Preprint of: Chenfeng Zhu and Andreas Reinhardt. "Reliable Streaming and Synchronization of Smart Meter Data over Intermittent Data Connections." In: *Proceedings of the IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm).* 2019, pp. 366–371.

Reliable Streaming and Synchronization of Smart Meter Data over Intermittent Data Connections

Chenfeng Zhu
Technische Universität Clausthal
Clausthal-Zellerfeld, Germany
zhuchenf@gmail.com

Andreas Reinhardt
Technische Universität Clausthal
Clausthal-Zellerfeld, Germany
reinhardt@ieee.org

***Abstract*—More and more smart meters are being rolled out in domestic and commercial buildings as well as industrial sites worldwide. They enable the timely and fine-grained monitoring of electrical energy generation and consumption. Besides storing measured data locally, most smart meters are equipped with communication interfaces to transfer collected readings to metering service providers. This not only allows for accurate billing, but also enables the extraction of additional information from collected data, particularly when they have been sampled at a high temporal resolution. The communication link used to exchange meter data can, however, be prone to disruptions and transmission errors. Consequently, while consumption data used for billing purposes might only be removed from a smart meter's internal buffer when they have been reported correctly, intermediate readings can be irrevocably lost during communication link outages. Because such readings are often useful for analytics purposes, their loss should be avoided. We hence propose a hybrid data transmission scheme that combines the real-time reporting of consumption readings with a background synchronization process that ensures the lossless exchange of data. We evaluate our design in a practical setting and demonstrate its efficacy in recovering data after the metering device has been physically disconnected from the network.**

I. Introduction

As part of the conversion of current power grids into *smart grids*, electromechanical energy meters are gradually being replaced by digital metering devices. Such *smart meters* can autonomously report consumption information to the metering service provider. Thus, they make the manual labor to report meter readings obsolete and simultaneously allow for a vastly increased temporal resolution of sampled data. In practice, smart meters not only report collected readings periodically (e.g., once every 15 minutes), but internally collect data at even greater temporal resolutions. Even though such high sampling rates are commonly unnecessary for billing purposes, the concept of Non-Intrusive Load Monitoring (NILM) relies on the wealth of information content in such data [1]. In fact, consumption data sampled at rates of 44 kHz [2], 100 kHz [3], 250 kHz [4], or even higher [5] have been presented for their use in NILM research.

Most smart meters use computer networking technologies, e.g., Powerline Communications (PLC) or cellular services [6], to forward collected data to the metering service provider. A commonality of such communication links is their limited real-time capability and the ensuing risk of data loss when network congestion or temporary outages occur. While this is less severe for low-volume data which can be buffered easily, data collected at higher temporal resolutions are commonly not retained on the smart meter due to the limited amount of available memory.

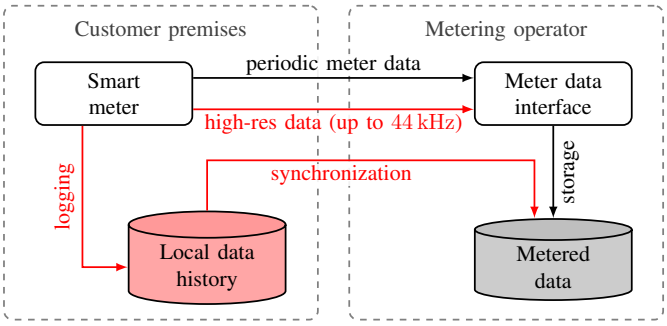

Fig. 1. Smart meter data are communicated to the metering operator by means of regular transmissions across a computer network. Highlighted components depict the additions proposed in this work, which targets to reliably make high-resolution data available to the metering operator as well.

Instead, they are often silently discarded when their transmission has failed. As ensuing gaps in the data may degrade the energy analytics accuracy, adequate means for recovery from such failure situations are essential. Especially when safety-critical services, such as predictive maintenance, are based on the data, the loss of relevant consumption characteristics due to link outages can often not be tolerated.

