Floating Gossip: Serverless Distributed Learning in Dynamic Scenarios

Gianluca Rizzo,*† Noelia Perez Palma,‡ Marco Ajmone Marsan,§ and Vincenzo Mancuso¶§
gianluca.rizzo@hevs.ch, noelia.perez3@um.es, ajmone@polito.it, vincenzo.mancuso@imdea.org
* HES SO Valais † Università di Torino ‡ University of Murcia § Institute IMDEA Networks ¶ University of Palermo

***Abstract*—This paper studies the performance of Floating Gossip, a novel decentralized approach for Gossip Learning at the network edge. Floating Gossip utilizes Floating Content to facilitate location-based probabilistic evolution of Machine Learning models, without external infrastructure support. We investigate dynamic scenarios requiring continuous learning, leveraging a mean field approach to analyze Floating Gossip's performance boundaries. Our focus is on the quantity of data that users can integrate into their models, as a function of key system parameters. Unlike previous studies that separately optimize communication or computational aspects of Gossip Learning, our methodology considers their combined effect. We validate our analysis through comprehensive simulations, demonstrating the high accuracy of our analytical model. Our methodology reveals Floating Gossip's effectiveness in training and updating Machine Learning models collaboratively, leveraging opportunistic exchanges between mobile users, while flexibly adapting to different user characteristics and mobility patterns. This research highlights Floating Gossip's potential for continuous, cooperative model training in dynamic, infrastructure-less environments, offering insight into its performance patterns and its potential in practical applications.**

I. Introduction

The last two decades have witnessed the introduction of several innovative technologies for wireless device-to-device (D2D) communications. The earliest technology on the market was Bluetooth, which appeared in 1999 and is now at version 6.0 [1]. Bluetooth was followed by WiFi Direct [2], and then by the LTE sidelink [3] and the 5G New Radio sidelink [4]. This diverse array of D2D protocols is poised to accelerate the emergence of innovative services that leverage direct data exchanges between users [5]. As a result, it is becoming increasingly feasible to implement context-aware, location-based applications—including mobile crowdsensing platforms and services like Gigwalk, Waze, and Millionagents [6]–[8]—where users can seamlessly share contextual and location information via D2D connections, or applications for the opportunistic diffusion and storage of information in a localized area, like in LINGER [9] and Floating Content (FC) [10]. At the same time, we are witnessing a growing trend towards services based on continuous ingestion of up-to-date contextual data—hereafter termed *observations*—used to train and update machine learning (ML) models on user equipment (UE) [11]. Typically, these services implement real-time, context-aware applications in which localized ML inference drives decision-making independently of centralized cloud infrastructure, allowing UEs to continuously update their strategies and optimize utility functions during task execution [12]. A relevant example context in which such applications can be useful is a post-disaster scenario, where infrastructure has been damaged, and rescue teams must collect and share information through D2D communications only. Another relevant scenario is vehicular sensing for Smart City applications, in which connected vehicles moving through urban areas collaboratively maintain models of traffic flow, accident locations, and road conditions, without centralized servers or raw data transmission, reducing latency and privacy exposure. Current approaches for training and updating ML models on UEs include federated learning (FL) [11] and gossip learning (GL) [13], [14], a decentralized, serverless alternative where models propagate in a peer-to-peer fashion and update locally via direct exchanges. These last are gaining traction due to their elimination of single points of failure, reduced infrastructure costs, enhanced privacy, and compliance with strict data sovereignty regulations, as data never leaves UEs. Recent results [14], [15] suggest that GL matches or outperforms FL in scenarios with evenly distributed data, while offering superior scalability and robustness in dynamic networks. A quite relevant aspect of GL is that it allows to preserve data privacy by leaving sensitive information at the UEs, thus limiting exposure to tampering, while ensuring scalability through lightweight model exchanges. As GL only transfers model updates rather than raw data streams, it helps reduce bandwidth and processing overhead.

In this paper, we study the effectiveness of D2D-based opportunistic services to support collaborative, gossip-based continual training of ML models in UEs that operate autonomously, without the need for a RAN infrastructure, without centralized support, and without exchanging any observations. Our main goal is to determine the limit performance of this approach to model training, in terms of the maximum amount of information that a group of UEs can autonomously learn and incorporate into their ML models, through decentralized, self-organized collaboration, leveraging their computational resources alongside D2D communications. We name the system that we propose and analyze Floating Gossip (FG), as it combines GL and FC, augmented with local computing at the UEs.

Numerous prior studies have examined individual aspects of the system under investigation. For instance, several works on Floating Content (FC) [16]–[18] have characterized the opportunistic D2D aspect, providing analytical models for the probability of a content (e.g., a message, or an ML model) to persist probabilistically in a given region. Other works

on GL have tackled the collaborative, serverless ML training aspects [19]–[21]. However, they have focused mainly on static scenarios with no churn and in which all nodes are always interconnected. Moreover, these results are related to very specific GL algorithms, ML architectures, and learning tasks, and thus they provide only partial indications about the limit performance of such learning approaches as a function of the main system parameters. Our modeling approach is instead based on the combination of a mean field approach and elements of queuing theory with results from information theory, validated through detailed simulation experiments. Specifically, the main contributions of this work are:

- We present Floating Gossip, a fully distributed Gossip Learning scheme for continual training of a set of models stored probabilistically among nodes in a region of space, without support from infrastructure-based communications;
- We derive a mean field-based analytical approach to model the limit performance of FG, in terms of *model staleness* (i.e., of average time elapsed since the collection of the most recent observation included in the model instance's training set), and of *learning capacity*, (i.e., of the maximum fraction of observations generated through the RZ which are incorporated in an ML model);
- We characterize the stability properties of an FG system, determining the operating conditions within which it can effectively incorporate the incoming observations.
- We assess numerically our results, validating our assumptions against simulation under different configurations, showing the accuracy of our approach. We characterize the limit properties of FG as a function of the main system parameters, and we evaluate the impact of internal infrastructure support to FG in the form of static nodes[1].

