Congestion in Wireless Sensor Networks and Various Techniques for Mitigating Congestion - A Review

V. Vijayaraja, Department of Computer Science Engineering Jaya Engineering College, Chennai, India
vvijay1975@gmail.com

Dr. R. Rani Hemamalini, Department of Electronics and Instrumentation Engineering, St. Peter’s University, Chennai, India
ranihema@yahoo.com

Abstract - In wireless sensor networks (WSNs), congestion occurs, for example, when nodes are densely distributed, and/or the application produces high flow rate near the sink due to the convergent nature of upstream traffic. Congestion may cause packet loss, which in turn lowers throughput and wastes energy. Therefore congestion in WSNs needs to be controlled for high energy-efficiency, to prolong system lifetime, improve fairness, and improve quality of service (QoS) in terms of throughput (or link utilization) and packet loss ratio along with the packet delay. In this paper the characteristics of WSNs are reviewed and challenges of reliable data transport over WSNs are discussed. The proposed reliable congestion control protocols and data transport protocols for WSNs are reviewed and summarized. The recent research progress in developing suitable transport protocols for WSNs, are surveyed. And finally, some future research directions of transport protocol in WSNs are discussed.

Keywords - Wireless Sensor Networks, WSNs, Congestion control, reliable transport protocols.

I. INTRODUCTION

Wireless sensor networks (WSNs) have attracted tremendous attention in both academia and industry in recent years. A WSN consists of one or more sinks and perhaps tens or thousands of sensor nodes scattered in an area. The upstream traffic from sensor nodes to the sink is many-to-one multi-hop convergent. A WSN consists of one or more sinks and perhaps tens or thousands of sensor nodes scattered in an area. The upstream traffic can be classified into four delivery models: event-based, continuous, query-based, and hybrid delivery [1][6][7].

In event-based delivery, a sensor node does event reporting if and only if target event occur. The sensor data for the event usually has very small size. Sensor nodes need periodically report to the sink and generate continuous data transmission in some cases. This is continuous delivery, sensory data is stored inside the networks and is queried by and the transmitted to the sink on demand. Practical applications might trigger hybrid data delivery including event-based, continuous, and query-based.

Due to the convergent nature of upstream traffic, congestion more probably appears in the upstream direction. In addition, upstream traffic could have high bit rate with the introduction and the development of wireless multimedia sensor networks. Such high speed upstream traffic is prone to cause congestion which will impair QoS of multimedia applications in WMSNs [7].

Different congestion control techniques (protocols) have been proposed for wireless sensor networks which are reviewed in this paper.

The reminder of this paper is organized as follows. In section 2, we present a brief summary about the congestion in wireless sensor networks. In section 3, we explain the Congestion control mechanism in wireless sensor networks. Section 4 gives an overview of various congestion control protocols in wireless sensor networks. Section 5 gives an outlook on various transport protocols for wireless sensor networks. Section 6 discusses some future aspects of congestion control. Section 7 concludes the paper.

II. CONGESTION IN WIRELESS SENSOR NETWORKS

Congestion control is another important issue that should be considered in transport protocols. Congestion is an essential problem in wireless sensor networks. Congestion in WSNs and WMSNs that can leads to packet losses and increased transmission latency has a direct impact on energy efficiency and application QoS, and therefore must be efficiently controlled [7][10].

Congestion may lead to indiscriminate dropping of data (i.e., high-priority (HP) packets may be dropped while low-priority (LP) packets are delivered). It also results in an increase in energy consumption to route packets that will be dropped downstream as links become saturated. As nodes along optimal routes are depleted of energy, only non optimal routes remain, further compounding the problem. To ensure that data with higher priority is received in the presence of congestion due to LP packets, differentiated service must be provided.

Congestion not only wastes the scarce energy due to a large number of retransmissions and packet drops, but also hampers the event detection reliability [1].