We hence present a holistic system design named *RoS/SoR* (Real-Time over SIP / Synchronization over Rsync) in this paper. It allows for the real-time streaming smart meter data at sampling rates of up to 44 kHz to the metering operator. Besides that, it features a synchronization component to retransmit missing fragments of the logged data after link failures have occurred. The key advantage of our design is that it almost completely eliminates the exchange of redundant information after temporary disconnections. Only data segments that could not be successfully transferred are re-uploaded by means of the *rsync* protocol. As we have designed RoS/SoR with its operability on embedded devices in mind, it can be seamlessly included into smart meter hardware at a later stage.

An overview of our system design and its integration into the legacy smart meter setup is shown in Fig. 1. Details about aspects related to RoS/SoR's implementation and its evaluation are presented in Secs. III and IV, respectively. We survey related work in Sec. II and conclude this paper in Sec. V.

II. Related Work

Communication protocols for high-resolution smart meter data must combine the properties of real-time streaming protocols

with a way to synchronize logged data if their real-time transmission has failed. Furthermore, their operability on smart meters, i.e., embedded systems with limited resources, is indispensable. Very few solutions currently provide this combination of features. Apache Flume[1] is a scalable and fault-tolerant framework to consolidate logging data from many streams. Even though it satisfies the requirements to streaming and synchronization, its operability on embedded systems has not been a design objective of Flume. Moreover, its recovery mechanisms cannot handle situations in which data could not be synchronized for extended periods of time. Given the scarcity of other protocols that synergistically combine real-time streaming with synchronization, we discuss protocols for both types of traffic separately as follows.

HLS [7] and DASH [8, 9] are multimedia streaming protocols. Their operation on top of the connection-oriented Transmission Control Protocol (TCP) protocol makes them resilient to sporadic packet losses, yet at the expense of packet retransmissions and the corresponding delays. Both protocols also adapt the bit rate of the data stream to the bandwidth limitations of the underlying link in order to avoid losses. As a consequence thereof, however, the input data must be available in different bit rates prior to their transmission. The required transcoding step puts a strong additional computational burden on the data source. The Stream Control Transmission Protocol (SCTP) is also a reliable transport protocol with support for rate-adaptive congestion control [10]. Moreover, it provides the possibility to handle multiple simultaneous and multiplexed streams across each connection. One of the most important impediments to the wider use of SCTP, however, is the slow uptake of SCTP support in embedded operating systems.

A more lightweight alternative for the end-to-end transmission of multimedia streams is the Real-time Transport Protocol (RTP) [11]. Operating on top of the User Datagram Protocol (UDP), RTP does not retransmit lost packets and thus caters to the real-time nature of transported traffic better. Many higher-layer protocols, e.g., the Session Initiation Protocol (SIP) [12] and the Real-time Streaming Protocol (RTSP) [13], use RTP to provide streaming functionalities. Due to its lightweight nature, RTP and protocols based on it appear as suitable candidates to stream smart meter data, yet without any provisions to encounter losses due to congestion or link outages.

Similar to RTP, the UDP-based Data Transfer (UDT) protocol provides abstractions for data streaming in high-performance environments [14]. A major obstacle for the use of UDT on smart meters, however, is its high resource demand. The Sampled Value Process Bus concept defined in IEC 61850-9-2 also addresses the real-time streaming of voltage and current samples [15]. Its focus, however, predominantly lies on monitoring components (e.g., substations) in the power grid, using dedicated and reliable networks. Its suitability to transfer data over networks with fluctuating link qualities is not inherently given.

In contrast to the real-time streaming protocols discussed so far, *synchronization* protocols target the efficient non-real-time synchronization of data between networked devices. The widely used *rsync* protocol [16] applies a specific encoding scheme to this end. Instead of exchanging the complete contents of a file to be synchronized, rsync limits the data transmissions to the actual differences between the file on both systems. Large bandwidth savings can thus be accomplished when only a small range of readings is missing from an otherwise complete data file. Alternatives to rsync include the Block Exchange Protocol (BEP) [17], which considers data in a block-wise fashion and prepends the actual synchronization step by the exchange of an index that lists changes to blocks. Only blocks with changes are subsequently synchronized to avoid redundant data transfers. BEP offers many features beyond the synchronization of files between two hosts, such as its operability in device clusters or authentication options. Its fundamental operation is thus highly similar to rsync, yet its extensions bear greater complexity and thus require more computational power to be executed. For the synchronization of database contents, the *FastSync* protocol can be used [18]. FastSync has not been designed to support the transfer of non-database contents, however. Consequently, as smart meters cannot be expected to rely on internal databases to archive readings, FastSync is not an applicable solution for the envisaged scenario.

