Opportunistic Data Aggregation in Delay Tolerant Networks

Wolf-Bastian Pöttner and Lars Wolf
Institute of Operating Systems and Computer Networks
Technische Universität Braunschweig, Braunschweig, Germany
Email: [poettner|wolf]@ibr.cs.tu-bs.de

***Abstract*—Delay Tolerant Networks are well suited for long-term statistical data collection, where the installation of infrastructure is impossible or not worthwhile. Those scenarios generally employ wireless links and are often based on limited energy resources such as batteries or battery-buffered solar power. Hence, reducing energy consumption is of foremost importance. In-network data aggregation reduces the amount of data that is to be transferred and therefore lowers the energy consumption.**

This paper presents an opportunistic, application-independent data aggregation approach for Delay Tolerant Networks. By employing a generic configuration mechanism, applications can specify which bundles to aggregate. Using a standardized data format for measurement values allows intermediate nodes to aggregate a given set of bundles using so-called aggregation functions. The evaluation shows significant reductions in terms of energy consumption in two realistic, exemplary scenarios.

I. Introduction

Delay Tolerant Networks (DTNs) are based on the store, carry and forward paradigm. They allow to transmit data even in challenged networks with intermittent connectivity and to exploit the mobility of nodes to transport data. The de-facto standard in delay-tolerant communication is the address-centric Bundle Protocol (BP) [1] which transports data in so-called bundles. In literature, we find a significant amount of DTNs that are used for long-term statistical data collection: ZebraNET [2] is used to collect movement patterns of Zebras in Kenya, Vineyard Computing [3] measures the exposure to the sun of vineyards and OptraCom [4] collects air pollution samples in the city. While those are just three examples, many similar applications send small sensor samples in BP bundles.

The significant overhead of the BP impacts the efficiency of such applications. Even if data rates are often small, the overhead increases the network load and is, hence, not desirable. Such networks often use low-power wireless links with nodes operating on batteries. Reducing the overhead also decreases the energy consumption and, thereby, increases network lifetime on batteries. Since low-power wireless links often provide a low bandwidth only, lowering the overhead increases the effective network capacity. Lowering the protocol overhead of the BP is one possibility but would break compatibility with existing BP implementations. In this paper we investigate, how data aggregation techniques can be applied to bundles in the BP to reduce the total amount of bundles in the network. In many applications, data is send to one or multiple sink nodes and statistically aggregated to allow drawing conclusions. Parts of the data processing can also be moved into the network to reduce the number of transmitted bundles and thereby also the energy consumption. By *data aggregation* we refer to the process of taking information from multiple bundles, combing them using an aggregation function and to forward only a single bundle to the next hop.

As we will show in the following Section II, existing data aggregation approaches are often based on a specialized network structure. However, in DTNs this type of structure may not exist or is costly to maintain. We therefore use an opportunistic, application-independent approach in which multiple bundles are aggregated into one bundle if a node happens to posses multiple bundles and the respective meta information for aggregation. The design of our approach is routing-protocol-agnostic (and thereby also structure-free) to allow for independent routing decisions within the DTN even when using data aggregation. We discuss the requirements towards such an aggregation mechanism and outline how configuration can be done in Section III. Our approach is application independent (apart from the configuration) and as BP-compliant as possible. We give insight into simulation results obtained using *The One* [5] simulator in Section IV and conclude the paper in Section V.

II. Related Work

Research in the field of Wireless Sensor Networks (WSNs) has produced a number of data aggregation approaches that can be classified according to their location in the stack and their topology requirements. Data aggregation can be performed in layer 2, layers 3 and 4 as well as layer 5 (application layer). Furthermore, aggregation protocols may require to set up a specific (virtual) network topology to aggregate data at specific points or be opportunistic in nature. In the following, representatives of each class are given. A survey of Aggregation Techniques for WSNs is given in [6].

