A Real-Time and Reliable Transport $(RT)^2$ Protocol for Wireless Sensor and Actor Networks

Vehbi Cagri Gungor, *Member, IEEE*, Özgür B. Akan, *Senior Member, IEEE*, and Ian F. Akyildiz, *Fellow, IEEE*

***Abstract*—Wireless Sensor and Actor Networks (WSANs) are characterized by the collective effort of heterogenous nodes called sensors and actors. Sensor nodes collect information about the physical world, while actor nodes take action decisions and perform appropriate actions upon the environment. The collaborative operation of sensors and actors brings significant advantages over traditional sensing, including improved accuracy, larger coverage area and timely actions upon the sensed phenomena. However, to realize these potential gains, there is a need for an efficient transport layer protocol that can address the unique communication challenges introduced by the coexistence of sensors and actors.**

In this paper, a Real-Time and Reliable Transport $(RT)^2$ protocol is presented for WSANs. The objective of the $(RT)^2$ protocol is to reliably and collaboratively transport event features from the sensor field to the actor nodes with minimum energy dissipation and to timely react to sensor information with a right action. In this respect, the $(RT)^2$ protocol simultaneously addresses congestion control and timely event transport reliability objectives in WSANs. To the best of our knowledge, this is the first research effort focusing on real-time and reliable transport protocol for WSANs. Performance evaluations via simulation experiments show that the $(RT)^2$ protocol achieves high performance in terms of reliable event detection, communication latency and energy consumption in WSANs.

***Index Terms*—Congestion detection and control, energy efficiency, real-time and reliable transport protocol, wireless sensor and actor networks.**

I. Introduction

WIRELESS sensor and actor networks (WSANs) are characterized by the collective effort of densely deployed sensor nodes and sparsely deployed actor nodes. In WSANs, sensor nodes collect information about the physical world, while actors take action decisions and perform appropriate actions upon the environment. The existing and potential applications of WSANs span a very wide range, including real-time target tracking and surveillance, homeland security, and biological or chemical attack detection [2]. Realization of these currently designed and envisioned applications, however, directly depends on real-time and reliable communication capabilities of the deployed sensor/actor network.

Recently, there has been considerable amount of research efforts, which have yielded many promising communication protocols for wireless sensor networks (WSNs) [1], [3], [16], [17]. The common feature of these protocols is that they mainly address the energy-efficient and reliable data communication requirements of WSN. However, in addition to the energy-efficiency and communication reliability, many proposed WSAN applications have strict delay bounds and hence mandate timely transport of the event features from the sensor field to the actor nodes [2], [6]. Consequently, the unique features and application requirements of WSANs call for a real-time and reliable data transport solution. The functionalities and design of a such solution for WSANs are the main issues addressed in this paper.

The major communication challenges for realization of a real-time and reliable transport in WSANs are outlined as follows.

- *Heterogeneous reliability requirements:* The transport paradigms of WSANs have different reliability requirements due to the node heterogeneities in the deployment field [2]. For example, while sensor–actor communication may not require 100% reliability due to the correlation among the sensor readings [1], [15], actor–actor communication requires 100% reliability in order to make a decision on the most appropriate way to collaboratively perform the action.
- *Delay bounds:* In WSANs, actor nodes need to immediately react to sensor data based on the application-specific requirements. Hence, real-time communication within certain delay bounds is a crucial concern to guarantee timely execution of the right actions.
- *Node mobility and route failures:* Actor nodes in WSANs might be highly mobile depending on the application requirements. The mobility may lead to route failures and packet losses that must be accurately captured by the transport layer solutions to avoid inaccurate congestion control.
- *Wireless channel errors:* The wireless channel errors in WSANs lead to bursts of packet loss [2], [6]. Despite the existence of channel coding schemes, packet-level transport layer reliability mechanisms are required.
- *Energy efficiency:* Although the primary objective of the transport protocols in WSANs is reliable event detection and timely execution of the right actions, this must be accomplished with minimum energy consumption due to limited energy resources of sensor nodes.

Despite the existence of several transport protocols in the current literature for wireless ad hoc and sensor networks [1],

Manuscript received June 27, 2005; revised May 16, 2006, and December 4, 2006; approved by IEEE/ACM Transactions on Networking Editor M. Krunz. This work was supported by the National Science Foundation under Contract ECS-0428329.

V. C. Gungor was with the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA. He is now with Eaton Corporation, Milwaukee, WI 53216 USA (e-mail: gungor@ece.gatech.edu).

Ö. B. Akan is with Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara 06531, Turkey (e-mail: akan@eee.metu.edu.tr).

I. F. Akyildiz is with the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: ian@ece.gatech.edu).

Digital Object Identifier 10.1109/TNET.2007.900413

[7], [10], [13], [16], [17], none of them addresses the application-specific delay bounds and the heterogeneous reliability objectives of WSAN applications. Consequently, there is a need for a new real-time and reliable communication protocol, which can efficiently address the unique challenges of WSANs outlined above.

In this paper, to address all above communication challenges, a real-time and reliable transport $(RT)^2$ protocol is presented for WSANs. $(RT)^2$ is a novel transport solution that seeks to *achieve reliable and timely event detection with minimum possible energy consumption and no congestion*. It enables the applications to perform right actions timely by exploiting both the correlation and the collaborative nature of WSANs. Furthermore, $(RT)^2$ addresses heterogeneous reliability requirements of both sensor–actor and actor–actor communication. More specifically, for sensor–actor communication, unlike traditional end-to-end reliability notions, $(RT)^2$ defines *delay-constrained event reliability* notion based on both *event-to-action delay bounds* and event reliability objectives. On the other hand, for actor–actor communication, it introduces 100% packet-level reliability mechanisms to avoid inaccurate action decisions in the deployment field. This way, the $(RT)^2$ protocol simultaneously addresses event transport reliability and timely action performance objectives of WSANs.

In general, compared to the existing transport layer proposals in the related literature, the main contribution of $(RT)^2$ is that *it concurrently provides real-time communication support and addresses heterogeneous transport reliability requirements* for typical WSAN applications involving reliable event detection and timely action objectives within a certain delay bound. To this end, the notion of *delay-constrained event reliability* distinguishes $(RT)^2$ from other existing transport solutions proposed for wireless ad hoc and sensor networks. To the best of our knowledge, reliable event transport has not been studied from this perspective before and hence $(RT)^2$ is the first solution attempt simultaneously addressing the real-time and reliable event transport and action performance objectives of WSANs.

The remainder of the paper is organized as follows. In Section II, we present the network architecture and describe the design principles and functionalities of the $(RT)^2$ protocol in detail. The protocol operation of $(RT)^2$ for sensor–actor and actor–actor communication is described in Sections III and IV, respectively. Performance evaluation and simulation results are presented in Section V. Finally, the paper is concluded in Section VI.

II. $(RT)^2$ Protocol Design Principles

Unlike traditional networks, the sensor/actor network paradigm necessitates that the event features are collaboratively estimated within a certain reliability and real-time delay bound. To achieve this objective with maximum resource efficiency, the $(RT)^2$ protocol exploits both the correlation and the collaborative nature of the network. In the following sections, we first describe the characteristics and challenges of both sensor–actor and actor–actor communication and then based on these characteristics, we discuss the main design components of the $(RT)^2$ protocol in detail. We also present a case study to gain more insight regarding the challenges of sensor/actor network.

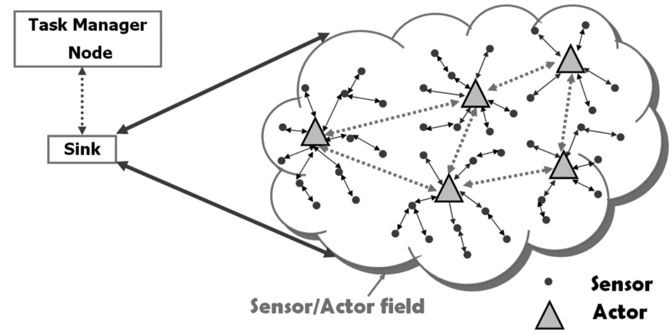


Fig. 1. Illustration of an integrated architecture of WSANs.