No single protocol presented in related work fully caters to the requirements of smart metering scenarios, where lossless data archival is mandatory and readings should be transmitted in real-time whenever a communication link is available. Multimedia streaming protocols are not designed to synchronize data in a lossless manner. In turn, synchronization protocols lack the potential for real-time streaming, rendering them equally inapplicable if used standalone. A combination of both traffic types is thus inevitable to meet all the requirements for smart meter data and enable NILM at scale.

III. Meter Data Streaming and Synchronization

In Sec. II we have shown that dedicated solutions for data streaming and data synchronization exist. However, we have not been able to identify a solution that caters to both traffic types while being sufficiently lightweight to be executed on smart meters. To accomplish the objectives of lossless smart meter data collection in a timely manner, a combination of streaming and synchronization is required. We realize such a synergistic fusion of protocols in RoS/SoR. More details about its components and their operation are provided as follows.

A. Considered Smart Metering Setup

Before motivating our choice of protocols and their adaptation to the considered use case, let us briefly summarize the envisaged usage scenario of RoS/SoR. As shown in Fig. 1, our system design is based on the extension of smart meters with a local data storage component. Given the anticipated nature of data to store (periodic samples of electricity consumption data), a stream of sequential write operations is expected. File-based solutions are well-suited to accommodate the storage of such data with little overhead, even on embedded systems.

The resource limitations present on smart meters make lightweight software implementations necessary. In contrast to this, significantly greater computational resources can be

[1] See https://flume.apache.org

assumed to be available on the metering service provider's systems. Besides enabling accurate billing, high computational power is also required for the offering of NILM services, e.g., to combine data from multiple smart meters through data fusion. In practice, such scalable storage and processing solutions are commonly implemented by means of large-scale database systems. We hence assume a database system to be present on the metering operator's side of the considered smart metering scenario as well.

The communication link between the smart meter and metering service provider can be realized over a wide range of technologies. Without loss of generality, we assume that both the smart metering devices and the hosts of the metering service providers support established Internet protocols (more specifically TCP and UDP over IP) for their networking.

B. Streaming Real-Time Energy Consumption Data

For the real-time transmission of high-resolution data from smart meters, a multimedia streaming protocol is required. In RoS/SoR, we have decided on SIP for this purpose because it combines real-time data streaming (via RTP) with communication primitives for the management of *sessions*. SIP sessions can be established, terminated, and resumed after interruptions. Participating devices can simply send an `INVITE` request to the recipient of their choice to initiate a session. Being a text-based protocol with implementations in numerous programming languages makes SIP a suitable and versatile choice.

The communication flow of SIP is depicted in Fig. 2. The session setup phase is composed of four messages to prepare sender and receiver for the impending session, whereas the subsequent data exchange takes place over RTP. The teardown part is not actively invoked by RoS/SoR, given that the continuous streaming of smart meter data is desired. In case of connection errors, SIP's session recovery feature is being used to eliminate the need to manually monitor the connection state.