AIDA [7] is an application-inspecific data aggregation approach located on an intermediate layer between the existing network- and data link-layer. Outgoing data is aggregated while waiting in the output queue of a node and deaggregated on the receiver node before being passed into the network layer. This generic aggregation approach allows to achieve certain reductions in overhead and does not rely on a specific network topology. However, the approach cannot be as effective as an aggregation mechanism that understands the application-specific data and protocols.

The authors of [8] aim at creating an aggregation tree with optimally selected aggregation nodes within the tree. Data is sent along the tree and aggregated at specific intermediate

nodes. Data aggregation is performed on layers 3 and 4 and the evaluation shows significant reductions in energy consumption. LEACH [9] is a cluster-based aggregation approach in which nodes form clusters and information from the cluster is routed over a so-called clusterhead. These clusterheads are also the aggregation points for data coming from their clusters. OPAG [10] is an opportunistic data aggregation approach that exploits multi-path data transfer through the network. By sending multiple copies of measurement values, costly retransmission of individual packets can be avoided. The approach computes an overlay spanning tree along which data is aggregated. A commonality of all those approaches is, that they construct a certain aggregation topology which is not suitable for an intermittently-connected, dynamic DTN in which a certain topology can never be guaranteed.

The authors of [11] explicitly avoid maintaining a network structure to lower the management overhead. By employing Data-Aware Anycast on the Medium Access Control (MAC) layer as well as Randomized Waiting in the application, potential for data aggregation is created. However, the solution has strong requirements such as known location of nodes as well as synchronized clocks that both cannot be guaranteed in a DTN.

Thus, to the best of our knowledge, it seems that work on opportunistic, application-inspecific data aggregation approaches for DTNs does not exist.

III. Design

This section outlines the overall design as well as design considerations for our BP data aggregation approach. The foremost goal is to aggregate multiple bundles into one to reduce protocol overhead while remaining largely compatible with the BP specification. More specifically, BP nodes that do not support aggregation must be able to successfully forward aggregated bundles. Furthermore, the aggregation approach shall be application independent in its design to enable widespread use. In general we assume, that data aggregation approaches are used in networks deployed for one primary use case being operated by one institution.

We realize this by using a standard data format (see Figure 1) as well as a configuration mechanism that allows to specify how to aggregate bundles. Our approach is opportunistic in so far, as nodes will reactively aggregate bundles if more then one bundle in their storage can be aggregated. This means, that if no opportunity for aggregation exists, bundles are forwarded as usual. However, to increase the likelihood of bundle to be aggregated, we introduce so-called delay functions to increase the time a bundle is stored on a node and hence increase the potential for aggregation.

In the context of data aggregation, we define the "aggregation degree" as the number of individual measurements that are contained in a bundle. Bundles start with an aggregation degree of 1 and if two bundles are aggregated together, the value increases to 2. If at a later stage another bundle is aggregated into the former bundle, then the aggregation degree is increased to 3. Please note, that a higher aggregation degree is desirable as it indicates how much overhead could be saved.

A. Aggregation Criteria

Aggregation-aware nodes check the bundles in their storage for aggregation potential for each new bundle that is stored. To check a pair of bundles to be aggregatable, the following basic criteria have to be met: both bundles have to use the standard data format so that the aggregation node knows what information to combine. Furthermore, both bundles have to share the same destination Endpoint Identifier (EID).

In addition, applications can configure a number of criteria to specify if and how bundles shall be aggregated. The configuration is always based on the destination EID and the data type and also contains the aggregation- and delay-function to use. Aggregation functions have to be known to all nodes in advance.

Aggregation can further be configured based on the following properties of the bundle:

- *Identity of the sender* allows to aggregate only bundles from a specific sender or a group of senders. Since the BP is an address-centric protocol, nodes are also referenced by their EID.
- The *Time Window* in which the measurements have been taken. Based on the bundle creation timestamp, a maximum difference between the timestamps of bundles that shall be aggregated can be defined. If more than two bundles shall be aggregated, the minimum and maximum respective creation timestamps may not differ by more than the time window.