A. Network Architecture

A typical network architecture of WSANs is shown in Fig. 1. In this network architecture, the sensors are energy constrained, multifunctional devices with limited processing and low range communication capabilities, while the actors are resource-rich nodes equipped with better processing capabilities, high transmission power and longer battery life time. Furthermore, in WSANs, a large number of sensors, i.e., on the order of hundreds or thousands, are randomly deployed in a target area to perform a collaborative sensing task. Such a dense deployment is usually not necessary for actors, because actors have higher capabilities and can act on large areas.

In WSANs, the collaborative operation of the sensor nodes enables *distributed sensing* of a physical phenomenon. After sensors detect an event occurring in the environment, the event data is distributively processed and transmitted to the actors, which gather, process, and eventually reconstruct the event data. We refer the process of transmission of event features from the sensor nodes to the actor nodes as *sensor–actor communication*. Once an event has been detected in the deployment field, the actors need to communicate with each other to make a decision on the most appropriate way to collaboratively perform the action. We refer to this process as *actor–actor communication*. Therefore, the operation of the WSANs can be considered a timely event detection, decision and acting loop.

Note that in an integrated network architecture of WSANs, event decision and coordination between different actor nodes in the vicinity of the phenomenon can be handled by a designated actor node. The selection of the designated actor and which actors will be involved in the action is a matter of application requirements and in responsibility of associated routing procedure and hence beyond the scope of this work.

B. Reliable Event Transport

The $(RT)^2$ protocol is equipped with different reliability functionalities to address heterogeneous requirements of both sensor–actor and actor–actor communication. Next, the main features of these reliability functionalities are described.

1) Sensor–Actor Transport Reliability: In WSANs, sensor–actor transport is characterized by the dense deployment of sensors that continuously observe physical phenomenon. Because of the high density in the network topology, sensor observations are highly correlated in the space domain. In addition, the nature of the physical phenomenon constitutes the temporal correlation between each consecutive observation of the sensor. Because of these spatial and temporal correlations along with the collaborative nature of the WSANs, sensor–actor transport does not require 100% reliability [1], [15].

Consequently, for sensor–actor communication, conventional end-to-end reliability definitions and solutions would only lead to over-utilization of scarce sensor resources. On the other hand, the absence of reliable transport mechanism altogether can seriously impair event detection. Thus, the sensor–actor transport paradigm requires a collective *event transport reliability* notion rather than the traditional end-to-end reliability notions. The (RT)^2 protocol also considers the new notion of *event-to-action delay bound* (described in Section II-C) to meet the application-specific deadlines. Based on both event transport reliability and event-to-action delay bound notions, we introduce the following definitions.

- The *observed delay-constrained event reliability* (DR_i) is the number of received data packets within a certain delay bound at the actor node in a decision interval i. In other words, DR_i counts the number of correctly received packets complying with the application-specific delay bounds and the value of DR_i is measured in each decision interval i.
- The *desired delay-constrained event reliability* (DR^*) is the minimum number of data packets required for reliable event detection within a certain application-specific delay bound. This lower bound for the reliability level is determined by the application and based on the physical characteristics of the event signal being tracked.
- The *delay-constrained reliability indicator* (δ_i) is the ratio of the observed and desired delay-constrained event reliabilities, i.e., $\delta_i = DR_i/DR^*$.

Based on the packets generated by the sensor nodes in the event area, the event features are estimated and DR_i is observed at each decision interval i to determine the necessary action. If the observed delay constrained event reliability is higher than the reliability bound, i.e., $DR_i > DR^*$, then the event is deemed to be reliably detected within a certain delay bound. Otherwise, appropriate action needs to be taken to assure the desired reliability level in sensor–actor communication. For example, to increase the amount of information transported from the sensors to the actor, reporting frequency of the sensors can be increased properly while avoiding congestion in the network. Therefore, sensor–actor transport reliability problem in WSANs is to *configure the reporting rate*, f, *of source nodes so as to achieve the required event detection reliability*, DR^*, *at the actor node within the application-specific delay bound*. The details of the (RT)^2 protocol operation for sensor–actor communication are described in Section IV.

2) Actor–Actor Transport Reliability: In WSANs, a reliable and timely actor–actor ad hoc communication is also required to collaboratively perform the right action upon the sensed phenomena [2]. The (RT)^2 protocol simultaneously incorporates *adaptive rate-based transmission control and (SACK)-based reliability mechanism* to achieve 100% packet reliability in the required ad hoc communication. To achieve this objective, (RT)^2 protocol relies upon new feedback based congestion control mechanisms and probe packets to recover from subsequent losses and selective-acknowledgments (SACK) to detect any holes in the received data stream. These algorithms are shown to be beneficial and effective in recovering from multiple packet losses in one round-trip time (RTT) especially [13]. The details of adaptive rate-based transmission and congestion control algorithms for actor–actor ad hoc communication are explained in Section IV. Next, event-to-action delay bound notion of (RT)^2 protocol is explained in detail.

C. Real-Time Event Transport

To assure accurate and timely action on the sensed phenomena, it is imperative that the event is sensed, transported to the actor node and the required action decision is taken within a certain delay bound. We call this *event-to-action delay bound*, Δ_{e2a}, which is specific to application requirements and must be met so that the overall objective of the sensor/actor network is achieved.

The event-to-action delay bound Δ_{e2a}, has three main components as outlined below.

1) **Event transport delay** (Γ^{tran}): It is mainly defined as the time between when the event occurs and when it is reliably transported to the actor node. In general, it involves the following delay components: *buffering delay, channel access delay, transmission delay*, and *propagation delay*.
2) **Event processing delay** (Γ^{proc}): This is the processing delay experienced at the actor node when the desired features of event are estimated using the data packets received from the sensor field. This may include a certain decision interval [1] during which the actor node waits to receive adequate samples from the sensor nodes.
3) **Action delay** (Γ^{act}): The action delay is the time it takes from the instant that event is reliably detected at the actor node to the instant that the actual action decision is taken. It includes the *task assignment delay*, i.e., time to select the *best*[1] set of actors for the task.

More specifically, while event transport delay (Γ^{tran}) and event processing delay (Γ^{proc}) occur during sensor–actor communication, action delay (Γ^{act}) is resulted from actor–actor communication in the deployment field. Let Δ_{e2a} be the event-to-action delay bound for the data packet generated by the detection of event. Then, for a timely action, it is necessary that the following relation holds:

$$\Delta_{e2a} \geq \Gamma^{tran} + \Gamma^{proc} + \Gamma^{act}. \quad (1)$$

Note that Γ^{tran} is directly affected by the current network load and the congestion level in the network. In addition, the network load depends on the event reporting frequency, f, which is used by the sensor nodes to send their readings of the event. Specifically, the buffering delay directly depends on the transport rate, and on the queue management and service discipline employed at each sensor node in the network.

In addition, since events occurring at further distances from the actor node are in general characterized by a higher average number of hops to reach the actor node, it is more difficult

[1]The best set of actors refers to the actors which are close to the event area, or which has high capability and residual energy, or which has small action completion time upon the sensed phenomenon [2].

to provide event-to-action delay bounds. Considering that the per-hop propagation delay does not vary, the buffering delay must be controlled in order to compensate for the increase in the event transport delay. To accomplish this objective, we introduce *Time-Critical Event First* (TCEF) scheduling policy. TCEF applies the general principles of earliest deadline first service discipline on each sensor node, which is shown to be the optimal scheduling policy, i.e., to have the widest scheduling region, when real-time deadlines in a system are considered [12].

To update the remaining time to deadline without a globally synchronized clock in the network, we measure the elapsed time for a packet at each sensor and piggyback the elapsed time to the event packet so that the following sensor can determine the remaining time to deadline without a globally synchronized clock. Then, by using these elapsed time measurements, the event packets are given high priority at the sensor nodes, as their remaining time to deadline decreases. This way, time critical sensor data obtain high priority along the path from the event area to the actor node and is served first, which is crucial to meet the application deadlines.