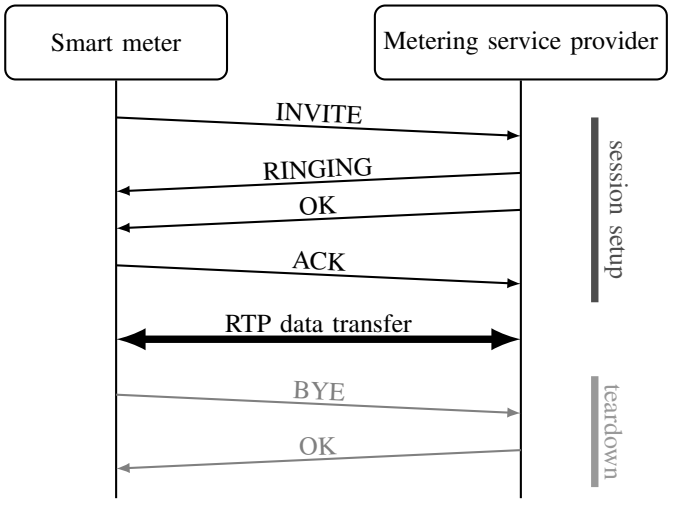

Fig. 2. A typical SIP communication flow is composed of session setup, real-time data streaming using RTP, and the termination of the session. In case of smart meters, the active termination of the session is not being used; rather, the SIP session remains open for an indefinite amount of time, given the continuous stream of energy consumption readings.

C. Synchronizing Buffered Consumption Data

The rsync algorithm was developed to ensure that local and remote files have identical contents [16]. Its operation is based on computing of hash values across contents of a file. The file is separated into blocks of constant size, and two hashes are computed for each block: A *weak* rolling hash to re-align data when parts have been omitted, as well as a *strong* hash to reliably confirm the identity of blocks. Both hash values are sent to the receiving system, which requests updated data blocks when the strong hashes differ. In case larger fractions are missing or different, rsync tries to re-align file contents based on matching the sequence of weak rolling hashes. Overall, its network traffic overhead is small: Besides exchanging hashes for each block of data, only missing data blocks are exchanged between both systems, and only after it has been determined which parts need to be synchronized.

In case of smart meters, the data to synchronize is the record of readings received from the metering component. To make RoS/SoR work in conjunction with smart meter data, we have used the rsync protocol with the following modifications.

1) For efficiency reasons, data on the smart meter are usually stored in the form of files (cf. Sec. III-A), whereas the data collected by the metering service provider will be placed in a database. Thus, the server-side database has been extended by a function to compute the rsync hash values of its stored (non-file) data.
2) Even though SIP and rsync support bidirectional data and synchronization flows, the data on the smart meter can always be assumed to be correct and as complete as possible (barring meter outages, during which no data can be collected at all). Thus, a unidirectional streaming and synchronization suffices to fulfill the requirements.
3) Once data has been collected by smart meters and written to data files, the data is immutable; it will only be deleted and/or overwritten by new data after the successful synchronization with the metering operator's database, or when the smart meter exhausts its storage space. Thus, hashes for previously logged data are unvarying. This allows for the buffering of pre-computed hash values to accelerate the synchronization process.
4) As the database server is much more computationally powerful than smart meters, the comparison of rsync hash values is executed on the database host, whereas the computation of the hashes is performed on the smart meter.
5) Energy consumption data is commonly annotated by the timestamp of its collection. Thus, instead of enforcing files of constant size (which accelerates the operation of rsync), files of constant duration are being used. This simplifies the recovery from missing data, given that rsync does not need to check for the occurrence patterns of rolling hashes in excessively long streams.

As a result of the above considerations, many parts are different from the original rsync workflow. We depict this in the sequence diagram for data synchronization in Fig. 3, which summarizes RoS/SoR's data synchronization operation.

