SERA: A Hybrid Scheduling Framework for M2M Transmission in Cellular Networks

UmaMaheswari Devi, Munish Goyal, Mukundan Madhavan, Ravi Kokku, Dilip Krishnaswamy
IBM Research, India

***Abstract*—**
Trends show that machine-to-machine (M2M) devices are going to grow by orders of magnitude, far surpassing the number of mobile devices. This unprecedented scale and the fact that M2M traffic typically consists of many small-sized transmissions make the data and signaling overhead of introducing M2M traffic into cellular networks a big concern. Fortunately, it is possible to exploit certain unique characteristics of M2M traffic, like periodicity and delay tolerance in its scheduling, to alleviate these concerns. In this paper, we propose SERA — a two-level Scheduled Randomization framework, which does precisely this, and efficiently integrates M2M traffic into cellular networks. Broadly, SERA consists of (i) a central controller that defines certain coarse-level transmission parameters to govern M2M traffic in the next scheduling period and (ii) a simple distributed randomized algorithm at each M2M device that governs fine-grained transmission decisions within the period. Using experiments and analyses, we show that compared to existing techniques for M2M traffic management, SERA can lower peak traffic load by 30-40%, bring down the total time spent under congestion by 30-40%, and that these gains are robust to errors in traffic prediction.

I. Introduction

Machine-to-machine (M2M) devices refer to smart end-devices that acquire parameters of interest in systems such as health-care, surveillance, and traffic-control, and transport the acquired values to back-end servers for improved decision making and control. With the increasing instrumentation of our surroundings, the number of M2M devices has been rapidly growing. It is expected that the number of wireless M2M devices would hit 50 billion by 2020 [1]. Cellular networks already carry a significant fraction of wireless M2M traffic, mainly due to their ease of deployment and accessibility [2].

Studies have shown that M2M transmissions exhibit certain distinct characteristics. (1) They typically consist of a larger number of short, distinct transactions than user traffic, thus exacerbating signaling overhead [3][4]. (2) Due to the largely time-triggered nature of their operations, they tend to be time-synchronized, typically at $\frac{1}{4}$-, $\frac{1}{2}$-, and 1-hr boundaries [2], resulting in bursty traffic patterns. (3) The data they carry is often utilized in batch, and hence, the level of urgency in their transmission is diverse. In contrast, user-traffic is often (but not always) urgent, with users actively waiting on it. A key aspect of embracing M2M devices in cellular networks is to realize the differences in M2M and user traffic, and accordingly, manage them in different ways.

While congestion alleviation in cellular networks is a classical problem, the distinct characteristics of M2M traffic present newer techniques for resource management. Traditional, largely, centralized approaches tend to create management bottlenecks with increasing scale, which is the case in M2M. Purely distributed scheduling algorithms, on the other hand, can scale with the number of devices but cannot easily know network load and requirements of other applications, and hence might not sufficiently alleviate congestion. Moreover, traditional congestion management is a reactive process due to the nature of user data; a better approach for M2M would be to leverage the periodicity and delay-tolerance inherent in a significant fraction of these devices to appropriately defer traffic generation at the source and prevent congestion in the first place. Congestion due to M2M devices in cellular access links, especially in the signaling and control plane, is already receiving considerable attention [5], [6], [7], [8].

In this paper, we explore a hybrid solution to achieve the benefits of both centralized and distributed approaches. To be both scalable and capable of using centralized knowledge on resource constraints and application requirements, we follow a design philosophy where (i) a central controller provides a coarse-level transmission plan (i.e., a schedule) for devices detailing a broad transmission strategy on *how to* attempt to transmit in a given interval and (ii) individual devices execute a local randomized scheduling algorithm that determines *when to* transmit exactly within the interval. In doing this, our solution called SERA, denoting Scheduled Randomization framework, takes into account estimates of non-M2M traffic, requirements of periodic M2M traffic, and the QoS requirements of M2M devices. Our design intrinsically leverages the facts that M2M QoS requirements are diverse, ranging from real-time to delay tolerances up to a few hours [6]; further, within the delay tolerance, our design allows data collected to be stored and aggregated for amortizing signaling overhead, and thus provides significant flexibility in scheduling control. We also note that our solution is equally applicable to signaling and data overload management. Given that M2M traffic is uplink-heavy [2], we focus on managing uplink transmissions.

Contributions: **(1)** In Sec. III-A, we present SERA, a scalable application- *and* network-aware traffic management framework for a diverse set of M2M devices. We detail its end-to-end design and discuss its ease of deployment at various points in the network. **(2)** In Sec. IV, we formulate the problem of coarse-level M2M traffic management at the central controller as one of optimizing upload transmission probabilities over time for heterogeneous classes of devices. We then propose efficient solutions to this problem under diverse realistic settings using a combination of algorithmic and analytical approaches. **(3)** In Sec. VI, we use a MQTT-based prototype [9] to demonstrate SERA's scalability and ease of deployment.

978-1-4244-8953-4/11/$26.00

We evaluate the efficacy of SERA in alleviating congestion using numerical simulations in Sec. V. In our experiments, congestion is lower under SERA for around 30-40% of the total time and the maximum of the traffic experienced over a specified fraction of time is also lower by similar extent. These results are robust to errors in traffic prediction.

II. Related Work

Projections on M2M traffic surge have triggered research and standardization efforts for containing congestion in cellular networks. 3GPP has an ongoing study in this regard. The proposals from 3GPP can be classified into *randomization*, *dedicated resource allocation*, *differentiated services provision*, and *pull-based* methods [7].

The *access class barring* (ACB) or *p-persistent* randomization scheme is a *reactive* scheme for limiting congestion by broadcasting to devices, an access probability p and an ACB time (time for which a device waits if access is disallowed) during times of overload. Setting p is studied in [10] assuming dedicated resources for M2M traffic, while [11] determines p using a PID controller based on congestion at a core network node. In contrast, we proactively shape delay-tolerant M2M traffic at the source to minimize instances of overload.