Note that although TCEF policy makes it possible to meet deadlines in the normal operating conditions of the network, in case of severe network congestion, it may become insufficient to provide delay-constrained event reliability. Hence, in addition to TCEF scheduling, the $(\mathrm{RT})^2$ protocol considers the event-to-action delay bounds and congestion conditions in its reporting rate update policies to assure timely and reliable event transport in WSANs (see Section III). It is also important to note that the measured elapsed time at each sensor node can give an idea of congestion level experienced in the network, since it represents both the buffering delay and the channel contention around the sensor node (see Section II-E).

D. Case Study

To investigate the relationship between the event-to-action delay and the event reporting rate, we develop an evaluation environment using *ns*-2 [14]. The parameters used in our case study are listed in Table I. In our simulations, 200 sensor nodes were randomly positioned in a 200 m $\times$ 200 m sensor field. Node parameters such as radio range and IFQ (interface queue) length were carefully chosen to mirror typical sensor mote values [11]. Event centers (X_{ev}, Y_{ev}) were randomly chosen and all sensor nodes within the event radius behave as sources for that event. In this case study, the actor node receiving the data is placed in the middle of the lower side of the deployment area. To communicate source data to the actor node, we employed a simple CSMA/CA based MAC protocol. For each simulation, we run 10 experiments and take the average of the measured values.

First, we investigate the impact of event reporting frequency on average sensor–actor communication delay and *on-time event delivery ratio*. Here, on-time event delivery ratio represents the fraction of data packets received within sensor–actor delay bound (which we refer to reliable packets) over all data packets received in a decision interval. The results of our study are shown in Fig. 2 for different number of source nodes, i.e, $n = 41, 62, 81, 102$. Note that each of these curves was obtained by varying the event reporting frequency, f, for a randomly chosen event center (X_{ev}, Y_{ev}) and corresponding number of sources, n. These values are tabulated in Table II.

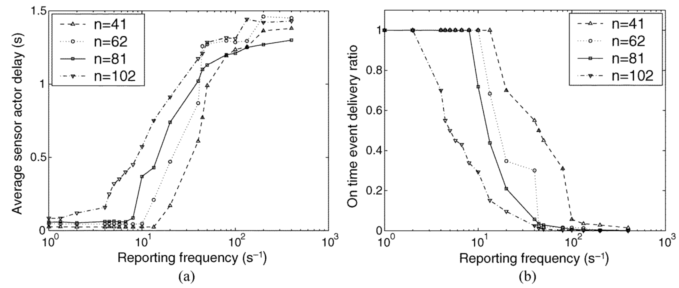


Fig. 2. Effect of varying reporting frequency of source nodes on (a) average sensor–actor delay and (b) on-time event delivery ratio.

TABLE I
ns-2 SIMULATION PARAMETERS

Area of sensor field	200x200 m^2
Number of sensor nodes	200
Radio range of a sensor node	20 m
Packet length	30 bytes
Interface queue (IFQ) length	65 packets
Transmit Power	0.660 W
Receive Power	0.395 W
Doze Power	0.035 W
Decision interval (τ)	1 s

TABLE II
RANDOMLY SELECTED EVENT CENTERS USED IN THE SIMULATIONS

Number of source nodes	**Event center** (X_{ev}, Y_{ev})	**Event radius**
41	(75.2, 72.3)	30m
62	(52.1, 149.3)	30m
81	(59.2, 68.1)	40m
102	(90.6, 119.1)	40m

As shown in Fig. 2(a) and (b), it is observed that as the event reporting frequency, f, increases, average sensor–actor transport delay remains constant and on-time event delivery is ensured, until a certain $f = f_{max}$ at which network congestion is experienced. After this point, the average sensor–actor transport delay starts to increase and on-time event delivery cannot be provided. This is obvious because the increased network load due to higher reporting frequency leads to increase in the buffer occupancy and network channel contention. Moreover, as the number of sources increases, on-time event delivery ratio cannot be provided even at lower reporting frequencies.

To further elaborate the relationship between observed delay-constrained event reliability, DR_i, and the event reporting frequency, f, we have observed the number of packets received at the actor node in a decision interval, τ. We make the following observations from Fig. 3.

i) Until a certain $f = f_{max}$, observed delay-constrained event reliability and no delay-constrained event reliability[2] coincides, beyond which delay-constrained event reliability significantly deviates from no delay-constrained event reliability.

ii) The observed delay-constrained event reliability, DR_i, shows a linear increase (note the log scale) with source

[2]No delay-constrained event reliability represents the number of event packets received at the actor irrespective of their packet delay.

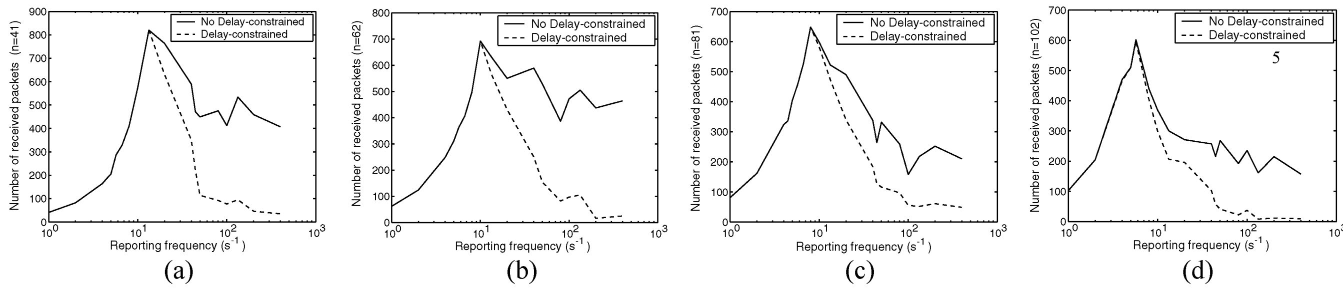


Fig. 3. The number of received packets at the actor node in a decision interval, when the number of sources (a) $n = 41$, (b) $n = 62$, (c) $n = 81$, (d) $n = 102$.

reporting rate, f, until a certain $f = f_{max}$, beyond which the observed delay-constrained event reliability drops. This is because the network is unable to handle the increased injection of data packets and packets are dropped because of congestion.

iii) Such an initial increase and a subsequent decrease in observed delay-constrained event reliability is observed regardless of the number of source nodes, n.

iv) f_{max} decreases with increasing n, i.e., network congestion occurs at lower reporting frequencies with greater number of source nodes.

v) After $f = f_{max}$, delay-constrained event reliability starts to drop significantly due to network congestion. Therefore, an accurate congestion detection mechanism is required to both provide delay-constrained reliability and an effective congestion control in the network.

In summary, with increasing reporting frequency, a general trend of an initial increase and a subsequent decrease (due to network congestion) in delay-constrained event reliability is observed in our preliminary studies, as shown in Fig. 3. Furthermore, when the application-specific delay bounds are considered, the observed delay-constrained event reliability decreases significantly with the network congestion, regardless of the number of source nodes. These observations confirm the urgent need for a delay-constrained reliable event transport solution with an efficient congestion detection and control mechanism in WSANs. In the following section, combined congestion detection mechanism of the (RT)^2 protocol is described in detail.

E. Congestion Detection and Control Mechanism

In WSANs, because of the memory limitations of the sensor nodes and limited capacity of shared wireless medium, congestion might be experienced in the network. Congestion leads to both waste of communication and energy resources of the sensor nodes and also hampers the event detection reliability because of packet losses [1]. Hence, it is mandatory to address the congestion in the sensor field to achieve real-time and reliable event detection and minimize energy consumption. However, the conventional sender-based congestion detection methods for end-to-end congestion control purposes cannot be applied here. The reason lies in the notion of delay-constrained event reliability rather than end-to-end reliability. Only the actor node, and not any of the sensor nodes, can determine the delay-constrained reliability indicator $\delta_i = DR_i/DR^*$, and act accordingly.

In addition, for efficient congestion detection in WSANs, the sensor nodes should be aware of the network channel condition around them, since the communication medium is shared and might be congested with the network traffic among other sensor nodes in the neighborhood [8]. Therefore, because of shared communication medium nature of WSANs, the sensor nodes can experience congestion even if their buffer occupancy is small.