If two bundles are considered to be eligible for aggregation, the measurement information is combined using the aggregation function (see Section III-C) according to the configuration. If a bundle matches the configured aggregation criteria, the delay function is applied to that bundle before it is forwarded (see Section III-D).

A set of aggregation criteria is combined to the aggregation configuration. This configuration is distributed in the network using a multicast bundle.

B. BP compatibility

In BP terms, a bundle is created by the sender and consumed by the receiver; the specification does not include any means to alter a bundle in-flight apart from bundle fragmentation. However, when taking two bundles with measurement information that happen to be stored at the same node at the same time, aggregating those bundles means to combine the information from both bundles and only forward one towards its destination. Hence, aggregating bundles is a significant change to the handling of bundles outlined in the BP specification [1].

When combining multiple bundles into one, a number of the fields of the Primary Bundle Block (PBB) have to be considered. First of all, two bundles can only be aggregated if they share the same destination EID. Each bundle is uniquely identified by the source EID, the creation timestamp and the sequence number. Our approach stores the necessary information to identify all bundles that are aggregated to allow identifying all original bundles that have been aggregated into any given bundle. The information identifying the original

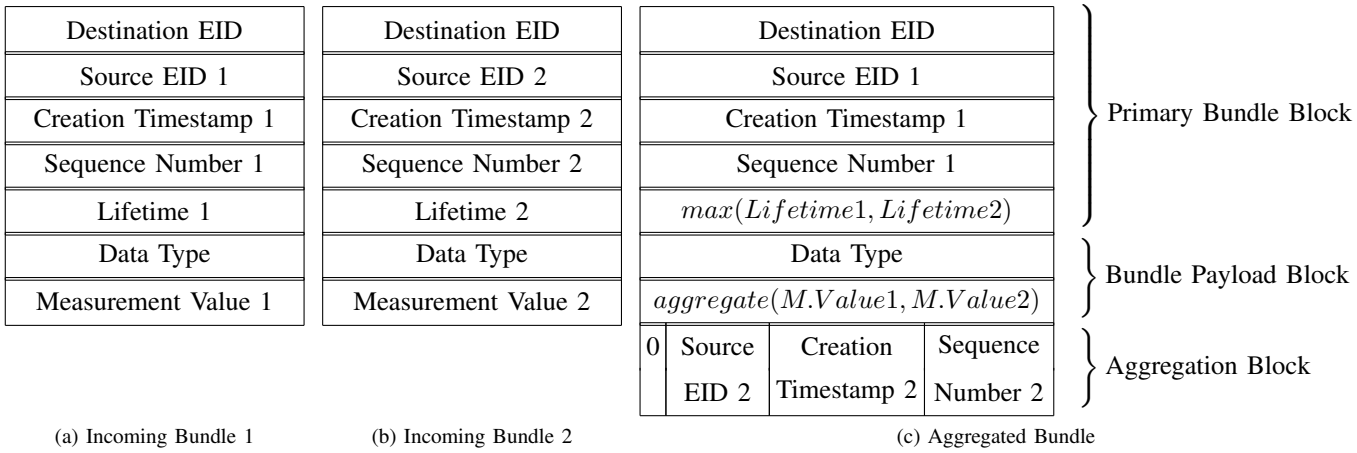

Figure 1: Bundle aggregation example with two (the same principle applies for more than two input bundles) incoming bundles and one outgoing bundle (shown bundles are simplified). Please note the matching *Destination EID* and *Data Type* fields.

bundles (source EID, timestamp, sequence number) is kept in the aggregation extension block (see Figure 1) which makes it compatible with all BP-compliant implementations. On nodes that do not support aggregation, two bundles may be considered to be identical due to a similar PBB since the nodes do not know about the extension block. Upon aggregation of two bundles, the header information of the outgoing bundle are selected in a deterministic way based on the incoming bundles.