Two types of congestion could occur in sensor networks. The first type is node-level congestion that is caused by buffer overflow in the node and can result in packet loss, and increased queuing delay. Packet loss in turn can lead to retransmission and therefore consumes additional energy. Not only can packet loss degrade reliability and application QoS, but it can also waste the limited node energy and degrade link utilization. In each
sensor node, when the packet arrival rate exceeds the packet-service rate, buffer overflow may occur. This is more likely to occur at sensor nodes close to the sink, as they usually carry more combined upstream traffic. The second type is link-level congestion that is related to the wireless channels which are shared by several nodes using protocols, such as CSMA/CD (carrier sense, multiple access with collision detection). In this case, collisions could occur when multiple active sensor nodes try to seize the channel at the same time [10]. Link level congestion increases packet service time, and decreases both link utilization and overall throughput and wastes energy at the sensor nodes. Both node level and link level congestions have direct impact on energy efficiency and QoS [1] [2].

III. CONGESTION CONTROL IN WIRELESS SENSOR NETWORKS

Congestion happens mainly in the sensors-to-sink direction when packets are transported in a many-to-one manner. Therefore, most of the proposed congestion control mechanisms are designed to lighten congestion in this direction [4].

Congestion control generally follows three steps: congestion detection, congestion notification, and rate adjusting [1] [3] [8] [10]. Congestion control protocol efficiency depends on how much it can achieve the following performance objectives: (i) First, energy-efficiency requires to be improved in order to extend system lifetime. Therefore congestion control protocols need to avoid or reduce packet loss due to buffer overflow, and remain lower control overhead that will consume additional energy more or less. (ii) Second, fairness needs to be observed so that each node can achieve fair throughput. Fairness can be achieved through rate adjustment and packet scheduling (otherwise referred to as queue management) at each sensor node. (iii) Furthermore, support of traditional quality of service (QoS) metrics such as packet loss ratio and packet delay along with throughput may also be necessary [1] [3] [8] [10].

Different congestion control techniques have been proposed for wireless sensor networks. The congestion control mechanisms all have the same basic objective: they all try to detect congestion, notify the other nodes of the congestion status, and reduce the congestion and/or its impact using rate adjustment algorithms [3] [8] [10]. There are several congestion control protocols for sensor networks. They differ in the way that they detect congestion, broadcast congestion related information, and the way that they adjust traffic rate [1].

IV. OVERVIEW OF VARIOUS CONGESTION CONTROL PROTOCOLS IN WIRELESS SENSOR NETWORKS

In this section, congestion control methods proposed for WSNs are reviewed. Typical WSNs work under light traffic load most of the time, but they can become congested when sudden events happen and bursts of traffic are injected from many sensor nodes. Congestion happens mainly in the sensors-to-sink direction when packets are transported in a many-to-one manner. Therefore, most of the proposed congestion control mechanisms are designed to lighten congestion in this direction [4] [8].

A. Event to sink reliable transport (STCP):

The ESRT protocol considers reliability at the application level and provides stochastically reliable delivery of packets from sensors to the sink. The congestion control mechanism in ESRT is designed for this purpose.

The motivation of ESRT is that in some applications the sink is only interested in reliable detection of event features from the collective information provided by numerous sensor nodes and not in their individual reports. If the event reporting frequency at the sensors is too low, the sink may not be able to collect enough information to detect the events reliably. On the other hand, if the reporting frequency is too high, it may endanger the event transport reliability by leading to congestion in the WSN. ESRT adjusts the reporting frequency such that the observed event reliability is higher than the desired value while avoiding congestion.

The event reliability is defined as the number of received data packets in a decision interval at the sink. The congestion detection in ESRT is through the local buffer level of the sensor nodes. The sensor nodes set the Congestion Notification (CN) bit in a packet’s header if congestion is detected. When the sink receives packets with CN bit marked, it knows that congestion has occurred in the WSN.

