Optimal cache partitioning in reliable data transport for wireless sensor networks

Nestor Tiglao, Antonio Grilo

Optimal cache partitioning in reliable data transport for wireless sensor networks

Nestor Tiglao[1,3], António Grilo[1,2]
[1]INESC-ID/INOV, Rua Alves Redol, N° 9, 1000-029 Lisboa, Portugal
[2]Instituto Superior Técnico, Avenida Rovisco Pais, 1, 1049-001 Lisboa, Portugal
[3]EEEI, University of the Philippines, Diliman, Quezon City, 1101 Philippines
Email: nestor.tiglao@inesc-id.pt, antonio.grilo@inov.pt

***Abstract*—Traditional reliable transport protocols have been designed to perform end-to-end delivery transparently to the intermediate nodes along the path (e.g., TCP). The resource-constrained nature of a Wireless Sensor Network (WSN) requires a new paradigm where intermediate nodes are able to cache packets and retransmit them on-demand during packet loss recovery in order to avoid incurring costly end-to-end retransmissions. This paper addresses the problem of cache partitioning optimization at the transport layer. We implemented several caching policies integrated into the DTSN transport protocol and analyzed their effect on overall network performance in terms transmission cost, throughput, and fairness. Our simulation results show that the choice of cache partitioning policy has a very significant impact on the performance and motivate the need to develop more advanced policies that adapt to dynamic network conditions and more complex topologies.**

***Index Terms*—reliable data transport, caching, optimization, WSN**

I. Introduction

A Wireless Sensor Network (WSN) is typically composed of small autonomous resource-constrained devices that transmit data from sensor nodes to one or more sink nodes. Since WSN nodes operate autonomously, power saving techniques are usually complemented with low power radio communications that lead to multihop data transmission from the sensor nodes to the sink nodes and vice versa. Besides leading to a lower throughput, multihop communications are also the cause of additional interference and hidden terminal problems due to spatial reuse, which, complemented with the fact that most WSN standards such as IEEE 802.15.4 [1] specify operation in unlicensed ISM bands, means that radio links are usually more error-prone than in typical WLANs.

Despite these limitations, certain WSN applications require complete data delivery. One example is the transmission of discrete audio and video snapshots for surveillance and monitoring purposes. Another example is a data collection application where sensor nodes accumulate readings over a period of time and reliably transmit them to a mobile data mule.

Corresponding Author: Nestor Michael C. Tiglao, PhD student, INESC-ID/INOV, Rua Alves Redol, N° 9, 1000-029 Lisboa, Portugal, Tel: +351 927 162113, Fax: +351 21 3100317, Email: nestor.tiglao@inesc-id.pt

The traditional design of reliable transport protocols is end-to-end delivery of data segments from the source to the destination transparent to the intermediate nodes along the path (e.g., TCP). Part of the transport layer's function is packet loss detection and recovery of lost segments, which can be performed either end-to-end or hop-by-hop. In the end-to-end approach, the end-points (sender or receiver) are responsible for loss detection and initiating loss recovery. On the other hand, in the hop-by-hop approach, the intermediate nodes along the path from the sender to the receiver are responsible for these functions.

The resource constraints present in WSNs require a different paradigm where intermediate nodes are able to cache packets and retransmit them on-demand in order to avoid incurring costly end-to-end retransmissions. In realistic WSN scenarios, intermediate nodes may have multiple concurrent flows that traverse them. Although previous works proposed the use of caching at intermediate nodes in the transport layer, they have analyzed its effect on a single flow only. To the best of our knowledge, this study is the first to identify and study the problem of optimal cache partitioning in the transport layer where the cache is partitioned for multiple concurrent flows.

This paper discusses the importance and impact of cache partitioning in reliable data transport protocols. We start by providing a brief survey of WSN transport protocols that employ the use of caching. Next, we introduce the optimal cache partitioning problem. Then, we show by simulations using two network topologies the performance of different caching policies and their effect on overall network performance in terms of transmission cost, throughput, and fairness.

The rest of this paper is structured as follows. In section II, we provide a survey of transport protocols that employ caching. In section III, we define the optimal cache partitioning problem and explain the caching policies we considered in this study. In section IV, we present and discuss our simulation results. Finally, section V concludes the paper.

