Size Efficient Messaging System (SEMS) Using Probabilistic Data Structures

Samuel Sungmin Cho[*], Benjamin Gooder[*], Myoungkyu Song[†]
[*]Northern Kentucky University, [†]University of Nebraska at Omaha
Email: [*]chos5@nku.edu, [*]gooderb1@nku.edu, [†]myoungkyu@unomaha.edu

***Abstract*—A Short Message Service (SMS) is a tool for personal communication systems using ubiquitous telephone networks. SMS also has been used for sharing information between intelligent devices, but the transferable payload size of a single SMS message is limited and can constrain its application. For example, when sharing context information between devices, a message size of more than 140 bytes needs be divided by the sender and reassembled by the receiver to fit into SMS messages. Although efforts have been made to reduce the context size using compression algorithms, their effectiveness is unreliable and their size reduction is, in many cases, not enough. We present SEMS (Size Efficient Messaging System) to reduce the footprint of context information to overcome this limitation. Using a probabilistic data structure, SEMS can significantly increase context representation size. SEMS also uses SMS to deliver the encoded message with far less overhead. Despite a slight potential decrease in data quality, our system and findings are shown to facilitate the sharing of context information in a highly size-efficient manner.**

I. Introduction

A Short Message Service (SMS) system is a popular tool for personal communication systems using ubiquitous telephone channels [1]. With the introduction of mobile devices, SMS has become one of the most frequently used systems of communication. Many services are integrated with the SMS system to enhance its usability: banks use SMS to send code to users to enhance security, hospitals use SMS to check doctors' appointments, and restaurants send SMS messages to confirm dinner reservations. The Internet of Things (IoT)[2] has introduced another usage of SMS: SMS messages are used to share context information among devices [3].

Context is any kind of information to represent a status of an entity[4][5]. Context awareness[6] is a critical part of pervasive computing including IoT[7] because intelligent system can make use of the context information to benefit users. Due to its ability to send text messages that both devices and humans can read and process, SMS is one of the simplest and effective ways of sharing context information [3].

However, SMS has constraints that restrict its capability as an effective communication system [1]. The payload size that SMS can transfer is limited: a maximum of 140 bytes can be used for a single text message within an SMS system. This size limitation is relatively innocuous when people share personal information with others, since a longer message gets split into multiple sub-messages without any practical problems. For context sharing among devices, however, the splitting, distributing, and combining of one context into multiple smaller contexts may introduce unnecessary complexities. Although reducing context size using compression can be a solution to this issue, size reduction using compression is effective only when the bit-pattern in the context message is regular. Therefore, one cannot guarantee the size reductions using compression algorithms.

In this paper, we present SEMS (Size Efficient Messaging System) to address this issue. SEMS uses probabilistic data structures to reduce the footprint of context information. Specifically, Bloomier Filters [8] are used to encode and decode context information. Furthermore, we use Folded Bloomier Filters (FBFs) [9] to overcome the limitation of Bloomier Filters, which restricts the size of values' encoding and enhances the size reduction rate even further. This dramatically eases the single message size constraints inherent to SMS communication systems.

However, probabilistic data structures have false positive probabilities that can lead to the possibility of misinterpretation of the encoded data when recovering original information. In this research, we shows that how we can manage the false positive probabilities of SEMS to be low enough to be used in context sharing using SMS systems.

Our contributions in this paper are as follows:

- We propose SEMS, which can overcome the payload limitation of a typical SMS system through the application of probabilistic data structures.
- We assess the benefits and challenges of using the SEMS by measuring both the size reduction efficiency and false positive probabilities when using SEMS.
- We demonstrate how SEMS can be used to share different types of context information flexibly and size-efficiently.

The rest of this paper is organized as follows: we explain related work in Section II. Then we introduce the theoretical research, concepts, and architecture of SEMS in Section III. In Section IV we show the experimental results by comparing how the original data in JSON format can be shared in a size-efficient manner using SMS with little or no degradation of data quality. Section V concludes this paper.