Other relevant header information in the PBB are the lifetime and the bundle priorities. Our approach uses the highest priority and highest lifetime of the incoming bundles for the outgoing bundle. This does not harm the network, as aggregation is usually only applied to a certain time window of original messages. Therefore, it is guaranteed that aggregated bundles have a bounded lifetime and will eventually expire. Using the highest lifetime further ensures, that no aggregated bundle is discarded before the original bundle would have been expired. It may happen however, that a subset of the information aggregated into a bundle is delivered to its destination after the lifetime for the original bundle would have been expired.

1) Bundle Status Reports and Custody Transfer: Two elements of the BP need special attention when combining multiple bundles into one: bundle status reports and custody transfer. Status reports may be requested by the sender and may be send by any node on the path for events such as receival, acceptance, forwarding, deletion and delivery. The PBB contains a special "Report-To" field which specifies the recipient of those reports. When combining two bundles, the report-to information including the flags which reports to send should be maintained. Furthermore, the report has to reference the original bundle, so that the original EID as well as creation timestamp and sequence number have to be present as well. In the scope of this paper, bundles requiring a report are not aggregated to circumvent problems associated with aggregating such bundles. To provide some level of functionality, nodes that aggregate multiple bundles into one could act as the recipient of the bundle and send all appropriate reports. However, we recommend not to use reports when sending bundles that are potentially aggregated.

If the end-to-end characteristics of the reports have to be kept at all cost, one would collect the report-related information (EID and flags) in the aggregation block in the bundle. An aggregation-aware destination or intermediate node can then send reports to all report-to addresses contained in the aggregation block. An aggregation-unaware node (standard BP implementation) would only send a report to the report-to EID contained in the PBB and not send reports to the other interested parties.

Custody transfer is another issue when aggregating bundles. The BP custody management involves a custodian for each custody-enabled bundle and this custodian is responsible for that particular bundle. This mechanism allows hop-by-hop reliability by ensuring, that a bundle will only be deleted by a custodian when another node has accepted custody for it. Unfortunately, the BP specification contains many blind spots with regard to custody transfer. It is rarely used in practice and often argued about[1]. We therefore exclude support for custody management in connection with bundle aggregation.

C. Aggregation Functions

To allow bundle aggregation, the measurement values contained in multiple bundles have to be combined into one single value that is then forwarded to the destination. The selection of the suitable aggregation function is of course application-specific. In general we provide a configuration mechanism to adapt to the specific application needs. In the context of this paper, we see an aggregation function $aggregate(...)$ as a mathematical function that takes ≥ 2 input values and calculates 1 output value. Examples of such functions are the minimum, maximum, average, standard deviation and the sum among many others.

In the scope of this paper we differentiate two types of aggregation functions: 1) *duplicate-sensitive* aggregation functions which will produce a wrong result if the same input value is put in twice. For the calculation of, e.g., a sum, putting

[1] http://www.ietf.org/mail-archive/web/dtn-interest/current/msg04337.html

in the same input value twice will actually produce a different (and wrong) result and has to be avoided. To avoid this, the aggregation block is appended to all aggregated bundles and allows to determine, if the measurement value of a bundle has already been taken into account for the aggregation. On the other hand, 2) *duplicate-insensitive* aggregation functions which will produce the same correct result, no matter how often a particular value has been repeated. When calculating, e.g., the minimum, having the same argument twice does not change the overall result.

D. Delay Functions

The approach of opportunistic data aggregation relies on the properties of DTNs: bundles are stored on a node for a certain amount of time which opens potential for aggregation. The longer a bundle remains on a node, the higher the potential for aggregation. Since DTNs are designed to handle potentially long delays, artificially increasing the delay of a bundle is legitimate (strictly speaking, this may depend in the specific use case). Our approach is to use so-called delay functions to keep bundles on a node for a certain amount of time. During this delay period, the bundle is not forwarded but potentially aggregated with more incoming bundles.