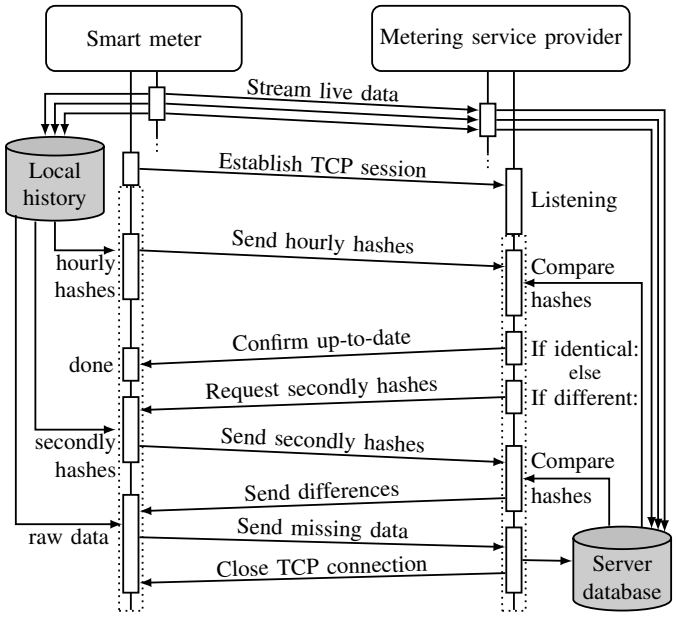

Fig. 3. Besides the constant streaming of live consumption data, the RoS/SoR synchronization flow runs an rsync-like comparison of hash values. Only when changes are detected in the detection of hourly hashes, the missing parts of the data are identified, individually transmitted, and inserted into the database.

D. System Integration

In our complete system design, RoS/SoR is composed of four software components that, when used in conjunction, accommodate the streaming and synchronization of smart meter data.

1) Readings captured on the smart meter are processed by the *streaming transmitter*. It collects data from the local metering component and forwards them to the SIP session for real-time streaming as well as writing them into local storage for their synchronization. Moreover, this component calculates and stores the hash values of all written data blocks to accelerate their synchronization.
2) Its direct counterpart is the *streaming receiver* component, which is running on the metering service provider's premises. It acts as an interface to the database in which the collected consumption data are being stored. It is primarily responsible for offering SIP sessions, receiving data through them, and inserting them into the database.

For the synchronization of data in case of intermittent network connectivity, two more components are used in RoS/SoR.

3) The *synchronization master* is executed on the metering service provider's side. It is responsible for maintaining a complete copy of the data from the smart meter in its database. When a client connects to the synchronization master, it determines the differences between the smart meter's data and its own copy (if any), and uses rsync to retrieve missing values as well as to correct erroneous data.
4) The *synchronization client* is executed on the smart meter and periodically connects to the synchronization master in order to determine whether further missing data needs to be retrieved. In our implementation the synchronization period is configurable and initially set to once per hour, which allows a substantial amount of data to be collected (almost 160 million samples when collecting data at $44\,\mathrm{kHz}$). In practice, the sizes per recorded file as well as the synchronization interval should be specified in accordance with the available storage space on the smart meter and the application requirements.

IV. Evaluation

After having described RoS/SoR's components, we proceed to demonstrate its efficacy in practice. To this end, we have set up a database server and an embedded sensing system with the functionality to generate waveform data at $44\,\mathrm{kHz}$. Both were connected via a $100\,\mathrm{Mbit/s}$ Ethernet link. As RoS/SoR is agnostic to the nature of the streaming data (i.e., it does not apply any data processing step specific to power consumption data), we have relied on synthetic data in our tests. To generate data at a constant rate, we have used the Ostinato packet generator[2]. Generated data are forwarded to the RoS (Real-time over SIP) component, as shown in the top of Fig. 3. Simultaneously, they are logged into a local file-based history, from where the periodically invoked SoR (Synchronization over Rsync) component ensures that all data are up-to-date in the metering service provider's database. For real-time streaming, the data generated by Ostinato was encapsulated into SIP traffic with help of the PJSIP library[3]. The server-side database was realized using a Cassandra[4] instance. Each row of data in the database's table contains a range specifying the period of collection as well as the list of values recorded during this period. Moreover, hash values for the synchronization component are computed upon data insertion, in order to accelerate the rsync synchronization.

A. Test Cases

We analyze two representative test cases which represent common error occurrences in practice, and determine how RoS/SoR reacts when these kinds of failures occur.

1) Network disconnection: While the connection is interrupted, RoS/SoR must reliably buffer data on the smart meter and synchronize them after the link is restored. We simulate this case by temporarily disconnecting the network cable connecting the devices.
2) Bandwidth limitations in the network: When the network only allows a lower throughput than required to stream the data, the transfer shall continue until the sender-side buffers are exhausted. The synchronization step will exchange all missing data when the bandwidth is no longer limited. This situation is emulated by sharing the same link between multiple RoS/SoR clients connected to the same synchronization master.