In ESRT, the WSN can stay in one of the 5 states: (No Congestion, Low Reliability) (NC, LR), (No Congestion, High Reliability) (NC, HR), (Congestion, High Reliability) (C, HR), (Congestion, Low Reliability) (C, LR) and Optimal Operating Region (OOR). Depending on the current state, the sink calculates an updated reporting frequency and then broadcasts this information to the sensor nodes. In (NC, LR) state, the reliability is lower than required, so the reporting frequency is increased. In both (NC, HR) state and (C, HR) state, the reporting frequency is decreased with respective factors. In (C, LR) state, the reporting frequency is decreased more aggressively. In OOR, the required reliability is attained with minimum energy expenditure.

Thus, the reporting frequency is left unchanged for the next decision interval. The self-adjusting feature in ESRT makes the network work as close as possible to the optimal operating point, where the required reliability is achieved without network congestion.

One disadvantage with this method is that all sensor nodes are controlled at once, treating regions of interest (where faster rates are appropriate) or regions with higher node density in the same way as uninteresting regions or...
regions with low node density. Also ESRT does not consider the case in which there are different types of events requiring different levels of reliability [4].

B. Fusion:
Fusion consists of three congestion mitigation techniques applied in different layers, that is, hop-by-hop flow control, rate limiting and prioritised MAC [4].

The hop-by-hop flow control resembles the backpressure scheme in CODA. The difference lies in that instead of using backpressure messages, in Fusion each sensor node sets a congestion bit in the header of every ongoing packet. By taking advantage of the broadcast nature of the wireless medium, the implicit feedback obviates the need for explicit control messages that can waste the network bandwidth. The congestion detection method in Fusion is also similar to that in CODA. Both buffer occupancy and channel utilisations are used to determine the congestion status. When a sensor node overhears a packet from its parent node (the node closer to the sink) with the congestion bit set, it stops forwarding data towards the sink. If a path persistently experiences congestion, hop-by-hop backpressure will eventually reach the source and the source will throttle the source event rate [4].

Rate limiting is a preventive scheme to avoid congestion. In a WSN with a large diameter or many hops, if packets are dropped due to congestion somewhere in the WSN, a significant amount of energy and bandwidth can be wasted. In Fusion, each node listens to the traffic its parent node forwards to estimate the total Number (N) of the unique sources routing through the parent node. A token bucket scheme is used to regulate the sending rate of each node. A node accumulates one token every time it hears its parent node has forwarded N packets, up to a maximum value. The node is allowed to send only when its token account is above zero, and each packet sending costs one token. The rate limiting method improves fairness, but it is only effective in the case that all nodes have the same traffic load and the routing tree is not significantly skewed [4].

Fusion also incorporates a prioritised MAC scheme to ensure that congested nodes receive prioritised access to the channel. A typical Carrier Sense Multiple Access (CSMA) MAC layer gives all nodes the same chance to compete for channel access. In WSNs, the parent node, which may gather traffic from several children nodes, tends to overflow if it does not have more chances to transmit its packets. Therefore, it has to drop the packets forwarded from the children nodes. The solution in Fusion is that the random back off time of each node is related to its local congestion state. Thus, the congested node has a better chance to win a contention so as to drain its buffer faster. Also, this mechanism allows the children nodes to learn the congestion information indicated by the congestion bit in its packets sooner [4].

C. EB:
Not much different from CODA and Fusion, EB also uses a backpressure scheme. EB considered congestion control in a tree routing structure from all data sources to a sink in a WSN [4].

It consists of three steps repeatedly run at each node.

1. Each node measures the average rate r at which packets can be sent from it.
2. The node divides the rate r among the number of children nodes, n, to give the per-node data packet generation rate rdata = r/n and adjusts the rate if its buffer is overflowing or about to overflow.
3. The node compares the rate rdata with the rate rdata, parent sent from its parent, uses the smaller rate among the two values and propagates it such that a data source does not send packets beyond the minimum rate supported by the nodes along the path to the sink [4].