In this regard, the (RT)^2 protocol uses a *combined congestion detection* mechanism based on both average node delay calculation and local buffer level monitoring of the sensor nodes to accurately detect congestion in the network. Note that average node delay at the sensor node gives an idea about the contention around the sensor node, i.e., how busy the surrounding vicinity of the sensor node. To compute the average node delay at the sensor node i, the sensor node takes exponential weighted moving average of the elapsed time. Recall that with the proposed mechanism in Section II-C, the calculation of the average node delay can be performed without globally synchronized clock in the network.

In combined congestion detection mechanism of the (RT)^2 protocol, any sensor node whose buffer overflows due to excessive incoming packets or average node delay is above a certain delay threshold value is said to be congested and it informs the congestion situation to the actor node. More specifically, the actor node is notified by the upcoming congestion condition in the network by utilizing the *Congestion Notification* (CN) bit in the header of the event packet transmitted from sensors to the actor node. Therefore, if the actor node receives event packets whose CN bit is marked, it infers that congestion is experienced in the last decision interval. In conjunction with the delay-constrained reliability indicator, δ_i, the actor node can determine the current network condition and dynamically adjust the reporting frequency of the sensor nodes.

To achieve timely execution of the right action upon the environment, actor–actor ad hoc communication must also be efficiently handled. In this respect, congestion control is also imperative for reliable and timely actor–actor ad hoc communication. Hence, combined congestion mechanism of the (RT)^2 protocol is also utilized for actor–actor ad hoc communication. The details of adaptive rate-based transmission and congestion control algorithms for actor–actor ad hoc communication are explained in Section IV.

III. (RT)^2 Protocol Operation for Sensor–Actor Communication

In this section, we describe the (RT)^2 protocol operation during sensor–actor communication. Recall that in the previous sections, based on the delay-constrained event reliability and the event-to-action delay bound notions, we had defined a new delay-constrained reliability indicator $\delta_i = DR_i/DR^*$, i.e., the ratio of observed and desired delay-constrained event reliabilities. To determine proper event reporting frequency update policies, we also define T_i and T_{sa}, which are the amount of time needed to provide delay-constrained event reliability for a decision interval i and the application-specific sensor–actor communication delay bound, respectively. In conjunction with the congestion notification information (CN bit) and the values of f_i, δ_i, T_i and T_{sa}, the actor node calculates the updated reporting frequency, f_{i+1}, to be broadcast to source nodes in each decision interval. This updating process is repeated until the optimal operating point is found, i.e., adequate reliability and no congestion condition is obtained. In the following sections, we describe the details of the reporting frequency update policies and possible network conditions experienced by the sensor nodes.

A. Early Reliability and No Congestion Condition

In this condition, the required reliability level specific to application is reached before the sensor–actor communication delay bound, i.e., $T_i \leq T_{sa}$, and no congestion is observed in the network, i.e., $CN = 0$. However, the observed delay-constrained event reliability, DR_i, is larger than desired delay-constrained event reliability, DR^*. This is because source nodes transmit event data more frequently than required. The most important consequence of this condition is excessive energy consumption of the sensors. Therefore, the reporting frequency should be decreased cautiously to conserve energy. This reduction should be performed cautiously so that the delay-constrained event reliability is always maintained. Therefore, the actor node decreases the reporting frequency in a controlled manner. Intuitively, we try to find a balance between saving energy and maintaining reliability. Hence, the updated reporting frequency can be expressed as follows:

$$f_{i+1} = f_i \frac{T_i}{T_{sa}}. \tag{2}$$

B. Early Reliability and Congestion Condition

In this condition, the required reliability level specific to application is reached before the sensor–actor communication delay bound, i.e., $T_i < T_{sa}$, and congestion is observed in the network, i.e., $CN = 1$. However, the observed delay-constrained event reliability, DR_i, is larger than the desired delay-constrained event reliability, DR^*. In this situation, the (RT)^2 protocol decreases reporting frequency to avoid congestion and save the limited energy of sensors. This reduction should be in a controlled manner so that the delay-constrained event reliability is always maintained. However, the reporting frequency can be decreased more aggressively than the case where there is no congestion and the observed delay-constrained event reliability, DR_i, is larger than the desired delay-constrained event reliability, DR^*. This is because in this case, we are farther from optimal operating point. Here, we try to avoid congestion as soon as possible. Hence, the updated reporting frequency can be expressed as follows:

$$f_{i+1} = min\left(f_i \frac{T_i}{T_{sa}}, f_i^{(T_i/T_{sa})}\right). \tag{3}$$

C. Low Reliability and No Congestion Condition

In this condition, the required reliability level specific to application is not reached before sensor–actor communication delay bound, i.e., $T_i > T_{sa}$, and no congestion is observed in the network, i.e., $CN = 0$. However, the observed delay-constrained event reliability, DR_i, is lower than the desired delay-constrained event reliability, DR^*. This can be caused by i) packet loss due to wireless link errors, ii) failure of intermediate relaying nodes, iii) inadequate data packets transmitted by source nodes. Packet loss due to wireless link errors might be observed in WSANs due to energy inefficiency of powerful error correction and retransmission techniques. However, regardless of the packet error rate, the total number of packets lost due to link errors is expected to scale proportionally with the reporting frequency, f. Here, we make the assumption that the net effect of channel conditions on packet loss does not deviate significantly in successive decision intervals. This is reasonable with static sensor nodes, slowly time-varying and spatially separated communication channels [1]. Furthermore, when intermediate nodes fail, packets that need to be routed through these nodes are dropped. This can cause a reduction in reliability even if enough number of data packets is transmitted by source nodes. However, fault-tolerant routing/re-routing in WSN is provided by several existing routing algorithms [3]. The (RT)^2 protocol can work with any of these routing schemes. Therefore, to achieve required event reliability, we need to increase the data reporting frequencies of source nodes. Here, we exploit the fact that the DR versus f relationship in the absence of congestion, i.e., for $f < f_{max}$, is linear (see Section II-D). In this regard, we use the multiplicative increase strategy to calculate updated reporting frequency, which is expressed as follows:

$$f_{i+1} = f_i \frac{DR^*}{DR_i}. \tag{4}$$

D. Low Reliability and Congestion Condition

In this condition, the required reliability level specific to application is not reached before sensor–actor communication delay bound, i.e., $T_i > T_{sa}$, and congestion is observed in the network, i.e., $CN = 1$. However, the observed delay-constrained event reliability, DR_i, is lower than the desired delay-constrained event reliability, DR^*. This situation is the worst possible case, since desired delay-constrained event reliability is not reached, network congestion is observed and thus, limited energy of sensors is wasted. Hence, the (RT)^2 protocol aggressively reduces reporting frequency to reach optimal reporting frequency as soon as possible. Therefore, to assure sufficient decrease in the reporting frequency, it is exponentially decreased and the new frequency is expressed by

$$f_{i+1} = f_i^{\frac{DR_i}{(DR^* * k)}} \tag{5}$$

where k denotes the number of successive decision intervals for which the network has remained in the same situation including the current decision interval, i.e., $k \geq 1$. Here, the purpose is to decrease reporting frequency with greater aggression, if a network condition transition is not detected.

E. Adequate Reliability and No Congestion Condition

In this condition, the network is within β tolerance of the optimal operating point, i.e., $f < f_{max}$ and $1-\beta \leq \delta_i \leq 1+\beta$, and no congestion is observed in the network. Hence, the reporting frequency of source nodes is left constant for the next decision interval:

$$f_{i+1} = f_i. \tag{6}$$

Here, our aim is to operate as close to $\delta_i = 1$ as possible, while utilizing minimum network resources and meeting event delay bounds. For practical purposes, we define a tolerance level, β, for optimal operating point. If at the end of decision interval i, the delay-constrained reliability indicator δ_i is within $[1-\beta, 1+\beta]$ and if no congestion is detected in the network, then the network is in (Adequate reliability, No congestion) condition. In this condition, the event is deemed to be reliably and timely detected and the reporting frequency remains unchanged. Thus, a greater proximity to the optimal operating point can be achieved with small β. However, the smaller the β, the greater the convergence time needed to reach corresponding (Adequate reliability, No congestion) condition. Therefore, a good choice of β is the one that balances the tolerance and convergence requirements and hence is mainly dependent on the application-specific requirements in terms of convergence time, the degree of energy conservation, expected lifetime, as well as desired delay-constrained reliability level.