II. Related Work

In the field of distributed learning, fully distributed, serverless schemes are of particular interest due to their scalability and robustness. Among these, Gossip Learning (GL) [13] is a collaborative machine learning paradigm primarily motivated by the demands for data privacy and scalability. In GL, model training occurs locally on each device's dataset, which is continually updated with new observations, and is complemented by the sharing and merging of models with other devices. We adopt GL under the constraint that model exchanges are restricted to device-to-device (D2D) communications within the boundaries of a finite region of space (the replication zone), introducing unique performance and computational limitations. A related approach has been explored in [19], where federated learning also leverages opportunistic model exchanges, demonstrating the effectiveness of encounter-based learning. However, their framework permits data exchanges over D2D links, thereby introducing potential privacy vulnerabilities. We focus on GL protocols which perform horizontal training, and in which each node trains (and exploits for inference) its own model, which in principle differs from that of other nodes. The study by Giaretta and Girdzijauskas [20] investigates GL within static networks (i.e., networks whose connectivity graph does not change over time), analyzing how data distribution, network connectivity, and communication speed influence system behavior. Building upon these aspects, our research extends the analysis to encompass the dynamics and peculiarities of opportunistic communications.

[1]We distinguish between external and internal infrastructure. The former is a general-purpose telecommunications infrastructure, such as an operator's RAN. The latter is a set of FG nodes placed in fixed positions within the RZ.

TABLE I: Features of Floating Gossip, Vertical Federated Learning, and Horizontal Federated Learning.

Feature	FG	HFL	VFL
Nodes contacts	Opportunistic	Fixed	Fixed
Data differs in	Sample space	Sample space	Feature space
Nodes exchange	Model	Model	Partial results
Nodes keep local	Samples+model	Samples	Samples+model
Nodes model	Local	Global	Local
Collaboration	No	No	Yes

Regarding data modeling, prevailing literature often assumes a fully distributed scenario in which each device possesses a single data point (henceforth denoted as *observation*) and participates in model training only once (see [21] and references therein). Notably, [20] is among the few to propose iterative training over multiple observations, offering a simulation-based assessment of its impact. Our approach is more general, as it permits devices to train on multiple, potentially non-unique observations, reflecting the realistic possibility that several devices may observe the same event. This scenario, where observations are not unique across devices, remains largely unexplored in existing studies.

Network connectivity is frequently idealized as stable, with minimal constraints from physical separation or infrastructure, commonly assuming environments such as data centers or multi-threaded computing systems [21]–[23]. Nevertheless, connectivity plays a crucial role in the convergence of GL, as evidenced in the context of stochastic gradient descent algorithms [24]. Our work focuses on opportunistic interactions among physically distinct and independently operating devices, entirely devoid of infrastructural support. This setting inherently limits the number of neighboring devices available for model exchange and introduces dynamic topologies with user equipment (UE) churn, making the timely and opportunistic exchange of models essential.

Communication speed and protocol overheads are often overlooked or underestimated in the literature. Some studies assume unrealistically high data rates or disregard the time required for connection establishment [25], while others nearly omit communication delays altogether [26]. In contrast, we explicitly consider the constraints imposed by D2D connectivity, including limited data rates and the non-negligible time required to establish connections. Additionally, we recognize that ongoing data exchanges may preclude the exploitation of new communication opportunities, leading to missed exchanges. Unlike prior approaches that typically optimize either communication or computation in isolation, our analysis centers on the combined effects of both factors.

It is important to note that comprehensive analytical models

for evaluating GL performance are lacking in the literature, with most studies relying on simulation or empirical trace analysis. In this work, simulations are employed solely to validate the intricate analytical model we develop. The closest methodological precedent is the seminal contribution by Chaintreau *et al.* [27], which applies mean field analysis to the epidemic message dissemination via gossiping. Our approach similarly leverages mean field techniques, but extends them to incorporate GL-specific characteristics of learning and computation, as well as the unique role of FC in maintaining model availability within designated areas of interest.

Similarities and differences exist between FG and FL, in both versions called horizontal FL (HFL) and vertical FL (VFL) [28], as illustrated in Table I for the case of a single ML model. First of all, while FG is based on FC, hence on opportunistic data exchanges due to nodes movement, FL normally assumes fixed connections among nodes. Second, in the case of VFL, different nodes train different features of the ML model, while in both FG and HFL all nodes train all features of the ML model. For this reason, FG and HFL exchange model parameters among nodes, while in VFL it is necessary to exchange intermediate results. The samples (that in this paper we call observations) used to train the model remain local to nodes in all cases, but in both FG and VFL also models are kept local to nodes. As a result, FG and VFL produce models that are local (possibly slightly different) to nodes, while HFL produces a shared global model. Inferences are necessarily collaborative in the case of VFL, because of the feature space partitioning, while in FG and HFL each node is capable of obtaining autonomously its own inferences.

III. System model

A. What is Floating Gossip?

We focus on a distributed Machine Learning (ML) scheme denoted as Floating Gossip (FG), resulting from the combination of the Floating Content [18], [29] scheme with Gossip Learning [20], [22]. Floating Content is a content distribution scheme that enables probabilistic storage of messages or ML models over a geographical area by exploiting moving nodes that traverse the area and opportunistically exchange content whenever they come sufficiently close for wireless communication to be possible. Gossip Learning is an ML training scheme according to which a set of nodes update a model via continual learning, by acquiring data in a distributed manner and exchanging and merging models trained on that data, rather than sharing the raw data itself, with significant advantages in terms of privacy, scalability, and robustness.

The combination of Floating Content and Gossip Learning thus brings to Floating Gossip (FG), i.e., to a distributed ML scheme in which nodes locally update and opportunistically share ML models trained on data collected at different times in different positions by different nodes, with no need for sharing the raw information. In FG, models are stored probabilistically in a given region of space and trained cooperatively by nodes in the region. Critically, models are scoped to the Replication Zone (RZ) and exist only within it. A model instance consists of coefficients and a training set; when a node exits the RZ, it discards all model instances and observations, as these models and data are no longer relevant outside their geographic scope. However, other nodes remaining in the RZ will continue to possess instances of the same model type, ensuring the model remains "afloat" in the region through their copies. If a node later re-enters the RZ, it receives a fresh default model instance (no prior knowledge preserved).