II. Survey of WSN reliable transport protocols that employ caching

In this section, we present a survey of the state of the art reliable transport protocols designed for WSNs that advocate the use of caching at intermediate nodes to improve network

performance. We classify these protocols according to the packet loss detection and signalling approach they use and explain how caching is used. We also comment of how cache additions and removals are implemented. A general survey of WSN transport protocols is found in [2], [3].

Pump Slowly Fetch Quickly (PSFQ) [4] is a protocol primarily designed for dynamic code distribution from the sink to the sensors and works using broadcast transmissions. The sink sends the fragments at a slow rate (pump slowly), allowing some time for the intermediate nodes to fetch missing fragments (if any) in a quick manner (fetch quickly). Intermediate nodes store received fragments and forward them to downstream nodes when they received in sequence. When an intermediate node detects an out-of-sequence segment, it does not forward it immediately but quickly issues a NACK in order to request missing fragments from its neighbors. Thus, loss recovery is local, not end-to-end. In between pumps, intermediate nodes may perform several fetches. Since PSFQ uses a large transmission interval and relies on local recovery of fragments, it might take a long time for the sensors to receive all the code fragments. The study in [4] considers a chain topology consisting of 13 nodes with only one source node.

Reliable Multi-Segment Transport (RMST) [5] is designed for guaranteed delivery of large blocks of data from sensors to sinks. In contrast to PSFQ, it uses unicast transmission for data frames and exploits MAC-layer retransmissions to achieve a higher hop-by-hop reliability. RMST is used in conjunction with directed diffusion and can operate in either cached or noncached mode. In cached mode, intermediate nodes cache incoming fragments and detect missing fragments. If there are missing fragments, an intermediate node sends a NACK packet indicating the missing fragments along the back-channel, which is just the reverse path from the sink to the sensor. Similar to PSFQ, loss recovery is also performed locally. RMST is designed to be used alongside Directed Diffusion and is not designed to provide time bounds. The study in [5] considers a scenario with 21 nodes arranged in a 3 x 7 grid with a single source and a single sink node placed at opposite locations in the grid.

Distributed TCP Caching (DTC) [6] enhances TCP in order to make it more efficient in WSNs. The provided guaranteed delivery service is similar to DTSN. DTC improves the transmission efficiency by compressing the headers and by using cache at selected intermediate nodes. DTC is fully compatible with TCP, leaving the endpoints of communication unchanged and it only requires changes in the logic of intermediate nodes. Although the paper only considers caching a single segment, the use of caching significantly improves the efficiency of end-to-end delivery, minimizing the energy spent with retransmissions. The caching mechanism may also be easily extended to use more than one segment per node. The authors also propose to use the TCP SACK option in order to optimize the use of the cache. In this proposal, the SACK block is used to carry information about segments in cache, allowing nodes farther from the destination to free their cache entries in case a node that is closer already has the segment in cache. In similar fashion to TCP, loss recovery in DTC is end-to-end. Only a single DTC flow is analyzed in [6].

Distributed Transport for Sensor Network (DTSN) [7] supports full end-to-end reliable delivery like TCP but employs selective repeat ARQ to improve energy efficiency. It uses both ACKs and NACKs sent from the receiver upon the request of the sender through an Explicit Acknowledgment Request (EAR). A NACK contains a bitmap of missing packets detected by the receiver. While relaying such NACKs, intermediate nodes will learn about the missing packets and check if those packets are present in their cache. If so, the intermediate nodes will retransmit those packets towards the receiver and modify the NACK bitmap accordingly before sending it towards the sender. Similar to DTC, loss recovery is end-to-end. The study in [8] only considers a scenario consisting of a single source and a single sink located several hops away in a linear chain topology.

A comparative summary of these transport protocols is provided in Table I. It shows that most transport protocols use NACKs for loss signaling except for DTC which uses ACK. Protocols that detect packet losses on a hop-by-hop manner like PSFQ and RMST populate the cache only when out-of-sequence fragments are detected and it is not clear how they can support multiple flows. In particular, PSFQ is designed specifically for dynamic code update. Furthermore, both these protocols were not designed to provide any time bounds. On the other hand, the protocols that implement end-to-end packet loss detection are designed to reduce the overall cost of transmitting packets from the source to the destination. In [6], DTC was implemented with a cache size of only one segment whereas previous works [7], [8] on DTSN used bigger cache sizes. However, they only considered a single DTSN flow. Whereas DTC was designed to make TCP compatible in wireless sensor networks, DTSN was designed from the very beginning to exploit intermediate caching albeit optional to improve the performance of reliable transport protocols in WSNs.

