

eXtend Collection Tree Protocol

Bruno P. Santos, Marcos A. M. Vieira, and Luiz F. M. Vieira

Universidade Federal de Minas Gerais, Brazil

Email: {bruno.ps, mmvieira, lfvieira}@dcc.ufmg.br

Abstract—In this work, we propose eXtend Collection Tree Protocol (XCTP), a routing protocol that is an extension of the Collection Tree Protocol (CTP). CTP is the de-facto standard collection routing protocol for Wireless Sensor Network (WSN). CTP creates a routing tree to transfer data from one or more sensors to a root (sink) node. But, CTP does not create the reverse path between the root node and sensors. This reverse path is important, for example, for feedback commands or acknowledgment packets. XCTP enables communication in both ways: sink to node and node to sink. XCTP accomplishes this task by exploring the CTP control plane packets. XCTP requires low storage states and very low additional overhead in packets. With the reverse path, it is possible to implement reliable transport layer protocols for Wireless Sensor Network (WSN). Thus, we designed Transport Automatic Piggyback Protocol (TAP2), a transport protocol with Automatic Repeat-reQuest (ARQ) error-control on top of XCTP. We implemented these protocols on TinyOS and evaluated on TOSSIM. We compared XCTP with CTP and Ad hoc On Demand Distance Vector (AODV) protocols. We conducted scalability and stress tests, evaluating them with different loads and number of nodes. Our results shows that XCTP is more reliable than CTP, delivering 100% of the packets. XCTP is faster to recovery from network failures and also stores fewer states than AODV, thus being efficient and agile.

Index Terms—Sensor Network, Routing, Wireless Communication.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) are composed of a large number of nodes with sensing, computation, and wireless communication capability. These networks have computing and communication energy constraints. Many applications in WSN need to transport large amount of data (image, audio, video monitoring). These applications are not tolerant to data loss, thus it is important to provide mechanisms to reliable collect data.

The WSNs have the following communication paradigms: *many-to-one* (data collection), *one-to-many* (data dissemination), and a more complex way that enables communication *any-to-any*. First two paradigms allow the collection and dissemination of data respectively. However, with routing on only one direction, it is infeasible to build reliable mechanisms to ensure the delivery of data end-to-end. *Any-to-any* communication paradigm allows communication between any pair of nodes in the network, but adds more complexity and also requires large amounts of memory to store all possible routes

In this work, we present eXtend Collection Tree Protocol (XCTP), a routing protocol that is an extension of the Collection Tree Protocol (CTP). CTP creates a routing tree to

transfer data from one or more sensors to a root (sink) node. But, CTP does not create the reverse path between the root node and sensors. This reverse path is important, for example, for feedback commands or acknowledgment packets. XCTP enables communication in both ways: sink to node and node to sink. XCTP requires low storage of states and very low additional overhead in packets.

Our main contribution are as follows:

- We propose eXtend Collection Tree Protocol (XCTP), which allows routing of messages in the reverse direction of CTP, using a few extra memory to store reverse routes.
- We compare the performance of XCTP, Ad hoc On Demand Distance Vector (AODV), and CTP. In the experiments, XCTP proved to be more reliable, efficient, agile, and robust.
- We show that it is possible to implement reliable data transport protocol over XCTP.

CTP optimizes data traffic towards the root thus achieves high packet delivery rate. However, our XCTP approach goes beyond, allowing bi-directional communication between sensor nodes and the root. XCTP and *any-to-any* routing protocols enable reliable communication. However, XCTP reduces the cost to store routes, since XCTP does not need to maintain routes to every peer.

Our work is organized as follows. In the next section, we present work related to XCTP. In Section III, we formally define the problem being solved in this work. We describe XCTP architecture in Section IV. We compare XCTP with AODV, CTP and present the simulation results in Section V: evaluation. Finally, we conclude in Section VI.

II. RELATED WORK

Table I
COMPARISON OF COMMUNICATION PARADIGMS.