B. The modeled system

We consider a region of space (termed *replication zone* - RZ) in a dynamic environment whose conditions change over time, such as a disaster-struck area in the immediate aftermath of the disaster. The region is populated by static or moving *nodes* representing wireless communication devices or *user equipments* (UE), such as smartphones or smart vehicles. Some of those nodes may be static, thus representing a form of internal infrastructure support to FG (rescue teams might populate the disaster area with some ad-hoc communication infrastructure). Moving nodes enter the RZ, cross it, and leave. The number of nodes in the RZ is assumed to be random, with a finite mean N. The node mobility patterns are assumed to be stationary, and such that at any instant, the node distribution over the RZ is uniform.

In our model, each node belongs to one of two classes (denoted with the labels s and f, respectively, for slow and fast), characterized by different node speeds (zero, in the case of static nodes) and/or computing capacities. We will mostly use the two classes to study the effect of static nodes on system performance. Note that the extension of our approach to an arbitrary number of classes is straightforward. Still, in what follows, we consider only the two-class case for simplicity of notation and exposition. We assume the RZ to be a finite region of the plane, of arbitrary shape and area A.

We assume that the FG scheme is based on a finite set of ML *model types*. Each model type is identified by a specific ML architecture (e.g., LSTM, DNN) and a specific configuration for that architecture (e.g., in terms of such hyperparameters as number of neurons, LSTM input sequence length, number of LSTM layers, learning rate). For each model type, nodes may store one or more ML *model instances*. A *model instance* is a version of the given model type, incorporating a given set of observations. For each model instance, at a given point in time, the *training set* is thus the set of observations it has incorporated through the FG scheme. Two instances of the same model type may differ in (i) model coefficients, and/or (ii) training set composition. For each model type, at each node, the *local model instance* is the instance that needs to be continuously updated, and which is exchanged with other nodes. When a node enters the RZ (or at the start of the FG scheme, for nodes located within the RZ), for each model type it receives a pre-configured instance (also called *default instance*), to which an empty training set is associated. Indeed, such an instance is not yet populated with fresh data from the RZ (for example, a rescue team entering a disaster area may be endowed with an ML instance that does not yet contain

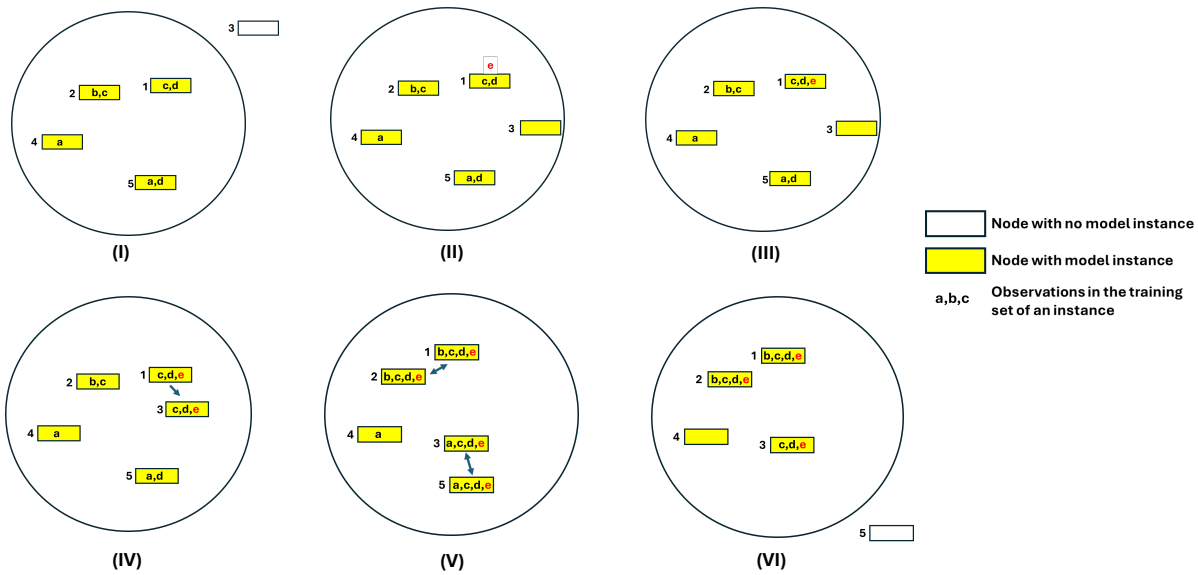


Fig. 1: Observation lifecycle in the FG scheme. (I) Nodes in the RZ (represented as a circle) have instances of the same ML model, with generally different training sets. Nodes outside have no instance; (II) Once node 3 enters the RZ, it acquires an instance of the ML model with an empty training set (a *default instance*). One observation (denoted with label e) is generated at node 1; (III) Node 1 incorporates the new observation into its model instance through local training; (IV) Node 3 comes in contact with node 1, and it sends its instances to node 3; (V) As nodes move, they come in contact and exchange their instances. The instances received by each node are merged with the local one, creating a new instance whose training set is the union of the training sets of the merged instances; (VI) When an observation becomes stale (observation a), it is removed from the training set of all instances. When a node (e.g., Node 5) leaves the RZ, it erases its instance.

recent data about the critical issues in the area). During their sojourn inside the RZ, nodes collect observations at various locations and points in time. Each observation is collected by a single node chosen uniformly at random among those in the RZ (extensions to arbitrary observation generation patterns are possible). Whenever a node collects a new observation, it incorporates it into one or more ML model instances by training, to update ML models used to make decisions or perform some tasks within the RZ. We assume that each model type is linked to a distinct process governing the arrival of observations within the RZ (such as, e.g., a distinct data source or a distinct physical process generating the observations). Specifically, we focus on the scenario where each observation arrival process is uniquely associated with a single model type. A model is thus a description of the environment according to one dimension, and the set of models provides an M-dimensional representation of the environment within the RZ. In what follows, observation arrivals follow a Poisson process with rate λ observations per second, equal for all processes. We assume all model instances from all types to have the same size, equal to L bits. We note, though, that our approach can be easily extended to more general settings, such as scenarios with different model sizes and different observation arrival rates. On the contrary, the assumptions of uniform distribution of nodes over the RZ and of Poisson observation arrivals are required for our mean field analysis approach, and can be considered as its main limitations.