In summary, although previous works have proposed the use of transport-layer caching, they did not consider multiple concurrent flows using the cache. However, this is a very relevant problem because in reality, intermediate WSN nodes may forward packets for multiple concurrent flows. In the next section, we describe the optimal cache partitioning problem.

III. The optimal cache partitioning problem

For a given intermediate node n with cache size C, the node allocates a fraction of its cache memory to one or more flows that has that node on its path. Given that flow i passes through node n, we denote this by ω_i^n. We also refer to this fraction as the caching weight allocated to flow i at node n. A cache partitioning policy (also referred to simply as caching policy in the remainder of the paper) is the rule that determines the cache allocation. An optimal cache partitioning policy is one that leads to a desired optimal end-to-end network

TABLE I
SUMMARY OF RELIABLE TRANSPORT PROTOCOLS THAT EMPLOY CACHING

Protocol	Loss detection	Loss signalling	Can use MAC-layer retx	Cache size	Caching policy
PSFQ [4]	Hop-by-hop	NACK	No	Not specified	Addition: out-of-sequence packets, Removal: in-sequence transmission
RMST [5]	Hop-by-hop	NACK	Yes	Not specified	Addition: out-of-sequence packets, Removal: in-sequence transmission
DTC [6]	End-to-end	ACK, SACK option	Yes	One	Addition: segments with lower sequence numbers cached closer to the sink, Removal: interception of an ACK
DTSN [7] [8]	End-to-end	NACK with loss bitmap, EAR	Yes	Variable	Addition: supports cache partitioning, Removal: interception of an ACK

performance. For simplicity, we assumed that the packet error rate (PER) is uniform across the whole network.

In this paper, we considered four different caching policies. First, the Uniform caching policy, which divides the cache equally among the flows that pass through a node.

Second, we developed a Heuristic caching policy that counteracts the negative effect of link error rates that accumulate with the number of hops. Intuitively, allocating a bigger cache to a flow will increase the probability of cache hits for that flow and could decrease the overall transmission cost (in terms of total number of transmissions) because retransmissions from the source will be reduced. To this effect, we consider a policy that assigns a higher caching weight to longer flows such that

$$\Omega_i^n = (l_i)^2 \tag{1}$$

where l_i is the length of flow i. To obtain the final caching weights, we normalize as follows

$$\omega_i^n = \frac{\Omega_i^n}{\sum\limits_{j \in F^+(n)} \Omega_j^n}. \tag{2}$$

where $F^+(n)$ is the set of all flows that pass through node n. Lastly, for comparison purposes, we also included two extreme caching policies that assign the full cache only to flow 1 (denoted as 100/0) and only to flow 2 (denoted as 0/100).

IV. SIMULATION RESULTS

We implemented DTSN on ns-2 [9] and considered two network topologies: a simple Pi network and a more complex grid network, on which to analyze the effect of different cache partitioning policies on the transport layer performance in terms of overall transmission cost, throughput, and fairness.

Transmission cost is defined as total number of hop-by-hop transmissions (including retransmissions) required to deliver all the packets from the source to the destination. For example, the total transmission cost of a single packet sent over 10 hops to the destination is 10, assuming there are no retransmissions. However, if the packet is successfully transmitted after two retransmissions at each hop, the total transmission cost will be 30.

We analyzed the effect of the transmission cost on the aggregate throughput. To measure fairness, we use Jain's fairness index [10],

$$\text{fairness} = \frac{(\sum x_i)^2}{n \cdot (\sum x_i^2)}$$

where n is the number of flows and x_i is the throughput of the ith flow. In addition to that, we also observed the effect of the caching policy on the cache hits at the intermediate nodes.