IV. (RT)^2 Protocol for Actor–Actor Communication

In WSANs, as discussed before, after receiving event information, actors need to communicate with each other to make decisions on the most appropriate way to perform the action. Thus, to timely initiate the right actions upon the sensed phenomena, the (RT)^2 protocol also addresses efficient actor–actor communication. In this section, we first describe the main design principles of the (RT)^2 protocol for actor–actor communication. Then, we describe the details of the (RT)^2 protocol operation during actor–actor communication.

A. (RT)^2 Protocol Overview for Actor–Actor Communication

In this section, we make an overview of the key design elements of the (RT)^2 protocol for actor–actor communication:

1) **Cross-layer interactions:** In the current literature on wireless ad hoc networks, some protocols providing an efficient coordination between communication layers are developed to react the network dynamics both accurately and timely [7], [18]. The (RT)^2 protocol also benefits from both cross-layer interactions and intermediate node feedback information to i) capture route failures accurately and timely, ii) get congestion notification and transmission rate feedback for both initial start up phase and steady state phase.
2) **Distinguishing cause of packet loss:** The (RT)^2 protocol distinguishes congestion and non-congestion related losses by the feedback information from both receiver and the intermediate nodes. In this context, the (RT)^2 protocol uses a *combined congestion detection* mechanism based on both the average node delay calculation and the local buffer level monitoring of the actor nodes to accurately detect congestion in the network (see Section II-E). When the actor node is notified about the congestion condition, it decreases the transmission rate accordingly to relieve the congestion as soon as possible.
3) **SACK-based reliability:** To provide reliable actor–actor communication, the (RT)^2 protocol relies upon probe packets to recover from subsequent losses and selective-acknowledgments (SACK) packets to detect any holes in the received data stream. Furthermore, to prevent congestion in the reverse path, SACK packets are delayed in the receiver, i.e., one SACK packet for every d data packets received. Hence, this delayed SACK strategy of (RT)^2 protocol enables the receiver to control the amount of the reverse path traffic accordingly.
4) **Adaptive rate-based transmission:** The (RT)^2 protocol periodically adjusts transmission rate based on bottleneck node information, i.e., congestion notification (CN), packet delay and the number flows passing through the node. Here, the packet delay represents the sum of queuing, channel access time and transmission time at the bottleneck node along the path. Note that we also compute exponential average of packet delays, i.e., D_i, at the intermediate nodes and the receiver to fine tune the fluctuations of the observed delay values, i.e., $Avg(D_i) = \alpha * Avg(D_i) + (1-\alpha) * Current(D_i)$. Moreover, based on the number of flows passing through the same node, a simple fair sharing principle is employed to equally distribute the network resources. Note that (RT)^2 can also work with other service disciplines such as per-flow quality-of-service (QoS) based disciplines, which can further improve the performance and are beyond the scope of the paper. In addition, to meet the application-specific delay bounds, the minimum transmission rate (R_{min}) is also determined according to the remaining time to event-to-action delay bound (see (7)). This way, the data rate is dynamically adjusted based on both the current conditions of the data path and event-to-action delay bounds.
5) **Flow control:** The (RT)^2 protocol performs flow control by observing the application processing rate R_p, which represents the application reading rate from the receiver buffer. Here, our objective is to limit the amount of data transmitted by the sender to a certain rate that the receiver can manage. In this regard, if R_p is smaller than the rate feedback R_f provided by intermediate nodes, the receiver sends R_p to the sender as a rate feedback. Thus, (RT)^2 also

provides flow control at the receiver while dynamically adjusting transmission rate.

B. $(RT)^2$ Protocol Operation for Actor–Actor Communication

In this section, we describe the protocol operation of $(RT)^2$ during actor–actor communication. The protocol operation is composed of two main states: i) start-up state, ii) steady state. In the following paragraphs, the operations at each state is described in detail.

1) **Start-Up State:** When establishing new connection between sender and receiver, the sender transports a probe packet towards the receiver to capture the available transmission rate quickly. Each intermediate node between the sender and receiver intercepts the probe packet and updates the bottleneck delay field of the probe packet, if the current value of delay information is higher than that of the intermediate node. Initially, the delay value of probe packet is assigned to zero. Therefore, after one round-trip-time, the sender gets estimated rate feedback from the receiver, which results in quick convergence to available transmission rate. Furthermore, this probing mechanism of start up phase is also applied after route changes.

2) **Steady State:** This state consists of four substates: i) Increase; ii) Decrease; iii) Hold; and iv) Probe. In the following, we describe the $(RT)^2$ protocol operations in each substate:

   i) **Increase:** In this state, the sender increases its transmission rate according to the feedback coming from the receiver. Once an increase decision for sender transmission rate is taken, only m fraction of the difference between transmission rate feedback (R_f) and sender current transmission rate (R_c) is performed. The appropriate fraction value (m) for the transmission rate increase is obtained as follows: If the hop count along the data path is greater than or equal to 4 for that connection, m is set to 4. Otherwise, if the hop count is less than 4, then m is set to the actual hop count value along the path. The inherent spatial reuse property of underlying CSMA/CA based MAC protocol requires this normalization in transmission rate. The details can be found in [5], [9]. Note also that to prevent fluctuations, transmission rate is only increased when a certain threshold (Δ_{rate}) is exceeded.

   ii) **Decrease:** In this state, the sender reduces its transmission rate according to the feedback coming from the receiver. Note that the transmission rate is decreased until the minimum transmission rate (R_{min}) is reached. R_{min} represents the minimum transmission rate requirement to transfer a certain amount of data within event-to-action delay bound. R_{min} can be calculated as follows:

$$R_{min} = \frac{B}{\Delta_{re2a}} \quad (7)$$

   where B represents the amount of packets that should be transmitted to the actor and Δ_{re2a} is remaining event-to-action deadline, which is the residual time of event-to-action delay bound Δ_{e2a} (see Section II-C), after the sensor–actor communication is performed.

   iii) **Hold:** In this state, the required transmission rate is reached. Sender does not change the transmission rate unless route failure or congestion occurs in the network.

   iv) **Probe:** In this state, the sender sends a probe packet to the receiver so as to monitor the available transmission rate in the network as in start up phase. This phase might occur due to route errors (RERR), which is common in ad hoc communication networks. When the route error is observed, i.e., RERR information is received from intermediate nodes, sender freezes its transmission and periodically starts to send the probe packet to get transmission rate feedback from the receiver.

Overall, the $(RT)^2$ protocol dynamically shapes data traffic based on both delay bounds and the current conditions of the network. Note that, in the protocol operation, the sender adjusts its transmission rate in response to the rate feedbacks from the receiver, which are sent with the period of T_{fdbk}. To prevent the sender from over-flooding the network in case all the feedback packets from the receiver are lost, the $(RT)^2$ protocol also performs a multiplicative decrease of transmission rate for each feedback periods, in which the sender does not receive feedback from the receiver up to a maximum of two feedback periods. After the second feedback period, if the sender still does not receive any feedback packet, it enters into probe state so as to monitor the available transmission rate in the network. In this respect, the periods of feedback (T_{fdbk}) and probe packets (T_p) should be larger than one round-trip-time (RTT) and small enough to capture the network dynamics.

For this purpose, the period of feedback packets (T_{fdbk}) and probe packets (T_p) are selected as $2 * RTT$. Note also that if the receiver rate feedback changes more than a certain threshold (Δ_{fdbk}), then the receiver immediately sends the rate feedback information to the sender without waiting for a feedback timer timeout event. Thus, the sender can adjust the transmission rate accordingly even for long RTT values. The algorithm of $(RT)^2$ for actor–actor communication is given in Fig. 4.