D. Congestion Control and Fairness (CCF):
CCF exactly adjusts traffic rate based on packet service time along with fair packet scheduling algorithms, while Fusion in performs stop-and-start non-smooth rate adjustment to mitigate congestion. CCF was proposed in as a distributed and scalable algorithm that eliminates congestion within a sensor network and ensures the fair delivery of packets to a sink node. CCF exists in the transport layer and is designed to work with any MAC protocol in the data-link layer. In the CCF algorithm, each node measures the average rate r at which packets can be sent from the node, divide the rate r among the number of children nodes, adjust the rate if queues are overflowing or about to overflow and propagate the rate downstream.

CCF uses packet service time to deduce the available service rate. Congestion information is implicitly reported. It controls congestion in a hop-by-hop manner and each node uses exact rate adjustment based on its available service rate and child node number. It can be shown that CCF guarantees simple fairness. CCF has two major problems. The rate adjustment in CCF relies only on packet service time which could lead to low utilization when some sensor nodes do not have enough traffic or there is a significant packet error rate. Furthermore, it cannot effectively allocate the remaining capacity and as it uses work-conservation scheduling algorithm, it has a low throughput in the case that some nodes do not have any packet to send [1] [10].

E. Congestion Detection and Avoidance (CODA):
CODA is an energy efficient congestion control scheme for sensor networks was proposed [1] [10]. CODA is designed to solve the congestion problem in the sensors-to-sink direction [4].
CODA comprises three mechanisms: (i) receiver-based congestion detection; (ii) open-loop hop-by-hop backpressure; and (iii) closed-loop multi-source regulation. CODA detects congestion based on queue length as well as wireless channel load at intermediate nodes. Furthermore it uses explicit congestion notification approach and also an AIMD rate adjustment technique [1] [4] [10]. CODA jointly uses end-to-end and hop-by-hop controls [1] [10].

In order to detect congestion, CODA uses a combination of the present and past channel loading conditions, and the current buffer occupancy at each receiver. If the buffer occupancy or the wireless channel load exceeds a threshold, congestion is inferred. To save energy, CODA uses a sampling scheme to activate local channel monitoring. Once congestion is detected, the node notifies its upstream neighbour nodes via a backpressure mechanism [4].

The backpressure mechanism operates in an open-loop hop-by-hop manner. A node broadcasts backpressure messages as long as it detects congestion. Backpressure messages are propagated upstream towards the source. A node that receives a backpressure message can adjust its sending rate in an Additive Increase Multiplicative Decrease (AIMD) manner or drop packets based on its local congestion control policy. It can decide whether or not to further propagate the backpressure message upstream based on its own local network conditions [4].

Besides the open-loop hop-by-hop backpressure mechanism, CODA uses a closed-loop end-to-end scheme to regulate multisource rates. When the event rate at a source is less than some fraction of the maximum theoretical throughput of the channel, the source regulates itself. When the threshold value is exceeded, the source is more likely to contribute to congestion and therefore closed-loop control is triggered. At this point, the source requires constant feedback (i.e. ACK) from the sink to maintain its sending rate. The source sets the ‘regulate bit’ in the event packets. Reception of packets with the regulate bit set forces the sink to send ACKs. When congestion builds up, ACKs can be lost. If the sources cannot receive enough ACKs from the sink, they drop their event rates [4].

The sink can infer that congestion happens if it cannot receive event packets at the desired reporting rate. In this case, the sink can stop sending ACKs to the source nodes. When the congestion has subsided the sink can start to transmit ACKs again and the event reporting rates of the resources will increase [4].

F. Adaptive Rate Control (ARC):

ARC mechanism was proposed which is most effective in achieving the goal of fairness, while being energy efficient for both low and high duty cycle of network traffic. The ARC does not have any congestion detection or notification mechanisms. Each intermediate node increases its sending rate by a constant $\alpha$ if it overhears successful packet forwarding by its parent node. Otherwise, the intermediate node multiplies its sending rate by a factor $\beta$.