In the second randomization approach, which comes closest to ours, the network specifies pre-agreed *grant times* / *forbidden times* to devices and devices independently randomize their accesses (uniformly) within the grant windows [12], [13]. Since the grant windows are typically long, uniform randomization by devices is oblivious to network load and cannot effectively smoothen aggregate load. There is also the option of specifying shorter communication windows (that are also only uniformly randomized) within each grant time to individual devices, albeit, at the expense of higher computation and communication overhead at the controller. No implementation of this specification could be found in the literature. For comparing it with ours, we implement it using (1) randomized access by devices at all times and (2) randomized access during times when load is below a threshold, with accesses forbidden during remaining times.

Dedicated resource allocation proposes splitting resources between M2M and user (non-M2M) traffic, either in a static or dynamic manner, to provide isolation for non-M2M traffic. Apart from requiring BS support and mechanisms for realization, with this approach, congestion is still a possibility within the M2M partition due to its nature. In differentiated services provision schemes, devices are prioritized, and accesses by lower-priority devices are rejected during times of congestion [7], [6]. These are also reactive methods that resort to service rejection, whose instantiations can be minimized by proactive traffic management. In pull-based schemes, the network explicitly grants access to each device via paging messages [4]. This is a centralized scheme and has been found to not scale with devices [7].

Other proposals found in the research literature largely fall under the categories of cooperative BS scheduling [14] (to better distribute traffic across BSs), BS uplink scheduling [15][16], aggregation and group-based scheduling [14], wherein multiple data units from a single or multiple devices are aggregated and uploaded in one go (to reduce control overhead and access delay). These are complimentary to our approach of load- and latency-based randomization.

III. System Architecture, Model, and Problem Formulation

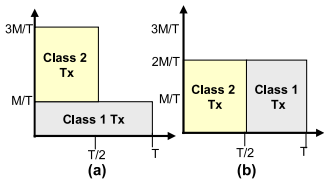


Fig. 1. Comparison of **(a)** application-oblivious and **(b)** application-aware approaches to M2M traffic control in the absence of non-M2M traffic.

Our approach, called SERA, proactively manages M2M traffic by deferring and shaping transmissions at the source itself in a network- and application-aware manner. As pointed out in [3] and [17], network- and application-blind approaches lead to increased resource consumption in both the network and the device. To see how being application aware can help even when non-M2M traffic is nil or evenly distributed, consider two classes of M devices each, both generating one unit of data every T time units. Let the delay tolerances of the first and second classes be T and $T/2$ time units, resp. If each device class uniformly randomizes transmissions within its window of delay tolerance, then the aggregate traffic would be unevenly distributed: as shown in Fig. 1(a), the traffic in the initial $T/2$ time units is three times that in the second half, which increases the probability of congestion in that time interval. On the other hand, if Class 1 devices are cognizant of the constraints of Class 2 and confine their transmissions to $[T/2, T)$, the aggregate traffic would be more evenly distributed, as shown in Fig. 1(b).

Having motivated SERA, we now describe its end-to-end system architecture.

A. End-to-End System Design

SERA consists of the following high-level system components, shown in Fig. 2: (1) A SERA controller for estimating traffic, and computing and publishing coarse first-level upload schedules to M2M devices; (2) end clients at M2M devices that use the schedule published by the controller to select an exact time for each upload; and (3) a message broker that serves as an intermediary for disseminating the upload schedules published by the controller to the clients.

SERA *controller:* The SERA controller consists of a traffic estimator/modeler, a first-level scheduler, and a policy publisher. We classify traffic that traverses the link managed by SERA into (i) M2M traffic that SERA schedules, which we will refer to simply as M2M load, and (ii) background (BG) traffic that SERA has no control over (and is adapted to). BG traffic can consist of both non-M2M traffic such as user traffic and M2M traffic with stringent real-time needs that is not amenable to deferrals. The traffic estimator builds models of both these traffic components, either by observing the traffic traversing the link and/or using declarations from end applications (mostly M2M) on their traffic profiles.

Data traffic volume is known to exhibit significant variation over time [18]. With the growth in (i) M2M devices that are not amenable to deferrals and hence contribute to BG traffic and (ii) chatty mobile apps with continuous online presence [19],

BG traffic in the control plane is also expected to exhibit variations over time. An example of the former is traffic from fleet tracking systems whose concentration at a BS varies with time and which perform data uploads every few seconds.

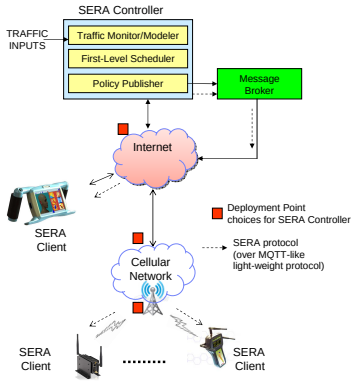


Fig. 2. SERA: end-to-end sys. arch.

The first-level scheduler takes traffic models from the estimator and produces first-level schedules for different classes of M2M devices. A first-level schedule is in the form of the fraction p_s of traffic from class c that can be transmitted in time slot s, specified for each class and slot. Fine-grained scheduling, so that the traffic actually conforms to the coarser first-level allocations, is the responsibility of the clients. E.g., suppose there are two device classes as in Fig. 1 and two slots of duration $T/2$ each. Here, the optimal schedule is to allocate slots 1 and 2 exclusively to the second and first classes, resp. Hence, the schedules published by the controller would be $\{(1,0.0),(2,1.0)\}$ for class 1 and $\{(1,1.0),(2,0.0)\}$ for class 2.

Estimates of BG traffic at a link can be obtained by observing the aggregate traffic or other means (e.g., from BS logs) and deducting the M2M load (estimated using the first-level schedule) from the aggregate. The SERA system can be bootstrapped by computing an initial first-level schedule assuming the aggregate load to be the BG load. Signaling traffic can be estimated from the distribution of the number of uplink connections requiring signaling for connection setup, which can be obtained from cellular network xDR reports.

The time slot granularity of first-level schedules depends on the granularity at which BG traffic varies and can be estimated. Since BG traffic evolves over time and can be impacted by sporadic events, coarse-level schedules are computed on a continuous basis, once every few hours, with the latest BG traffic estimate. Scehdules may also be computed and pushed to clients when sudden changes are detected to the BG profile. Our approach is applicable for M2M devices whose delay tolerance is higher than the duration of a time slot. Several categories of M2M devices, such as metering and building security systems, meet this requirement.

Policy publisher is client facing and publishes the schedules computed by the scheduler to M2M devices.