Dissemination	Collection		Any-to-Any
Unreliable	Unreliable	Reliable	Reliable
Directed Diffusion [1]	CTP [2]	XCTP	AODV [3], DYMO [4], DSR [5]
(DIP, DRIP, DHV) [6]	MultiHopLQI [7]		Hydro [8]
Deluge [9]	MintRoute [10]		RPL [11]

We present in Table I the main related protocols. We classified them according to the communication paradigm (*any-to-any*, *many-to-one*, *one-to-many*). Table I shows that XCTP is, to the best of our knowledge, the only Reliable Collection protocol. In other words, it is a data collection protocol that also allows unicast routes root-to-node. Besides

that, it offers an interface that facilitates the development of reliable end-to-end transport protocols.

From the protocols presented in Table I, Directed Diffusion [1], (DIP, DRIP, DHV) [6] are used for dissemination of small data packets in the network. DIP, DRIP, DHV offers eventual consistency models and use timers based on Trickle [12]. DRIP treats each information as a separated entity, which allows more control of when and how fast the data will be disseminated. DIP and DHV treat data as a group, meaning that control and dissemination parameters are applied equally for all data.

CTP and Deluge are related protocols. CTP is a data collection protocol that uses Expected Transmissions (ETX) metric to estimate the link quality and route cost. Data and control packets are used to obtain the link quality. XCTP is an extension of the CTP protocol. Besides creating unicast routes to a data collection point, XCTP also creates unicast routes from the root to the sensors. Deluge is a protocol that operates under the *one-to-many* paradigm, which has the objective to propagate large amount of data, as is the case, when reprogramming the network nodes.

Hydro [8] and RPL [11] are protocols that aim at maintaining *any-to-any* communication in WSN. Hydro differs from our approach, which focus in creating unicast routes to exchange messages in both directions root-to-node and vice-versa. Routing Protocol for low-power and lossy networks (RPL) disseminates Destination Advertisement Object (DAO) messages through the network to announce routes for various destines inside a RPL network. XCTP, different from RPL, does not need extra control messages, and avoids extra costs to keep state with *peer-to-peer* routes that are under-utilized in WSN.

AODV [3] and Dynamic Source Routing (DSR) [5] are on-demand routing protocols for *any-to-any* communication. AODV floods the network with messages RREQ to build a path till the destination. On the other hand, DSR protocol uses the packet header to store the route path. Unlike DSR, our protocol does not store any routing information in the packet header. AODV protocol has some similarity with XCTP in the strategy of storing the reverse path. However XCTP, unlike AODV, does not save routes that are not reverse among the sensor nodes and the base station. Dymo [4] is the AODV successor, however it is optimized for MANETs.

None of the protocols here related allow sending unicast messages in *root-to-node* direction and vice versa, except those *any-to-any* protocols that require large amount of information to be stored or control messages.

III. PROBLEM

The data delivery reliability is one of the most challenging problems in WSN, due to frequent link instability in fractions of a seconds, many times in less than 1s [13]. Therefore, it brings basic requirements that are fundamental for any routing protocol for low-power and lossy networks (L2Ns): 1) **reliability**, it should deliver the largest possible amount of packets, when there is route between the participants

communication; 2) **robustness**, the protocol should operate in different topologies, loads, amount of sensor nodes and in the presence of failures; 3) **efficiency**, the protocol must deliver packets with the least amount of transmissions, save energy and keep the least amount of possible states.

An alternative to collecting data in a reliable, robust and efficiently mode is using XCTP. This protocol balances the compromises imposed by the three fundamental goals of WSN. XCTP adjusts the communication model for data collection to provide routes that enable feedback commands, confirmation messages or control messages to be exchanged in bi-directional form between any sensor node and the base station. XCTP allows data transport protocols with confirmation be built. Thus, XCTP enables reliable data deliver.

IV. SOLUTION

A. XCTP Architecture

Here, we describe XCTP architecture. To accomplish the task of forwarding packets also in the reverse direction of the standard CTP data flow, we had to modify CTP architecture by adding new features to the protocol rules as well as incrementing the packet format. We did a minor modification in the data packet by adding a new field. We also created a reverse flow table. The protocol rules were modified at the control and data planes. The data plane was changed to query the reverse flow table. The control plane is responsible for the construction and modification of this table. The control plane was modified to manipulate the reverse fluxes and also to react appropriately to the two main events:

- 1) **Reverse flow:** correct and efficient installation of the reverse flow rules;
- 2) **Topological changes:** nodes must appropriately react when loops occur or when CTP unicast routes change.

