JUDISHARE: Judicious Resource Allocation for QoS-based Services in Shared Wireless Sensor Networks

Victor Cionca, Ramona Marfievici, Roland Katona, Dirk Pesch
Nimbus Research Centre, Cork Institute of Technology, Cork, Ireland
{firstname.lastname}@cit.ie, Roland.Katona@mycit.ie

***Abstract*—In shared wireless sensor networks (WSNs), multiple users request access to sensing resources, often with varying sampling rates and QoS requirements. To accommodate a request, appropriate sensing, computing and communication resources need to be allocated across the network. Traditionally, each request is mapped to a dedicated set of resources, even when the requests are similar. The need to reduce resource usage has led to data virtualization techniques that focus primarily on merging the requests ignoring the QoS requirements. In this paper, we present a QoS-aware resource allocation approach, JUDISHARE, that *merges* requests, where possible, if they are compatible in their requirements, providing judicious reuse of both sensing and communication resources through a mixture of data virtualization and Virtual Network Embedding (VNE). We show that JUDISHARE respects QoS requirements and reduces resource usage to up to 60%. This, in turn, allows up to 50% more requests to be accommodated onto the network, even when the network resources are fully utilized.**

I. INTRODUCTION

Shared Wireless Sensor Networks (WSNs) [1] enable multiple applications or requests to be concurrently accommodated on the network, often with differing quality of service (QoS) requirements (e.g., reliability, latency). Compared to single-purpose deployments, they provide increased flexibility and can reduce redundancy in deployed resources.

Supporting multiple QoS aware requests on the network is equivalent with building flows with constraints to satisfy the QoS requirements. Here a flow is an end-to-end communication path from sensor source to sink. The optimal allocation of flows is commonly performed by a Virtual Network Embedding (VNE) algorithm [2], whose objective is to maximise the number of accepted requests while minimising resource cost. One problem is that VNE handles all the requests distinctly even when they target the same data source, and allocates distinct flows and resources. Because the communication capacity of the WSN is limited by the capacity of the sink, the number of requests that can be accepted is also limited [3]. If similar requests could be *merged* to use the same flow, it would reduce resource usage. Data virtualization can achieve this [4], but does not consider QoS. What we propose in this paper is an algorithm that *judiciously* uses the available network resources and respects QoS constraints, by *merging* requests not only based on their requested data source but also on the QoS requirements. Where possible, new requests will be accommodated onto existing flows by carefully modifying the flow properties so as to not violate QoS requirements. The goal is to reduce the resources needed to support a set of requests. This frees up network capacity so that additional requests can be accepted, which may lead to increased revenue [3].

Motivating scenario. The motivation for the work described here emerges from our ongoing project Service-centric networks for URban Feedback systems (SURF) [5]. One of the project's goals is to devise an approach for multi-purpose, multi-user, WSNs that can concurrently run multiple sensing applications with varying QoS requirements while optimizing resource utilization and network lifetime. To give an example, consider an environment monitoring WSN deployed throughout a city. Researchers interested in the relationship between disease and urban environment need a sampling rate of the order of days with best effort communication; an environmental agency interested in monitoring the air quality demands a sampling rate of the order of hours and high reliability; the city council interested in the link between traffic and air pollutants runs with periods of minutes with high reliability, while in case of a sudden air contamination event, the city council can ask for a change in sampling rate to tens of seconds with high reliability and low latency; finally, a private company offering services for selecting the least polluted route wants to run a service with sampling in the tens of seconds with high reliability and low latency. These requests and their QoS requirements have to be accommodated by the deployed WSN.

Contribution. We propose JUDISHARE, the first approach, to the best of our knowledge, that performs QoS-aware data virtualization, merging not only the sensing data, but also the communication flows of multiple compatible data requests. We demonstrate through simulation that JUDISHARE reduces the utilization of network resources and can accept more requests than if only application embedding is performed. Furthermore, we conduct experiments that show that when merging with JUDISHARE, the end-to-end reliability requirements of WSN applications can be maintained at rates as high as 100pkt/s.

