On the Scalable Collection of Metering Data in Smart Grids through Message Concatenation

Babak Karimi, Vinod Namboodiri, and Murtuza Jadliwala
Department of Electrical Engineering and Computer Science
Wichita State University
Wichita, Kansas
Email: {bxkarimi, vinod.namboodiri, murtuza.jadliwala}@wichita.edu

Abstract—Advanced Metering Infrastructure (AMI) initiatives are a popular tool to incorporate changes for modernizing the electricity grid, reduce peak loads, and meet energy-efficiency targets. There is the looming issue of how to communicate and handle consumer data collected by electric utilities and manage limited communication network resources. Several data relay points are required to collect data distributedly and send them through a communication backhaul. This work studies the smart meter message concatenation (SMMC) problem of how to concatenate multiple small smart metering messages arriving at data concentrator units (DCUs) in order to reduce protocol overhead and thus network utilization. This problem needs to deal with the added constraint that each originating message from its source may have its own stated deadline that must be taken into account during the concatenation process. Six heuristic algorithms are proposed and evaluated to gain a better understanding of the best data volume reduction policies that can be applied at data concentrators of smart grids.

I. INTRODUCTION

Advanced metering infrastructure (AMI) uses technology to capture and transmit energy use to a collection point on an hourly or sub-hourly basis in contrast to standard meters that provide a daily energy usage total and a cumulative monthly bill [1]. With the introduction of AMI technology, two-way communication between a “smart” meter and the grid operator’s control center, as well as between the smart meter and consumer appliances, would be facilitated for demand-response, dynamic pricing, system monitoring, cold load pick-up, and greenhouse gas-emission mitigation [2]. The information communication and control layer of the smart grid brings about numerous advances, including the empowerment of customers to actively participate in the maintenance of the supply-demand balance around the clock and the resulting reliability improvement in electricity service. There are many benefits to grid operators, consumers, and society as a whole from adopting AMI technologies [3].

Besides AMI, there are many other applications that will be enabled by information flow across the electric power grid. These include distributed generation, state estimation of the power distribution system, demand-side management, to name a few. A big challenge for smart grid application scenarios, and the information-sharing framework that enables them, will be handling the massive amount of data that is expected to be collected from data generators and sent through the communication backhaul to the grid operator. For example, by current standards, each smart meter sends a few kilobytes of data every 15-60 minutes to a smart meter [4], [5]. When this is scaled up to many thousands, existing communication architectures will find it difficult to handle the data traffic due to limited network capacities, especially in limited bandwidth last mile networks[6], [7]. Future applications may require data to be collected at a finer granularity, thus adding to the challenge [8]. Network capacity is a precious resource for electric utilities because they are either leasing such networks from third-party providers [9], or building infrastructure themselves and leasing bandwidth out (especially at the backhaul) to recuperate investment costs [10]. In either case, it is in the interest of electric utilities to reduce the volume of information transported through these networks for smart grid applications while ensuring application QoS requirements are met.

One approach to reduce data volume given some application sampling rate is to concatenate multiple messages into a larger packet to reduce protocol overhead due to packet headers. This approach has the potential to reduce network capacity requirements significantly (quantified later in this paper) due to the small size of messages sent in smart metering networks with packet headers possibly being of a comparable size to the underlying message to be sent. Such concatenation of messages can be done by each smart meter itself. However, each meter may not generate messages frequently enough to be able to have the chance to concatenate enough packets to reduce overheads significantly and also meet their stated application deadlines. Each meter is also expected to be relatively constrained (compared to a concentrator) in terms of data storage capabilities to keep a large window of packets from which to aggregate. Thus, a better approach is to concatenate messages at an intermediate point upstream from individual meters.