Model instances are exchanged opportunistically between nodes through D2D communications, over links of capacity C bits per second, and thus with no global connectivity among them. After the exchange, at each node involved, all model instances of the same type are *merged* into a single new local instance incorporating all the observations from the merged instances, analogously to what is done in the parameters server in Federated Learning schemes [19].

Observations become obsolete and thus not useful after a given amount of time. The *age* of an observation is the time since its generation. An age limit (denoted as observation lifetime τ_l) is defined for each observation, beyond which it is not considered useful, and thus discarded (if already incorporated in a training set). Thus, models have to be continuously updated by incorporating new observations as long as they are generated, while stale observations are discarded. Moreover, we assume that models are relevant only in the region where observations are generated. UEs moving out of the RZ forget their local model instance(s) and all collected observations. If a node re-enters the RZ at a later time, it does so with only a default instance for each model type. This could be implemented, e.g., by having a common metadata structure for every model instance, incorporating training set and generation time for each observation within it.

Such a scheme therefore implements an incremental or continual learning scenario [30], where appropriate training, merging, and forgetting mechanisms ensure that only non-stale data are incorporated in models and that phenomena such as catastrophic forgetting are avoided.[2] As our modeling approach is general enough to apply to a large set of learning architectures, in the rest of this paper, we will not refer to any

[2]In this sense, our study captures the limit behavior of actual continual learning architectures.

TABLE II: Main notation used in the paper.

Notation	Parameter
A	Area of the Replication Zone (m^2)
N	Mean number of nodes in the Replication Zone
L	Model size ($bits$)
M	Number of model types
g	Mean contact rate per node (s^{-1})
λ	Per-model observation generation rate (s^{-1})
α	Node departure rate (s^{-1})
t_0	Transfer setup time (s)
τ	Observation age (s)
τ_l	Observation lifetime (s)
T_T	Training time (s)
T_M	Merging time (s)
T_L	Model transfer time (s)

specific ML model architecture. Note that the observation lifetime parameter τ_l serves a dual purpose. First, it ensures that observations that have become stale due to temporal dynamics are automatically discarded (e.g., in a disaster area, environmental sensor readings become invalid after a significant time). Second, it implicitly accounts for data drift scenarios where the underlying statistical properties of the environment change over time. By bounding the age of observations incorporated into models, the system prevents catastrophic drift where models are trained on observations from fundamentally different environmental regimes. The specific value of τ_l should be set based on domain knowledge about the rate of environmental change in the target application.

Table II summarizes the main notation used in the paper. Fig. 1 illustrates the key aspects of FG operations.

C. Training and merging tasks

Model *training* operates a transformation that takes as input a model instance and a new set of one or more observations and gives as output a new model instance, with a training set given by the union of the training set of the old model instance and the new set of observations. We assume that model training is performed one observation at a time, though our approach easily extends to the case in which all observations in the new set are incorporated as a batch in the model instance. The incorporation of an observation in a model instance via training lasts T_T seconds.

Each node is equipped with a wireless transceiver for wireless D2D communications. We say that two nodes are *in contact* when they can directly exchange information via wireless communications. When two nodes come in contact with each other within the RZ, they may exchange their model instances. We assume that data exchanges are one-to-one (i.e., unicast) and that the duration of the connection establishment process is constant (and denoted as t_0). We assume nodes can engage in at most one exchange at a time. We say that a node is *busy* at a given time when it is involved in an exchange of model instances with another node. Once an exchange of model instances between two nodes is complete, the nodes disconnect and are again available for new exchanges with other nodes. The time necessary for the transfer of a model instance is $T_L = \frac{L}{C}$. We assume that the order in which instances are transferred is random, as optimization of transfer scheduling is out of the scope of the present work.

After an exchange, nodes merge each received instance with the local instance of the corresponding model type. Specifically, the *merging* operation is a transformation that takes as input two instances of a model and gives a new local model instance as output, whose training set is the union of the training sets of both input instances. The merging of two instances lasts T_M seconds.

For what concerns the specific merging algorithm, the coefficients of the ML model obtained through merging are derived as a weighted average of the coefficients of the merged model instances, typically, as is typical of Artificial Neural Networks (ANNs). However, our approach does not depend on the choice of a specific merging algorithm. Rather, it applies to any ML architecture for which the merging operation is well-defined.

Two instances of the same model type with the same training set are considered to be identical, independently of the specific sequence of training and merging operations that have produced them, although in practice this may not always be the case. For this reason, if two nodes come in contact and their model instances have identical training sets, they are not exchanged. Whenever a node has to exchange model instances, for each model type, it sends only the non-default local instance it possesses, if it has one. Thus, in general, all model instances received from other nodes are not exchanged. However, when a node possessing only a default instance for a given model type receives an instance of that same model type from another node, it considers the received instance as the new local instance for that model type, and it starts exchanging it whenever possible. Algorithm 1 illustrates the mechanism by which model instances are exchanged in FG.

Computing tasks are managed via two FIFO queues: one for training, one for merging. Since observations arrive sequentially while model instances arrive asynchronously, tasks are queued individually and processed one-by-one. A single shared service with finite capacity processes both queues, with merging tasks being served with non-preemptive priority over training tasks. Under high computational load, prioritizing merging ensures recently incorporated observations can be disseminated to other nodes in the RZ. Conversely, high-priority training would render most tasks obsolete before dissemination completes. However, our approach easily generalizes to other scheduling strategies.

IV. Performance Analysis of Floating Gossip

As explained above, the main goal of the FG scheme is to enable the collaborative elaboration of "up-to-date" models, i.e., models that incorporate as much as possible of the relevant data collected by users about a given context, and to make them available to these same users in a timely fashion, without external infrastructure support.