The remainder of the paper is organized as follows. Section II surveys the related work. We present our approach in Section III, and its performance evaluation in Section IV. We conclude in Section V.

II. Related Work

Several works have focused on solutions for efficient sharing of WSNs as a means of reducing network traffic and energy consumption [1], [4], [6]–[8]. TinyDB [6] is a query processing system where requests are submitted to the sink as SQL queries. To reduce the power consumption, TinyDB performs query optimization at the sink as well as on the node. TinyDB addresses single queries (sequential query handling). Müller et. al. [1] consider a limited form of multi-user sharing where different users request data at different data rates from different sensors. The proposed system merges user queries into a network query to be executed by the nodes. A chain of operators process the flow of tuples from the query while adapting the data rate and filtering out tuples and attributes not requested by the user queries. By using the greatest common divisor of the sampling periods associated with user queries, the network operates more efficiently while still providing all requested data. The two-tier multiple query optimization system from [7], rewrites a set of queries into an optimized set at the first tier and shares sensor readings among similar queries over time and space at the second tier. A multi-query optimization approach is proposed by [4] but only for queries requesting the same packet rate from the sensor nodes. TaskCruncher [8] builds an interval-coverage graph for overlapping sampling requirements and models tasks as data flow graphs enabling the dynamic optimization of the graph structure upon tasks entering or leaving the system. All these works focus on efficient sharing of resources while minimising network traffic and improving energy consumption, but do not consider QoS requirements. JudiShare is an approach to preserve the various QoS requirements while sharing resources efficiently.

III. JudiShare Approach

The WSNs that we consider for JudiShare are large networks with heterogeneous sensing capabilities that concurrently support multiple QoS-aware data requests for distinct data sources. Users submit data requests that specify the data source as a node location or address, with additional requirements for sensor sampling rate and QoS, expressed in terms of reliability and latency. In response, sensing and computation resources are reserved on the source nodes and a communication flow consisting of communication links is allocated as a path between the data source and the network sink, such that the requested requirements are respected.

Setting up and maintaining concurrent data flows with end-to-end QoS requirements requires a global view of the network state to make optimal network-wide decisions. As such, we adopt a centralised network control model, where the controller maintains a complete model of the network state, i.e., the connectivity map and link quality between nodes, using Smog [9], a solution developed in our group. Based on the network state, the controller decides which and how many user requests are accepted on the network, and which resources are used.

JudiShare works only with communication resources, which must be derived from the user requirements. First it determines the requested data rate, based on the given sampling rate and a known application payload size. The data rate is then converted into *communication quota*, which represents a percentage of the maximum data rate achievable on a link. Considering the use of a time-slotted deterministic MAC (Medium Access Control) protocol[1], the maximum link data rate (which is the 100% communication quota) can be derived from the maximum data transmissible in a slot. After extracting the requested communication quota for each user request, JudiShare will try to merge new user requests onto others that are already in the network and have similar demands. To reserve resources in the network in an optimal, QoS constrained, manner, JudiShare makes use of a Virtual Network Embedding (VNE) algorithm, which is discussed next.

A. Virtual Network Embedding

Virtual Network Embedding (VNE) is an algorithm for resource allocation in virtualized enterprise networks [2], that was adapted in our previous work [3] to WSNs. It is used here to map the JudiShare user requests to network resources. The input user requests are represented as a set of source and destination node pairs[2] in the network topology, with associated node and communication requirements such as location, bandwidth or QoS as problem constraints. The network topology has computing and link capacity constraints that are obtained with the Smog mechanism [9] discussed above. The VNE algorithm determines, for each input request, a communication flow (i.e., network path) between the source and destination nodes that satisfies the problem constraints. The objective of the VNE is to maximise the number of requests admitted in the network while minimising the amount of resources allocated. Note that the algorithm only provides a solution to network resources (as in layer three routing flows), but not link resources (as in layer two Medium Access Control). However, the use of the upper-bounded communication quota ensures a feasible allocation at the link layer. This greatly reduces the complexity of the VNE problem from a joint routing and scheduling problem to a constrained multi-commodity routing problem.