II. Related Work

Standardization of communication between devices is one of the largest challenges facing pervasive computing [10]. The universality of SMS offers the infrastructure necessary to facilitate the easy implementation and high dependability of inter-device communication [1]. Despite these strengths, SMS has a limit of 140 bytes [3]. If an SMS exceeding this size is sent, it can be segmented into multiple (multipart) SMS messages that can arrive at the receiving device in any order to be then properly reordered. However, this approach using multipart message can complicate the process of encoding and decoding the original message, as it introduces the partitioning and necessary reordering of the original message when sending the multiple sub-messages. Furthermore, this process is a source of potential problems since JSON [11], a commonly used text format in inter-device communication, can easily exceed 140 bytes [11].

Efforts have been made to reduce the size of context information in a number of ways. Compression is popularly used in the encoding of message subsections. Encoding scheme types vary greatly in that they target different subsets of the original message. Our study [9], we showed that the zip compression method that uses Huffman coding [12] can reduce the footprint size of context information.

However, context information size can be further reduced using probabilistic data structures. Bloom Filters [13] map input to binary values—-true or false—-to reduce the set associations. Bloomier Filters [8] are a generalization of Bloom Filters, with the introduction of a table to store various values that require more than simple set associations. Folded Bloomier Filters (FBFs) were introduced to overcome the limitation of Bloomier Filters [9], [14]; Bloomier Filters use a fixed size table that not only limits the maximum bit-width of the stored values, but also wastes the table space when the stored values have shorter bit-width than the table width. Using FBFs, we can use any type of information including strings, integers, and floating point numbers. It was shown that FBF can achieve a size reduction of up to 87.42% in context footprint size with a near zero false-positive rate [9] When we consider the relationships among entities, the false-positive rate can be even further reduced [15].

III. SEMS Architecture

The SEMS architecture uses an intelligent algorithm to enhance the usability of SMS. Fig. 1 shows an overall architecture of SEMS. The SEMS architecture assumes JSON is being used to share context information as in ❶ and ⓫. The JSON format is flexible and simple with its dictionary structure that maps keys (e) to values ($f(e)$). JSON uses text for both keys and values. As a result, the footprint for data representation using JSON tends to be large, especially when the schema (the structure of keys) is complex and lengthy.

This example shows an example of context information represented in JSON format to be shared using SMS.

```
{
  "date": [15, 10, 11],
  "latitude": [30, 25, 38, 5],
  "longitude": [-17, 47, 11, 0],
  "temp": 91
}
```

When the size footprint of the JSON is more than 140 bytes, the context information cannot be shared without splitting the JSON into smaller JSON sub-files in normal cases. However,

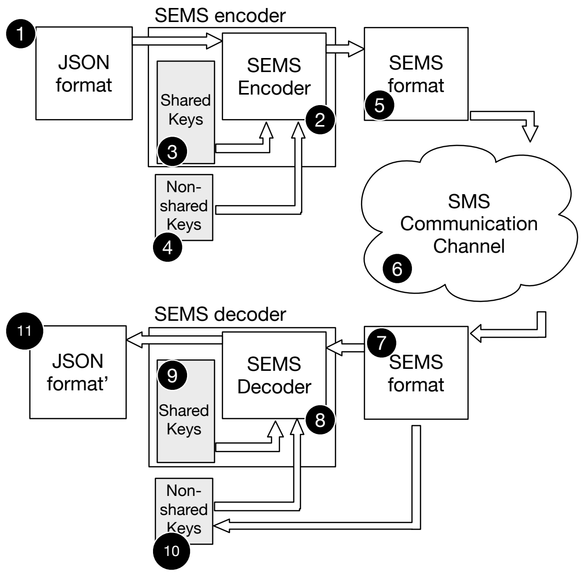
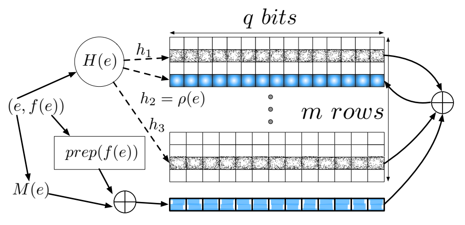
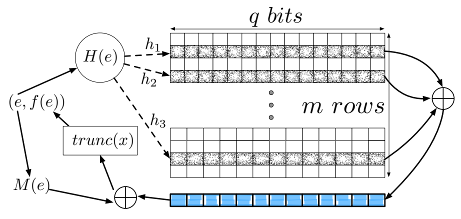