Denote by Y the probability that at a node, the local model instance is not merged with a received model instance because the training set of the former is a superset of the training set of the latter. We say that the system is in the *substable regime* when $Y \approx 0$. In practice, in the substable regime,

Algorithm 1 ModelExchange(i, j)

Require: nodes i, j
Ensure: Updated model instances at both nodes
if i and j are in wireless D2D range AND neither is busy **then**
  establish connection with duration t_0 ▷ *connection setup time*
  set i and j in the busy state
else
  return ▷ *no action*
end if
models_to_exchange $\leftarrow \emptyset$
for each model type $m \in \{1, 2, \ldots, M\}$ **do**
  instance$_i \leftarrow i$.local_instance$[m]$
  instance$_j \leftarrow j$.local_instance$[m]$
  if instance$_i$.training_set $\neq$ instance$_j$.training_set **then**
    if instance$_i$ is not DEFAULT **then**
      models_to_exch.add$(m, \text{instance}_i, i \rightarrow j)$
    end if
    if instance$_j$ is not DEFAULT **then**
      models_to_exch.add$(m, \text{instance}_j, j \rightarrow i)$
    end if
  end if
end for
for each exchange $(m, \text{inst}, \text{sender} \rightarrow \text{receiver})$ in models_to_exch **do**
  transfer_time $\leftarrow L/C$
  if tr_time $\leq$ remaining_c_time **then**
    transfer inst from sender to receiver
    receiver.received_instance$[m] \leftarrow$ inst
    remaining_c_time $\leftarrow$ r_contact_time $-$ tr_time
  else
    break ▷ *transfer interrupted, connection ends*
  end if
end for
Terminate connection
Remove i and j from the busy state

computing load is low enough to ensure timely observation incorporation in ML models. In other words, the system is in the substable regime when models are updated at a rate that is high enough for pairs of nodes exchanging their model instances to incorporate several observations which are not yet part of the training set of the local instance, e.g., because they have recently been generated and thus not yet disseminated into the system. In scenarios where these conditions are not verified, the substable regime assumption is conservative for what concerns the effectiveness of the FG system in training ML models, as it maximizes the queue length for both training and merging tasks. In the rest of our analysis, we will assume the system is in the substable regime.

We now introduce some key parameters describing the dynamics of the FG system, namely the *model availability* and the *node busy probability*.

Definition 1. (**Model availability**) *In a RZ of area A, the availability of a model of type m at time t in class j nodes, (denoted as $a_{m,j}(t, A)$, $j \in \{s, f\}$) is the mean fraction of class j nodes possessing a non-default instance (i.e., an instance with a non-empty training set) of that model type.*

Model availability generally takes different values for different model types in the same RZ. Let us consider the case in which the population of nodes with a non-default instance of a given model type is stationary (as the arrival process of observations is stationary, such a stationary state for the node population exists) for all model types. Let us denote the mean fraction of class j nodes busy at time t with $b_j(t, A)$. Moreover, let $b_j(t, A) = b_{jj}(t, A) + b_{ji}(t, A)$, where $b_{jj}(t, A)$ (resp. $b_{ji}(t, A)$) is the fraction of nodes which are busy due to contacts between two class j nodes (resp. between a node of class j and one of class i), with $i, j \in \{s, f\}, i \neq j$. Let us introduce the steady-state quantities $a_{m,j}(A)$ and $b_j(A)$. As we assume λ to be the same for all model types, $a_{m,j}(A) = a_{m',j}(A)$, $\forall m, m'$. Thus, we drop the model type index m from the notation of model availability. Let $T_{T,j}$ and $T_{M,j}$ denote the training and merging times for a node of class $j \in \{s, f\}$, respectively. We thus have the following result. It shows that the system stability condition for nodes of class j is of the threshold type with respect to the per-model observation generation rate λ (i.e., $\lambda < \lambda_{\max}$), and that the threshold value depends on the ratio between the average number of nodes in the RZ, N and the number of ML models M, on model training and merging times on nodes of class j (respectively, $T_{T,j}$ and $T_{M,j}$), and on the mean arrival rate of merging tasks at nodes of class j (r_j).

Theorem 1. ***(Stability condition, mean training and merging delay)*** *$\forall j \in \{s, f\}$, the mean arrival rate of merging tasks at a class j node, denoted as r_j, is given by $r_j = Mg(1 - b_j(A))[a_j(A)S_{jj}v_{jj}(1 - b_j(A)) + a_i(A)S_{ji}v_{ji}(1 - b_i(A))]$ where S_{ji} (for any j and i, including i=j) is given by*

$$S_{ji} = \int_{t_0}^{+\infty} \min\left(1, \left\lfloor \frac{(t - t_0)}{T_L} \right\rfloor \frac{L}{\gamma_{ji}}\right) f_{ji}(t)dt$$

where t_0 is the transfer setup time, and $T_L = \frac{L}{C}$ is the mean time necessary for the transfer of a single model instance between two nodes. f_{ji} is the PDF of contact duration between a node of class j and one of class i. When the condition

$$\frac{M\lambda}{N} T_{T,j} + r_j T_{M,j} < 1 \tag{1}$$

holds, then the system is said to be stable, and the mean time required by an instance of the m-th model received by a class j node to be incorporated via training (respectively, merged), denoted as $d_{I,j}$ (resp. $d_{M,j}$), is given by

$$d_{M,j} = T_{M,j} + \frac{0.5(\rho_{M,j}T_{M,j} + \rho_{T,j}T_{T,j})}{1 - \rho_{M,j}} \tag{2}$$

$$d_{I,j} = T_{M,j} + \frac{0.5(\rho_{M,j}T_{M,j} + \rho_{T,j}T_{T,j})}{(1 - \rho_{M,j})(1 - \rho_{M,j} - \rho_{T,j})} \tag{3}$$

where $\rho_{T,j} = \frac{M\lambda T_{T,j}}{N}$, and $\rho_{M,j} = r_j T_{M,j}$.

Proof. As a consequence of the substable regime assumption, we can assume that the probability that two models are not merged because their training sets are identical or one is a subset of the other is negligible. Thus, the mean amount of model instances that a node j may receive on a contact with another node j (resp. i) is Ma_j (resp. Ma_i), if every contact

brings a full exchange of all that can be exchanged. To this we add as a factor the S_{jj} (resp. S_{ji}), which model the likelihood for a model to be exchanged, given the distribution of the duration of a contact between two nodes from class j (resp. between a node from class j and a node from i).

Then, the mean arrival rate of merging tasks at a class j node is derived by accounting for g (the mean contact rate) and v_{jj} (resp. v_{ji}), i.e. the fraction of those contacts involving two class j nodes (resp. a class j and a class i node).

Finally, of all contacts, those that may lead to an exchange of models are those in which neither node is already busy exchanging model instances. The probability of such an event is $(1-b_j(A))^2$ for the j-j case and $(1-b_j(A))(1-b_i(A))$ for the j-i case.

