better than DTSN$^+$ and Ca-RC, respectively. The addition of ETO component of RT-CaCC significantly maximized cache utilization by adapting to variations in RTT due to local retransmissions. Although the minimal throughput gain can be attributed to conservative congestion window, this can be further improved by setting the appropriate P_{cache} (Fig. 3).

5.3 Link Contentions with Multiple Flows

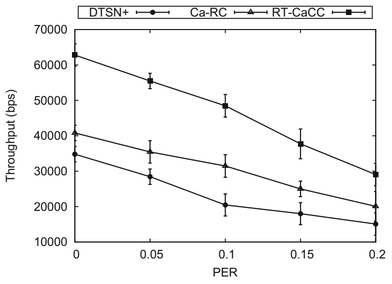
Figure 4 shows the results of varying the number of flows in contending flow topology. In terms of delay, the cache-aware congestion control performed well with 34.76% and 27.97% average gain difference than DTSN$^+$ and Ca-RC,



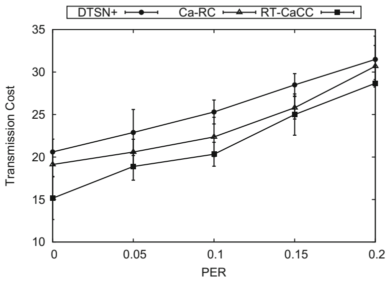
(a)



(b)



(c)



(d)

Fig. 3. RT-CaCC performance with varying PER: (a) cache hits, (b) end-to-end delay, (c) throughput and (d) transmission cost

respectively, at worst case. It shows that the cache-aware retransmission timeout mechanism of RT-CaCC is effective enough to reduce the end-to-end delay which is very important in an event-based applications. In addition, RT-CaCC was able to prevent the network from collapsing and recover immediately specifically at higher buffer overflows. It is mainly evident from the high throughput performance gain obtaining 20.92% and 10.13% as compared with DTSN$^+$ and Ca-RC, respectively. In terms of cache hits, RT-CaCC achieved 18.71% and 9.97% gain improvement against DTSN$^+$ and Ca-RC, respectively. It can be deduced that using cache elimination policy such as implicit NACK notification is effective in mitigating link-level congestion from contending flows than its predecessor protocols.

5.4 Bottleneck Link with Multiple Flows

Two RT-CaCC flows were also evaluated in a bottleneck topology entering the link while varying cache size values. Figure 5 shows the results wherein 20% and 10% cache hit improvement gain were achieved by the RT-CaCC as compared with DTSN$^+$ and Ca-RC, respectively. The protocol also gained 47% and 25% end-to-end delay improvement gain against DTSN$^+$ and Ca-RC, respectively. Further, it also achieved 14% and 5% throughput improvement gain compared

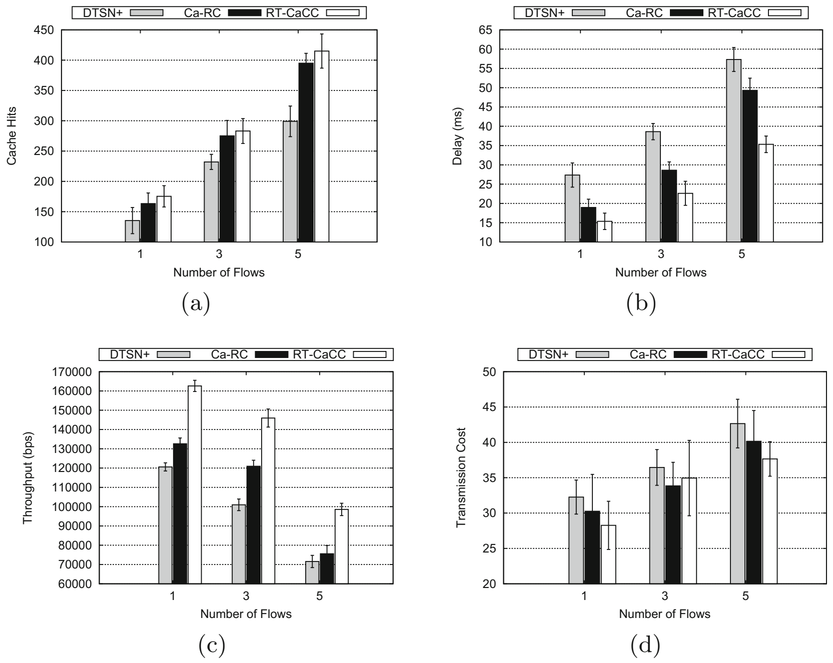


Fig. 4. RT-CaCC performance with varying number of flows: (a) cache hits, (b) end-to-end delay, (c) throughput and (d) transmission cost