Note that actor–actor communication in WSANs is similar to the communication paradigm of ad hoc networks due to the small number of resource-rich actor nodes being loosely deployed. In the related literature, there are several transport protocols dealing with ad hoc networks [4]. In general, these solutions are either window-based [7] or rate-based protocols [13]. Although these solutions may improve TCP performance to a certain extent, they do not address the unique requirements of WSANs completely. In Table III, we summarize the main differences between $(RT)^2$ protocol and the previously developed ad hoc transport protocols [7], [13]. To evaluate the performance of $(RT)^2$ during actor–actor communication, we also compare $(RT)^2$ with these ad hoc transport solutions in the following section.

V. $(RT)^2$ Performance Evaluation

Here, we present the performance evaluation of the $(RT)^2$ protocol. In Section V-A, we report the performance results for

TABLE III
SUMMARY OF DIFFERENCES

Issue	TCP	TCP-ELFN[7]	ATP[13]	(RT)^2
Transmission type	Window-based	Window-based	Rate-based	Rate-based
Explicit congestion detection	Not addressed	Not addressed	Not addressed	Combined congestion detection
Rate feedback	Not addressed	Not addressed	Periodic with constant period	Periodic with dynamic period
Timing constraints	Not addressed	Not addressed	Not addressed	Addressed with R_{min}
Route failures	Packet Loss	ELFN feedback	Intermediate node feedback	Intermediate node feedback

(RT)^2()
Sender:
/*Transmission Rate Update Procedure*/
$packet_{fdbk}$ received with R_f
If (CONGESTION)
  /*Decrease transmission rate*/
  $R_c = \frac{R_c}{2}$
If (NO CONGESTION)
  If ($HopCount$ > 4)
    m=4
  else
    m=$HopCount$
  If ($\frac{R_f - R_c}{R_c}$ > Δ_{rate})
    R_c=R_c + $\frac{R_f - R_c}{m}$
  else If (R_c > R_f)
    R_c=max(R_f, R_{min})
else
  $Hold\ R_c$
Intermediate Node:
/*Calculate D_i for individual packet*/
$Avg(D_i)$=$\alpha * Avg(D_i)$+$(1-\alpha) * Current(D_i)$
If ($Avg(D_i)$ > D_{th})
  stamp D=$Avg(D_i)$
Receiver:
$Avg(D)$=$\alpha * Avg(D)$+$(1-\alpha) * Current(D)$
When T_{fdbk} expires or Δ_{fdbk} exceeded
  /*Compute rate feedback*/
  $R_f = \frac{1}{Avg(D)}$
  /*Flow control at the receiver*/
  If (R_f > R_p)
    $R_f = R_p$
  stamp R_f on $packet_{fdbk}$
  send $packet_{fdbk}$ to sender

Fig. 4. The (RT)^2 protocol for actor–actor communication.

the sensor–actor communication, while in Section V-B, we discuss the performance results for the actor–actor communication.

A. Sensor–Actor Communication

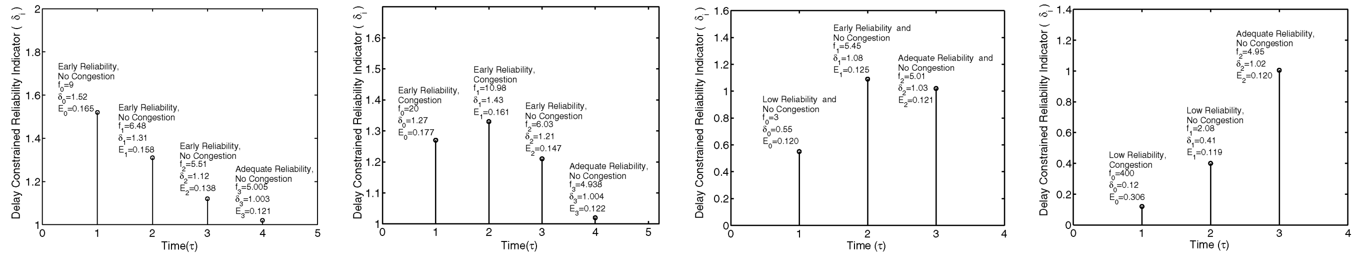
To evaluate the performance of the (RT)^2 protocol during sensor–actor communication, we developed an evaluation environment using *ns*-2 [14]. For sensor–actor communication scenario, the number of sources, sensor–actor delay bound and tolerance level were selected as $n = 81$, 1 s and $\epsilon = 5\%$, respectively. The event radius was fixed at 40 m. We run 10 experiments for each simulation configuration. Each data point on the graphs is averaged over 10 simulation runs. We use the same sensor node and simulation configurations provided in Table I in Section II-D.

Moreover, in this simulation scenario, the actor nodes, which receive data packets from sensors, stop their movements once they start to receive data. This way, the possible packet losses and extensive message exchange due to the associated actor node movement are avoided. Thus, the limited energy resources of the sensors are saved. Note that the other actor nodes, which can involve the action but do not receive data from sensors, may continue their mobility and the impacts of the actor mobility on network performance are investigated in Section V-B in detail. For sensor–actor communication case, the main performance metrics that we use to measure the performance of (RT)^2 protocol are the convergence time to (Adequate reliability, No congestion) condition from any other initial network conditions and average energy consumption per packet (E_i) for each decision interval i.

The (RT)^2 protocol convergence results are shown in Fig. 5 for different initial network conditions. As observed in Fig. 5, (RT)^2 protocol converges to (Adequate reliability, No congestion) condition starting from any of the other initial network conditions discussed in Section III. Thus, (RT)^2 is self-configuring and can perform efficiently under random, dynamic topology frequently encountered in WSAN applications. Moreover, the average energy consumed per packet during sensor–actor communication, i.e., (E_i), is also observed. As shown in Fig. 5, E_i decreases as the (No congestion, Adequate reliability) state is approached which shows that energy consumption of the sensor nodes is also decreased while providing reliability constraints and delay bounds. Due to energy limitations of sensors, this result is also important for the proper operation of WSAN. Performance of reporting frequency update policies for sensor–actor communication are given as the trace values and states listed within Fig. 5.

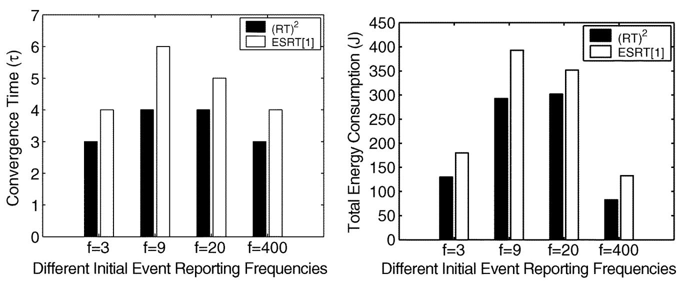
To further investigate (RT)^2 protocol convergence results, we have compared (RT)^2 protocol and event-to-sink reliable transport (ESRT) [1] protocol in terms of convergence time to the (Adequate reliability, No congestion) condition and total energy consumption. The reason for comparison with ESRT is that both of them is based on event transport reliability notion unlike the other transport layer protocols addressing conventional end-to-end reliability in WSNs. As shown in Fig. 6, the convergence time and total energy consumption of the (RT)^2 protocol are much smaller than those of ESRT for different initial network conditions. This is because ESRT does not consider application-specific delay bounds while avoiding network congestion and adjusting reporting rate of sensor nodes.

To elaborate the relationship between the event-to-action delay notion and the (RT)^2 protocol operation, in Fig. 7, we have also observed the delay distributions of the event packets received at the actor node, when there is a transition from (Low reliability, Congestion) condition to (Adequate reliability, No congestion) condition. As seen in Fig. 7, when the (Adequate reliability, No congestion) condition is approached, the delay of the event data packets also decreases. This is because the (RT)^2



(a) (b) (c) (d)

Fig. 5. $(\mathrm{RT})^2$ trace for (a) early reliability and no congestion, (b) early reliability and congestion, (c) low reliability and no congestion, and (d) low reliability and congestion.