Each class j node models as an $M/D/1$ queue with two customer types: merging tasks (high priority) and training tasks (low priority). From the stability of non-preemptive priority queues, we derive the stability condition and mean delay expressions.

For the high-priority type, the arrival rate is $r_j T_{M,j}$, and it is $\frac{M\lambda}{N}$ for low priority. From the condition of stability of an $M/D/1$ queue with non-preemptive priority, we derive the condition $r_j T_{M,j} + \frac{M\lambda T_{T,j}}{N} < 1$. The derivation of the delay formulas follows standard queuing theory results for such systems [31]. □

The stability condition implies that both the queue of training tasks and that of merging tasks are stable. However, when the stability condition is not satisfied, queues do not grow indefinitely. Indeed, in that case, there is not enough service capacity for serving all of the training tasks waiting in the queue, and several of them will expire before being served (a training task expires whenever the observation associated with it reaches its maximum age). As a consequence, fewer training tasks are executed, and the condition $Y \approx 0$ (substable regime assumption) does not hold anymore. This entails a decrease in the mean amount of merging tasks waiting to be executed at any point in time in the system, and thus of the overall computing load at each node.
In any case, when condition 1 is not satisfied, even if both queues remain bounded, the fraction of training and merging tasks that are never executed, and thus the fraction of arriving observations not incorporated in a model, becomes substantially large. That is, the system is no longer able to incorporate effectively the collected observations.

Let us consider an observation that has been incorporated into the m-th model at a class j node, and let τ denote its age. With $o_j(\tau)$ we denote the mean *observation availability*, i.e., the mean fraction of class j nodes with the m-th model whose local instance includes the given observation at age τ, at the mean field limit.

Definition 2. (**Observation availability**) *When the system is stable, given a class j node in the RZ, a given model type, and a given observation of age τ, at the mean-field limit, the mean observation availability $o_j(\tau)$ is the probability to find that observation of age τ in the training set of an instance of that model type at that node.*

As a consequence of what we have assumed for observations, $o_j(\tau) = 0$ for $\tau > \tau_l$. For a node with a non-default model instance in the RZ, a *return to default* (RTD) event takes place whenever all the observations included in its training set expire. In this case, the instance is considered equivalent (e.g., in terms of accuracy) to the default instance for that model type. For a single class j node, with $P_{d,j}(t)$ we denote the *RTD probability* for class j, i.e., the probability that the model at the node will return to default in the interval $(t, t+dt)$.
The following result states that, $\forall m$, steady state values of $a_{m,j}(t,A)$, $b_{jj}(t,A)$ and $b_{ji}(t,A)$ depend neither on specific trajectories (i.e., on the amount of initial seeders), nor on the time at which each model has been seeded, but only on properties of the system once all transients are exhausted.

Theorem 2. ***(Mean model availability at the mean field limit)*** *For $t \to \infty$, $\forall m$, when the system is stable, the mean values of the variables $a_{m,j}(t,A)$, $b_{jj}(t,A)$, and $b_{ji}(t,A)$ (with $i,j \in \{s,f\}, j \neq i$) at the mean-field limit (denoted as a_j, b_{jj} and b_{ji}, respectively) are given by the unique solution of the following fixed point problem in a_j, b_{jj} and b_{ji}:*

$$\frac{b_{jj}}{T_{S,jj}}\frac{a_j(1-a_j)}{E_j}S_{jj} + \frac{b_{ji}}{T_{S,ji}}\frac{a_i(1-a_j)}{E_{ji}}S_{ji} +$$

$$+\min\left(\frac{\lambda}{D_j A}, \frac{1-r_j T_{M,j}}{T_{T,j}}\right)(1-a_j) - \left(\frac{\alpha_j}{D_j A} + P_{d,j}(t)\right)a_j = 0$$

$$\frac{2v_{jj}g}{D_j}(1-b_{jj}-b_{ji})^2\left[1-(1-a_j)^{2M}\right] - \frac{b_{jj}}{T_{S,jj}} - \frac{\alpha_j}{AD_j}2b_{jj} = 0$$

$$\frac{v_{ji}g}{D_j}(1-b_{jj}-b_{ji})(1-b_{ii}-b_{ji})\left[1-(1-a_j)^M(1-a_i)^M\right] +$$

$$-\frac{b_{ji}}{T_{S,ji}} - \frac{\alpha_j}{AD_j}b_{jj} - \frac{\alpha_i}{AD_j}b_{ji} = 0 \tag{4}$$

with

$$\begin{aligned}
&E_j = 1-(1-a_j)^2 1_{M=1}\\
&E_{ji} = 1-(1-a_j)(1-a_i)1_{M=1}\\
&T_{S,jj} = \int_0^{+\infty} \min\left(t, \frac{\gamma_{jj}}{L}T_L + t_0\right) f_{jj}(t)dt\\
&T_{S,ji} = \int_0^{+\infty} \min\left(t, \frac{\gamma_{ji}}{L}T_L + t_0\right) f_{ji}(t)dt\\
&\gamma_{ji} = \frac{ML(a_j+a_i)}{E_{ji}}, \quad \gamma_{jj} = \frac{2MLa_j}{E_j}
\end{aligned} \tag{5}$$

where $1_{M=1}$ is the indicator function. Independent of the initial condition, any trajectory of the system converges to the unique solution of the fixed-point problem.

Proof. To derive the mean value of the variables $a_{m,j}(t,A)$, $b_{jj}(t,A)$ and $b_{ji}(t,A)$ at the mean field limit, we first derive the differential equations for the drift at time t [32], [33], which describes the average local variation of these quantities over time. To this end, we assume the system satisfies the

homogeneous conditions. These imply that (1) at $t = 0$ the mean number of nodes per unit surface possessing a given model is the same for all models; (2) at any time instant, nodes possessing a given model are uniformly distributed within the region; and that (3) the probability of a node to have that model is independent from the probability of any other node to have the same model.