Message broker: In order to scale to tens of thousands of devices and be reliable in the face of network disconnections, the controller communicates to clients via an intermediary message broker. Off-the-shelf standard brokers based on light-weight protocols such as *message queue telemetry transport* (MQTT) [9] or Android *Cloud to Device Messaging* (C2DM) framework [20] and tailored for M2M applications can used. Both the SERA controller and clients should be designed as clients of the message-broker system (MQTT clients in the case of MQTT-based brokers), with the controller acting as the message publisher, and clients as message subscribers.

SERA *clients:* SERA clients subscribe to the first-level schedule policy published by the controller (that pertains to the class they belong to) and use it for performing second-level, fine-grained scheduling. To ensure that the aggregate class traffic distribution over time is in accordance with the fractions indicated in the first-level schedule, each device uses the fraction p_s indicated for slot s as the probability with which slot s is chosen for any upload.

Let T denote the number of slots in a scheduling period (see Sec. III-B) and F_i, the sum of the probabilities (total prob.) assigned to slots 1 to i, for $i > 0$. Thus, F_{kT}, $k > 0$, denotes the total prob. of the first k scheduling periods. Let $F_0 = 0$. Our goal is to design a client-side algorithm for determining exact upload slots so that the following hold: (1) The probability with which a device uploads in slot s is p_s. (2) The average number of uploads in the k^{th} period equals $F_{kT} - F_{(k-1)T}$, the total prob. of period k. (3) Given an interval spanning slots $t_1 \ldots t_2$, such that $F_{t_1-1} \leq m$ and $F_{t_2} \geq m+\ell$, where m and ℓ are non-negative integers and $0 \leq t_1 \leq t_2$, there are at least ℓ uploads in slots $t_1 \ldots t_2$. If $F_{kT} = k$ holds for all k, that is, the total prob. per period equals 1.0, one upload is attempted per period.

While it is natural to surmise that a straightforward procedure in which a device generates a random number r per slot s and uploads in that slot if $r < p_s$ should suffice, such a procedure cannot ensure the third condition above. Hence, the client procedure is instead as follows: For each period k, each device generates $\lceil F_{kT} \rceil - \lceil F_{(k-1)T} \rceil$ random numbers. Let r_i denote the i^{th} random number generated by a device and let $R_i = i - 1 + r_i$. A device's i^{th} random number corresponds to its i^{th} upload opportunity, which is attempted in slot t for which $F_{t-1} \leq R_i < F_t$ holds. It can be shown that this procedure ensures that all the three conditions above are met. Within the chosen slot, to avoid intra-slot synchronization, devices perform uniform randomization to pick the exact upload time.

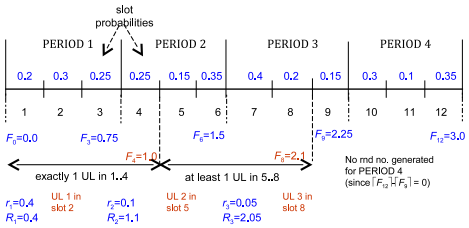


Fig. 3. Illustration of client-side algorithm.

Consider the example in Fig. 3. Here, $T = 3$ and the total prob. per period is 0.75. For Per. 1, the device generates $\lceil F_3 \rceil = \lceil 0.75 \rceil = 1$ random number, r_1, which is 0.4 in this example. Since $F_1 = 0.2 < 0.4 < F_2 = 0.5$, the first upload is scheduled in slot 2. Similarly, the second and third random numbers are generated for the second and third periods, resp., and lead to uploads in slots 5 and 8, resp. Since $\lceil F_{12} \rceil - \lceil F_9 \rceil = 0$, no random number is generated for the fourth period. The total prob. in the first four (resp., 12) slots is 1.0 (resp., 3.0), and exactly one (resp., three) upload is scheduled in that interval.

Assumptions/remarks: We assume that M2M devices whose traffic is scheduled are either stationary or if mobile, maintain association with a given BS during their next period. Lastly,

we note that SERA offers considerable flexibility in where and how it can be deployed. Deployment can be tightly integrated with the BS or over-the-top (OTT) with the controller and broker located either in the Internet or cellular network.

B. Problem Formulation for the First-Level Scheduler

We are given a set of M2M devices, possibly of varying data generation and upload frequencies, delay tolerances, and transaction lengths. Given the distribution of background (BG) load at a network link over time, the problem is to determine how traffic from the various device classes should be distributed over time so that M2M device upload completion times are commensurate with their delay tolerances and the possibility of congestion at the link over all slots is minimized.

To model the problem, let the number of device classes in the M2M system be N_C. Let time be divided into discrete *slots* of uniform length. Class c, $1 \leq c \leq N_C$, consists of M_c devices, generating data uploads with a mean rate of λ_c per scheduling period T_c (slots). A Class c device's payload size is denoted Z_c and its mean upload completion time (i.e., mean latency) by D_c (periods). A device can hold on to multiple acquired samples or messages and upload them in one shot to conserve energy or reduce signaling overhead, depending on the latency that can be tolerated. In that case, Z_c and D_c are for the aggregated payloads. If data traffic (resp. signaling traffic) is managed, then Z_c (resp. λ_c) would increase (resp. decrease) with aggregation. The average rate at which the uploads generated by a Class c device need to be serviced so that a mean latency of D_c can be maintained is given by μ_c, and we denote the lower and upper bounds of service rate by μ_c^ℓ and μ_c^h, resp. We assume that device payloads are much smaller than the capacity of a slot and there is only a finite number of distinct payload sizes.

Let L_s denote the mean BG traffic volume or load in slot s (which is assumed to be specified), and C, the total capacity per slot, both normalized by the minimum payload size over all classes. The objective is to manage the uploads from the M2M devices so that their mean latency times are as specified and congestion probability over all slots is minimized.