(a) (b)

Fig. 6. Comparison of $(\mathrm{RT})^2$ and ESRT[1] for sensor–actor communication in terms of (a) convergence times to (Adequate reliability, No congestion) condition, and (b) total energy consumption.

protocol takes event-to-action delay bounds into account, while adjusting reporting rate of sensor nodes and avoiding network congestion.

B. Actor–Actor Communication

In this section, we present the performance results of the $(\mathrm{RT})^2$ protocol during actor–actor communication. For the simulations, we set up an evaluation environment using *ns*-2 [14]. The simulations for this scenario are performed for a 200 m $\times$ 200 m field with 10 actor nodes, distributed randomly over the field. In addition, to take into account the mobility of the actors during actor–actor communication, we have used the random way-point model. In this mobility model, we consider maximum speeds of 1 m/s, 5 m/s, 10 m/s, 15 m/s and 20 m/s for mobile actor nodes. The packets are 1000 bytes. Other simulation parameters are the same as those listed in Table I in Section II-D.

For actor–actor communication scenario, the performance of the $(\mathrm{RT})^2$ protocol is evaluated and compared against TCP-NewReno, TCP-ELFN [7] and ATP [13]. The main performance metrics that we employ to measure the performance of the $(\mathrm{RT})^2$ protocol are aggregate throughput, and average packet delay. Here, the aggregate throughput reflects the number of packets successfully received at the destination. By average packet delay, we refer to average latency of data packets during actor–actor communication. All the simulations last for 1000 s. We run 10 experiments for each simulation configuration and each data point on the graphs is averaged over 10 simulation runs.

1) Aggregate Throughput: In Fig. 8, we present the aggregate throughput results of the $(\mathrm{RT})^2$ protocol and other ad hoc transport protocols, i.e., TCP-NewReno, TCP-ELFN [7] and ATP [13]. Here, different number of flow connections are used and source-destination pairs are randomly chosen from 10 actor nodes. In terms of aggregate throughput, the $(\mathrm{RT})^2$ protocol outperforms other transport protocols under comparison, since $(\mathrm{RT})^2$ dynamically shapes data traffic according to the channel condition and intermediate node feedbacks. In addition, proper reaction of $(\mathrm{RT})^2$ to congestion and non-congestion related losses, such as route failures, avoids any performance degradation during actor–actor communication. For example, for 5 flow connection and 10 m/s speed, we obtain that the aggregate throughput achieved by $(\mathrm{RT})^2$ during actor–actor communication is around 40%, 30% and 15% higher than that of TCP-NewReno, TCP-ELFN and ATP, respectively. Note also that rate-based transport protocols, i.e., $(\mathrm{RT})^2$ and ATP, outperform window-based transport protocols, i.e., TCP-ELFN and TCP-NewReno, mainly because rate-based schemes capture the available bandwidth more quickly compared to window-based schemes.

2) Average Delay: In Fig. 9, we also show the average packet delay results of the $(\mathrm{RT})^2$ and the other transport protocols. As shown in Fig. 9, for all simulation configurations, the average packet delay values of $(\mathrm{RT})^2$ are much lower than those of other protocols, since $(\mathrm{RT})^2$ captures the available bandwidth in the network quickly and does not allow a burst of packet transmissions with explicit congestion notification and rate feedback based mechanisms. For example, for 10 flow connection and 15 m/s speed, the average packet delays achieved by $(\mathrm{RT})^2$ are approximately eight, seven and five times lower than that of TCP-NewReno, TCP-ELFN and ATP, respectively. This is so crucial because of timely event detection and action performance objectives of the WSANs.

Note that, in these experiments, we do not assume that the underlying layer protocols, i.e., network, MAC, and physical layer protocols, provide any additional support for meeting application-specific real-time delay requirements. Intuitively, we anticipate that the performance of $(\mathrm{RT})^2$ protocol further improves, when deployed on top of lower layer communication protocols, which also provide real-time support. The evaluation of such scenario is left as a future study mainly due to lack of space.

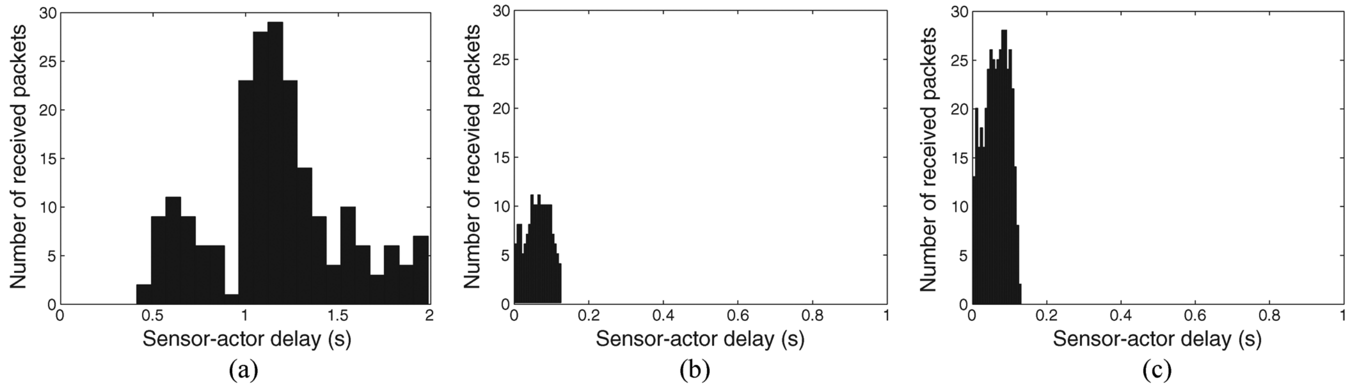


Fig. 7. Packet delay distribution in (a) (Low reliability, Congestion), (b) (Low reliability, No congestion), and (c) (Adequate reliability, No congestion) conditions.

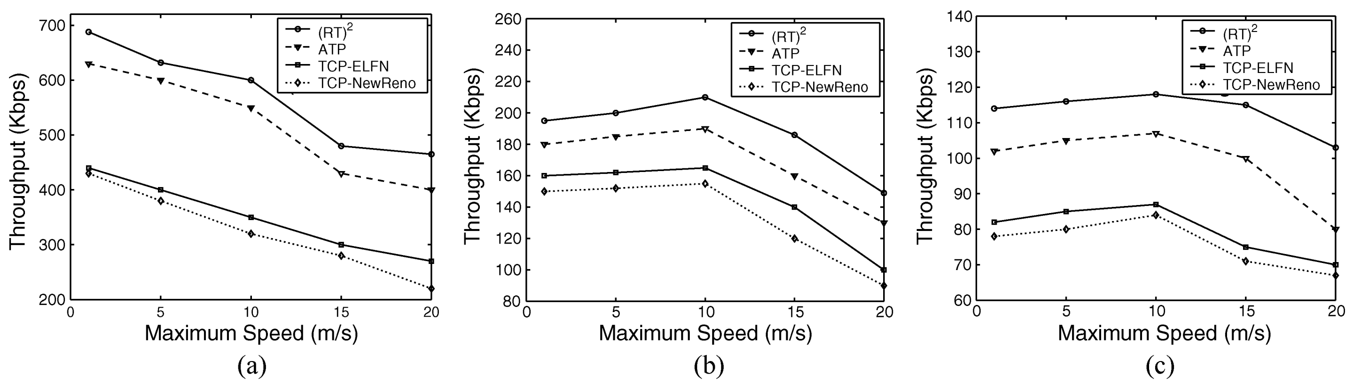


Fig. 8. Aggregate throughput for (a) 1 flow connection, (b) 5 flow connection, and (c) 10 flow connection, when the maximum speed of the actors are varying.