In this work the VNE algorithm uses offline request processing, which means that requests are received and processed in batches. It is implemented with a greedy heuristic similar to that in [11], where the input requests are first sorted in non-decreasing order of their requested communication quota, then, in turn, embedded on the network using a shortest path algorithm. The latter minimises the number of hops allocated subject to a maximum bound on link capacity (100% quota) and minimum bound on end-to-end reliability (specified by the user). The link capacity constraint must take into account and prevent internal radio interference which occurs naturally in a WSN but does not occur in the traditional wired VNE. To this extent, allocating communication quota over a link also affects all neighbouring links that interfere with it. So

[1] A deterministic protocol is required to achieve QoS requirements [10].

[2] The source is the sensing source specified in the user request. The destination is the network sink.

instead of having 100% quota *per link* as in wired VNE, WSN VNE allocates a maximum of 100% quota for the entire interference neighbourhood of a link. This has two effects. First, it increases the complexity of the problem, compared to wired VNE. Second, it greatly reduces the total amount of communication capacity that a network can accommodate. However, because JUDISHARE merges requests, it manages to overcome this limitation.

B. Judicious Resource Allocation

Concept. The benefits of managing a multi-tenant WSN infrastructure as described in this paper depend on the number of user requests serviced and the amount of network resources used to service those requests. Therefore the objective is to maximise the user request admission ratio while minimising the amount of allocated network resources, specifically, communication quota. The VNE algorithm considers that all user requests are distinct. Even when requests indicate the same sensing source, the resulting flows start and end at common nodes, but the communication resources allocated are distinct. Each flow has its own, private, quota on the allocated network links. If two flows intersect and share links the amount of quota that must be reserved on those shared links is the sum of the quotas requested by each flow. However, if the QoS requirements of the user requests permit, multiple requests could all make use of the same communication resources. Consider two requests, U_1 and U_2, for a sensor on a node that is one hop away from the sink. The requested packet rates are $P_1 = 2pkt/s$ and $P_2 = 1pkt/s$ and the minimum reliability $R_1 = R_2$. Assuming an application payload of $20Bps$ and a maximum link rate of $100Bps$ (hypothetical example) the quota requested is $Q_1 = 40\%$ and $Q_2 = 20\%$. The VNE algorithm would allocate two flows over the link, one with 40% quota, one with 20% quota, so a total of 60% quota that corresponds to $3P_2$ packet rate. Instead of allocating total resources corresponding to $3P_2$ packet rate at reliability R_1, we observe that both user requests can instead be serviced by a single flow with packet rate $2P_2$ and reliability R_1. At the sink, the data for U_2 can be obtained by down-sampling the incoming flow and taking every second sample. So instead of allocating 60% quota only 40% would be allocated. This saves 20% network quota on that link that could be used to accommodate a third user request, and therefore increase revenue.

As already discussed in Section II, merging data requests to reduce resource consumption has been explored before in the context of data virtualization in [7]. However, this body of work does not consider the QoS requirements of the user requests. If user requests specify QoS requirements there are situations when they cannot be merged. If in the above example U_2 that requests $1pkt/s$ needs higher reliability than U_1, it cannot be serviced anymore by the flow of U_1 because that might lose more packets than allowed by U_2. In this case the two user requests cannot be merged directly in the common data virtualization manner. However with JUDISHARE we show that there are ways of merging flows that can satisfy QoS requirements.