Fig. 1: SEMS Architecture



(a) Encoder



(b) Decoder

Fig. 2: BF/FBF Encoder and Decoder Process

this division of representation of context information requires rules and algorithms for the proper segmentation of JSON files. Furthermore, these partial JSON files should then be combined and their final representation validity should be verified upon delivery. These added difficulties can add unnecessary burden when sharing context information. The SEMS encoder ❷ uses probabilistic data structures, i.e., FBF, to address this issue.

Bloomier Filters encode a pair of (key, value) information into a table. The first diagram of Fig. 2 shows their encoding process [8][1]. Bloomier Filters have a table of size m by q: m is the number of table rows, and q is the maximum width of any associated value. This table has a constraint $m \geq n$ where n is the number of stored key value associations (e, $f(e)$). To reduce the table size even further, we can introduce m bits

[1]The diagrams in Fig. 1–4 are from our previous work [9]

of an array to indicate which table rows are occupied. This makes the footprint of $(q \times n) + m$. Compared to the JSON footprint size, this table size provides a great size reduction. FBF further reduces the table size by folding the table.

Fig. 3 shows a Bloomier Filter table example with three values of different types: `{"name":"Jim", "age":7, "time": 12,00}`. The time is encoded into two bytes of data (0x03, 0x00). The string value of "Jim" becomes (3, 'J', 'i', 'm'); The first value 3 indicates the length of a string. The grayed bytes show prepended zeroes, which add unnecessary table space.

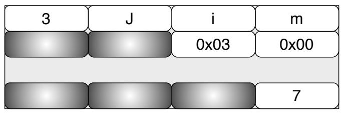


Fig. 3: BF table without folding

Fig. 4 shows the same values folded twice into a table of width one byte width. FBF uses the same data structure, encoding, decoding algorithms, and table at its core, but pre-processes the associations of $(e, f(e))$ to reduce the table size [9].

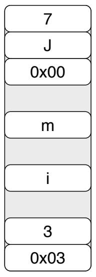


Fig. 4: FBF table with folding

FBF does not store a schema of the JSON representation to reduce the information footprint. The value that matches a key is stored in one of the locations that the Hash ($H(e)$) function generates. In this example, out of the three generated locations, the 2^{nd} hash output ($h_2 = \rho(e)$) becomes the location where the $f(e)$ is finally stored after (1) it is prepended, (2) XOR'ed (exclusive or) with $M(e)$, and XOR'ed with the table values that h_1 and h_3 point to.

This can be a problem, in terms of flexibility, when we recover the JSON from the SEMS, as the receiver cannot retrieve values without the schema information from the sender. Another problem when we use FBF probabilistic data structures is the possible false positive probability, where the key that is not a part of schema wrongly generates a value that is accepted as a valid one. As an example, the second diagram of Fig. 2 shows a decoding process where any random locations from key e can be XOR'ed to generate a value $f(e)$ that might be accepted as a valid information. This can be another problem, in terms of data quality, when we recover a wrong JSON representation from the transferred SEMS format using SMS messages.