In Figure 1, we show the relationships between modules. Major changes are highlighted in gray. The Router module is responsible for filling the Forward and Reverse tables. These tables indicate what is the next hop for the data packet to be transmitted. We did not modify the Link Estimator module. This module estimates the quality of the links to the neighboring nodes. The quality of the links are estimated using beacons and data packets. The Forward module queries the Forward and Reverse tables, and determines any router inconsistencies to inform the Router module. It also keeps a packet queue for transmission and check for duplicate packets. The Link Layer module contains the features used in radio communication. Finally, the Upper Layer module is the interface provided to implement components that utilizes XCTP.

B. Changes in data packet

To allow the navigation of the reverse data packet, we added a 16 bits packet field to the data packet to represent the address of the message destination. Figure 2(a) shows the new data packet format. The packet fields are: *P* allows node to request routing information to other nodes; *C* indicates congestion notification; *Time Has Live (THL)* each node, when receiving a packet, increments this field; *ETX* routing metric for routes

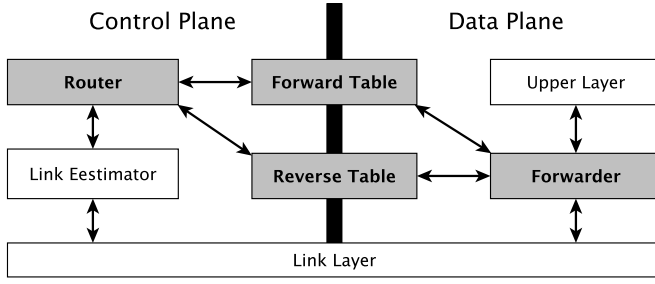


Figure 1. Arquitetura do protocolo XCTP.

construction and loop detection; *origin* address of the source node; *destination* address of the destination node; *seq. num.* sequence number; *collect ID* collection tree identifier; *Payload* packet data content.

We also created an Acknowledgment (ACK) packet. The ACK packet has a subset of the data packet fields, as illustrated in Figure 2(b). The *New Features* field 16 bits is reserved for future features. The ACK packet is useful as acknowledgment message for end-to-end transport protocols implemented over XCTP.

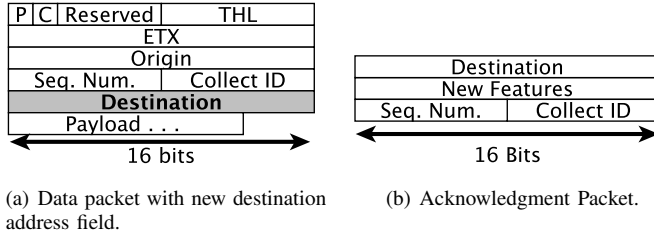


Figure 2. Packet formats for XCTP protocol.

C. Reverse Flow

The control plane is responsible for the manipulation of the XCTP reverse table. The reverse table has the following fields: *addr dest*: XCTP tree descendent (but not 1-hop neighbor); *next hop*: neighbor address to reach destination; *TTL*: route time to live, where we can apply removing policies. The Router module implements the basic operations Creation, Read, Update, and Delete over the reverse table.

1) *Creation*: The table starts empty. When a sensor node forwards a message to the root, the reverse route is installed. Since the link estimator module stores information about the 1-hop node neighbors, the router module does not insert entries in the reverse table of 1-hop neighbors. Figure 3(a) illustrates this situation: where node C sends data to the root, the intermediate node (that is not a 1-hop neighbor of node C) intercepts the packet from source C and install a reverse flow.

2) *Read*: Router module provides an interface to query the table. This mechanism is used for data and control planes. In Section IV-E, we provide details of using this interface.

3) *Update e Delete*: Router module provides mechanisms for updating and removing installed rules. These functions are called when there is a topology change (see Section IV-D).