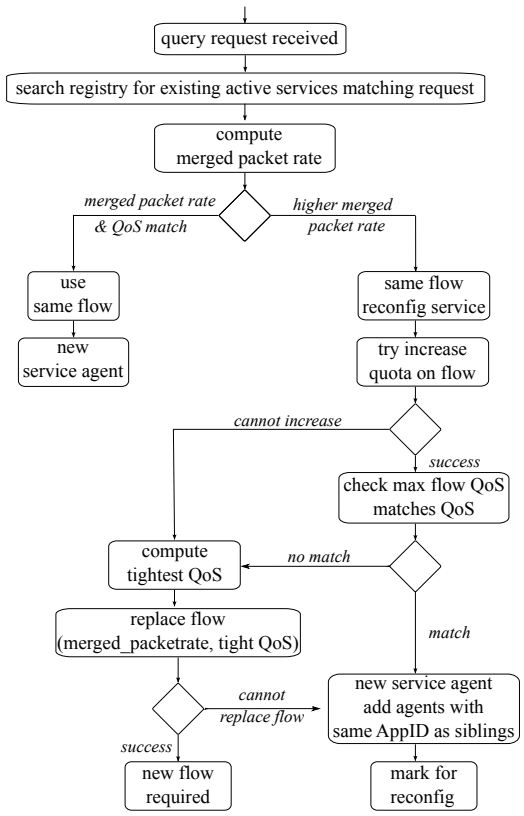

Fig. 1: JUDISHARE approach.

Operation. JUDISHARE works with a VNE algorithm and its goal is to reduce, by merging, the number of input user requests that the VNE considers, resulting in the allocation of fewer communication flows in the network, and therefore in reduced resource consumption. The algorithm is executed when the central controller receives a request. If there are no flows in the network from the requested source node, the new flow is embedded using the VNE algorithm. Otherwise JUDISHARE processes the request, following the flow-chart in Fig. 1.

JUDISHARE tries to merge new user requests into existing flows, by merging the packet rate and the QoS of the two. Packet rates, expressed as packets per time, are merged by computing their Least Common Multiple (LCM), which is the inverse of the method employed in [1], where packet *periods* and their Greatest Common Divisor are used. If the two packet rates are co-primes, the merged packet rate would equal the product of the rates. However, the maximum that VNE allocates is the *sum* of the rates, so we set that as the maximum value. If the merged rate is greater than the sum of the rates JUDISHARE will treat the two requests as distinct and allocate them individually using VNE. Using LCM to find the merged packet rate guarantees that the merged flow can be used to serve all user requests associated with it, from a packet rate point of view. The QoS is expressed as end-to-end reliability (i.e., packet delivery ratio PDR) and latency. An existing flow A can support a new request B's QoS requirements if the reliability of A is higher than that of B and the latency of A is lower than that of B. Similarly, two QoS values can be merged, creating

a *tight* QoS, with the highest reliability and lowest latency.

We consider three cases. i) In the simplest case, the merged packet rate is lower than the existing one, QoS is supported, and the new request can be merged onto the existing flow without making any changes. ii) If the merged packet rate is higher than the existing one, the algorithm tries to increase the rate of the existing flow. It does this for all links of the flow as well as their interfering links. If the increase does not violate the maximum link capacity on all affected links and the QoS of the existing flow supports the requested QoS, the two can be merged. This case relies on the assumption that increasing the packet rate on a flow does not affect its QoS. In Section IV-B we show this to be essentially the case. iii) If the flows cannot be merged, JUDISHARE tries to replace the existing flow with a new one that has the merged packet rate and the *tight* QoS. This is achieved by running the VNE algorithm, using all flows already embedded in the network as an input, except the flow under consideration, which is replaced with the new merged flow. If this also fails, a new flow is defined for the user request.

IV. EVALUATION

We first present results of an extensive simulation campaign demonstrating JUDISHARE's performance in accommodating a set of requests on a shared WSN, using as benchmark a VNE-only solution. Secondly, we ran experiments to validate that merging flows and an increase in the packet rate does not negatively impact communication QoS, i.e., reliability.

A. Processing Requests

The behaviour of JUDISHARE when processing requests is compared with a VNE-only solution. The performance is measured in terms of the number of user requests *supported* and the number of flows *embedded* onto the network. For VNE, the two values are identical, however JUDISHARE is able to support more requests over the embedded flows, by merging requests and reusing flows. The performance is also quantified in terms of communication quota, supported and embedded, as well as energy consumption.