To address the first issue—the flexibility problem—the SEMS encoder uses frequently used keys, and these keys are already shared between a sender ❸ and a receiver ❾. In a scenario where we should share a key-value pair whose key is not a member of shared keys, we keep a record of non-shared keys ❹ and send them over a SMS communication channel ❻ as a part of the SEMS format ❺ ❼. The shared keys ❾ of the receiver are used to recover JSON representation ⓫. When the SEMS format ❼ includes the non-shared key, they are also used to reconstruct JSON representation ❿ by the SEMS decoder ❽. This approach not only reduces the information, but also enables sharing information flexibly.

To address the second issue—the data quality problem—SEMS uses various types of filters to remove false positives. As innate filters, $prep$, $M(e)$, and $trunc(x)$ functions in BF/FBF decoder and encoder are used to reduce the false positive probability. Additionally, correlational filters can reduce the false positives dramatically. As an example, to constitute a location, we need both 'latitude' and 'longitude' information. Using this correlational information, we can more accurately detect if the location information recovered from a SEMS format is false positive or not. The probability that both 'latitude' and 'longitude' are false positives becomes significantly lower [15] than the probability where only 'latitude' or 'longitude' information is false positive. Further, we can enhance the data quality by filtering out the invalid data from situational information [9].

IV. Experiments

In this section, we use three different scenarios to assess the performance of SEMS both in size reduction and data quality (false positive probabilities). The size reduction rate shows how much SEMS can communicate information in a size-efficient manner, and data quality shows how the SEMS format can retain its quality.

In the first scenario (EMS), a person calling emergency medical services to pick them up from their home and transport them to the hospital uses a biotech sensor patch to take a set of their vital signs and sends SEMS so that emergency medical services personnel can have a better understanding of this person's condition prior to being on the scene.

```
{
  "name": "William",
  "date": [15, 10, 11],
  "past medical history": "Cardiac arrest",
  "medications": "Clonazepam",
  "heart rate": 85,
  "blood Pressure Systolic": 135,
  "blood Pressure Diastolic": 85,
  "sp02": 95,
  "respirations": 12,
  "allergies": "sulfa drugs",
  "latitude": [-17, 47, 11, 0],
  "longitude": [-17, 47, 11, 0]
}
```

In the second scenario (weather), a ship is sent a weather report specific to its location that updates it on the condition of the water and current climate.

```
{
  "date": [15, 10, 11],
  "latitude": [30, 25, 38, 5],
  "longitude": [-17, 47, 11, 0],
  "temp": 91,
  "significant wave height": 12,
  "swell height": 30,
  "swell period": 41,
  "swell direction": "west",
  "tide data": 18,
  "wind speed": 16,
  "visibility": 50,
  "unique weather condition code": "TG188",
  "weather description": "stormy",
  "pressure": 43,
  "cloud cover": "humidity",
  "water temp": 43,
  "humidity": 70,
  "max temp": 95,
  "min temp": 70
}
```

In the final scenario (farming), a farmer has placed down a number of sensors in a field to monitor changes in soil acidity across specific predetermined sectors of the field. The farmer receives each sensor's individual reading as well as an indication of the crop being grown in the specific sector.

```
{
  "time": [11, 21],
  "date":[15, 10, 11],
  "field health index": 85,
  "sensor group": "soil ph",
  "sensor 1 location": 5,
  "sensor 1 crop": "soybeans",
  "sensor 1 value": 5.9,
  "sensor 2 location": 3,
  "sensor 2 crop": "barley",
  "sensor 2 value": 43,
  "sensor 3 location": 1,
  "sensor 3 crop": "wheat",
  "sensor 3 value": 6.8,
  "sensor 4 location": 2,
  "sensor 4 crop": "sorghum",
  "sensor 4 value": 5.7,
  "sensor 5 location": 7,
  "sensor 5 crop": "soybeans",
  "sensor 5 value": 43,
  "sensor 6 location": 4,
  "sensor 6 crop": "barley",
  "sensor 6 value": 6.8,
  "sensor 7 location": 8,
  "sensor 7 crop": "soybeans",
  "sensor 7 value": 5.7
}
```

In each of the scenarios listed above, the intended message is much greater than a single SMS message's 140 byte limit. Therefore, unless otherwise specified, SEMS encodes the data representation and yields a binary message of a size smaller than the SMS limit. This is then decoded via SEMS on the receiving device for queryable access to the data representation and, thus, foregoing the complex partitioning and reordering of a multipart SMS message.