In particular, our approach uses the following three delay functions:

- *Static delay* will delay the bundle for a fixed amount of time on each node.
- *Static delay if below threshold* will delay bundles for a fixed amount of time, if the aggregation degree is below a threshold. If the aggregation degree increases to match or exceed the threshold, the bundle may be forwarded before the fixed delay has elapsed.
- *Variable delay* calculates the fixed delay per bundle by dividing the static delay by the aggregation degree to forward bundles that have already been aggregated faster. $\text{delay} = \frac{\text{Static Delay}}{\text{Aggregation Count}}$

E. Aggregation Data Format

To enable intermediate nodes to aggregate bundles, a common data format has to be used. While we cannot standardize a data format, we discuss the elements it has to contain. To identify the type of the measurement value, a "data type" field has to be included as shown in Figure 1. The content of this field specifies the type of data (temperature, pressure, etc.) and hence allows aggregating nodes to decide, whether two bundles contain the same type of data. Also, the raw measurement value has to be contained in each bundle. We argue, that in terms of BP compatibility, the Bundle Payload Block (BPB) is the proper place to store such information.

However, to know which bundles have already been aggregated into a bundle at hand, those bundles have to be uniquely identified. We therefore insert a custom aggregation block into each aggregated bundle that contains the source EID, the timestamp and the sequence number of all bundles that have been aggregated into the bundle at hand (see Figure 1). This allows aggregation-aware nodes to decide, whether two bundles are carrying the same information even if the source EID, the sequence number of the timestamp of the PBB are identical.

If a node receives a bundle which is an aggregate of one or multiple original bundles, the node has to check whether the information contained in the bundle at hand is already present in its storage. If all original bundles have been aggregated into bundles that are currently stored on the node, then the newly received bundle will be silently discarded. Otherwise, the bundle may be aggregated with existing bundles if - and only if - a duplicate-insensitive aggregation function is used. Otherwise, multiple bundles have to be kept in the storage to avoid producing erroneous results.

IV. Evaluation

To evaluate the potential of opportunistic data aggregation in DTNs we have conducted a series of simulations using *The One* [5] simulator. The primary focus are the number of bundle transmissions between nodes (correlating with bandwidth utilization and energy consumption) and the end-to-end bundle delivery rate. We expect, that enabling aggregation will significantly reduce the number of transmissions while keeping the original bundle delivery rate. We have furthermore focussed on the aggregation degree per bundle as well as the end-to-end delivery delay.

We modified the simulator to support our aggregation protocol. Namely, we have modified the *sourceMessage* class to extend the data format and the bundle identification. Incoming bundles are accepted only if previously unknown information is contained. Furthermore, we have modified the bundle router to perform reactive bundle aggregation.

A. Evaluation Scenarios

For the evaluation we have used two scenarios:

Scenario 1 is called "Vineyard" and contains 20 nodes that are modelled after the scenario described in [3]. 16 nodes are fixed in a vineyard and create one measurement bundle per minute each. The bundle is addressed to a fixed sink node that is out of range for all nodes on the vineyard. 3 worker nodes are constantly moving over the vineyard and come into range of the sink during breaks and after the work day is over. We assume, that nodes are WSN nodes such as INGA [12] with a range of 50 m, a throughput of 250 kbit/s and storage capacity of 1 MBytes. The simulation time is 14 hours and we simulate duplicate-sensitive and -insensitive aggregation functions.

Scenario 2 is called "Braunschweig" and is based on the scenario used in [13], [14]. 28 nodes are following a realistic mobility pattern of rail cars for inner-city transportation and each generates one bundle every minute. All bundles are addressed to a stationary sink node, where most of the rail cars pass by from time to time. We assume nodes that are equipped with IEEE 802.11 transceivers offering a range of 300 m and a throughput of 11 MBit/s. Buffers are configured to 2000 MBytes which can easily be realized using an SD card. The simulation was performed for 5 hours and we simulate duplicate-sensitive and -insensitive aggregation functions.