Experimental setup. The evaluation was conducted by simulating the allocation of batches of user requests over a network of 50 nodes. We assumed only one type of sensor in the network, generating packets of 50B for each sample, and all packets are sent to the network sink (no data aggregation used). For the communication quota, we considered a maximum link data rate of 624Bps, obtained experimentally using ContikiOS on the TelosB nodes without a MAC and in-line with results from [12]. Various sizes for the request batches were evaluated, conducting 100 runs for each size. In each run, the network topology was randomised to eliminate topology bias created by the relative position of the source node and the sink in the topology. The request parameters were also varied between runs. The source node identifier was randomly sampled from the network nodes. Packet rate was uniformly sampled from the set {1pkt/s, 1pkt/5s, 1pkt/10s, 1pkt/30s, 1pkt/min, 1pkt/10min, 1pkt/30min, 1pkt/hour}. The user requested end-to-end reliability was generated by sampling a normal distribution with mean 0.9 and standard deviation 0.1, taking only values less than 1. Latency was uniformly sampled from [100...5000] ms.

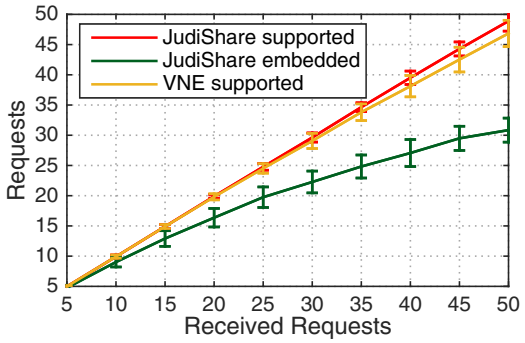

Fig. 2: Number of *supported* and *embedded* requests when the requested location is random.

Results. Fig. 2 shows the improvement that JUDISHARE provides over a VNE-only solution. Since the source node of a request is randomly sampled the probability of a specific node being requested is always the same $(1/50)$. However, with the number of nodes constant, as the number of user requests increases, the probability of source nodes being requested multiple times also increases. Therefore, as the number of user requests increases JUDISHARE is able to merge more and more requests embedding fewer flows into the network and consuming fewer network resources. This way more network capacity remains available after the embedding and that capacity can be used to accommodate additional user requests. Fig. 2 shows that JUDISHARE embeds roughly 33% less flows than the VNE-only solution. The figure also shows that JUDISHARE can support 4% more user requests than the VNE-only solution however this is not necessarily meaningful because a flow requesting 1% quota is easier to support than one requesting 10%. For this reason we also analysed the amount of communication quota that can be supported by the two solutions. Furthermore, we wanted to increase the demand on the network, because the linear trend shown by the *supported* requests with VNE and JUDISHARE in Fig. 2 did not satisfy the expectation. In the considered network setup the single sink limits the total data rate of the network to the maximum data rate at the sink. While the network sink provides 100% quota, paths longer than one hop see that value halved, or more, due to intra-flow interference. As all user requests require data collection at the sink, neither VNE-only nor JUDISHARE are able to allocate more capacity network-wide than the maximum capacity at the sink. Therefore, the total accepted communication quota in the network should have an *asymptotic* trend, not a linear one.

We stressed the maximum capacity of the network in two ways. Firstly, we increased the number of user requests from 55 to 95 in increments of 10; to increase the impact, the packet rate was also slightly increased, not considering packet rates of less than 1pkt/min. Using this approach, the total quota supported in the network was measured, and the results are shown in Fig. 3. It is clear that the VNE-only solution reaches a maximum value of around 50% for the total embedded quota. VNE is therefore subject to the maximum network capacity as explained above.