Both CODA and ARC employ AIMD-like (Additive Increase Multiplicative Decrease) coarse rate adjustment. [1] [10]

G. PCCP:

PCCP is designed with such motivations: 1) In WSNs, sensor nodes might have different priority due to their function or location. Therefore congestion control protocols need guarantee weighted fairness so that the sink can get different, but in a weighted fair way, throughput from sensor nodes. 2) Congestion control protocols need to improve energy-efficient and support traditional QoS in terms of packet delivery latency, throughput and packet loss ratio.

PCCP tries to avoid/reduce packet loss while guaranteeing weighted fairness and supporting multipath routing with lower control overhead. PCCP consists of three components: intelligent congestion detection (ICD), implicit congestion notification (ICN), and priority-based rate adjustment (PRA).

ICD detects congestion based on packet inter-arrival time and packet service time. The joint participation of inter-arrival and service times reflect the current congestion level and therefore provide helpful and rich congestion information. To the best of our knowledge, jointly use of packet inter-arrival and packet service times as in ICD to measure congestion in WSNs has not been done in the past.

PCCP uses implicit congestion notification to avoid transmission of additional control messages and therefore help improve energy-efficiency. In ICN, congestion information is piggybacked in the header of data packets. Taking advantage of the broadcast nature of wireless channel, child nodes can capture such information when packets are forwarded by their parent nodes towards the sink.

Finally, PCCP designs a novel priority-base rate adjustment algorithm (PRA) employed in each sensor node in order to guarantee both flexible fairness and throughput, where each sensor node is given a priority index. PRA is designed to guarantee that: (1) The node with higher priority index gets more bandwidth; (2) The nodes with the same priority index get equal bandwidth. (3) A node with sufficient traffic gets more bandwidth than one that generates less traffic. The use of priority index provides PCCP with high flexibility in weighted fairness. For example, if the sink wants to receiver the same number of packets from each sensor node, the same priority index can be set for all nodes.

On the other hand, if the sink wants to receive more detailed sensory data from a particular set of sensor nodes, such sensor nodes can be assigned a higher priority index and therefore allocated higher bandwidth. [1]

H. Siphon:

The Siphon is another congestion control protocol and it is based on the use of virtual base stations. There
are two detection techniques in Siphon protocol: node-initiated congestion detection and physical sink initiated “post-facto” congestion detection [4] [9].

In the first mechanism, all locations and levels of congestion in a node are determined. When a virtual sink observes a congestion situation near it, it sends a message that notifies that situation. The most important is that the traffic is redirected to other areas of the network so that the node can flow all the data that are causing the congestion.

In the second mechanism, the physical base stations will interfere directly in the congestion detection through monitoring the reliability and data reception quality. When these data are outside the normal range a signal is then sent to a nearby virtual sink that can transmit to the network. This method has the advantage that it is not necessary that all nodes need to make congestion detection.

The redirection of data is done using redirection bit. There are two ways of using this bit, one is enable the bit only when there is a congestion detection (on-demand redirection) and the other is the bit is always on (always-on redirection). A sensor receives a data packet with the redirection bit active, then it will forward it to the VS nearest you, if the bit is not active then the traffic goes unchanged. If a virtual sink receives a package that has been redirected will have to send it to the near virtual sink and has recently received a message that includes a signature byte. When everything runs normally and the virtual sinks are connected to physical base stations that receive packets they redirect the packets and will put them back on the network [4].

I. SenTCP:

SenTCP is an open-loop hop-by-hop congestion control protocol proposed for WSNs. SenTCP uses the average local packet service time and average local packet inter-arrival time to jointly estimate the current local congestion degree in each intermediate sensor node. It uses hop-by-hop congestion control. In SenTCP, each intermediate sensor node sends a hop-by-hop feedback signal backward to its children node. The feedback signal, which carries local congestion degree and buffer occupancy ratio, is used by the children sensor nodes to adjust their sending rates in the transport layer [4].

J. Congestion control for multiclass traffic (COMUT):

COMUT is a distributed cluster-based congestion control mechanism for supporting multiple classes of traffic in WSNs. COMUT is based on the self-organisation of the networks into clusters. Each cluster autonomously and proactively monitors congestion within its localised scope. The clusters then exchange appropriate information to facilitate system-wide rate control [4].