B. Bundle Delivery Rate and Delay

The bundle delivery rate is the fraction of original bundles that have been delivered to the sink node within the simulation time compared to the total number of created bundles. In

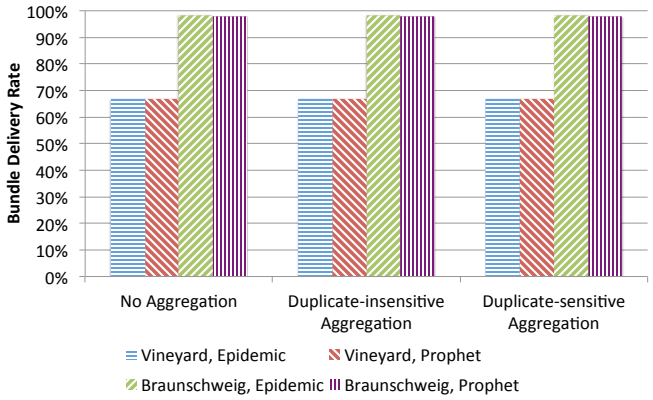

Figure 2: Bundle delivery rate for two routing protocols and two scenarios (higher is better).

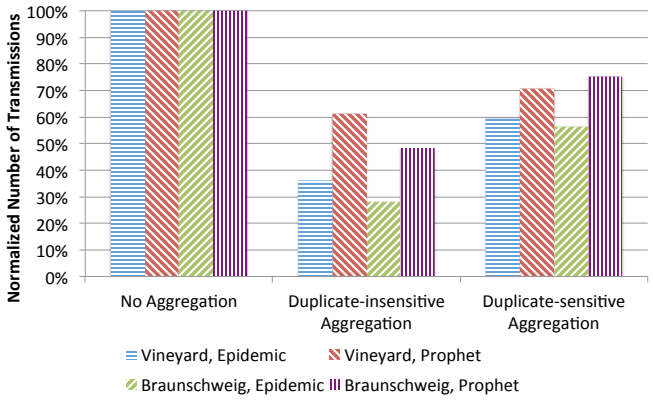

Figure 3: Total number of bundle transmissions for two routing protocols and two scenarios, normalized to "No Aggregation" (lower is better).

	Vineyard	Braunschweig
No Aggregation	378.39 s	3168.40 s
Duplicate-insensitive Aggr.	379.29 s	3168.49 s
Duplicate-sensitive Aggr.	378.97 s	3164.83 s

Table I: Average end-to-end Delay for epidemic routing in two scenarios (lower is better).

the context of bundle aggregation, we compare the number of measurements delivered to the sink with the number of measurement send by the nodes. Since measurements are aggregated on the way, one incoming bundle may actually include n measurements and then this is counted as n original bundles.

The results in Figure 2 show, that 66.81 % of the measurements are delivered when using Epidemic routing in the vineyard scenario without aggregation. Enabling aggregation increases the ratio of delivered bundles slightly to 66.86 %. All bundles that have not been delivered are still stuck in the bundle storage of the nodes. Due to the movement pattern of the workers on the vineyard, bundles that are generated after the workers have finished their work will not be picked up during the simulation time and hence cannot be delivered. While the delivery ratio for the Braunschweig scenario is significantly higher (98.42 % with Epidemic routing), aggregation also does not impact the delivery ratio significantly. Using Prophet routing slightly decreases the delivery ratio to 97.85 % independent whether aggregation is used or not.

The end-to-end delay is the time between creation of a bundle and its delivery to the sink node. In the context of data aggregation, we again look at the delay of individual measurements, even if those are delivered to the sink in aggregated form. If a measurement is delivered multiple times to the sink, we count the earliest delivery (the lowest delay) as delay for that particular measurement.

The results in Table I show the average end-to-end delay. As we learn, data aggregation has no significant impact on the end-to-end delay of bundles.