We do not particularly consider system failures for the following reasons. If the smart meter stops and/or reboots due to a hard- or software issue, energy consumption data will be irrevocably lost until the sensing process is operable again. Thus, RoS/SoR cannot recover from this situation. In contrast, the outage of the metering service provider's server will occur to the smart meter

[2]Available for download at https://ostinato.org/.
[3]Available for download at https://pjsip.org/.
[4]Available for download at http://cassandra.apache.org/.

as if the communication link is down. Thus, when the server program stops, data must be buffered on the smart meter until a reboot of the server allows for their retrieval (or until the smart meter runs out of buffer space).

Across all tests, we allow the system to reach a stable state before disrupting the connection. This ensures that any observed traffic is related to the recovery from link disruptions, and no pending synchronization traffic falsifies the results.

B. Results and Observations

In our first set of test cases, we check RoS/SoR's reaction to network disconnections. To replicate this effect in practice, we have manually unplugged the network connection between server and client, either on the client side (first two cases) or on the server system (third case). This emulates the requirements of our first test case, in which we observe RoS/SoR's reaction to connection interruptions.

Case 1: We simulate a transient disconnection by unplugging the network cable for less than one second before reconnecting it. In this case, the network adapter on the system-under-test reports the connection to still be active. As long as the network link is considered to be available by the operating system, the streaming transmitter continues to send data. Due to the connection-less nature of the underlying RTP protocol, however, the readings contained in these packets will not reach the streaming receiver. This loss of streaming data is later compensated for during the periodically invoked synchronization process. While the server did not receive any data sent during this period, the connection was transparently recovered as soon as the network cable was re-connected.

Case 2: When unplugging the network cable until the operating system reports the network adapter to be in "inactive" or "disconnected" state, the SIP library raises an exception and terminates the session. The streaming transmitter monitors the network state and sets up a new session as soon as a connection is available again. Once the cable had been re-connected, the duration to re-establish the network connection consistently ranged between 2–3 seconds. Across all conducted experiments, the streaming transmitter had re-established the SIP session shortly thereafter.

Case 3: When disconnecting the network cable on the server side, the SIP connection is terminated on the smart meter due to a timeout. The smart meter continues to collect data and writes them into data files, yet the streaming process is inactive until the SIP session has been re-established. Once the server connection is established again, the client re-initiates the SIP session; as per above observation, this takes up to 3 seconds. Data collected during the unavailability of the SIP session is synchronized during the next synchronization phase.

C. Performance Testing

Bandwidth restrictions can exist on the computer network connecting the smart meter and the metering service provider's systems. We assess the throughput over a link of unconstrained bandwidth first to establish a baseline reference, before considering bandwidth limitations in our evaluation. Our first test thus serves as a means to verify to what extent the embedded smart metering system can stream data in real-time. To determine its throughput, we use the following two metrics:

TABLE I
RESULTS FOR STREAMING DATA OF DIFFERENT RATES USING ROS/SOR.

sampling rate	samples / hour	R_{client}	R_{server}
1 Hz	3600	100.0 %	100.0 %
10 Hz	36 000	100.0 %	100.0 %
44 Hz	158 400	99.994 949 %	100.0 %
100 Hz	360 000	99.985 000 %	100.0 %
440 Hz	1 584 000	99.981 565 %	100.0 %
1000 Hz	3 600 000	99.857 666 %	100.0 %
4400 Hz	15 840 000	99.587 575 %	99.999 797 %
10 000 Hz	36 000 000	98.902 211 %	99.999 730 %
16 000 Hz	57 600 000	98.963 600 %	99.999 466 %
32 000 Hz	115 200 000	96.292 703 %	99.973 462 %
44 000 Hz	158 400 000	98.139 782 %	99.747 838 %

$$R_{client} = \frac{n_{logged}}{n_{generated}} \tag{1}$$

$$R_{server} = \frac{n_{received}}{n_{logged}} \tag{2}$$

The client logging ratio R_{client} modeled by Eq. (1) describes the ratio between the number of samples generated (by Ostinato) to the number of entries logged on the smart meter. Under ideal conditions, this ratio should be $100\,\%$, yet under high loads the data generation rate can exceed the system's computational power and occasional entries be skipped. To assess the impact of the communication link, we relate the number of samples collected in the metering service provider's database to the actual number of sent values. This ratio is referred to as R_{server} and defined in Eq. (2).