There is a trade-off between agility and efficiency regarding the maintenance of routes in L2Ns. Agility refers to how fast the network can react to a topological change, while efficiency is the energy consumption and the number of packets sent to keep the network operational. The network requires high frequency of the beacons to keep routes updated. This increases the agility of the network but, on the the hand, it reduces efficiency. CTP uses the Trickle algorithm [12] to increase the number of beacons when the network is unstable and exponentially reduces the number of beacons when the network is stable, thus keeping a tradeoff balance between speed and efficiency. XCTP uses the data packets to create the reverse route, thus, there is no need for extra beacons.

D. Topology Changes

A routing system must know when and where to change the reverse routes of the data plane so it can correctly react to the network topology dynamics. XCTP control plane reacts and changes the data plane for reverse routes when there is the occurrence of loops or link failures.

To maintain the consistency of routes, each sensor node keeps the estimated route cost to the base station. Moreover, this information is attached to the control and data packets (see Figure 2(a)). XCTP uses ETX as the metric cost. The route cost is always increasing towards the leaf nodes of the routing tree and this invariant must always be maintained. Loops are detected when this invariant is broken. In this case, the reverse flow table entry is removed.

Figures 3(a) and 3(b) illustrates this situation. In Figure 3(a), we show the initial flow table. Then, as shown in Figure 3(b), there is a link failure which causes a loop between nodes A, B, and D. In the event of a loop, the data plane marks, in the reverse flow table, the sensor nodes that were descendants and now are parents in the routing tree. Therefore, the action taken when loops are detected by XCTP data plane is to signal the control plane for the loop detection so that the appropriate reverse flow table entries are cleared. The reverse flow entries are reconstructed when there are new data packets in the network.

In case of link exchanges due to the dynamics of link quality, the control plane must update the data plane reverse flow entries to reflect this new routing tree configuration. The reverse flow table is updated when a data packet from an already installed flow is intercepted but it was routed through a different neighbor. Figures 3(c) and 3(d) illustrates this case. Data packets from node D towards the root was forwarded by node B and changed to be forwarded by node C due to changes in link quality. Thus, the data plane of node A must be updated to reflect this new configuration: the reverse flow should be forwarded by node C.

E. API

Here, we describe the XCTP Application Programming Interface (API). The CTP protocol does not require a destination

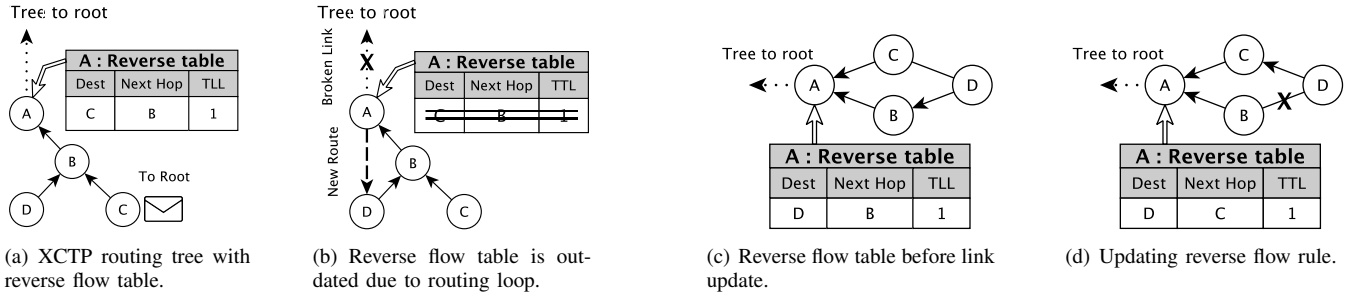


Figure 3. Control plane reactions over the data plane rules when detecting a routing loop and updating the routing paths.

address. XCTP, on the other hand, needs a destination address to provide unicast routing to a specific sensor node. XCTP integrates an interface that includes the destination address as well as routines for handling and the reverse and forward tables. The routines are:

- **addr sendTo(target, pkt):** where *target* is the destination address of XCTP *pkt* packet.
- **addr nextHop(target):** where *target* is an optional parameter. If *target* is instantiated, nextHop(target) routine queries the Reverse Table, otherwise the message is towards the base station.
- **loopDetect():** this routine signals the control plane when a loop is detected (see Section IV-D).
- **snoopNewPkt(pkt):** when intercepting a data packet from a new flow, the control plane must signal to update the reverse and forward tables.