Such an intermediate point where message concatenation can be done is at data concentrator units (DCUs) (or some similar entity, sometimes also called a data aggregator) that collect data from many smart meters and forward them upstream. Figure 1 depicts this concept and shows the DCU’s role at the power-distribution level of the power grid. Data concentrators or aggregators can play an important role in reducing network capacity requirements by reducing packet protocol overhead through message concatenation algorithms applied along the data collection tree. Such algorithms and policies, however, do not exist currently and need to be developed keeping in mind the unique characteristics of metering data like variable packet sizes, stochastic arrivals, and the presence of messages with and without deadlines. Current DCUs on the market lack the ability to reduce the volume of data flowing through them and

This work was supported in part by the Power Systems Engineering Research Center (PSERC) Project S-54.
real-time aggregation capabilities. They only provide simple integration of sensing and WAN communications options with the intention to follow the PRIME standard [11], [12] which gives the utilities the freedom to choose meters from various vendors and avoid being reliant on proprietary solutions from a single source.

In this paper we design and comparatively evaluate a suite of online message concatenation algorithms at DCUs in the AMI scenario that minimize usage of network capacity in transporting data through the meter data collection network while meeting quality-of-service (QoS) constraints imposed by applications on individual messages. The specific contributions of this work include:

1) A formulation of the message concatenation problem at DCUs in smart metering networks to minimize network capacity utilization
2) Multiple heuristic online message concatenation algorithms that can be employed at DCUs for the message concatenation problem
3) A comparative performance evaluation of proposed heuristic message concatenation algorithms

Our results indicate that the proposed concatenation heuristic algorithms can easily reduce data volume by 10-25%, with greater benefits seen for scenarios with greater data traffic rates. These benefits are obtained operating only on packet headers without compressing or aggregating the underlying information in messages.

II. PROBLEM STATEMENT

A. Motivation

In most communication protocol suites (e.g. TCP/IP) used for sending smart metering messages, the small size of packets will result in a high amount of protocol overhead due to packet headers. For example, for messages of size 100 bytes from the source smart meter, there may be 40-60 bytes of additional header overheads due to TCP/IP protocols and specific versions used. If a data concentrator collects multiple packets and strips off all individual headers and includes only one header for the larger aggregated message, there could be significant reductions in network capacity utilization. Studying the messaging format as shown in Figure 2 for the ANSI C12 smart meter communications standard provides an idea of message sizes involved and the amount of protocol overhead to expect. Each smart meter generated message includes parameters like meter identification number, equipment status, type of message, among others. This information is enough to uniquely identify a message source with no additional protocol header information required for source identification. Thus, source protocol headers can be stripped away to rely only on a common aggregated packet header to route the packet to the destination.

In Table 1 (abstracted from [6]) basic message types along with their properties are listed. It can be seen that messages can be of various sizes, and can have loose or strict deadlines, or no deadlines at all. Some messages may be generated randomly at any time to indicate critical events that need to be responded to immediately. Data concentrators will have the challenge of handling these varying message sizes that may or may not have deadlines, with possibly stochastic arrivals, at the same time guaranteeing that each message meet any specified deadline. Stochastic message generation and critical events with short deadlines exclude the use of polling based algorithms to collect data at DCUs.

B. The Smart Metering Message-Concatenation Problem

The smart metering message concatenation (SMMC) problem considered in this paper is as follows. A DCU receives different types of messages from smart meters with a stochastic arrival process (we will discuss this arrival process later in Section IV). Each message can be of different size and comes with a deadline by which it must reach the common destination that is the utility control center. Each message has protocol overhead as it is packaged into a packet before being sent to the DCU. The DCU can either send each packet to the destination as it arrives as a single message or wait and concatenate multiple messages before sending them out over the backhaul to the destination. The objective considered is to minimize the number of individual packets sent upstream by the DCU so as to reduce network capacity requirements of the backhaul. The constraint set is that all packets meet their deadline (if
A. Nature of the SMMC Problem

B. Heuristic Algorithms

the idea described above. Messages whose deadline is about to expire. Figure 3 depicts they decide to fill in the concatenated packet in addition to the other messages (in addition to the message whose deadline is about to expire). They differ in terms of what other messages (in addition to the message whose deadline is about to expire) are put in the packet being sent out.