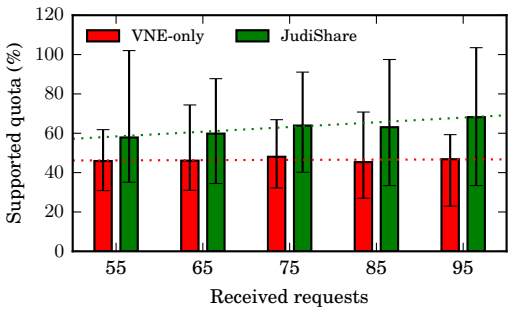

Fig. 3: Average supported quota in the network. Error bars show min and max values.

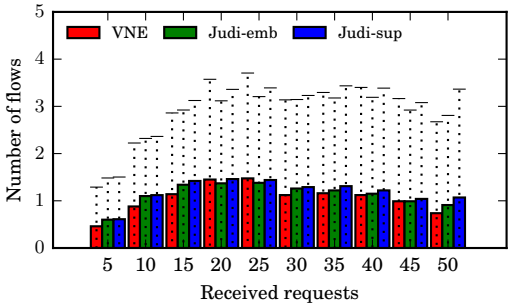

Fig. 4: Request processing results at high packet rates.

JUDISHARE is subject to the same limitation in terms of *embedded* flows. However, by reusing flows, the quota that can be *supported* can be further increased, and, as the figure shows, the impact is substantial, with JUDISHARE supporting over 100% quota in some extreme cases. Furthermore, JUDISHARE maintains an increasing trend for supported quota as the number of requests increases.

Secondly, we stressed the maximum network capacity by increasing the requested sampling rate and in turn the packet rate. Fig. 4 shows the result of processing the input requests where the packet rate is sampled uniformly from {1pkt/s, 1pkt/0.2s, 1pkt/0.1s, 1pkt/0.04s, 1pkt/0.02s}. The results are very different because at 1pkt/0.1s the requested quota is already 80%, which takes up most of the network capacity. Consequently, both JUDISHARE and the VNE-only solution manage on average to support only a single request.

B. Impact on Energy Consumption

Using the setup from Section IV-A, we further studied the impact that JUDISHARE has on energy consumption. Fig. 5 depicts the ratio of total energy consumption between JUDISHARE and the VNE-only solution, along with similar ratios for the average flow length and the total allocated quota. The total energy consumption is computed as $\sum_{link \in flow} E(quota(link))$ for all flows *embedded*. The figure shows the following. Before reaching the maximum network capacity (left side, as in Fig. 2) JUDISHARE allocates more quota and consumes more energy than VNE, in fact quota and energy are highly correlated. The two solutions allocate flows of roughly the same length; JUDISHARE will in most cases reuse existing flows embedded with VNE, but in some cases it will replace existing flows with new ones that have merged packet rate and tight QoS (as in case

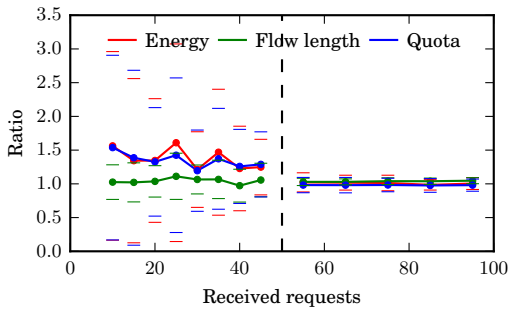

Fig. 5: Resource consumption ratio, JUDISHARE over VNE. The left side of the dotted line shows an under-utilized network. On the right side it is fully utilized.

iii from Section III-B), which will be longer and thus increase the ratio to > 1. Once the network capacity limit is reached (right side, as in Fig. 3), as expected, both solutions reach the same embedded quota, therefore the energy consumption will be very similar (ratio $\in [0.98, 1.02]$).

C. Impact on End-to-End Reliability

As JUDISHARE merges communication flows under the assumption that the QoS characteristics of the flow, mainly reliability, are unaffected when the packet rate is increased, we wanted to verify if this was correct as it is well known that higher packet rates can impact PDR negatively [13].