Let X_s be a random variable denoting the total number of uploads that are attempted by all devices of all classes put together in slot s. One way of minimizing congestion probability over all slots is by minimizing the maximum aggregate load over all slots, i.e., minimizing $\max_s X_s + L_s$.This can be accomplished in one way by minimizing the extent by which $X_s + L_s$ differs from a threshold level, say capacity C, over all s. Hence, we minimize $\sum_{s=1}^{N_S} E[(X_s - K_s)^2]$, where $K_s = C - L_s$ denotes the capacity available in slot s after accounting for BG load. $E[(X_s - K_s)^2] = E[X_s^2] + K_s^2 - 2K_sE[X_s] = Var[X_s] + (E[X_s])^2 + K_s^2 - 2K_sE[X_s]$. Thus, the mean square deviation or error in slot s is dependent on the probabilities assigned in that slot to the various classes, and our goal is to assign probabilities (for various (class,slot) pairs) such that total deviation over all the slots is minimized. The optimization problem TBU (Time Balanced Uploads, since minimizing the mean square error roughly balances traffic over time) can hence be stated as follows.

$$\text{TBU:} \quad \underset{\text{per-slot device class probabilities}}{\arg\min} \sum_{s=1}^{N_S} Var[X_s] + (E[X_s])^2 + K_s^2 - 2K_sE[X_s] \quad (1)$$

$$\text{s.t.} \quad \text{constraints on per-slot device class probabilities}$$

In the next section, we show how the above problem can be further expanded and reduced to a form that can be solved.

IV. Computing Load- and Latency-Dependent Slot Transmission Probabilities

A. Managing Homogeneous Devices

We first consider the simpler case in which devices are homogeneous, belonging to a single class. Let the number of devices be M, each with average upload request rate λ per period and mean upload latency D. Let each period be $T = N_S$ slots long. First-level scheduling, in this case, is performed successively over a sequence of periods. For simplicity, we assume that the upload size of each device is unity. (Non-unit upload sizes are considered later.)

We now show how μ, the total prob. over a period for ensuring D, can be determined using λ and D. Consider the upload system at each device and let the number of requests uploaded (or served) from the device per period be at least σ. Hence, assuming that arrivals are Poisson (as generally assumed for M2M transmissions [8]), the upload system at each device can be modeled as a distinct $M/D/1$ queue, with λ as the arrival rate and σ as the deterministic service rate. For an M/D/1 queue, $E[N_R]$ is given by $E[N_R] = \frac{\rho(2-\rho)}{2(1-\rho)}$, where ρ, the system utilization, is given by $\frac{\lambda}{\sigma}$. By Little's theorem [21], we have $E[N_R] = \lambda \cdot D$, where N_R is the number of pending requests and $E[N_R]$, its long-term average. Hence, σ, the desired deterministic service rate can be obtained using $\lambda \cdot D = \frac{\rho(2-\rho)}{2(1-\rho)} = \frac{\lambda}{2\sigma}\frac{2\sigma-\lambda}{\sigma-\lambda}$. Formulas can similarly be derived for other arrival- and service-time distributions.

For the above approach to be applicable, we are left with showing that a deterministic service rate of σ can be provided for the device's upload system. From the properties of the client algorithm provided in Sec. III-A, it follows that if μ (total per-period prob.) is integral, then exactly μ uploads are guaranteed per period. Hence, a simple approach would be to set $\mu = \lceil \sigma \rceil$. Alternatively, one can reduce the conservativeness for small $\sigma < 0.33$ by making use of the property that at least one upload can be scheduled in $\lceil \frac{2}{\mu} \rceil \leq \frac{2}{\mu} + 1$ periods (for proof, see [22]). Thus, each device can be thought of as receiving deterministic service of at least $\frac{\mu}{2+\mu}$ per period. μ is then easily determined from $\sigma = \frac{\mu}{2+\mu}$.

Having determined μ, we now show how per-slot probabilities can be determined. Let p_s denote the probability with which any device attempts to upload in slot s. Hence, per-slot upload for a device follows a Bernoulli distribution, and the total number of uploads in slot s is a sum of M Bernoulli r.v.'s, following a binomial distribution $B(M, p_s)$ with mean Mp_s and variance $Mp_s(1-p_s)$. Therefore, when the devices are homogeneous belonging to a single class, the objective function (1) is given by $Var[X_s] + (E[X_s])^2 + K_s^2 - 2K_sE[X_s] = Mp_s(1-p_s) + (Mp_s)^2 + K_s^2 - 2K_sMp_s$. The optimization problem for homogeneous devices may then be stated as follows.

$$\text{TBU\_SC:} \quad \underset{p_1,\ldots,p_{N_S}}{\arg\min} \sum_{s=1}^{N_S} (M^2 - M)p_s^2 - M(2K_s - 1)p_s + K_s^2$$

$$\text{s.t. } \mu^\ell \leq \sum_{s=1}^{N_S} p_s \leq \mu^h, \qquad p_s \geq 0, \quad \forall 1 \leq s \leq N_S$$

μ^ℓ and μ^h are lower and upper bounds on the desired service rate, and will both equal μ, if a strict service rate is desired. An upper bound may be specified if providing a lower average latency (than D) is desirable, when feasible.

The optimal solution to TBU_SC is given by the following theorem. Its proof is omitted due to lack of space and is available in the full version [22]. Using the three cases in the theorem, TBU_SC can be solved efficiently in $O(N_S)$ time.

Theorem 4.1: The optimal solution $\vec{p^*} = \{p_1^*, \ldots, p_{N_S}^*\}$ to TBU_SC is given by:

$$\vec{p^*} = \begin{cases} \max(0, \frac{2K_sM-M+\eta_1}{2M^2-2M}) \text{ ,if } \exists \eta_1 > 0 : \\ \qquad \sum_s \max(0, \frac{2K_sM-M+\eta_1}{2M^2-2M}) = \mu^\ell \\ \max(0, \frac{2K_sM-M-\eta_2}{2M^2-2M}) \text{ ,if } \exists \eta_2 > 0 : \\ \qquad \sum_s \max(0, \frac{2K_sM-M-\eta_2}{2M^2-2M}) = \mu^h \\ \max(0, \frac{2K_sM-M}{2M^2-2M}) \text{ ,otherwise} \end{cases}$$

B. Extension to Heterogeneous Devices

As before, let X_s denote the aggregate M2M load in slot s. The high-level objective in this case is also given by that in TBU. One difference in comparison to the single class problem is the assignment of distinct per-slot probabilities to distinct classes. Hence, X_s (the r.v. denoting the total number of uploads in slot s) is not a single binomial r.v., but a sum of N_C binomial r.v.'s, one per class. Let $p_{c,s}$ denote the probability with which devices of Class c attempt to upload in slot s, and $X_{c,s}$, the r.v. denoting the number of uploads by Class c in slot s. Then, the objective function in (1) can be expanded as $Var[X_s] + (E[X_s])^2 + K_s^2 - 2K_sE[X_s] = (\sum_{c=1}^{N_C} M_c \cdot p_{c,s})^2 - \sum_{c=1}^{N_C} M_c \cdot p_{c,s}^2 + (1-2K_s)\sum_{c=1}^{N_C} M_c \cdot p_{c,s} + K_s^2$. For derivation, see [22].