to DTSN$^+$ and Ca-RC, respectively. This only shows that the ETO mechanism which uses congestion RTT is effective enough to detect and mitigate congestion due to buffer overflow. It can also be deduced that the value of CS affects the behavior of RT-CaCC protocol. Ideally, larger CS should be used since it will also increase the number of retransmissions for successful recovery of packet losses. However, in real-life scenario, smaller CS should be used due to the constrained characteristics of WSN.

5.5 Congestion Window Response

The congestion window behavior of RT-CaCC was also observed in a bottleneck link topology with increasing number of flows. Figure 6 shows the congestion window response of RT-CaCC and DTC protocols. DTC [2] was a modified TCP with caching mechanism which uses the traditional TCP-AIMD algorithm as its congestion control mechanism. Flows were injected in the bottleneck link incrementally every after 100 s simulation time. It can be observed that from 100 s to 200 s wherein a single flow of traffic is entering the link, less were the number of retransmission timeout occurrence for both protocols. In addition, both protocols frequently reached the upper window limit BDP_{ave} which indicates that the bandwidth allocation was maximized. It can be seen that the BDP_{ave}

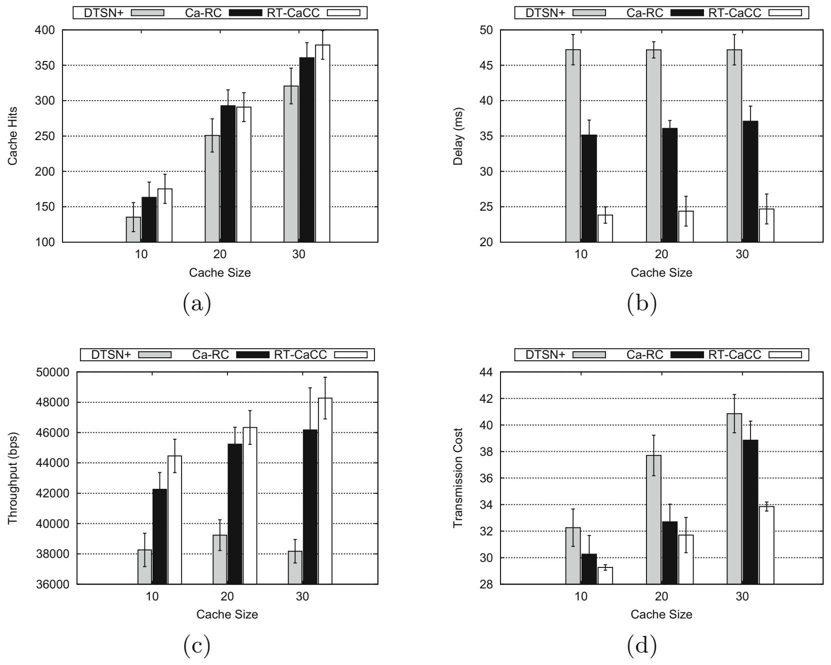


Fig. 5. RT-CaCC performance with varying cache sizes: (a) cache hits, (b) end-to-end delay, (c) throughput and (d) transmission cost

is always almost equal to B which ensures full buffer utilization. From 200 s to 300 s wherein an additional flow was injected in the bottleneck link, RT-CaCC registered fewer number of retransmission timeout than DTC. From 300 s to 400 s wherein three simultaneous traffic flows were entering the link, RT-CaCC