The results confirm, that data aggregation has no significant effect on the bundle delivery ratio and the end-to-end delay. This is the expected behaviour in networks operating below the network capacity limit. Only if storage or link capacity become the limiting factor, data aggregation is expected to increase bundle delivery ratio and decrease end-to-end delay by lowering the overall network load.

C. Number of Transmissions and Aggregation Degree

The number of bundle transmissions in the network are an approximation for energy consumption and bandwidth utilization of the nodes as transmitting more bundles (or radio frames) consumes more energy. Unfortunately, the number of transmissions is no accurate measure for energy consumption, as also the length of packets has to be considered. However, *The One* offers no means to count the amount of transmitted bytes as the BP overhead is not modelled properly. Although BP data aggregation increases the packet size (see Figure 1), one aggregated bundle is still significantly smaller than two original bundles. Hence we can conclude, that decreasing the number of transmissions also reduces energy consumption.

We have summed up all bundle transmissions in the network and plot the results in Figure 3. The transmissions are normalized to the respective combination of scenario and routing protocol without using aggregation ("No Aggregation"). The results show, that data aggregation with a duplicate-insensitive aggregation function has the highest potential in terms of energy savings. When looking at the vineyard scenario with Epidemic routing, our data aggregation approach reduces the number of transmissions to 36.14 % (reduced from 177209 to 64501 transmissions). When using epidemic routing, the reduction to 28.26 % (down to 69409 transmissions from 245637 transmissions) is even more striking when looking at the Braunschweig scenario. The numbers show, that duplicate-insensitive aggregation together with Epidemic routing reduces energy consumption significantly.

The reduction is lower when using Prophet routing in both scenarios (down from 51501 to 31625 for vineyard and down from 179270 to 86685 transmissions for Braunschweig respectively). The reason is, that Prophet produces less bundle

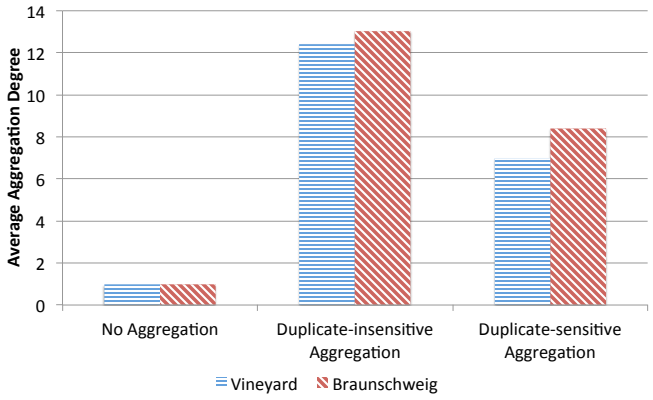

Figure 4: Average aggregation Degree for epidemic routing in two scenarios (higher is better).

copies and hence less potential for aggregation. In a duplicate-sensitive setting (such as the sum of all values), the benefits are lower but still significant. The reason is, that if two bundles with aggregation degree higher than 1 (and non-disjoint original bundles) are stored at the same node, those bundles cannot be aggregated using a duplicate-sensitive aggregation function to avoid corrupting the data. Therefore, duplicate-insensitive data aggregation can produce more utility from the aggregation potential and hence less transmissions are required.

The aggregation degree is the number of original bundles that have been aggregated into a given bundle. The results in Figure 4 show the average aggregation degree for all bundles that are delivered to the sink for Epidemic routing in our two scenarios. The graph confirms, that the aggregation degree for all bundles delivered to the sink is 1 when using no data aggregation. Enabling duplicate-insensitive aggregation increases the aggregation degree to 12.39 and 13.03 for the Vineyard and Braunschweig scenario respectively. However, as we have seen before, duplicate-sensitive aggregation functions have less potential for aggregation and, hence, feature a lower bit still significant aggregation degree of 6.99 and 8.40 respectively.