All flows in the simulations are DTSN flows with an ACK window of 20 and all caching nodes have a cache size of 20. Each flow transmits 1000 packets of 500 bytes each towards the destination. In order to focus on the transport layer performance, we used statically assigned fixed routes instead of using dynamic routing. For each scenario, we performed a maximum of 20 simulations runs. For simplicity, we assume a constant packet error rate (PER) on all links in the network. The underlying MAC layer is CSMA/CA-based and the ARQ mechanism performs a maximum of 3 retries. In choosing the PER values, we consider a worst-case scenario where a successful transmission is made at the last ARQ retry. Thus, the raw PERs are actually much higher.

A. Simple network

The Pi network consists of 16 nodes arranged in a Π symbol configuration with two network flows as indicated in Fig. 1. Flow 1 is 5 hops long while Flow 2 is 16 hops long. These two flows share a common set of intermediate caching nodes, namely 2, 3, 4, and 5. In order to focus on the performance of caching partitioning policies, only those nodes are enabled to cache packets. Table II shows the normalized caching weights for the different caching policies.

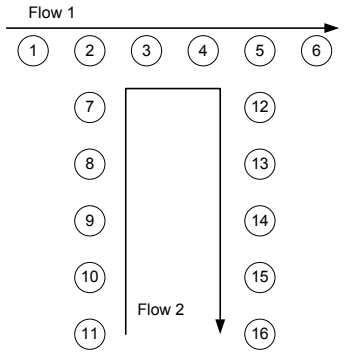


Fig. 1. Pi network topology

Transmission cost: Fig. 2 shows that the 0/100 caching policy gives the lowest total transmission cost among all the caching policies since it only caches flow 2 and avoids costly

TABLE II
PI NETWORK: NORMALIZED CACHING WEIGHTS (ω_i^n) AT NODES 2, 3, 4 , AND 5

Caching policy	Flow 1	Flow 2
Uniform	0.5	0.5
Heuristic	0.129	0.871
0/100	0	1.0
100/0	1.0	0

retransmissions from nodes closer to the source (node 11). The Heuristic policy provides a very good approximation to the performance of the 0/100 policy and has a gain of 15% over the Uniform policy at PER=0.10, increasing to 34% at PER=0.3. The 100/0 policy gives the poorest performance because it prefers flow 1 to the detriment of flow 2, incurring the worst overall transmission cost.

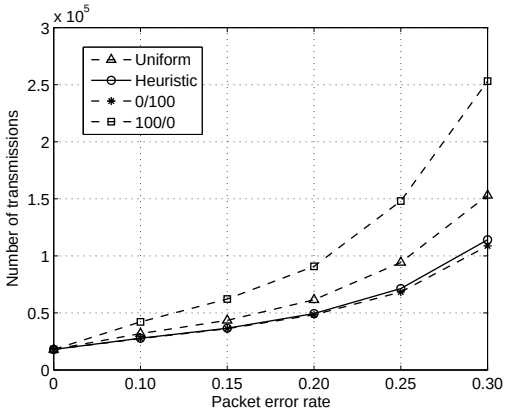


Fig. 2. Transmission cost in the Pi network as a function of PER

Throughput: Table III shows the aggregate throughput as a function of PER. It shows the the Heuristic policy provides highest throughput compared with the other policies up to PER=0.15. This is because at lower PERs, packets take less time (i.e., fewer transmission attempts) to reach the caching nodes and since the Heuristic policy assigns a higher weight to flow 2, packets from flow 2 benefit from the retransmission of cached packets resulting in higher overall throughput.

However, at higher PERs, the overall throughput of the Heuristic policy is lower than the Uniform and 100/0 policies. This is because the packets take more time to reach the caching nodes. Thus, even though flow 2 has a higher caching weight than flow 1 in the Heuristic policy, it is not able to benefit as much from intermediate cache retransmissions as in the lower PERs. As a result, this leads to a lower overall throughput performance. Fig. 5 (Heuristic policy vs. Uniform policy) shows how flow 2's throughput improvement is reduced in higher PERs.

Among all the policies, the 100/0 policy provides the highest overall throughput while the 0/100 gives the lowest overall throughput. Take note that the throughput of the 0/100 policy is always lower than that of the Heuristic policy since 0/100 only caches flow 2 packets while the Heuristic policy also caches flow 1 packets.

We noted that although the per-flow transmission cost has an inverse relationship with per-flow throughput (not shown in this paper), this relationship does not necessarily apply to the overall transmission cost and aggregate throughput as shown by the results. This is because of the presence of nodes that do not cache packets, which has a significant impact on the overall network performance.