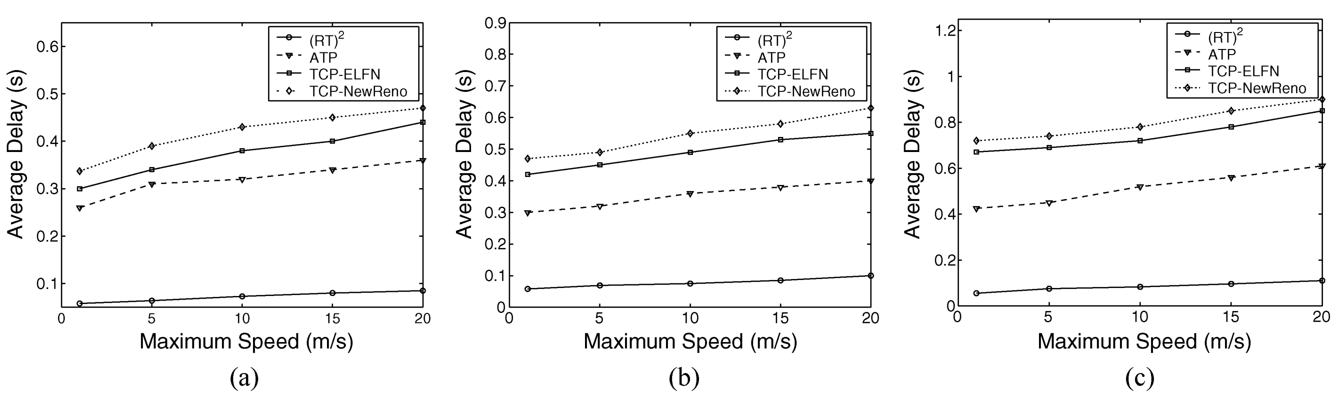


Fig. 9. Average packet delay for (a) 1 flow connection, (b) 5 flow connection, and (c) 10 flow connection, when the maximum speed of the actors are varying.

VI. Conclusion

To address the communication challenges introduced by the coexistence of sensors and actors in WSANs, a Real-Time and Reliable Transport $(\mathrm{RT})^2$ protocol for WSANs is presented in this paper. $(\mathrm{RT})^2$ dynamically adjusts its protocol configurations to adapt to heterogeneous characteristics of WSANs. It also enables the applications to perform right actions timely by exploiting both the correlation and the collaborative nature of WSANs. The objective of $(\mathrm{RT})^2$ is to reliably and collaboratively transport event features from the sensor field to the actor nodes and to timely react to sensor information with a right action. In this respect, the $(\mathrm{RT})^2$ protocol simultaneously addresses congestion control and timely event transport reliability objectives in WSANs. To the best of our knowledge, this is the first research effort focusing on real-time and reliable event transport and action performance objectives of WSANs. Performance evaluation via simulation experiments show that $(\mathrm{RT})^2$ achieves high performance in terms of reliable event detection, communication latency and energy consumption in WSANs.

References

[1] O. B. Akan and I. F. Akyildiz, "ESRT: Event-to-sink reliable transport for wireless sensor networks," *IEEE/ACM Trans. Networking*, vol. 13, no. 5, pp. 1003–1016, Oct. 2005.

[2] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad Hoc Netw. J.*, vol. 2, no. 4, pp. 351–367, Oct. 2004.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw. J.*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[4] I. Chlamtac, M. Conti, and J. Liu, "Mobile ad hoc networking: Imperatives and challenges," *Ad Hoc Netw. J.*, vol. 1, no. 1, pp. 13–64, 2003.

[5] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 1744–1753.

[6] V. C. Gungor, M. C. Vuran, and O. B. Akan, "On the cross-layer interactions between congestion and contention in wireless sensor and actor networks," *Ad Hoc Netw. J. (Elsevier)*, vol. 5, no. 6, pp. 897–909, Aug. 2007.

[7] G. Holland and N. H. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," presented at the ACM MOBICOM, Seattle, WA, 1999.

[8] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in sensor networks," presented at the ACM SENSYS, Baltimore, MD, Nov. 2004.

[9] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless network," presented at the ACM MOBICOM, Rome, Italy, Jul. 2001.

[10] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 7, pp. 1300–1315, Jul. 2001.

[11] MICA Motes. [Online]. Available: http://www.xbow.com/Products/Wireless_Sensor_Networks.htm

[12] M. Schwartz, *Mobile Wireless Communications*. New York: Cambridge Univ. Press, 2004.

[13] K. Sundaresan, V. Anantharaman, H. Y. Hsieh, and R. Sivakumar, "ATP: A reliable transport protocol for ad hoc networks," presented at the ACM MOBIHOC, Annapolis, MD, Jun. 2003.

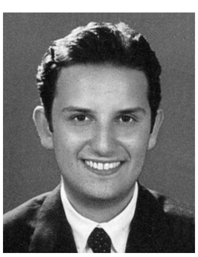
[14] *ns*-2, The Network Simulator. [Online]. Available: http://www.isi.edu/nsnam/ns/index.html

[15] M. C. Vuran, O. B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: Theory and applications wireless sensor networks," *Comput. Netw. J.*, vol. 45, no. 3, pp. 245–259, Jun. 2004.

[16] C. Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion detection and avoidance in sensor networks," presented at the ACM SENSYS, Los Angeles, CA, Nov. 2003.

[17] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy, "Pump-slowly, fetch-quickly (PSFQ): A reliable transport protocol for sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 862–872, Apr. 2005.

[18] X. Yu, "Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness," presented at the ACM MOBICOM, Philadelphia, PA, Sep. 2004.



Vehbi Cagri Gungor (S'06–M'07) received the B.Sc. and M.Sc. degrees in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2001 and 2003, respectively. He received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, in 2007 after working at the Broadband and Wireless Networking Laboratory (BWN-Lab).

Currently, he is a Communication Systems Architect at Eaton Corporation. His current research interests are in wireless ad hoc and sensor networks, wireless mesh networks, WiMAX, IP networks, and electric utility automation.

Özgür B. Akan (M'00-SM'07) received the B.Sc. and M.Sc. degrees in electrical and electronics engineering from Bilkent University and Middle East Technical University, Ankara, Turkey, in 1999 and 2001, respectively. He received the Ph.D. degree in electrical and computer engineering from the Broadband and Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, in 2004.

He is currently an Associate Professor with the Department of Electrical and Electronics Engineering, Middle East Technical University. His current research interests include sensor networks, wireless communications, and bio-inspired communications.

Dr. Akan serves as an Associate Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, an Editor for *ACM-Springer Wireless Networks Journal*, Area Editor for *Ad Hoc Networks Journal (Elsevier)*, and participates in the organization of many international conferences.

Ian F. Akyildiz (M'86–SM'89–F'96) received the B.S., M.S., and Ph.D. degrees in computer engineering from the University of Erlangen-Nuernberg, Germany, in 1978, 1981, and 1984, respectively.

Currently, he is the Ken Byers Distinguished Chair Professor of the School of Electrical and Computer Engineering, and Director of the Broadband and Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, GA. His current research interests are in next generation wireless networks, sensor networks, and wireless mesh networks.

Dr. Akyildiz is the Editor-in-Chief of *Computer Networks* (Elsevier), and the founding Editor-in-Chief of *Ad Hoc Networks Journal* (Elsevier). He has been a Fellow of the ACM since 1996. He received the Don Federico Santa Maria Medal for his services to the Universidad of Federico Santa Maria in Chile in 1986. He served as a National Lecturer for ACM from 1989 until 1998 and received the ACM Outstanding Distinguished Lecturer Award for 1994. He received the 1997 IEEE Leonard G. Abraham Prize award (IEEE Communications Society) for his paper entitled "Multimedia Group Synchronization Protocols for Integrated Services Architectures" published in the IEEE JOURNAL OF SELECTED AREAS IN COMMUNICATIONS (JSAC) in January 1996. He received the 2002 IEEE Harry M. Goode Memorial Award (IEEE Computer Society) with the citation "for significant and pioneering contributions to advanced architectures and protocols for wireless and satellite networking." He received the 2003 IEEE Best Tutorial Award (IEEE Communications Society) for his paper entitled "A survey on sensor networks," published in *IEEE Communication Magazine* in August 2002. He also received the 2003 ACM Sigmobile Outstanding Contribution Award with the citation "for pioneering contributions in the area of mobility and resource management for wireless communication networks."