Thus, the interface *sendTo(idNode, pkt)* should be used when the base station needs to send a packet to a specific node.

Algorithm 1 describes this routine. On line 1, we check if it is a data or acknowledgment packet because only these two packet types should travel on the reverse path. Then, the destination address is extracted. If the destination is the node itself (line 2), the packet has reached its destination and it should be properly processed. If the destination is one of its descendants (line 4), the packet is forwarded. Otherwise the recipient is not in any of the routing tables. In this case, there are two approaches: discard the packet or forward the packet to the root (line 7). In the second case, since the root knows the entire network topology, the root can forward the packet or just discard it. If the packet does not have a valid address, XCTP routes the messages directly to the base station (lines 10-12).

XCTP permits the *any-to-any* communication paradigm. This is possible due to how the reverse route is constructed (line 7 of Algorithm 1). If a node X wants to directly connected to a node Y, node X can use the routine *sendTo(Y, pkt)*. Node Y will receive the message from an ancestral of node X or, in the worst case, the message will go to the root and towards node Y.

F. Transport layer over XCTP

Using XCTP API, we implemented a reliable transport protocol, called Transport Automatic Piggyback Protocol (TAP2).

Algorithm 1 Internal operation sendTo() interface.

```

1: if isDataXCTP(pkt) or isAckXCTP(pkt) then
2:   if pkt.destination = my.addr then
3:     // Process package locally.
4:   else if pkt.nextHop = nextHop(pkt.dest) then
5:     // Send unicast message to neighbor in reverse flow.
6:   else
7:     // Drop pkt or forwards to the root.
8:   end if
9: else
10:  // Normally forwards packets through tree XCTP.
11:  pkt.nextHop = nexHop()
12:  forward(pkt)
13: end if

```

TAP2 uses piggyback and Automatic Repeat-reQuest (ARQ) error-control mechanism for packet retransmission. Other transport protocols for WSN such as [14]–[16] can also be implemented over XCTP. However, the requirement of a few computing resources and its simplistic implementation were reasons why we chose this approach in our work.

V. EVALUATION

In this section we analyze XCTP and compare it with two protocols: CTP and AODV. The objective of this analysis is to show that the protocol is working properly, as well as to evaluate XCTP performance when compared with the current state-of-the-art protocols. We analyzed according to the following items: 1) favoring the construction of data transport protocols; 2) robustness in the presence of faults in different topologies; 3) scalability.

A. Simulation

Of the protocols shown in Table I, Directed Diffusion [1], Deluge [9], (DIP, DRIP, DHV) [6] are used just for data dissemination in the network and do not serve to compared with XCTP. Collection Tree Protocol (CTP) [2] is one of the newest protocols and it presents better results than MultiHopLQI [7] and MintRoute [10] in data collection. To the best of our knowledge, there are no stable and open source implementations to the community of protocols Dymo [4], DSR [5], RPL [11] and Hydro [8]. Therefore, we made comparisons with AODV [3] and CTP.

Protocols XCTP, CTP and AODV were implemented in the TinyOS [6] platform. We also performed experiments with Tymo, a TinyOS version of protocol Dymo. However, Tymo implementation did not show to be stable as reported in [17].

We run the experiments on the simulator for L2Ns TOSSIM [18]. We consider the base station to be a PC without memory restrictions and which can hold information about the entire network topology. To generate the topology and the connectivity model of the sensor nodes, we used the LinkLayerModel tool provided by TinyOS, with parameters appropriate to have nodes randomly deployed in an open field region and low asymmetric links. The simulations use as scenario the default parameters presented in Table II, when we vary some of these parameters we describe the changes.

Table II
SIMULATION PARAMETERS

Parameter	Value
Root	1 center
Number of sensors	100
Radio range	100m
Density	10 sensors/m ²
Number of flows	1
Bytes transmitted	1024B
Path Loss Exponent	4.7
Power decay (dB)	55.4
Shadowing Standard Deviation (dB)	3.2

B. Simulation Results

Figure 4 shows the percentage of delivery of a file of size 512KB sent from the root to a sensor node 5 hops away. XCTP+TAP2 and AODV+TAP2 reach 100% of data delivery, because they allow feedbacks to be sent from data received between the two involved in the communication. We observed that CTP can transfer only 96.5% of the 512KB, because it is not possible to request lost data or confirm received messages, since there are routes only towards the root. The impossibility of requesting the remaining fragments results in malfunctioning the application that is intolerant to data loss. We conclude that XCTP and AODV favor the development of data transport protocols, providing routes that allow feedback messages to be exchanged among sensor nodes and the base station.