Results of this performance test when using a set of eleven different data generation rates are shown in Table I. The table shows that a greater loss of samples occurs when the smart meter must generate more data at the same time as logging it to a file, as expressed by the values for R_{client}. In contrast, the combination of real-time streaming in RoS/SoR leads to no losses when data are sampled up to $1000\,\mathrm{Hz}$. Even beyond this sampling rate, losses amount to less than $0.1\,\%$ of the data, when using sampling rates of up to $32\,\mathrm{kHz}$. Only when sampling at the maximum rate RoS/SoR has been designed for, i.e., $44\,\mathrm{kHz}$, losses of $0.25\,\%$ are experienced. In other words, approximately $400\,000$ out of a total of $158\,400\,000$ samples are omitted from their real-time transmission and need to be exchanged during the next synchronization phase. Besides a slight reduction of R_{server} with increasing sampling rates, no outliers were observed in our analysis of the rate of successfully transferred values using real-time streaming.

In a second experiment, we have used an even more computationally restricted hardware platform for the smart meter. As a result, only two streaming frequencies were supported without overloading (and thus crashing) the system: $1\,\mathrm{Hz}$ and $10\,\mathrm{Hz}$. Results for this platform are shown in Table II. Again, all collected data could be successfully streamed to the server, as indicated by $R_{server} = 100\,\%$, while minimal losses were experienced on the data collection platform itself.

TABLE II
RESULTS FOR USING ROS/SOR ON A LOW-POWER DEVICE.

sampling rate	samples / hour	R_{client}	R_{server}
1 Hz	3600	100.0 %	100.0 %
10 Hz	36 000	99.944 444 %	100.0 %

TABLE III
RESULTS WHEN TWO CLIENTS STREAM DATA AT 44 kHz USING ROS/SOR.

Device	R_{client}	R_{server}
Client A	96.998 727 %	99.808 904 %
Client B	97.033 017 %	98.645 684 %

In a third and final experiment, a horizontal scalability test was conducted using two smart meter devices and a single database server. Again, a real-time streaming rate of 44 kHz was used, and both smart meters were sharing a $100\,\mathrm{Mbit/s}$ Ethernet connection. The test results, shown in Table III, confirm the viability of RoS/SoR even in the presence of higher network loads. However, the reception rates drop lower than for the single-client case due to the increased cross-traffic on the communication link.

D. Analysis of the Results

The results may convey the impression that local data logging already leads to losses (as indicated by values of R_{client} below 100 %). Even though these would be compensated through the periodic synchronization step, the explanation for this observation is a different one: In repeated tests, Ostinato did not generate exactly the requested number of packets, but exhibited slight variations. As the packet generator was mainly used to conduct evaluations for different data generation rates, this effects can be easily mitigated in practice by using data from an actual smart meter. With regard to the networking aspects, slight losses only became noticeable for sampling rates of 4.4 kHz and above. At lower frequencies, all the available data was transmitted to the server and stored without loss, proving the efficacy of RoS/SoR for such settings.

V. CONCLUSIONS

The global rollout of smart meter will lead to a torrent of data to collect, consolidate, and process on the systems of metering service providers. In particular, with the rise of energy analytics services like NILM, data of sampling rates in excess of 1 Hz are often needed to reach high accuracy levels. At the same time, communication links interfacing smart meters to data collection points might be lossy, intermittent, or limited in their available bandwidth. These features are rarely considered in today's communication protocols for smart meters, however. As a consequence, we have presented RoS/SoR, a combined solution to provide real-time streaming of captured data while ensuring their lossless synchronization during link outages. Through its reliance on well-established Internet protocols, it is compatible with a wide range of Internet-connected devices. Our evaluations show that RoS/SoR successfully delivers most of the streaming data, even for sampling rates as high as 44 kHz. During link disruptions, data are buffered locally on the smart meter and synchronized at a later time using the rsync algorithm in conjunction with pre-computed data hashes.