V. Conclusions

DTNs are increasingly popular for long-term statistical data collection, especially in the context of low-power WSN nodes [15]. However, the protocol overhead of the BP is significant when small sensor measurements are transferred. For a bundle with 4 bytes of payload, at least 18 bytes are added for the BP header. While the overhead is a clear disadvantage of the BP, the interoperability with existing DTN networks and implementations make its use still worthwhile. To reduce the impact of the overhead, we have proposed and investigated an opportunistic approach for BP data aggregation in this paper.

A generic and application-independent configuration mechanism allows applications to configure which bundles shall be aggregated on nodes in the network based on source EID, data type and time window. Applications can specify which aggregation function to use and if bundles shall be delayed in intermediate nodes to increase their likelihood of being aggregated. The specific protocol to configure the aggregation mechanism is out of scope of this paper due to space constraints.

Simulations in *The One* have shown, that data aggregation significantly reduces the number of bundle transmissions in the network. Since transmitted data costs energy, employing data aggregation also reduces the consumed energy. The simulations have furthermore revealed, that aggregating bundles does not have significant effects on the end-to-end delay and the bundle delivery rate. Finally, in two exemplary scenarios, an average aggregation degree between 6.99 and 13.03 could be achieved.

References

[1] K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC 5050 (Experimental), Internet Engineering Task Force, Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5050.txt

[2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet," *SIGPLAN Not.*, vol. 37, pp. 96–107, Oct. 2002.

[3] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard computing: sensor networks in agricultural production," *Pervasive Computing, IEEE*, vol. 3, no. 1, pp. 38 – 45, Jan. 2004.

[4] S. Lahde, M. Doering, W.-B. Pöttner, G. Lammert, and L. Wolf, "A practical analysis of communication characteristics for mobile and distributed pollution measurements on the road," *Wireless Communications and Mobile Computing*, vol. 7, no. 10, pp. 1209–1218, Dec. 2007.

[5] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[6] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network Aggregation Techniques for Wireless Sensor Networks: A Survey," *Wireless Commun.*, vol. 14, no. 2, pp. 70–87, Apr. 2007.

[7] T. He, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "AIDA: Adaptive Application-independent Data Aggregation in Wireless Sensor Networks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 2, pp. 426–457, May 2004.

[8] D. Hoang, R. Kumar, and S. Panda, "Optimal data aggregation tree in wireless sensor networks based on intelligent water drops algorithm," *Wireless Sensor Systems, IET*, vol. 2, no. 3, pp. 282–292, Sep. 2012.

[9] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *Wireless Communications, IEEE Transactions on*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[10] Z. Chen and K. Shin, "OPAG: Opportunistic Data Aggregation in Wireless Sensor Networks," in *Real-Time Systems Symposium, 2008*, Nov. 2008, pp. 345–354.

[11] K.-W. Fan, S. Liu, and P. Sinha, "Structure-Free Data Aggregation in Sensor Networks," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 8, pp. 929–942, Aug. 2007.

[12] F. Büsching, U. Kulau, and L. Wolf, "Architecture and Evaluation of INGA - An Inexpensive Node for General Applications," in *Sensors, 2012 IEEE*. Taipei, Taiwan: IEEE, Oct. 2012, pp. 842–845.

[13] M. Doering, T. Pögel, and L. Wolf, "DTN Routing in Urban Public Transport Systems," in *Proceedings of the 5th ACM Workshop on Challenged Networks*, ser. CHANTS '10. New York, NY, USA: ACM, 2010, pp. 55–62.

[14] T. Pögel, "Optimized DTN-Routing for Urban Public Transport Systems," in *17th GI/ITG Conference on Communication in Distributed Systems (KiVS 2011)*, ser. OpenAccess Series in Informatics (OASIcs), vol. 17. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, pp. 227–232.

[15] W.-B. Pöttner, F. Büsching, G. von Zengen, and L. Wolf, "Data Elevators: Applying the Bundle Protocol in Delay Tolerant Wireless Sensor Networks," in *The Ninth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2012)*, Las Vegas, Nevada, USA, Oct. 2012.