TABLE III
PI NETWORK: AGGREGATE GOODPUT IN BPS, AS A FUNCTION OF PER

0	0.1	0.15	0.2	0.25	0.3
Uniform policy					
12471.94	7924.46	6132.49	4632.19	3527.92	2818.12
Heuristic policy					
12471.94	8133.72	6148.52	4398.66	3135.82	2391.98
0/100 policy					
12471.94	8096.86	6130.53	4330.08	3073.84	2295.99
100/0 policy					
12471.94	7525.11	5950.42	4899.97	4031.12	3506.02

Cache hits: In order to study the dynamic effects of the caching policy, we look at the average cache hits at the intermediate caching nodes in the network. Comparing the Uniform policy with the Heuristic policy at PER=0.2, the average cache hits for flow 1 have decreased, while those for flow 2 have increased as shown in Fig. 3(a) and Fig. 3(b), respectively. A higher caching weight allocates more cache space, and vice-versa. Consequently, a larger cache allocation leads to higher potential cache hits, and vice-versa.

Fairness: Fig. 4 shows that the Heuristic caching policy achieves better fairness compared with the Uniform policy.

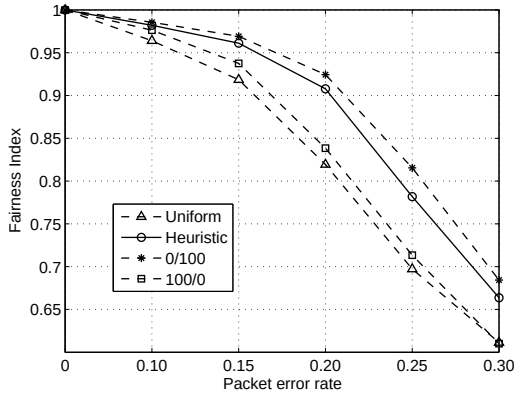
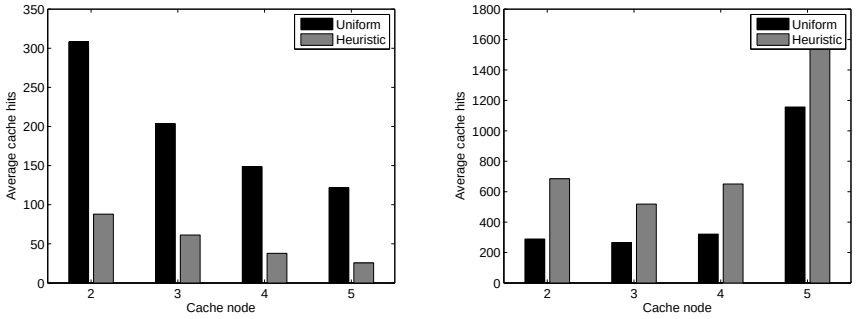


Fig. 4. Fairness in the Pi network as a function of PER

Fairness can be verified by looking at the effect of the caching policy on the individual throughput of flows 1 and 2 shown in Fig. 5. Maximum fairness of 1 is achieved when both flows have equal throughput, which is the case when PER=0 (i.e., no packet loss). For higher PERs, the throughput difference between the two flows for the Heuristic policy is significantly less than the Uniform policy.



(a) Flow 1 (b) Flow 2

Fig. 3. Average cache hits at nodes 2, 3, 4, and 5 in the Pi network at PER=0.2

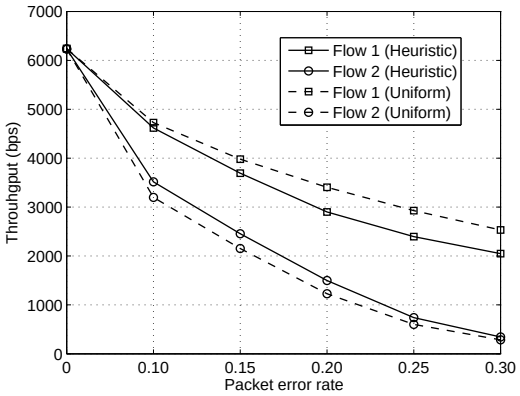


Fig. 5. Goodput in the Pi network as function of PER

In summary, for lower PERs (i.e., <0.15), the Heuristic caching policy provides the optimal performance in terms of overall throughput while the Uniform policy surpasses it at higher PERs. However, overall the Uniform policy has the lowest fairness while 0/100 policy has the highest fairness and the lowest throughput. Considering this throughput-fairness tradeoff, the Heuristic policy achieves a good balance between the both policies.