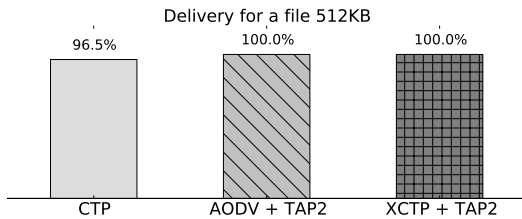
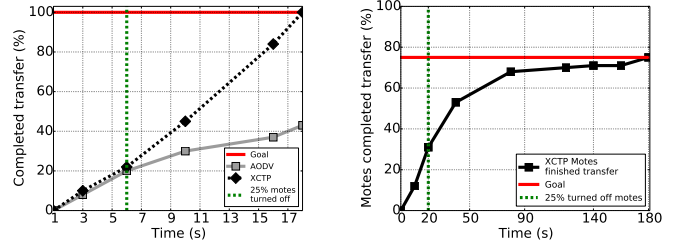


Figure 4. Transferring a 512KB file to a sensor node 5 hops away from the base station.

1) *Robustness*: To evaluate XCTP in terms of robustness, we elaborated experiments with different amounts of active

flows (sensor nodes transmitting data to the root) and inserted network failures.

Initially we compared XCTP and AODV with respect to the reaction in the presence of network failures. In this scenario, 5 sensor nodes transfer, each one, 1KB of data to root reliably and after 6s from the beginning of transmission we disable 25% of the network, without creating disconnected components in the network. Figure 5(a) shows in y-axis the percentage of flows that have completed the transfer of 1KB of data, the x-axis shows the elapsed time. In the first seconds of the simulation the two approaches are similar, being XCTP a little faster due to proactive construction of routes. After the shutdown of part of the network at 6s, XCTP reacts quickly finding new routes to the root, the 5 sensor nodes operating with the protocol XCTP complete the transfer in approximately 18s. The protocol AODV reacts slowly to topological change. In this simulation scenario, AODV took over 360s to complete the data transfer.



(a) Protocol reactions XCTP and AODV in the occurrence of network failures. (b) XCTP reaction in the presence of failures. 75 flows are active, each sending 1KB.

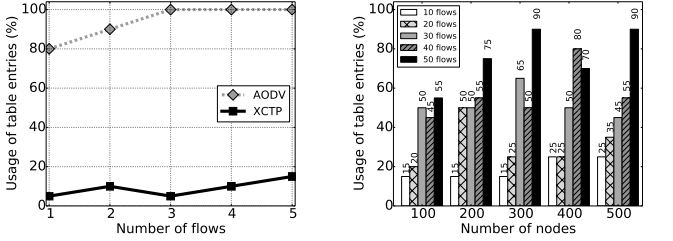
Figure 5. Experiments on the robustness of the protocols XCTP and AODV.

In the second experiment, we have a scenario with 100 sensor nodes, and 75 of them have active flow with the root, after transferring for 20s we shut down 25% of the network. Figure 5(b) shows in x-axis the time elapsed since the beginning of data transmission, and in the y-axis the percentage of sensor nodes that ended the data transfer to the root. XCTP, even after the shutdown of part of the network, quickly rebuild routes to the root and continues to transfer data. With 3 minutes of simulation, all nodes end the data transfer. This shows that XCTP is agile even in presence of faults and with many concurrent flows in the network. AODV is not shown in Figure 5(b) because it can not operate with higher number of 5 flows in the network. In the next section, we detail the scalability of simultaneous flows streams of AODV and XCTP.