Although the shared schema should cover the majority of cases, if the sending device wants to add a key to the receiving device's schema, then SEMS would allow a user to send a non-shared part of the schema (ie. a new key) in addition to the expected data representation according to the shared schema. Further experimental comparison was done to determine the effects of sending additions to the schema that were not previously shared using the example scenarios and respective JSON data above.

As an example of this dynamic schema addition, consider the second scenario regarding the weather report. If a vessel is looking for location-specific information regarding the waters, then local authorities can update the schema by sending the ship the originally requested for weather report data representation and a schema addition such as restricted areas, ports of interest, where fuel can be bought locally, or a notification of dangerous areas. In this way, the limitation inherent to only sending encoded data representation of previously shared schema is overcome with the ability to send this non-shared key in addition to the encoded data representation sent.

Table I shows the size reduction rate of the three given scenarios assuming a shared schema between the sender and receiver in comparison to the size reduction rate achieved via Zip compression. Table II shows the size reduction rate of three scenarios assuming a shared schema between the sender and receiver with the addition of a key not in the previously shared schema in comparison to the size reduction rate achieved via Zip compression.

TABLE I: Reduction rate (%) using shared keys

	EMS		Weather		Farming	
	Size	Rate (%)	Size	Rate (%)	Size	Rate (%)
Original	284	0.0	384	0.0	563	0.0
Zip	185	34.86	246	35.94	192	65.90
SEMS	80	71.83	75	80.47	129	77.09

TABLE II: Reduction rate (%) using shared and non-shared keys

	EMS		Weather		Farming	
	Size	Rate (%)	Size	Rate (%)	Size	Rate (%)
Original	288	0.0	388	0.0	567	0.0
Zip	185	35.76	249	35.82	199	64.90
SEMS	84	70.83	79	79.64	133	76.54

The false positive probability of FBF depends on various factors such as the table width or the data type of a value. Table III shows false positive probability examples of various data types [9], [15]. fp_{innate} is the false positive probability that each type innately has, and $fp_{correlate}$ is the probability where the type has a correlation with other types. As an example 'Lattitude' has a relationship with 'longitude' to specify a location. So, The $fp_{correlate}$ is significantly lower than fp_{innate}. Likewise, 'time' data type needs 'date' type to specify exactly when an event is occurs. From our previous research [15], the theoretical $fp_{correlate}$ is too low to detect in a real-world situation when the schema have a relationship among themselves.

TABLE III: False positive (*fp*) probabilities (FBF)

Type	fp_{innate}(%)		$fp_{correlate}$(%)	
	theory	exp.	theory	exp.
Boolean	0.39	0.38		
Latitude	1.5	1.5	0.045	0.051
Time	2.2	1.9	0.12	0.11

The false positive probability of a string is calculated from the fact that a series of random bits can be decoded into a valid string. We showed that $1/256 \times \alpha^{n'} \times \frac{1-\alpha^{(256-n')}}{1-\alpha}$ is the false positive probability of a string, where n' is the minimum allowable string length, and $\alpha = 95/256$ is the ratio of ASCII printable characters to non-printable ones [9]. This results in the 0.085% with $n' = 2$ and 0.0043% with $n' = 5$. When you use a string for representing numbers only, the probability even drops to 8.17×10^{-4} with $n' = 2$ and 7.30×10^{-8} with $n' = 5$.