Let $N_{a,m,j}$ denote the mean number of class j nodes in the region possessing a non-default instance of the $m-th$ model at time t, and $N_{b,j}$ the mean number of class j nodes in the region which are busy at time t. Let $N_{b,j} = N_{b,jj} + N_{b,ji}$, where the two quantities are the number of class j nodes which are busy in a contact of the j-j type and j-i type, respectively. Analogously, let $b_j(t, A) = b_{jj}(t, A) + b_{jj}(t, A)$. For ease of notation, in what follows we drop the dependency on t and A. *Drift term* $\frac{dN_{b,jj}}{dt}$ *(resp.* $\frac{dN_{b,ji}}{dt}$*)*. The probability that the two nodes in contact are not busy is $(1 - b_j)^2$ for the j-j case, and $(1 - b_j)(1 - b_i)$ for the other case. Moreover, note that every contact of the j-j type leads to an increase of 2 in the number of class j busy nodes, while contacts of the j-i type bring an increase of only one node each. Note that for a contact between two nodes to bring those nodes to the busy state, at least one of them must have a non-default model. I.e., contacts between nodes possessing only default models do not count (with probability $1 - (1 - a_{m,j})^{2M}$ for the j-j case, and $1 - (1 - a_{m,j})^M(1 - a_{m,i})^M$ for the other case). Thus, the overall rate of increase of busy nodes is $2v_{jj}gA(1 - b_j)^2\left(1 - (1 - a_{m,j})^{2M}\right)$ for the j-j case, and $v_{ji}gA(1 - b_j)(1 - b_i)\left(1 - (1 - a_{m,j})^M(1 - a_{m,i})^M\right)$ for the j-i case.

The first loss component models the end of the busy state due to the completion of the model exchange process among nodes in contact. As the mean time required to complete such exchanges is $T_{S,jj}$ (resp. $T_{S,ji}$), by Little's law we have that such events take place with a mean rate $\frac{AD_j b_{jj}(t,A)}{T_{S,jj}}$ (respectively, $\frac{AD_j b_{ji}(t,A)}{T_{S,ji}}$).

The second loss component accounts for class j busy nodes who stop being busy because they leave the RZ due to their mobility. Let α_j denote the mean rate at which class j nodes exit the RZ. We assume this rate to be equal to the one at which class j nodes enter the region, so that the mean amount of class j nodes does not change over time. Specifically, we have a contribution from nodes in a contact of the type j-j (from which *two* class j busy nodes stop being busy, even if only one of the two leaves the RZ). Then we have another contribution from class j nodes in a contact of the type j-i (from which only a single class j busy node stops being busy) and due to the departure of the class j node. Finally, we have another contribution from class i nodes in a contact of the type j-i, due to the departure of the class i node. Thus we have a term of the form $-\alpha_j 2b_{jj}$ for nodes in a contact of the type j-j, and $-\alpha_j b_{ji} - \alpha_i b_{ji}$ for nodes in a contact of the type j-i. Putting all together, and dividing by the mean number of class j nodes (i.e. by AD_j), we get $\frac{db_{jj}}{dt} = 2v_{jj}\frac{g}{D_j}(1-b_{jj}-b_{ji})^2\left(1-(1-a_{m,j})^{2M}\right) - \frac{b_{jj}}{T_{S,jj}} - \frac{\alpha_j}{AD_j}2b_{jj}$, and $\frac{db_{ji}}{dt} = v_{ji}\frac{g}{D_j}(1 - b_{jj} - b_{ji})(1 - b_{ii} - b_{ji}) \cdot \left(1 - (1 - a_{m,j})^M(1 - a_{m,i})^M\right) - \frac{b_{ji}}{T_{S,ji}} - \frac{\alpha_j}{AD_j}b_{jj} - \frac{\alpha_i}{AD_j}b_{ji}$.

Drift term $\frac{dN_{a,m,j}}{dt}$. By the definition of model availability, $N_{a,m,j} = a_{m,j}AD_j$, where D_j is class j mean node density, assumed to be constant over time. The drift of $N_{a,m,j}$ is given by the sum of three components. The first is due by those class j nodes which, before a contact, owned only a default instance for the m-th model. For contacts of the j-j type, the end of a contact corresponds to a decrease of 2 units in the number of busy nodes. However, it might lead to an increase of at most one unit in the number of nodes of the j-th class that own a nondefault model instance. Thus, the rate of such an event is half of the one in the drift equation for $N_{b,jj}$, and is equal to $0.5\left(\frac{N_{b,jj}}{T_{S,jj}}\right)$. When $M = 1$, two nodes in contact might start an exchange (1) because one possesses a default model, and the other a non-default one (probability $2a_{m,j}(1 - a_{m,j})$); or (2) because both have a non-default model, but with a different training set, and one is not a subset (proper or improper) of the other. In this last case, $N_{a,m,j}$ won't increase. We neglect this, assuming (conservatively, as for what concerns performance of the system) that if both nodes have an instance for the same model (probability a_j^2), they always exchange it both ways. Thus, the mean fraction of all contacts of the j-j type which increase $N_{a,m,j}$ is $\frac{2a_{m,j}(1-a_{m,j})}{1-(1-a_{m,j})^2}$. Putting all together, and multiplying it by the probability that the contact duration is sufficient to allow such model exchange (S_{jj}) we have, for $M = 1$, $\frac{N_{b,jj}}{T_{S,jj}}\frac{a_{m,j}(1-a_{m,j})}{1-(1-a_{m,j})^2}S_{jj}$. For $M > 1$, two nodes in contact might still exchange instances other than those of the $m-th$ model, even if they have no instance for the $m-th$ model (probability $(1 - a_{m,j})^2$). Thus, the mean fraction of all contacts of the j-j type which increase $N_{a,m,j}$ is $2a_{m,j}(1 - a_{m,j})$, and for $M > 1$ we have $\frac{N_{b,jj}}{T_{S,jj}}a_{m,j}(1 - a_{m,j})S_{jj}$. Bottom line, by using the indicator function, for contacts of the $j - j$ type the first contribution can be expressed as

$$\frac{N_{b,jj}}{T_{S,jj}}\frac{a_{m,j}(1 - a_{m,j})}{1 - (1 - a_{m,j})^2 1_{M=1}}S_{jj} \tag{6}$$