In COMUT, sensors are self-organised into clusters. Each cluster is governed by a sentinel. Each sensor in a cluster estimates its local traffic load within a specific time interval and reports it to its sentinel via a local broadcast. Upon collecting all the load values from the sensors, the sentinel estimates the cluster’s traffic intensity. If the intensity is higher than a threshold, the cluster is regarded as congested. The sentinel periodically sends its locally computed traffic intensity values to other sentinels and finally to the sources. To regulate the source rate, the AIMD policy is used. COMUT considers two types of packet flows, high importance flows and low importance flows. To differentiate the two types of flows, upon congestion, for a low importance flow, the rate is dropped to a minimum value (instead of multiplicative decrease) if packets of higher importance exist along the congested path [4].

K. Congestion control for sensor networks (CONCISE):

Different from the backpressure approaches to regulate sources’ rate every time congestion occurs, an adaptive data-aggregation is used in CONCISE to reduce the amount of data travelling throughout the networks. Depending on the congestion level experienced at each aggregate node, CONCISE varies the degree of aggregation of data packets. The aggregation of data can reduce the amount of traffic thus reducing congestion [4].

Using adaptive aggregation can only partially resolve the congestion problem because there is a limit on the aggregation degree. If the limit is exceeded, some information about the events in a WSN may be lost. In addition, aggregation of data at a high degree may consume excessive processing and hence energy resources in sensor nodes. CONCISE can be used as a complement to the other congestion control methods [4].

L. Congestion control from sink to sensors (CONSISE):

Different from the aforementioned congestion control mechanisms, CONSISE works in the sink-to-sensors direction. Although congestion mostly happens in the sensors-to-sink direction, it is argued that congestion can happen in the sink-to-sensors direction as well subject to some factors, such as reverse path traffic from the sensors to the sink and broadcast storm problem which refers to the higher level of contention and collision occurring due to a series of local broadcast. Thus, congestion in the sensors-to-sink direction will not be rare if WSN is built over a CSMA/CA type of MAC and flooding-based routing protocols [4].

CONSISE adjusts the sending rate of each sensor node to utilise the available network bandwidth depending on the congestion level in the local environment. The CONSISE algorithm is run in the sensor nodes and the sink periodically. At the end of each epoch, a node determines its sending rate and provides explicit feedback to its upstream node (closer to the sink). The upstream node then adjusts its sending rate correspondingly. To receive packets as fast as possible, a downstream node determine which one is its preferred upstream node and send a notification to it. If the upstream node does not obtain a notification as to whether it is a preferred upstream node over an epoch, it gradually decreases its
sending rate. Thus, the preferred upstream node can send data at a higher rate without experiencing heavy congestion due to competitions from other non-preferred nodes [4].

V. OVERVIEW OF VARIOUS TRANSPORT PROTOCOLS FOR WIRELESS SENSOR NETWORKS

In this section, three reliable data transport protocols, namely PSFQ, RMST and, GARUDA are reviewed. All three provide guaranteed reliability for data transport. PSFQ and GARUDA support sink-to-sensors (one-to-many) data transport while RMST supports sensors-to-sink (many-to-one) data transport.

A. Pump slowly fetch quickly (PSFQ):

The PSFQ protocol is designed to deliver a number of packets from a single source node to a subset of receiver nodes or to all nodes within a WSN. It is mainly used in the sink-to-sensors direction. PSFQ provides guaranteed reliability for data delivery. It can be used to transport management data or retasking codes in sensors [4] [5].

PSFQ comprises three protocol components: message relaying (pump operation), relay-initiated error recovery (fetch operation) and selective status reporting (report operation).

The pump operation disseminates the packets (e.g. segments of the retasking codes) by broadcasting to the target nodes (sensors) from the user node (sink). To enable local loss recovery and in-sequence data delivery, a data cache is created at each intermediate node. To schedule the forwarding of the packets, two timers, T1 and T2 are used.