We now turn to the problem of balancing the total aggregate traffic over a scheduling interval $\mathscr{I}$, whose length, N_S, is a multiple of the period of each of the classes. Let R_c denote the number of periods for Class c in $\mathscr{I}$, and $F_{c,k}$ and $L_{c,k}$, the first and last slots of the k^{th} period of Class c, resp. Then, to ensure that each device of Class c is served at a rate of μ_c in each period, we require that $\sum_{i=F_{c,k}}^{L_{c,k}} p_{c,i} = \mu_c$ holds. (μ_c can be determined from λ_c and D_c as discussed in Sec. IV-A). Hence, the multi-class per-slot probability determination problem may be formulated as follows.

$$\text{TBU\_MC:} \quad \underset{p_1,\ldots,p_{N_S}}{\arg\min} \sum_{s=1}^{N_S}(\sum_{c=1}^{N_C} M_c \cdot p_{c,s})^2 - \sum_{c=1}^{N_C} M_c \cdot p_{c,s}^2 + (1-2K_s)\sum_{c=1}^{N_C} M_c \cdot p_{c,s} + K_s^2$$

$$\text{s.t.} \sum_{s=F_{c,k}}^{L_{c,k}} p_{c,s} = \mu_c,\ p_{c,s} \geq 0, \forall 1 \leq c \leq N_C, 1 \leq s \leq N_S, 1 \leq k \leq R_c$$

The above is a quadratic program that can be solved either using standard solvers or using the single-class approach for each class independently, in multiple rounds. The first round of computation for each class is performed by assuming some initial probabilities for the remaining classes. These initial probabilities are successively refined (using prob. computed for the other classes in the previous round) until they converge. A sub-optimal solution may be obtained by determining slot probabilities for the various classes in sequence. While computing probabilities for Class k, expected load for Classes $1 \ldots k$ (computed using their slot prob.) is added to the BG load L_s (which is used in determining $K_s = C - L_s$).

C. Generalizations and Variants

Non-unit upload sizes: To minimize the amount of time that a device stays in active transmission mode and conserve battery life, the device may want to aggregate and upload multiple data units in one shot. When the upload size $Z > 1$, the objective function for homogeneous devices is given by $2(M^2 - M)Z^2p_s^2 - M(2K_sZ - Z^2)p_s + K_s^2$. Since going from unit to non-unit sizes only changes the constant factors of various terms, the approach of Sec. IV-A still remains applicable. Similar is the case with heterogeneous devices. More details can be found in [22]. On the other hand, if signaling traffic is managed and aggregation is for lowering signaling overhead, then Z would not be impacted, but λ would be lowered, leading to a smaller μ and lower congestion.
Providing deterministic guarantees: If devices have deterministic traffic profiles in each period, deterministic worst-case guarantees can be provided for upload completion (latency) by ensuring the constraint that the total probability assigned to the slots in each period equals the number of data units.
Delay-error tradeoff: In the solutions to the optimization problems, error is contained by spreading the load as evenly as possible over time, subject to constraints. If sufficient capacity is available and may remain unused, it may often be desirable to complete device uploads early, subject to not exceeding capacity thresholds. This can be accomplished by adding an extra cost term $\gamma \sum_{s=1}^{N_S} x_s \cdot s \cdot P(X_s = x_s)$ to the objective function, where γ denotes the incremental cost per slot of delay, and an additional capacity constraint $\sum_c p_{c,s}M_c \leq K_s$. These modifications will result in a variant of our original formulation, but with quadratic / affine constraints and can hence, be easily solved using methods described earlier.

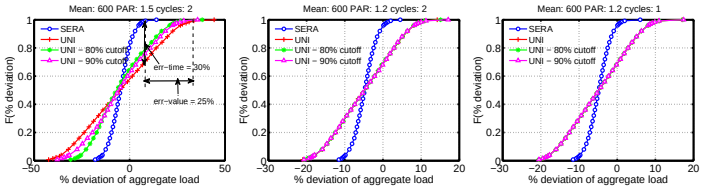
Other aspects such as permitting latency to be less than period, analysis for steady-state conditions, and reducing error are discussed in [22].

V. Numerical Simulations

Setup: The scheduling interval is set to 48 slots. In practice, slot duration could range from 5 to 30 mins depending on the timescale at which background (BG) traffic can be estimated at the link that is managed (as discussed in Sec. III-A). Thus, the total duration can range from four hrs to a day.

The capacity C of each slot is set to 900 units.[1] The mean of the per-slot BG load follows a triangular profile with one or two cycles over the entire interval (to mimic the common diurnal load pattern). The peak-to-average ratio (PAR) of the profile is varied between 1.0 and 1.5, while the mean over all the time slots is maintained a constant at 600 units. We also tested with mean set to 450 and 300 and found only little variation in the results. Deviation of the actual per-slot BG load from the profile is simulated by drawing BG load for each slot from normal distribution with mean taken from

[1]In practice, slot capacity would depend on the link bandwidth and minimum M2M upload size. Slot capacity was limited to 900 units due to the need to repeat each case studied around 100,000 times. We verified that the results extend to larger capacity slots using sanity tests.



(a) (b) (c)

Fig. 4. Comparison of SERA, UNI, and UNI-THR for homogeneous devices for varying peak-to-average ratio (PAR) of background (BG) traffic. BG traffic was generated using a triangular profile with two cycles in **(a)&(b)** and one cycle in **(c)** and a 10% standard deviation around the mean in all.

the triangular profile and standard deviation (STDEV) ranging between 10% and 25% of the mean. The number of M2M devices is so chosen that the aggregate (aggr) traffic is close to saturation. In all experiments, we assume devices to have deterministic profiles with deterministic upload rates and data ready for upload at period boundaries.