2) *Scalability*: To show that XCTP is scalable, we performed analyzes on the size of XCTP Reverse Table and compared with the routing table of AODV. We did not compare with CTP because CTP table has constant size and stores only the next hop towards the root. Figure 6(a) shows the comparison between XCTP and AODV in the use of routing tables varying the number of flows. We notice that with 5 flows AODV consumes 100% of the routing table. When many flows coexist and the routing table is full, AODV is obliged to

dismiss requests for new routes. Thus, AODV needs to wait for timeouts from old routes to expire so that new routes are installed, resulting in high reaction time for network fault and it prevents higher number of concurrent flows. Unlike AODV, XCTP consumes approximately 83% less from the table than unless AODV for the same amount of flows.

Figura 6(b) shows that XCTP is robust and scalable. XCTP operates under the Reverse Table limit for different number of concurrent flows, with different topologies and amounts of sensor nodes in the network.



(a) Consumption of the routing table by varying the number of flows in the network for the protocols XCTP and AODV.

(b) XCTP Reverse table use by varying the number of flows and sensor nodes in the network.

Figure 6. Scalability experiments for XCTP and AODV.

3) *Memory Consumption:* Table III presents the RAM and ROM footprint sizes of the components in our protocol stack with and without TAP2. XCTP adds little more than 1KB of code to CTP, requiring smaller amounts of RAM than when compared with AODV. Regarding the protocols in conjunction with TAP2, XCTP consumes less RAM than AODV.

Table III
CODE AND MEMORY FOOTPRINT IN BYTES.

	CTP	XCTP	XCTP+TAP2	AODV	AODV+TAP2
RAM	1505	1812	1968	2119	2545
ROM	16204	17942	18435	13868	14562

VI. CONCLUSION

In this work, we present XCTP, a reliable, robust, scalable, and efficient protocol for WSN. XCTP solves the problem of reliable data collection, extending the de-facto standard collection routing protocol CTP. XCTP allows bi-directional exchange of messages between a node and the base station, extending the range of previously impossible applications with the CTP. For example, we show that XCTP favors the construction of transport protocols, unlike CTP. In our experiments, XCTP reduces the number of states stored when compared with AODV. This indicates that XCTP is an alternative for applications in L2Ns that are intolerant to data loss.

REFERENCES

[1] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *Networking, IEEE/ACM Transactions on*, 2003.

[2] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, November 2009.

[3] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*. IEEE, 1999.

[4] "Dynamic manet on-demand (aodvv2) routing draft-ietf-manet-dymo-26." <http://tools.ietf.org/html/draft-ietf-manet-dymo-26>, 2013.

[5] D. Johnson, Y. Hu, D. Maltz *et al.*, "The dynamic source routing protocol for mobile ad hoc networks," RFC 4728, Tech. Rep., 2007.

[6] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "Tinyos: An operating system for sensor networks," in *Ambient intelligence*. Springer, 2005.

[7] "The multihopliqi protocol." <http://www.tinyos.net/tinyos-2.x/tos/lib/net/lqi/>, 2014.

[8] S. Dawson-Haggerty, A. Tavakoli, and D. Culler, "Hydro: A hybrid routing protocol for low-power and lossy networks," in *Smart Grid Communications (SmartGridComm)*. IEEE, 2010.

[9] A. Chlipala, J. Hui, and G. Tolle, "Deluge: data dissemination for network reprogramming at scale," *University of California, Berkeley, Tech. Rep.*, 2004.

[10] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. ACM, 2003.

[11] T. Winter, "Rpl: Ipv6 routing protocol for low-power and lossy networks," 2012.

[12] P. A. Levis, N. Patel, D. Culler, and S. Shenker, *Trickle: A self regulating algorithm for code propagation and maintenance in wireless sensor networks*. Computer Science Division, University of California, 2003.

[13] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β -factor: measuring wireless link burstiness," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008.

[14] J. Paek and R. Govindan, "Rcrt: Rate-controlled reliable transport for wireless sensor networks," in *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*. ACM, 2007.

[15] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: A reliable bulk transport protocol for multihop wireless networks," in *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*. ACM, 2007.

[16] X.-S. Wang, Y.-Z. Zhan, and L. min Wang, "Stcp: Secure topology control protocol for wireless sensor networks based on hexagonal mesh," in *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, 2008.

[17] "Tymo: Dymo on tinyos." <http://tinyos.stanford.edu/tinyos-wiki/index.php/Tymo>, 2008.

[18] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003.