The false positive probability of SEMS also depends on various factors including data types, filters, and table width. When a false positive rate of i_{th} pair (key, value) in SEMS format is fp_i, we can calculate the maximum false positive

Scenario	n'	**max** fp **(%)**	**min** fp **(%)**
EMS	2	0.34	5.56×10^{-30}
	5	0.017	5.95×10^{-68}
Weather	2	0.35	7.68×10^{-52}
	5	0.017	1.62×10^{-118}
Farming	2	0.69	2.33×10^{-62}
	5	0.03	2.12×10^{-142}

TABLE IV: Fasle positive probability (%) using only string types

probability as $\sum_i (fp_i)$ when each member has no relationship among themselves (and thus $fp_{correlation} = fp_{innate}$). When all of the members have relationships among themselves, the minimum false positive probability is $\Pi_i(fp_i)$.

Table IV shows the maximum and minimum false positive probabilities. Both for the cases where the minimum string is 2 ($n' = 2$) and 5 ($n' = 5$), we have significant size reduction. For the false positive probabilities, even though the max false positive probabilities look somewhat high in theory. However, in reality, most of the false positive strings are unreadable and random, e.g., "IxP_zp" or "zPAkxp", so we can easily remove them to reduce the probability to practically zero. The minimum false positive probability is astronomically small. This experiment shows that we can practically reduce false positives by introducing simple relationships among entities or by making minimum string lengths large.

V. Conclusion and Future Work

SMS messages have a single message size limitation that, when exceeded, partitions the message's contents into multiple sub-messages. This leads to increased complexity when sharing encoded data representation between devices.

In this work, we propose a system used to share context information of different types in a size-efficient manner between devices. We propose SEMS and reveal its novel approach to largely overcome the size limitation inherent to SMS systems via the application of probabilistic data structures. We show the size reduction achievable when using SEMS given different types of context information, and we quantitatively analyze the false positive probability inherent to the system. To improve upon our work, we plan the implementation of an enhanced system of larger scope with the infrastructure and interfaces necessary to support the sharing of context information in a real-world environment.

References

[1] J. Brown, B. Shipman, and R. Vetter, "SMS: The Short Message Service." *Computer*, vol. 40, no. 12, pp. 106–110, 2007.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey." *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] J.-S. Lee and U. Chandra, "Mobile phone-to-phone personal context sharing," in *ISCIT'09: Proceedings of the 9th International Symposium on Communications and Information Technology, 2009*. IEEE, 2009, pp. 1034–1039.

[4] A. K. Dey, "Understanding and Using Context." *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[5] M. Chalmers, "A Historical View of Context." *Computer Supported Cooperative Work*, vol. 13, no. 3-4, pp. 223–247, 2004.

[6] J. Kolari, T. Laakko, T. Hiltunen, and V. Ikonen, "Context-Aware Services for Mobile Users," VTT Publications, Tech. Rep. 539, 2004.

[7] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[8] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal, "The Bloomier filter: an efficient data structure for static support lookup tables." in *SODA'04*. Society for Industrial and Applied Mathematics, 2004, pp. 30–39.

[9] S. Cho and C. Julien, "CHITCHAT - Navigating tradeoffs in device-to-device context sharing." in *PerCom'16: Proceedings of the 2016 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 2016, pp. 1–10.

[10] A. Ferscha, "What is Pervasive Computing?" Universitatsverlag Rudolf Trauner, Tech. Rep., Jun. 2003.

[11] N. Nurseitov, M. Paulson, R. Reynolds, and c. izurieta, "Comparison of JSON and XML Data Interchange Formats - A Case Study." in *CAINE*, 2009.

[12] C. McAnlis and A. Haecky, *Understanding Compression*, ser. Data Compression for Modern Developers. O'Reilly Media, Inc., Jul. 2016.

[13] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors." *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[14] S. Cho, "Navigating Tradeoffs in Context Sharing Among the Internet of Things," Ph.D. dissertation, The University of Texas at Austin, The University of Texas at Austin, Aug. 2016.

[15] S. Cho and C. Julien, "Size Efficient Big Data Sharing Among Internet of Things Devices," in *PerCom Workshop BICA*, Jan. 2017, pp. 1–6.