A user node broadcasts a packet to its neighbours every T1 until all the data segments have been sent out. The time T1 between the different packets is relatively large (pump slowly) and each packet has a sequence number. When a node receives a new packet not yet seen, the node stores the packet in its cache. If the new packet is received in-sequence, the node waits for some random time between T1 and T2 and forwards it further. However, the packet will not be forwarded when the node notices that four or more of its neighbours have already forwarded the same packet, since the expected additional coverage achieved by forwarding the packet one more time tends to be small.

The fetch operation is an act of requesting a retransmission from neighbouring nodes once a loss is detected by a sequence gap in the packets at a receiving node. When a packet is received out-of-sequence, it is also stored. However, instead of forwarding the packet, the node requests immediate retransmission of the missing packets from any upstream neighbour using a NACK message indicating the missing packet(s) (fetch quickly). As soon as the node receives the missing packets, it starts forwarding the segments in-sequence by the pump operation.

To avoid the message implosion problem, NACK messages are not propagated. That is, a node does not relay a NACK message to its upstream neighbours unless the number of times the same NACK is received exceeds a predefined threshold, while the missing packets requested by the NACK message are not found in the node’s cache.

The fetch operation by means of a NACK corresponds to a retransmission request and is triggered by missing sequence numbers in a node’s cache. When the last segment of an information block is missing, there is no higher sequence number and this method of detecting losses fails. To avoid this problem, PSFQ includes a proactive fetch operation such that a node can also enter the fetch mode and send a NACK message for the next segment or the remaining segments if the last segment has not been received and no new segment is delivered after a period of time.

The report operation is an optional operation designed specifically to feedback data delivery status to the sink. The sink node sets the ‘report bit’ in its injected packet whenever it needs to know the latest status of the surrounding nodes. To reduce the number of report messages, only the last hop nodes will respond by sending a report message to its parent node. Each node along the path towards the sink node will piggyback its report message by adding its own status information to the report.

B. Reliable multisegment transport (RMST):

Reliable multisegment transport (RMST) The RMST scheme is mainly used in the sensors-to-sink direction and it provides guaranteed reliability of packet delivery. It is designed to complement directed diffusion by including a reliable data transport service on top of it. Directed diffusion is used to discover paths from sensors to the sink.

Similar to PSFQ, RMST is a NACK-based protocol, which employs primarily timer-driven loss detection and repair mechanisms. The basic mechanism for loss detection is a watchdog timer. RMST provides both noncaching mode and caching mode. In the caching mode, the sink and all intermediate caching nodes in the reinforced path from source sensor node to sink cache the segments and check the cache periodically (controlled by the timer) for missing segments. When a node detects missing segments, it generates a NACK message which travels back to the source along the reinforced path. The first node having the missing segments in its cache forwards them again towards the requesting node and thereafter to the sink. If the node can retrieve all requested segments from its cache, then it drops the NACK packet. Otherwise the NACK is forwarded further. In its non-caching mode, only the sinks set the timer and monitor the integrity of a RMST entity in terms of received segments. The intermediate nodes do not maintain any cache. The NACK packet travels back to the source node, which retransmits the missing segments [4] [5].

Besides the NACK-based mechanisms, RMST also combines several mechanisms in other layers to enhance reliability. At the application layer, redundancy is used:
the source transmits the whole data block periodically until the sink explicitly unsubscribes. At the MAC layer, retransmissions of lost MAC frames are used [4].

C. GARUDA:

Similar to PSFQ, GARUDA provides data transport from a sink to all or part of the sensors in a WSN with guaranteed reliability.