In all six schemes the aggregation process starts with the main procedure which is required to be run consistently over the DCU. The DCU-OMA (Optimized Message Allocation) procedure is an infinite loop as defined in Procedure 1. The “Classifier” module checks the arrived messages to see whether they are best-effort or having a specific deadline (if the selected heuristic requires to differentiate between them). Two different queues observable in Figure 3 are formed based on the classification done. All deadline messages are kept in a priority queue sorted by earliest deadline. It is assumed there are two queues in the system, one for the messages with specific delay objective ($Q_d$) and the other for those without delay objective (the best effort messages, $Q_{nd}$). The call to function $bufferQueues$ inserts the messages with deadline to the first queue using sub-function $enqueue_d(M)$ and buffers the best-effort messages into the second queue using sub-function $enqueue_{nd}(M)$. If no classification is required then all arrived messages will be sorted and placed in a single buffer using $enqueue(M)$.

### Procedure 1 Main

1. procedure DCU-OMA
2. $alg ← selectAlgorithm()$
3. repeat $▷$ forever
4. $bufferQueues(alg)$$\triangleright$ Buffer messages that have missed their deadline 5. $checkQueues()$$\triangleright$ Check the message deadline
6. until DCU fails.
7. end procedure

### Module 1 - Classifier

8. function $bufferQueues$
9. if $alg$ is “EDF-FCFS” or “EDF-KN” then
10. $enqueue(M)$$\triangleright$ Insert deadline messages as much as possible inside the packet and the remaining space will be filled through knapsack selection over best-effort messages that have been queued.
11. else
12. if delay objective is best-effort then
13. $enqueue_{nd}(M)$$\triangleright$ Reverse order of knapsack process in EDF-KDKB working first on the queued deadline messages and then on the message deadlines if needed.
14. else
15. $enqueue_d(M)$$\triangleright$ Insert messages from the second queue with any available space filled with non-deadline messages in the non-deadline queue through the knapsack algorithm.
16. end if
17. end if
18. end function

### Module 2 - Scheduler

19. function $checkQueues$
20. while Queue(s) are not empty do
21. $drop(M_{missed})$$\triangleright$ Drop messages that have missed their deadline
22. $EDFsort$$\triangleright$ Earliest Deadline First(deadlines in ascending order)
23. $pkt ← createPacket()$$\triangleright$ Earliest Deadline First(deadlines in ascending order)
24. $sendPacket(pkt)$$\triangleright$ Earliest Deadline First(deadlines in ascending order)
25. if $addNDM.size > τ$$\triangleright$ If no classification is required then all arrived messages will be sorted and placed in a single buffer using $enqueue(M)$.
26. $addNDM()$$\triangleright$ Sort deadline messages by earliest deadline
27. $sendPacket(pkt)$$\triangleright$ Send the packet
28. end if
29. end while
30. end function

### Table II. The Proposed Concatenation Heuristics

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDF-DKB</td>
<td>Inserts deadline messages as much as possible inside the packet and the</td>
</tr>
<tr>
<td>EDF-SDKB</td>
<td>Only a single deadline message sits inside the packet with any available</td>
</tr>
<tr>
<td>EDF-FCFS</td>
<td>Messages will be placed in the packet according to their arrival sequence</td>
</tr>
<tr>
<td>EDF-KN</td>
<td>Messages are chosen from a common pool of deadline and best-effort</td>
</tr>
<tr>
<td>EDF-KDKB</td>
<td>A sequence of knapsack selections first on all queued deadline messages</td>
</tr>
<tr>
<td>EDF-KBKB</td>
<td>Reverse order of knapsack process in EDF-KDKB working first on the</td>
</tr>
</tbody>
</table>

1A formal proof is omitted due to space limitations. For interested readers, a similar problem is the scheduling with release times and deadlines on a minimum number of machines (SRDM) [14].