B. Grid network

Next, we considered a network topology with 60 nodes arranged in 10x6 grid. The network traffic consists of eight horizontal 5-hop flows and four vertical 9-hop flows as shown in Fig. 6. In this case, all the intermediate nodes cache one horizontal and one vertical flow that cross them. Table IV summarizes the normalized caching weights at the intermediate nodes.

Transmission cost: In the Grid network, the Heuristic policy incurs the lowest transmission cost among all the policies with the 0/100 policy closely matching its performance. Compared with the Uniform policy, it has a performance gain of 2% at PER=0.1, increasing to 14% at PER=0.3. Similar to the Pi network, the 100/0 policy gives the worst performance. However, as we will see in the throughput and fairness performance results, the Heuristic policy does not provide the highest overall throughput nor fairness.

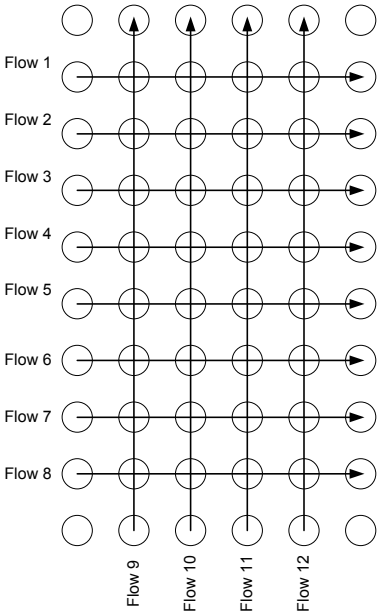


Fig. 6. Grid network topology

TABLE IV
GRID NETWORK: NORMALIZED CACHING WEIGHTS AT ALL CACHING NODES

Caching policy	Horizontal flow	Vertical flow
Uniform	0.5	0.5
Heuristic	0.236	0.764
0/100	0	1.0
100/0	1.0	0

Throughput: Table V shows that the Uniform policy gives higher throughput compared with the Heuristic policy, although it does not exceed it by more than 5%. The 0/100 policy gives the lowest overall throughput while the the 100/0 gives the highest overall throughput. However, there is a tradeoff between throughput and fairness as shown in the fairness performance.

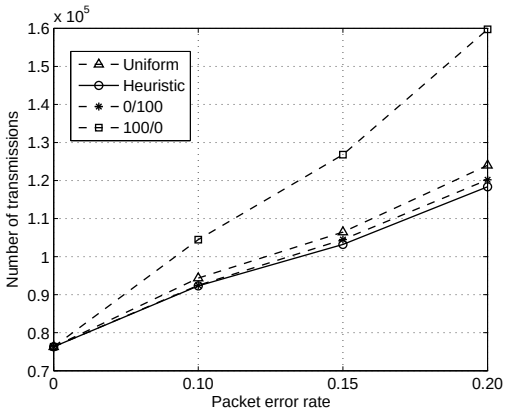


Fig. 7. Transmission cost in the grid network as a function of PER

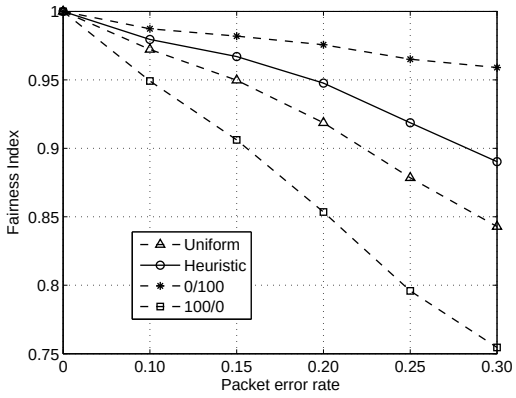


Fig. 9. Fairness in the grid network as a function of PER

TABLE V
GRID NETWORK: AGGREGATE GOODPUT IN BPS, AS A FUNCTION OF PER

0	0.1	0.15	0.2	0.25	0.3
Uniform policy					
69549.75	35687.32	28137.17	21432.09	16048.26	12200.09
Heuristic policy					
69549.75	35562.57	27589.19	20793.84	15376.22	11680.04
0/100 policy					
69549.75	35422.91	27083.02	20327.38	14913.03	11227.75
100/0 policy					
69549.75	35954.89	28612.49	22614.69	18517.77	15507.27