GARUDA also uses a NACK-based scheme. However, since NACK-based request schemes do not suffice for a single packet delivery (or when all packets in an information block are lost) without any additional support, GARUDA uses an ACK-based scheme as an alternative for the first packet. GARUDA addresses the reliable delivery of the first packet using a Wait-for-First-Packet (WFP) pulsing scheme. In this scheme, a small finite series of short duration pulses are repeated periodically. Due to the unique property of the pulse (larger amplitude and shorter period) the receiving node, regardless of whether it is currently idle or receiving a regular data packet, can sense the pulse. When a sink wants to send the first packet, the sink transmits the finite series of WFP pulse on a periodic basis. The sensor nodes within the transmission range of the sink, upon reception of the pulses, also start pulsing with the same periodicity between two series of pulses. This process is repeated until all the nodes start pulsing. The sink after pulsing for a finite time, transmits the first packet as a regular data packet transmission [4] [5].

When the first packet has been delivered, a core of the WSN is constructed. The core is an approximation to the minimum dominating set of the WSN topology and the members of this set act as recovery servers for downstream core members and neighbouring non-core members.

Only the nodes that have a hop distance to the sink of the form $3i$, where $i$ is an integer, can elect themselves as core nodes. A candidate core node refrains from becoming a core node when there are enough core nodes in its neighbourhood. On the other hand, non-core nodes that have no core node within their transmission range can request a candidate core node to become a core node. All core nodes know at least one upstream core node from which they request retransmissions.

GARUDA performs a two-phase loss recovery mechanism: loss recovery for core nodes first and loss recovery for non-core nodes afterwards.

When a core node receives an out-of-sequence packet, the core infers a loss. A core node sends a request to an upstream core node only if it is notified that the missing packet is available at the upstream core node. The notification is through a scalable Availability Map (A-map) that is included in every packet and conveys meta-level information representing availability of packets with a bit set. The downstream node can use the knowledge included in the bitmap to suppress NACKs for packets also missing in its upstream node.

A non-core member snoops all (re)transmissions from its core node. Once it observes an A-map from its core node with all the bits set (i.e. the core node has all the packets it needs), it enters the non-core recovery phase by initiating retransmission requests to the core node [4].

VI. FURTHER DISCUSSIONS

In Sections 3 and 4, we have provided a brief overview of the existing congestion control protocols and reliable data transport proposed for WSNs. In most protocols, caching in intermediate nodes is used. Caching data in the intermediate nodes can reduce the total number of data retransmissions thus conserving power and reducing delay. However, since the memory in sensor nodes is generally small, it must be used wisely. In addition, keeping multiple copies of data packets in the memories of many sensor nodes may waste energy and resources.

In order to detect congestion, some protocols use buffer occupancy and some others use channel load or both. Buffer occupancy-based methods are the simplest while channel load provides more accurate information in some cases. However, monitoring and calculation of channel load can be costly in terms of power consumption. In these transport protocols, some use hop-by-hop control while the others use end-to-end control. In the end-to-end control, most control tasks are relied upon the sink, which in general is less restricted by processing resources. However, as congestion tends to happen in the proximity of the sink, the control messages issued by the sink may be throttled. On the other hand, hop-by-hop control needs the intermediate nodes to participate in the control process, which may increase the workload and power consumption on these resource-limited nodes.

In summary, these methods have made significant attempts to tackle the transport problems in WSNs from different aspects and in different circumstances. However, more research efforts are still necessary. Future research on transport protocols in WSNs should consider further improvement in the following aspects:

1. A unified transport protocol providing both sensors-to-sink and sink-to-sensors data transport is desired.
2. Interaction between reliability provision and congestion control needs to be investigated. A unified protocol providing both reliability and congestion control is desired.
3. Application of cross-layer design methodology should be considered. Cross-layer design is more specific to applications and networks.
4. Energy efficiency needs to be more emphasized in the future transport protocols for WSNs.
5. QoS provisioning in WSNs needs to be considered. In a WSN, both delay sensitive and loss sensitive data can coexist. The data in a WSN may be prioritised in terms of reliability and delay requirements. Future transport protocol design should take the different QoS requirements and fairness into account.

VII. CONCLUSION

In this paper, recently proposed congestion control and reliable data transport protocols specifically designed for WSNs have been reviewed. Although some progress has been achieved, more research efforts are needed to continue to improve data transport in WSNs.

REFERENCES