Experimental setup. We used a small-scale setup with two TelosB nodes placed indoors at 25 m apart and TRIDENT [14], a tool for connectivity assessment. The experiment was divided into rounds, during which each node sent 200 packets. For each inter-packet interval (IPI): {1000, 500, 250, 125, 100, 50, 25, 20, 15, 10, 5} ms we interleaved rounds in which nodes were communicating at {0, −1, −8, −10, −13, −15} dBm corresponding to power levels {31, 28, 14, 11, 9, 7} of the TelosB. We report results from 66 rounds.

Results. Fig. 6 illustrates the extent to which the PDR of a link can be affected by the packet rate. It clearly shows that the range of PDR variation is less dramatic at high transmit powers with lower bounds at 99% at the lowest IPI, i.e., 5 ms. On the other hand, for low transmit powers, the PDR is affected more by the decrease of the IPI, dropping to 98.5% at a 10 ms IPI and 93% in the worst case scenario at 5 ms IPI.

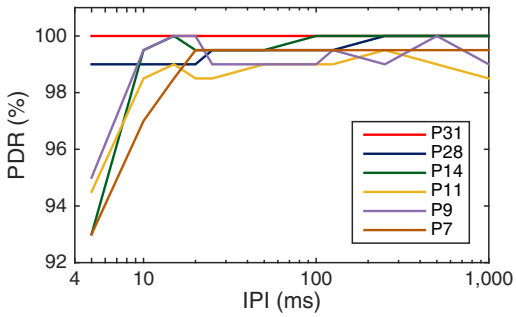

Fig. 6: PDR of a link as the packet rate (IPI) increases.

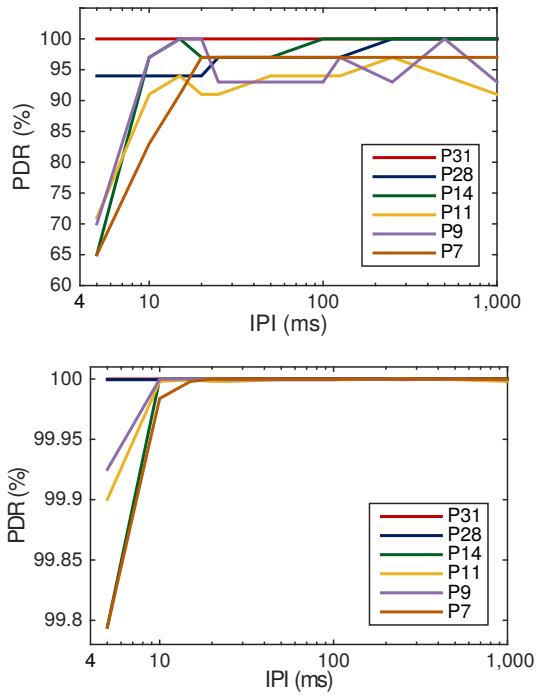

Fig. 7: End-to-end PDR of a 6-link path with no (top) and two (bottom) retransmissions as IPI increases.

Building on these results, we define $\mathcal{R}_P$ as the end-to-end reliability of path P, computed as the product of per-hop reliabilities $\mathcal{R}_l$. $\mathcal{R}_l$ of a link l is given by $\mathcal{R}_l = 1-(1-PDR)^{N+1}$ [15], where PDR is the reliability of the link and N represents the number of maximum retransmissions per packet. For a network path composed of 6 links with characteristics reported above, we computed $\mathcal{R}_P$ with no and with two retransmissions, as depicted in Fig. 7. JUDISHARE considers two retransmissions since this is the default value used by the Contiki OS. With two retransmissions, for all tested power levels, $\mathcal{R}_l$ drops as low as 93%, which translates into $\mathcal{R}_P$ decreasing to 99.79%. When there are no retransmissions, at high powers the decrease of $\mathcal{R}_l$ to 94% does not have a dramatic impact on the $\mathcal{R}_P$. In the worst case scenario, at the lowest power and the highest possible packet rate supported by a TelosB node, i.e., IPI of 5 ms, $\mathcal{R}_P$ drops to 65% when no retransmissions are in place. These encouraging results show that JUDISHARE would succeed in satisfying the reliability requirements for rates as high as 100pkt/s.