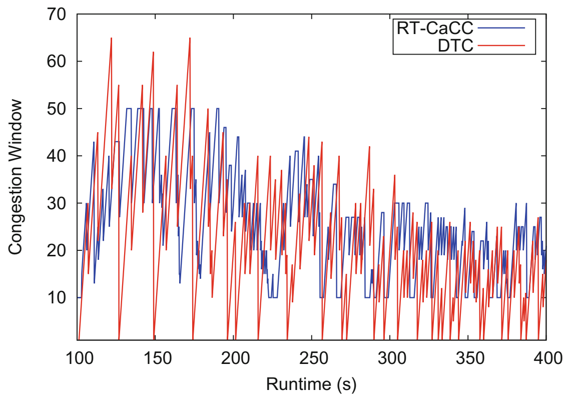


Fig. 6. Congestion window with increasing number of flows in a bottleneck link

obtained better bandwidth usage than DTC at high level of congestion. In addition, RT-CaCC congestion window is less aggressive than DTC which lead to better resource allocation and optimum cache utilization.

RT-CaCC was also evaluated with other cache-based transport protocols like DTC and ERCTP. These protocols also implement congestion control mechanisms but not cache-aware. Due to page limitation, the results of the comparison will be presented in the extended version of this work in another paper.

6 Conclusion

This work developed a cache-aware congestion control mechanism that uses cache management policies such as cache insertion and elimination policy as well as cache size allocation to mitigate packet losses from poor wireless link and congestion in the network. RT-CaCC outperformed the baseline protocols in terms of cache utilization, end-to-end delay and throughput from 10% to 50% average improvement gain. This only shows that using a cache-aware approach can effectively respond to packet losses either due to poor channel link or congestion in the network. Further, limiting the lower and upper bounderies of the congestion window during high level of packet losses guaranteed optimum usage of cache memories while preventing the network from overshooting. In the future, RT-CaCC can be evaluated in a more challenging network scenario as well as incorporating a cache-aware cross-layer approach with the routing protocol in WSN such as RPL.

Acknowledgment. The authors would like to acknowledge the support of the University of the Philippines Diliman and the Department of Science and Technology (DOST) through the Engineering Research and Development for Technology (ERDT) Program.

References

1. Akyildiz, I., Vuran, M.C.: Wireless Sensor Networks. Wiley, New York (2010)
2. Dunkels, A., Alonso, J., Voigt, T., Ritter, H.: Distributed TCP caching for wireless sensor networks. In: Proceedings of the 3rd Annual Mediterranean Ad-Hoc Networks Workshop (2004)
3. Braun, T., Voigt, T., Dunkels, A.: TCP support for sensor networks. In: Fourth Annual Conference on Wireless on Demand Network Systems and Services, WONS 2007, pp. 162–169, January 2007
4. Sharif, A., Potdar, V.M., Rathnayaka, A.J.D.: ERCTP: end-to-end reliable and congestion aware transport layer protocol for heterogeneous WSN. Scalable Comput.: Pract. Exp. **11**(4), 359–372 (2010)
5. Paek, J., Govindan, R.: RCRT: rate-controlled reliable transport protocol for wireless sensor networks. ACM Trans. Sens. Netw. **7**, 20:1–20:45 (2010)
6. Alipio, M.I., Tiglao, N.M.C.: Analysis of cache-based transport protocol at congestion in wireless sensor networks. In: 2017 International Conference on Information Networking (ICOIN), pp. 360–365, January 2017

7. Alipio, M.I., Tiglao, N.M.C.: Improving reliable data transport in wireless sensor networks through dynamic cache-aware rate control mechanism. In: 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), October 2017 (to appear)
8. Tiglao, N.M.C., Grilo, A.M.: Cross-layer caching based optimization for wireless multimedia sensor networks. In: 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 697–704, October 2012
9. Tiglao, N.M.C., Grilo, A.M.: An analytical model for transport layer caching in wireless sensor networks. Perform. Eval. **69**(5), 227–245 (2012)
10. Meneses, D., Grilo, A., Pereira, P.R.: A transport protocol for real-time streaming in wireless multimedia sensor networks. In: 2011 7th EURO-NGI Conference on Next Generation Internet Networks, pp. 1–8, June 2011