We compare SERA with uniform randomization, called UNI, in which each device picks a slot for upload with uniform probability, and uniform randomization with cutoff, called UNI-THR, in which slots with BG traffic exceeding a threshold (set to $0.8C$ or $0.9C$) are excluded for M2M and the remaining slots are chosen with uniform probability. (Reducing the threshold any further only worsens the UNI-THR scheme.) UNI and UNI-THR mimic the uniform randomization of device uploads with grant and forbidden times specified by the 3GPP proposals [12]. As explained in Sec. II, the other proposals in the literature are complementary to ours or do not proactively minimize congestion while accounting for varying BG load and hence are not directly comparable.

For a given slot, we refer to the deviation of its aggr load from the threshold capacity as *error*, and compare the cumulative distribution function $F(.)$ of this slot-wise error for competing schemes. (Relative errors of the various schemes are insensitive to the threshold.) Specifically, we study the following two metrics :

- For a given value of *error* e,[2] we determine the difference in % of slots whose error exceeds e in two schemes. We denote the max of this difference as err-time $= \max_e F_{SERA}(e) - F_u(e); u \in \{\text{UNI}, \text{UNI-THR}\}$. Intuitively, err-time is represented by the maximum vertical gap between the CDF curves of two competing schemes.
- For a fixed percentile (pctl) of slots, we compare the difference in value of error under competing schemes. We denote the max of this difference as err-val $= \max(e_1 - e_2)$ s.t. $F_{SERA}(e_1) = F_u(e_2); u \in \{\text{UNI}, \text{UNI-THR}\}$. This represents the average extra load faced by a given % of slots across the schemes and is the widest horizontal gap between their CDF curves, as shown in Fig. 4(a).

Results for Homogeneous Devices: Figs. 4(a) & (b) present single-class results for two different PARs of 1.5 and 1.2, resp., when the BG load contains two cycles. These figures plot the CDF of the % by which the aggr load in a slot deviates from the threshold.

Referring to Fig. 4(a), which is for a PAR of 1.5, at the 99^{th} pctl, error is around 25% higher under UNI than under SERA. Thus, err-val is at least 25%. Also, $\sim$30% slots see higher aggr load than that at the 99^{th} pctl under SERA, and so, err-time is at least $\sim$30%. The differences narrow down, but not significantly, as we move to the UNI-THR schemes, in which slots with high BG load are cutoff to M2M: err-time is $\sim$25% and 20% with 90% and 80% cutoff, resp., (at 99^{th} pctl of SERA), and err-val is $\sim$20% (at 99^{th} pctl).

As the BG load PAR decreases to 1.2, the load variability, and hence the scope to smoothen it using differential prob. decreases. err-val reduces to 10%, which is still significant. However, err-time is still 30% for both UNI and UNI-THR (observed at SERA's 95^{th} pctl). Note also that since the peak BG load is only 720 units (which is $0.8C$), the UNI-THR schemes are identical to UNI. The differences are further lowered as PAR is reduced to 1.1 (not plotted due to lack of space). err-val and err-time are reduced to 5% and 15%, resp.

Reducing the number of cycles for a given PAR in the BG profile reduces the degree of fluctuation in the BG load. However, the performance difference of the schemes is unaffected by the number of cycles. Inset (c) shows an example.

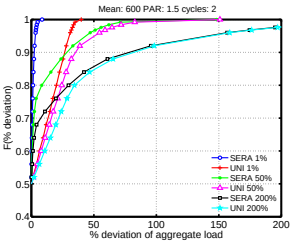
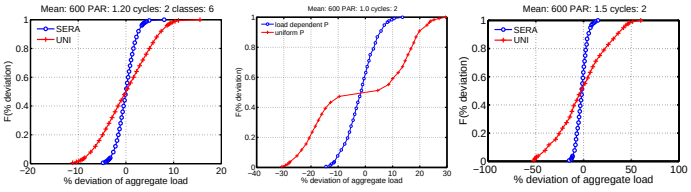


Fig. 6. Comparsion of SERA and UNI for varying degree of predictability of BG traffic.

Limited BG load predictability: In Fig. 6, we study the SERA's limits by increasing the degree of unpredictability of the actual load by increasing STDEV to 50% and 200%. When STDEV is 50%, err-val and err-time are still around 30% and 22%, resp., observed at SERA's $\sim 80^{\text{th}}$ pctl. Also, loads at the 99^{th} pctl still differ by 15%. However, the % of slots under UNI that exceed SERA's 99^{th} pctl load is reduced to almost zero. Increasing STDEV to 200% lowers both err-val and err-time to around 12%, which occurs at the 70^{th} pctl. Thereafter, the diff. narrows down and the two plots converge. Thus, the performance of SERA is higher than that under UNI by non-trivial amounts even when the actual load deviates significantly from the predicted, by as much as 50%. More importantly, SERA *always performs as well as or better than* UNI.

Results for Heterogeneous Devices: Fig. 5 plots the results for heterogeneous devices. We only plot a subset of the results presented for single-class, since the results follow similar patterns. Inset (a) shows the results when there are six classes. Their data upload periods are set to 4, 6, 8, 12, 24, and 48, and latency tolerances equal periods. BG PAR is set to 1.2. The difference between SERA and UNI is very similar to that observed with a single class under similar BG load.



(a) (b) (c)

Fig. 5. Comparison of SERAand UNIfor heterogeneous devices. **(a)** Latency = Period. **(b) & (c)** Latency < Period. Standard deviation for BG is 10% of the mean.

[2] $e > \{\hat{e} : F_{SERA}(\hat{e}) = F_u(\hat{e}); u \in \{\text{UNI}, \text{UNI-THR}\}\}$, that is, e is after the cross-over point $\hat{e}$ of the error CDF's of competing schemes.

Insets (b) and (c) correspond to multiple classes, but with latencies of some of the classes differing from their upload periods. In inset (b), the BG load is uniform (PAR is 1.0), but still there is a huge performance difference. This shows that SERA has applicability even in the absence of variability in background traffic as explained in Sec. III. Inset (c) compares the schemes for multiple classes with latencies different from periods when BG PAR is 1.5. The difference in performance is diminished in comparison to that in inset (b) as some of the unevenness of M2M device load is smoothened by the BG load, but the magnitude of the difference is still comparable to that with a single class.