![Fig. 3. Data concentrator schematic concept.](image-url)
Module “Scheduler” always checks the queues and does the appropriate operation when they have some messages inside them. Here the EDF algorithm is utilized. A packet is formed “just-in-time” before the deadline of the first message in the priority queue expires, with each of the heuristic algorithms having a different approach on how to maximize the size of the packet that is sent out.

### Procedure 2 Packet Creation

1. **createPacket**
   - $TPS \leftarrow 0$
   - $i \leftarrow 1$
   - $deadline \leftarrow currentTime + D_i$
   - switch alg do
     - **case 1 - EDF-DKB**
       - $deque(M_i)$ → Put the first messages with deadline in the packet
       - while $size(TPS + M_i + Header) \leq MTU$ do
         - if (deadline is not passed) then
           - $deque(M_i)$
         - append($TPS, M_i$)
       - $end$ while
     - **case 2 - EDF-SKDB**
       - Do the same procedure of EDF-DKB but with a single loop
     - **case 3 - EDF-FCFS**
       - while $size(TPS + M_i + Header) \leq MTU$ do
         - $deque(M_i)$
         - $TPS \leftarrow append(TPS, M_i)$
       - $end$ while
     - **case 4 - EDF-KN**
       - while (deadline is not passed) do
         - $updateDeadline$
       - $end$ while
     - **case 5 - EDF-KDKB**
       - while (deadline is not passed) do
         - $waitList = deque(M_i(Q_d))$
       - $updateDeadline$
     - **case 6 - EDF-KBKD**
       - $deque(M_i(Q_d))$ → Put the first messages with deadline in the packet
       - while (deadline is not passed) do
         - $waitList = deque(M_i(Q_d))$
         - $updatedDeadline$
   - $end$ while

During the main procedure the most important call is to the createPacket procedure. Let’s assume $M_i$ is the $i_{th}$ message in the deadline queue. Function $deque(M_i)$ moves message $M_i$ from the queue of DCU to the outgoing packet whereas function $reque(M_i)$ inserts back the message $M_i$ to the end of the queue if it was not selected during knapsack selection. Finally it is assumed that at any given time $TPS$ is the current “Total Packet Size” inside the DCU. That is, packet is being formed using procedure 2 which when called will create the packet depending on the heuristic algorithm selected.

Since messages with delay objective have always the higher priority to the best-effort one, it is important to know when and how to add non-delay objective messages to the outgoing packet. Procedure 3 utilizes the 0-Knapsack algorithm [13] to accurately append best-effort messages to the packet when possible.

### Procedure 3 Best-effort Messages Addition

1. **addNDM**
   - $B[i] \leftarrow benefitFP(M_i(Q_{nd}))$
   - $TPS_{nd} = knapsack_{0 \rightarrow 1}(B[i], Q_{nd})$
   - append($TPS, TPS_{nd}$)
   - $return TPS$
   - $end procedure$

Function $benefitFP(N)$ in Procedure 3 is checking the resulting benefit out of placement of all the messages in a waiting list during packet creation and stores them in an array of positive integer values, $B[i]$ (the inverse value of remaining space after placement of a particular message in packet). This array is then used in the 0-Knapsack algorithm.

### C. Reference Algorithms

1) **Mixed Integer Linear Program (MILP) Algorithm**

To get a solution for the SMMC problem one can use mathematical optimization algorithms. We have formulated the SMMC problem as a mixed-integer linear programming which optimally schedules the messages in addition to the EDF message to begin a packet. The problem is formulated as:

**Indices**
- $i$: index of messages
- $j$: index of packets

**Parameters**
- $M_i$: Size of $i_{th}$ message in the buffer
- $MTU$: Maximum Transfer Unit
- $H$: Size of the header

**Objective**

$$\max \sum_{i,j} M_i X_{ij}$$

**Constraints**

$$\sum_{i,j} M_i X_{ij} + H \leq MTU \quad \forall (i,j)$$

$$X_{ij} \in \{0, 1\} \quad \text{Binary Constraint}$$

The given constraint in the model specifies the maximum packet size to be defined. Since this is a maximization problem, the model will assign messages to the packets as much as possible unless it exceeds the maximum transfer unit when including the packet header size $H$. The deadline constraint is already met by following the EDF approach of starting a packet when a deadline approaches. The drawback of this approach in practice (as opposed to our heuristics) is the brute force nature of this MILP solution procedure which makes it practically infeasible for real-time applications and those that involve large-scale data.
2) Theoretical Optimal: This method is the theoretically minimal number of packets that needs to go out of a DCU for a given number of messages generated from the smart meters over a period of time. This value is not constrained by arrival times or deadlines of messages; it is computed over all generated messages and maximum packet size MTU that includes a header size $H$. This value can be mathematically calculated through the equation:

$$\min(N_{\text{packets}}) = \left\lceil \sum_{i=1}^{n} \frac{\text{size}(M_i)}{\text{MTU} - \text{size}_{\text{header}}} \right\rceil$$

where $n$ is the total number of arrived messages during a time interval, and $M_i$ is the size of a message $i$. MTU and header size $H$ are the parameters defined according to the backhaul technology. Although this solution is not feasible in practice, it gives a theoretical reference for the performance evaluation of any SMMC algorithms, not limited EDF based heuristics.

IV. Evaluation

A. Methodology

We outline below more details about the simulation environment, message arrival process, and distribution of various message types.

1) Simulation Environment: A discrete-event simulator was developed using MATLAB to evaluate the proposed heuristic algorithms and compare to the reference algorithms. The network topology consisted of a group of smart meters generating messages as poisson arrival process and sending messages to the DCU to be routed to the control center. Due to the assumption of each meter generation as a poisson process, we can sum individual average message generation rates to get an average arrival rate at the DCU of $\lambda$ which is used as a parameter. We have considered three different $\lambda$ values of 0.1, 0.5, and 1 at the DCU which would correspond to 90, 450, and 900 smart meters sending 1 message on average every 15 minutes. The service capacity of the DCU is considered to be infinite. An assumption made in this work is that processing delays at the DCU can be ignored in constructing a concatenated packet from a queue of messages. The simulation time interval was kept at 300 seconds.

2) Message types distribution: During a day, different types of the messages may be exchanged between smart meters and the utility control center through the AMI. In our evaluations we have considered all seven basic types of messages listed in Table I and first reported in [6]. Based on geographic location, power distribution infrastructure, and utility preferences, the transmission of messages could come from different distributions of these basic message types which will have an impact on the performance of our proposed heuristics. In our evaluations we used different Beta distributions across these message types by varying shape parameters $\alpha > 0$ and $\beta > 0$.

Assuming that the arrival probability of different message types is $p_1, p_2, \ldots, p_n$ with $0 < p_i \leq 1$ for $i = 1 \cdots n$ and $\sum_{i=1}^{n} p_i = 1$ we can compute each as the result of difference between cumulative probabilities of $n$ intervals as follow:

$$p_i = \text{betacdf}(i) - \text{betacdf}(i - 1)$$

For our experiments we generated five different message type distribution using the shape parameters mentioned in Table III to test the performance of our proposed algorithms.

TABLE III. PRE-DEFINED MESSAGE ARRIVAL DISTRIBUTIONS

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform ($\lambda = 1, \beta = 1$)</td>
<td>The traffic would have almost equal percentage of all message types.</td>
</tr>
<tr>
<td>More smaller ($\alpha = 0.18, \beta = 0.25$)</td>
<td>There is higher percentage of large message size and very few numbers of small size messages.</td>
</tr>
<tr>
<td>More larger ($\alpha = 0.18, \beta = 0.25$)</td>
<td>Most of the times there are incoming messages with deadline restriction.</td>
</tr>
<tr>
<td>More deadline ($\alpha = 1, \beta = 1.8$)</td>
<td>Most of the times there are incoming messages with deadline.</td>
</tr>
<tr>
<td>More best-effort ($\alpha = 2.5, \beta = 0.5$)</td>
<td>There are very few numbers of messages with a deadline and so many best-effort messages.</td>
</tr>
</tbody>
</table>