Cache hits: Analyzing the effect of the caching policy on the short flows, the average cache hits for the short flows in the Uniform policy have decreased under the Heuristic policy. We show this through the representative short flows 1 and 2, in Figs. 8(a) and 8(b), respectively. For the long flows, the cache hits increased under the Heuristic policy for all nodes except the nodes one hop away from the source as shown by representative flows 9 and 10 in Figs. 8(c) and 8(d), respectively.

Fairness: Fig. 9 shows that the Heuristic caching policy provides better fairness compared with the Uniform policy with the performance gain of the former increasing with increasing PER. The 0/100 has the highest fairness level while the 100/0 has the worst fairness. Considering the tradeoff between throughput and fairness, the Heuristic policy provides the optimal performance between both extremes.

Fig. 10 confirms that the Heuristic policy reduces the throughput difference between the long and the short flows. This is consistent with the effect observed in the Pi network.

In summary, while the 100/0 policy provides the highest overall throughput it also has the worst fairness. On the other hand, the 0/100 policy has the lowest throughput but obtains the highest performance. The Heuristic and Uniform policies have throughput and fairness performance in between those two extreme policies. Compared with Uniform policy, the Heuristic achieves better fairness while maintaining the overall throughput (i.e, within 5% of the Uniform policy).

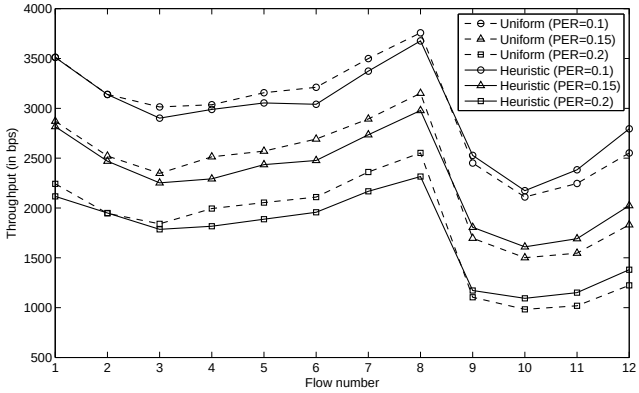
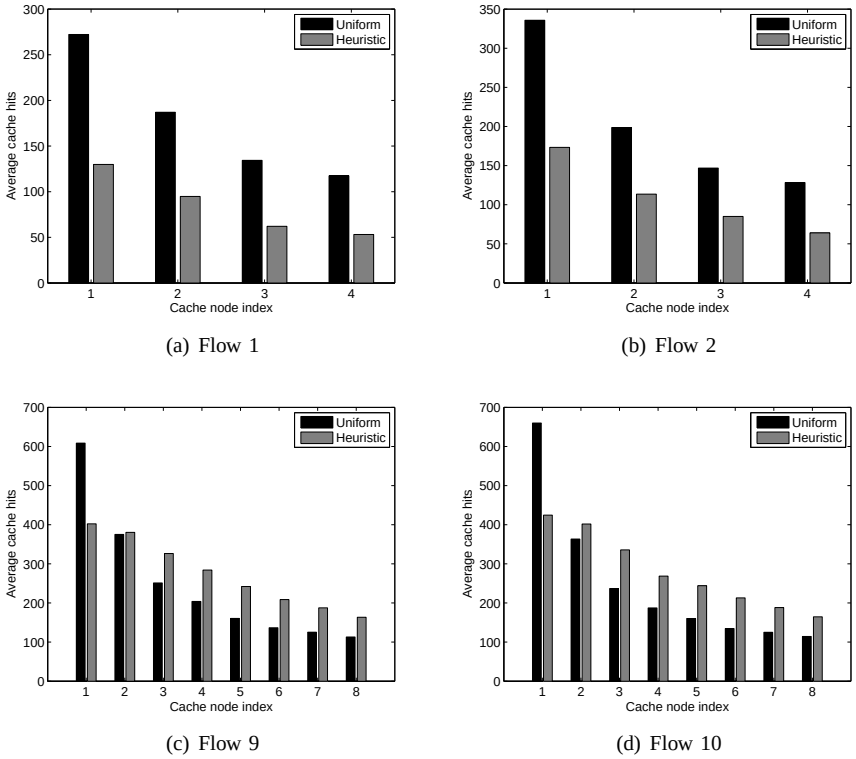