V. CONCLUSIONS

In shared WSNs, the goal of the infrastructure provider is to maximise profits, by accommodating as many users as possible and reducing the resource usage and associated cost. Existing data virtualization techniques reduce resource usage by merging data requests, however, they do not consider the QoS requirements of the requests, which, in our view, need to be taken into account in a shared WSN. We developed JUDISHARE, a QoS aware request-merging mechanism that allows same source sensing requests to reuse communication flows, if they have matching QoS requirements. JUDISHARE runs on a central network controller and relies on a Virtual Network Embedding algorithm to map requests to network resources. Through simulations we validated that JUDISHARE, by merging communication flows, greatly reduces the quantity of resources used in the network. This, in turn, allows more user requests to be accommodated. The most significant finding is that JUDISHARE, by merging requests, can break the limitation imposed by the network sink of a maximum of 50% communication quota and can push the supported communication quota to even more than 100% of the sink's capacity. Moreover, using real nodes, we investigated the impact that flow merging, i.e., increasing packet rate, has on a flow's end-to-end reliability, and concretely demonstrated that for packet rates of up to 100pkt/s the QoS is not affected. A practical use of JUDISHARE would be its integration with a scheduling mechanism in a system where network-wide decisions are taken based on QoS application requirements and the current network state.

Acknowledgements. This work was funded by Science Foundation Ireland (SFI) under grant 13/IA/1885.

REFERENCES

[1] R. Muller and G. Alonso, "Efficient Sharing of Sensor Networks," in *Proc. of the Int. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006.

[2] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, 2013.

[3] R. Katona, V. Cionca, D. O'Shea, and D. Pesch, "Exploring the Economical Benefits of Virtualized Wireless Sensor Networks," in *Proc. of Int. Sym. on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2017.

[4] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman, "Multi-query Optimization for Sensor Networks," in *Proc. of the Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, 2005.

[5] "Service-Centric Network for Urban Scale Feedback Systems (SURF)," Accessed: 2017-09-21. [Online]. Available: http://www.nimbus.cit.ie/portfolio-items/surf/

[6] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, 2005.

[7] S. Xiang, H. B. Lim, K. L. Tan, and Y. Zhou, "Two-Tier Multiple Query Optimization for Sensor Networks," in *Proc. of the Int. Conf. on Distributed Computing Systems (ICDCS)*, 2007.

[8] A. Tavakoli, A. Kansal, and S. Nath, "On-line Sensing Task Optimization for Shared Sensors," in *Proc. of the Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2010.

[9] P. Corbalan, R. Marfievici, V. Cionca, D. O'Shea, and D. Pesch, "Into the SMOG: The Stepping Stone to Centralized WSN Control," in *Proc. of the Int. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, 2016.

[10] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Real-Time Scheduling for WirelessHART Networks," in *Proc. of Real-Time System Symposium (RTSS)*, 2010.

[11] J. Lu and J. Turner, "Efficient Mapping of Virtual Networks Onto a Shared Substrate," WUSTL, Tech. Rep. WUCSE-2006-35, 2006.

[12] F. Osterlind and A. Dunkels, "Approaching the Maximum 802.15.4 Multihop Throughput," in *Proc. of the Workshop on Embedded Networked Sensors (HotEmNets)*, 2008.

[13] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "An Empirical Study of Low-power Wireless," *ACM Trans. Sens. Netw.*, vol. 6, no. 2, 2010.

[14] T. Istomin, R. Marfievici, A. L. Murphy, and G. P. Picco, "TRIDENT: In-field Connectivity Assessment for Wireless Sensor Networks," in *Proc. of the Extreme Conf. on Communication and Computing (ExtremeCom)*, 2014.

[15] M. Zimmerling, "End-to-end Predictability and Efficiency in Low-power Wireless Networks," Ph.D. dissertation, ETH Zurich, Swizerland, 2015.