We next study some of the tradeoffs described in Sec. IV-C.

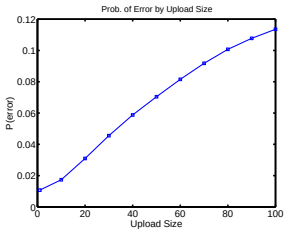
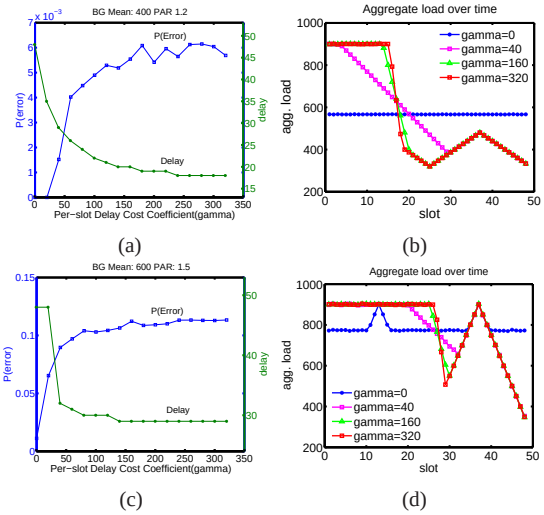


Fig. 7. P(Error) by upload size.

Upload Size vs. Error Tradeoff: To study how error increases with upload size, we systematically scaled up the upload size from 1 to 100, while scaling down the total number of uploads by the same factor to maintain constant M2M device load. BG traffic was generated with mean of 600 and PAR of 1.5. The probability of error is plotted against upload size in Fig. 7. In our setup, we observe that error probability increases linearly with the upload size, roughly according to $P(err) = 0.01 + 0.0011U$, where U is the upload size. Thus, the error probability increases $\sim 0.1U$-fold for a U-fold increase in the upload size.



(a) (b) (c) (d)

Fig. 8. Error-delay tradeoff study. **(a) & (b)** BG traffic Mean: 400 PAR: 1.2; **(c) & (d)** BG traffic Mean: 600 PAR: 1.5.

Delay vs. Error Tradeoff: It may sometimes be desirable to complete device transmissions early, for instance, to enable the link to be work-conserving. To study the impact of early upload completions on error probability, we performed simulations with a delay cost term $\gamma \sum_{s=1}^{N_s} sE(X_s)$ added to the objective function, as explained in Sec. IV-C. For simplicity, we considered a single-class setup, and analyzed for (i) a moderately-loaded scenario (where upload advancements are feasible) and (ii) a heavily-loaded scenario (where upload advancements are hard). The mean of the BG load was set to 400 and 600, and the PAR to 1.2 and 1.5, in the first and second scenarios, resp. Figs. 8(a) and (c) show how error and delay vary with γ (cost co-efficient) in the two cases. Here, delay denotes the latest slot in which an M2M device transmission is scheduled. As γ increases, delay decreases, but because more number of slots are saturated, error probability also increases. The decrease in delay with γ is more gradual in the first case, due to the availability of sufficient capacity. Similarly, the maximum error probability is much lower in that case. The point to operate at should be chosen based on the tolerable error and delay.

Insets (b) and (d) show how the aggr load is distributed over time for different values of γ for the two scenarios mentioned above. The load distribution is even for $\gamma = 0$ (except for the peaks carried over from the BG traffic in the second case). As γ increases, the number of slots operating close to capacity, all of which are confined to the initial part of the interval, increases. Thereafter, the load decreases linearly with varying slopes (the higher the γ, the higher the slope) until all of M2M device uploads complete. The traffic in the slots after the latest M2M transmission time is just the BG traffic.

VI. Prototype Implementation

Fig. 9 gives a block-level description of a prototype that we have implemented for demonstrating and testing scalability of SERA. On the server-side, the prototype consists of SERA controller, which is responsible for computing upload probability configurations. The prototype uses MQTT (Message Queue Telemetry Transport) [9] to communicate these configurations to M2M devices. MQTT is a publish-subscribe based protocol that uses a message broker for distributing messages between publishers and subscribers. Subscribers subscribe to specific topics of interest, and publishers associate messages that they generate to specific topics. Based on the topic of a message, the broker delivers the message to all the relevant subscribers. In our prototype, we use an open-source MQTT broker called Mosquitto [23].

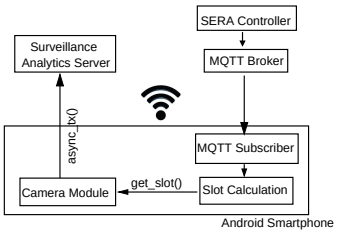


Fig. 9. Block Diagram of SERA prototype.

On the client-side, we use an Android smartphone's camera as our M2M device. The camera is configured to periodically upload surveillance images to a Surveillance Analytics Server in the backhaul for archiving and analysis. The phone also contains a MQTT subscriber that is subscribed to the Mosquitto broker's messages under the topic "Upload Config." Whenever SERA controller updates upload probability configuration values, a message is published under this topic, and in turn, sent to the subscriber through the broker. The MQTT subscriber passes on these configurations to a *slot calculation* module, which uses them to pick upload slots once in subsequent scheduling periods (using methods from Sec. III).

We now present an experiment on our prototype that illustrates the scalability of SERA in a real network. To this end, we study the rate at which our prototype's MQTT broker can disseminate messages to its subscribers. Since it is impractical

to deploy thousands of sensors for such stress testing, we retransmit the same MQTT message many times to a single subscriber and study the underlying rate of transmission. To ensure that the client-side does not bottleneck this study of the server's dissemination capacity, we use a 4-core T420 laptop instead of an Android smartphone as the subscriber client. Upon transmission, we measure the average MQTT dissemination throughput as $R_{dis} = \frac{\text{Number of messages}}{\text{Time taken}}$. Fig. 10 shows values of R_{dis} for an increasing number of slots per scheduling period. We point out that the MQTT message size is a direct function of the number of slots per scheduling period, a fact illustrated by this plot.