Fig. 10. Goodput in the Grid network for the different flows. Flows 1 to 8 are the short flows while flows 9 to 10 are the long flows.

V. CONCLUSION AND FUTURE WORK

We have studied the impact of cache partitioning at intermediate nodes on the performance of WSN transport protocols. Different cache partitioning policies were implemented and integrated with the DTSN transport protocol. Simulation results show that throughput and fairness can only be achieved if the cache partitioning policy takes into account the different lengths and channel conditions experienced by concurrent flows.

The network topologies we considered in this work are limited to two concurrent flows and the packet error rate is assumed uniform over the whole network. We intend to pursue this work further considering more complex and realistic network scenarios. We would like to explore the use of optimization techniques to develop more advanced cache partitioning policies (e.g., based on Game Theory) and implement distributed algorithms to better adapt to dynamic network conditions. Furthermore, we envisage that transport-layer cache partitioning can be applied to wireless mesh networks and vehicular networks.



(a) Flow 1 (b) Flow 2

(c) Flow 9 (d) Flow 10

Fig. 8. Average cache hits at caching nodes in the grid network at PER=0.2. The cache node index (x-axis) value denotes the position of the caching node relative to the source (i.e. 1 is the node one hop away from source and so on). The highest cache node index denotes the caching node one hop away from the destination.

ACKNOWLEDGMENT

This work has been partially supported by the European Community Seventh Framework Programme under grant agreement no. 225186, project WSAN4CIP. Nestor Tiglao would like to acknowledge the support of the Philippine Department of Science and Technology - Science Education Institute (DOST-SEI) and the University of the Philippines Engineering Research and Development for Technology (UP ERDT).

REFERENCES

[1] IEEE Standard for Information Technology Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4-2003.

[2] I. Akyildiz, T. Melodia, and K. Chowdhury, ”A survey on wireless multimedia sensor networks,” Computer Networks, vol. 51, no. 4, pp. 921-960, 2007.

[3] C. Wang, K. Sohraby, B. Li, M. Daneshmand, and Y. Hu, ”A Survey of Transport Protocols for Wireless Sensor Networks”, IEEE Network, IEEE, May/June 2006.

[4] C. Y. Wan, A. Campbell, and L. Krishnamurthy, ”PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks”, Proceedings of WSNA 2002, Atlanta, Georgia, USA, 2002.

[5] F. Stann, and J. Heidemann, ”RMST: Reliable Data Transport in Sensor Networks”, Proceeding of the 1st IEEE International Workshop on Sensor Net Protocols and Applications, IEEE, Anchorage, Alaska, USA, May 2003, pp. 1-11.

[6] A. Dunkels, J. Alonso, T. Voigt, and H. Ritter, ”Distributed TCP Caching for Wireless Sensor Networks”, Proceedings of the 3rd Annual Mediterranean Ad-Hoc Networks Workshop, Bodrum, Turkey, June 2004.

[7] Marchi, B.; Grilo, A.; Nunes, M.; , ”DTSN: Distributed Transport for Sensor Networks,” Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on , vol., no., pp.165-172, 1-4 July 2007.

[8] F. Rocha, A. Grilo, P.R. Pereira, M.S. Nunes, and A. Casaca, Performance Evaluation of DTSN in Wireless Sensor Networks, Wireless Systems and Mobility in Next Generation Internet: 4th International Workshop of the EuroNGI/EuroFGI Network of Excellence Barcelona, Spain, January 16-18, 2008 Revised Selected Papers, Springer-Verlag, 2008, pp. 1-9.

[9] ”The Network Simulator - ns-2,” http://www.isi.edu/nsnam/ns/.

[10] R. Jain, ”The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,” Wiley-Interscience, New York, NY, April 1991, ISBN:0471503361.