A planned future extension to this work is the evaluation to what extent data compression can be applied to accelerate the synchronization phase even more. Based on the observations in [19], the recurrent nature of voltage and current waveforms might lead to considerable bandwidth and time savings.

ACKNOWLEDGMENTS

This research was supported by Deutsche Forschungsgemeinschaft (DFG) grant no. RE 3857/2-1.

REFERENCES

[1] G. W. Hart, "Nonintrusive Appliance Load Monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, 1992.

[2] M. Kahl, A. Ul Haq, T. Kriechbaumer, and H.-A. Jacobsen, "WHITED – A Worldwide Household and Industry Transient Energy Data Set," in *Proceedings of the 3rd International Workshop on Non-Intrusive Load Monitoring (NILM)*, 2016.

[3] T. Picon, M. Nait Meziane, P. Ravier, G. Lamarque, C. Novello, J.-C. Le Bunetel, and Y. Raingeaud, "COOLL: Controlled On/Off Loads Library, a Public Dataset of High-Sampled Electrical Signals for Appliance Identification," *arXiv preprint arXiv:1611.05803 [cs.OH]*, 2016.

[4] T. Kriechbaumer and H.-A. Jacobsen, "BLOND, a Building-Level Office Environment Dataset of Typical Electrical Appliances," *Scientific Data*, vol. 5, no. 180048, 2018.

[5] S. Gupta, M. S. Reynolds, and S. N. Patel, "ElectriSense: Single-point Sensing Using EMI for Electrical Event Detection and Classification in the Home," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (UbiComp)*, 2010.

[6] N. Uribe-Pérez, L. Hernández, D. De la Vega, and I. Angulo, "State of the Art and Trends Review of Smart Metering in Electricity Grids," *Applied Sciences*, vol. 6, no. 3, 2016.

[7] R. Pantos and W. May, "HTTP Live Streaming," Request for Comments, RFC 8216, 2017.

[8] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, 2011.

[9] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP – Standards and Design Principles," in *Proceedings of the 2ond Annual ACM Conference on Multimedia Systems (MMSys)*, 2011.

[10] R. Stewart, "Stream Control Transmission Protocol," Request for Comments, RFC 4960, 2007.

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Request for Comments, RFC 3550, 2003.

[12] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," Internet Request for Comments, RFC 3261, 2002.

[13] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, "Real-Time Streaming Protocol Version 2.0," Internet Request for Comments, RFC 7826, 2016.

[14] Y. Gu and R. L. Grossman, "UDT: UDP-based Data Transfer for High-speed Wide Area Networks," *Computer Networks*, vol. 51, 2007.

[15] V. Skendzic, I. Ender, and G. Zweigle, "IEC 61850-9-2 Process Bus and Its Impact on Power System Protection and Control Reliability," in *Proceedings of the 9th Annual Western Power Delivery Automation Conference*, 2007.

[16] A. Tridgell and P. Mackerras, "The rsync Algorithm," Department of Computer Science, Australian National University, Tech. Rep. TR-CS-96-05, 1996.

[17] Syncthing. (2016) Block Exchange Protocol v1. [Online]. Available: https://docs.syncthing.net/specs/bep-v1.html

[18] Oracle Corporation. (2010) Sun Java System Mobile Enterprise Platform 1.0. [Online]. Available: https://docs.oracle.com/cd/E19957-01/820-3751/6nf8v5huf/index.html

[19] M. Ringwelski, C. Renner, A. Reinhardt, A. Weigel, and V. Turau, "The Hitchhiker's Guide to Choosing the Compression Algorithm for Your Smart Meter Data," in *Proceedings of the 2nd IEEE Conference and Exhibition / ICT for Energy Symposium (ENERGYCON)*, 2012.