Importantly, we note that R_{dis} is around 9430 messages/second for realistic configurations. Let us consider a typical cellular network which has about $100,000$ BSs on the whole. If we assume that each BS caters to about $10,000$ M2M devices, this gives us a worst-case of 1 billion devices in the whole network. If we assume a typical SERA scheduling period of 4 hours (with 15 minute scheduling slots) and staggered configuration updates to devices, this gives us a time-frame of 4 hours within which each M2M device's configuration must be updated once.

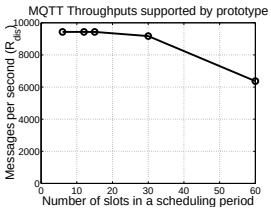


Fig. 10. MQTT message dissemination rate

Fig. 10 shows that our prototype MQTT broker, which is bound to 2 CPU cores of a Lenovo T420 laptop, can deliver about 9430 messages/second for the above configuration. Since the dissemination process is highly parallelizable and its rate is mainly bottlenecked by processing speed, one can expect an eight-fold increase in R_{dis} (75440 messages/second) on a medium grade 16-core server. This implies a dissemination time of $\frac{1\,\text{billion}}{75440} = 13226$ seconds (roughly 3 hours and 40 minutes) for the entire network, which is well within the 4-hour deadline. This complexity will be further reduced if we remove the stringent condition of having a single dissemination point and allow, say, a simple architecture where multiple SERAcontrollers, located in SGSNs control a different subset of base-stations. For a typical density of 100 SGSNs for a network, the dissemination latency in this architecture will come down to $\frac{13226}{100} \approx 133$ seconds (order of minutes). We point out that while the above analysis does not include the computational needs of the SERA optimization algorithm, optimization complexity scales with the number of slots in a scheduling period and not with the number of devices associated, making its computational requirement independent of network scale. These observations lead us to conclude that the cost of deploying SERA for an entire network is quite low and comparable to the cost of a medium grade server, making SERA's deployment practically feasible.

VII. Conclusion

The number of machine-to-machine (M2M) devices connected to the Internet is increasing at a rapid rate. In this paper, we propose a two-level scheduling framework called SERA, for embracing M2M device traffic onto cellular networks. SERA leverages the distinct characteristics of M2M traffic to pro-actively shape traffic generation at the source and reduce resource wastage at both the network and device. To be both capable of scaling with the projected device growth and shaping M2M traffic in a network- and application-aware manner, SERA adopts a hybrid model consisting of a central and distributed components. SERA is independent of lower-layer technology, easy to deploy, and provides multiple deployment options. Due to technology independence, SERA is not restricted to cellular networks and may be used for M2M traffic shaping and management over any bottleneck network link or resource, such as data center ingress, WiFi, or ZigBee links — some of which are also known to be impacted by the M2M device surge. Joint optimization over multiple such networks, along the lines of SERA, is an interesting problem, which we suggest as future work.

References

[1] OECD, "Machine-to-machine communications: Connecting billions of devices."

[2] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "A first look at cellular machine-to-machine traffic large scale measurement and characterization," in *SIGMETRICS*. ACM, 2012, pp. 65–76.

[3] 3GPP TR 37.868 V11.0.0, "Study on ran improvements for machine type communications," Sep. 2011.

[4] M. Hasan, E. Hossain, and D. Niyato, "Random access for machine-to-machine communications in LTE-Advanced networks: Issues and approaches."

[5] 3GPP TS 22.368 V11.6.0, "Service requirements for machine-type communications stage 1," 2012.

[6] P. Jain, P. Hedman, and H. Zisimopoulos, "Machine type communications in 3GPP systems."

[7] M. Cheng, G.-Y. Lin, H.-U. Wei, and A. Hsu, "Overload control for machine-type-communications in LTE-Advanced systems."

[8] A. G. Gotsis, A. S. Lioumpas, and A. Alexiou, "M2M scheduling over LTE: Challenges and new perspectives," *IEEE Vehicular Technology Magazine*, Sep. 2012.

[9] "Official mqtt webpage." [Online]. Available: http://mqtt.org

[10] S. Duan, V. Shah-Mansouri, and V. Wong, "Dynamic access class barring for m2m communications in lte networks," in *Globecom Workshops*, 2013, pp. 4747–4752.

[11] A. Ksentitni and Y. Hadjadj-Aoul, "Cellular-based machine-to-machine: Overload control."

[12] 3GPP TR 23.888 V11.0.0, "System improvements for machine-type communications," 2012.

[13] T. Taleb and A. Kunz, "Machine type communications in 3GPP networks: Potential, challenges, and solutions."

[14] A. Lo, Y. W. Law, and M. Jacobsson, "Enhanced LTE-Advanced random-access mechanism for massive machine-to-machine communications," in *27th Meeting of Wireless World Research Forum*, Oct.

[15] A. S. Lioumpas and A. Alexiou, "Uplink scheduling for M2M communications in LTE-based cellular systems," in *International Workshop on Machine-to-Machine Communications*, 2011, pp. 353–357.

[16] A. G. Gotsis, A. S. Lioumpas, and A. Alexiou, "Evolution of packet scheduling for machine-type communications over LTE: Algorithmic design and performance analysis," in *Second International Workshop on Machine-to-Machine Communications*, 2012, pp. 1620–1625.

[17] H. Viswanathan, "Getting ready for M2M traffic growth," May 2011.

[18] Y. Zhang and A. Arvidsson, "Understanding the characteristics of cellular data traffic," in *CellNet*, 2012.

[19] A. Aucinas, N. Vallina-Rodriguez, Y. Grunenberger, V. y Erramilli, K. Papagiannaki, J. Crowcroft, and D. Wetherall, "Staying online while mobile: The hidden costs," in *CoNext*, 2013, pp. 315–320.

[20] "Android cloud to device messaging framework." [Online]. Available: https://developers.google.com/android/c2dm/

[21] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley, 2002, ch. 8, Continuous-Time Markov Chains.

[22] U. Devi and M. Mukundan and M. Goyal and R. Kokku and D. Krishnaswamy, "SERA: A scheduling framework for m2m transmission in cellular networks," 2014. [Online]. Available: http://resweb.watson.ibm.com/researcher/files/in-umamadev/M2M-long.pdf

[23] "Official page of mosquitto mqtt broker." [Online]. Available: http://mosquitto.org