B. Simulation Results

100 runs of simulations were conducted with mean value plotted in our results along with 95% confidence intervals. Each scheme was evaluated in terms of the overall reduction in bytes of data transmitted out into the backhaul network by the DCU as compared to the overall incoming data in bytes from smart meters, including all headers. Each packet header was assumed to be fixed size of 50 bytes corresponding to the 40-60 byte range for TCP and IP headers. Figure 4 displays the output of our proposed algorithms and reference algorithms over five message types distributions with 95% confidence intervals. Results are shown for packet arrival rates at the DCU of $\lambda = 1, 0.5, \text{ and } 1$. It can be seen that overall data volume reduction varies from 5-25% depending on message type distribution, message arrival rate at DCU, and specific algorithm used. Three questions answered are:

1) How do the proposed heuristic algorithms stack up against each other and reference algorithms? Taking a look at the bar charts in Figure 4 one can observe that the algorithm EDF-KN has the best performance among all other heuristic algorithms and comes very close to the performance of the MILP across all $\lambda$ and message type distributions. This is due to the fact that EDF-KN is using a common pool of messages whether they be deadline or best effort, giving more options to maximize packet size before it is sent out. Since typically there are enough queued messages before a deadline reaches, the algorithm has a good collection of options to maximize the packet before sending it out.

2) What is the impact of message type distribution?: The uniform distribution of all message types serves as the reference case to compare other distributions. For the more deadline case with a majority of all messages having deadlines, overall data volume reduction is smaller for all algorithms. Presence of more messages with deadlines than best-effort necessitates packets to be sent out of the DCU without having the luxury of waiting for the right combination to maximize packet size. On the other hand when there are most best-effort packets present, algorithms can wait longer before being forced to send out packets; this allows each packet to be larger, and hence reduces packet overheads. The case for more smaller size messages is similar to the more deadline message case in that it helps reduce packet overheads significantly through concatenation as

where betacdf is the cumulative density function of beta distribution used with parameter $\alpha$ and $\beta$.

Prior work in [16] supports this assumption that smart meters message generation can be modeled as a poisson process.

This assumption would not make a difference in comparing all the proposed heuristics if queue sizes remain relatively small; however for large queue sizes, there may be an impact that does not come out in our evaluation results in the following section.
header sizes are comparable to data sizes. Smaller messages are also easier to pack into a packet. Conversely, the more larger messages case results in greater difficulty to packet messages into a packet; also larger underlying message sizes already have reduced overhead making much improvements through concatenation difficult.

3) What is the impact of $\lambda$?: The value of $\lambda$ signifies the packet arrival rate at the DCU; hence, larger values indicate that more messages are arriving at the DCU and more packets need to be sent out after concatenation. With greater data volume, there are more opportunities for each concatenation algorithm to find a best fit of messages in an outgoing packet from the DCU to reduce overall protocol overhead. The EDF-KN data volume reduction approaches very close to that of even the theoretically optimal solution with increasing $\lambda$. Thus, greater the rate of packet arrivals, the proposed EDF-based concatenation algorithm over a common queue of messages maximizes the reduction in data volume.

**V. CONCLUSION AND FUTURE WORK**

This paper demonstrated that message concatenation algorithms can be an important element of data concentrators deployed in smart grids to solve the looming challenge of transporting massive data volumes through last mile bandwidth-constrained backhaul networks. Effective message concatenation algorithms at DCUs (such as the EDF-KN algorithm proposed in this paper) were shown to be able to reduce overall data volume by 10-25% for each DCU. This reduction was achieved just by a reduction in protocol overhead with no compression of the original data sent by smart meters. This leaves much room to develop additional data concentration mechanisms at DCUs.

**REFERENCES**