Let us consider now contacts of the j-i type. The end of one such contact corresponds to the decrease of *one* unit of the number of busy nodes of class j, and thus, to an increase of *at most* one unit of the number of nodes of the $j - th$ class which own a non-default model instance. Thus, the rate of such an event is the same as the one in the drift equation for $N_{b,ji}$, i.e., $\frac{N_{b,ji}}{T_{S,ji}}$. When $M = 1$, with similar considerations we have that the mean fraction of all ended contacts of the j-i type which increase $N_{a,m,j}$ is $\frac{a_{m,i}(1-a_{m,j})}{a_{m,i}+a_{m,j}-a_{m,i}a_{m,j}}$. Putting all together, and multiplying it by S_{ji} we have, for $M = 1$,

$$\frac{N_{b,ji}}{T_{S,ji}}\frac{a_{m,i}(1 - a_{m,j})}{a_{m,i} + a_{m,j} - a_{m,i}a_{m,j}}S_{ji} \tag{7}$$

For $M > 1$, taking into account that the probability that the two nodes have no instance for the $m - th$ model is $(1 - a_{m,j})(1 - a_{m,i})$), and using the indicator function, we have

$$\frac{N_{b,ji}}{T_{S,ji}}\frac{a_{m,i}(1 - a_{m,j})}{1 - (1 - a_{m,i})(1 - a_{m,j})1_{M=1}}S_{ji} \tag{8}$$

with $i, j \in \{s, f\}$. Putting all together, we have that the first contribution to the drift is given by the sum of (6) and (8).
The second term is given by the rate at which nodes (from all classes together) receive an observation for the $m-$th model (given by λ), multiplied by the fraction of those nodes which belongs to class j (given by $\frac{D_j}{D}$), and by the probability that those class j nodes do not possess the $m-$th model yet (i.e., $(1 - a_{m,j})w$). The term $\alpha_j w a_{m,j}$ accounts for the rate at which class j nodes with an instance for model m leave the RZ. Finally, the second loss term in this equation is due to the rate at which non-default models turn to default models in class j nodes. This is given by $D_j A a_{m,j} P_{d,j}(t)$. Putting all together, and dividing by the mean number of class j nodes (i.e. by AD_j), we get $\frac{da_{m,j}}{dt} = \frac{b_{jj}}{T_{S,jj}} \frac{a_{m,j}(1-a_{m,j})}{1-(1-a_{m,j})^2 1_{M=1}} S_{jj} + \frac{b_{ji}}{T_{S,ji}} \frac{a_{m,i}(1-a_{m,j})}{1-(1-a_{m,i})(1-a_{m,j})1_{M=1}} S_{ji} + \lambda \frac{1}{DA}(1-a_{m,j})w - \frac{\alpha_j}{D_j A} w a_{m,j} - a_{m,j} P_{d,j}(t)$. From these drift expressions, by applying standard results, it is easy to prove the convergence to the mean field limit.

Let us define the *busy period* the fraction of contact time between two nodes spent transferring model instances. Then we have that, when the busy period duration is larger than zero, the mean amount of models transferred is well approximated by $\gamma_{ji} = \frac{ML(a_j+a_i)}{1-(1-a_j)(1-a_i)1_{M=1}}$. To derive this, we note that we assume that the nodes have always at least one model to transfer. Then for the case j-i, $M = 1$, the nodes transfer a single model (of size L bits) with probability $a_j(1-a_i)+a_i(1-a_j)$, and two models (one per direction of transfer) with probability $a_i a_j$. Thus $\gamma_{ji,M=1} = \frac{L(a_j+a_i)}{a_j(1-a_i)+a_i}$. In the j-j, $M = 1$ case we have, similarly, $\gamma_{jj,M=1} = \frac{2La_j}{a_j(1-a_j)+a_j} = \frac{2L}{2-a_j}$. In the case j-i, $M > 1$, for each model, the nodes transfer a single instance with probability $a_j(1-a_i)+a_i(1-a_j)$, two instances (one per direction of transfer) with probability $a_i a_j$, and none with probability $(1-a_j)(1-a_i)$. As the event of possessing an instance of a given model is assumed to be independent from the event of possessing an instance for other models, we have $\gamma_{ji,M>1} = ML(a_j + a_i)$. Finally, in the case j-j, $M > 1$, we have, similarly, $\gamma_{jj,M>1} = 2MLa_j$. □

Note that this result relies on a mean-field approximation, whose accuracy improves with system scale—particularly as the mean number of contacts per node in the RZ increases—remaining reliable even when key assumptions, such as uniform node distribution, are mildly violated. With α_j, $j \in \{s, f\}$, we denote the mean rate at which nodes enter the RZ. Definition 2 implies that, given a class j node, a model and an observation of age τ, the probability that it is included in an instance of that model on that node is $wa_j o_j(\tau)$. Thus, a performance metric more closely related to the ability of the FG systems to incorporate information into models in a timely fashion is the mean observation availability at the mean field limit. The following result shows that the mean observation availability at the mean field limit can be characterized through a compact system of delay-differential equations (DDEs). These DDEs capture how quickly a given observation propagates across node classes via opportunistic contacts, training, and merging, accounting for node churn and observation aging. Intuitively, higher contact rates and shorter delays enhance availability by accelerating diffusion. At the same time, the results reveal a coupling between classes. That is, observations originating from fast-moving nodes reach slow/static ones more gradually.

Theorem 3. ***(Mean observation availability at the mean field limit)*** *When the stability condition holds, at the mean-field limit the mean observation availability at age $\tau \leq \tau_l$ for classes i and j, $i, j \in \{s, f\}, j \neq i$, is given by*

$$o_j(\tau) = \frac{D_j}{D} o_j^P(\tau) + \frac{D_i}{D} o_j^S(\tau) \tag{9}$$

$$o_i(\tau) = \frac{D_j}{D} o_i^S(\tau) + \frac{D_i}{D} o_i^P(\tau) \tag{10}$$

Where $o_j^P(\tau)$, $o_i^P(\tau)$, $o_j^S(\tau)$, and $o_i^S(\tau)$ are the solutions of the following system of delay-differential equations that model the evolution over time of the average observation availability at class j nodes when the observation arrives at a node in either class j or class i:

$$\frac{do_j^P(\tau)}{d\tau} = B_{jj} a_j (1-a_j) o_j^P(\tau) + B_{ji} a_i (1-a_j) o_i^S(\tau) + \\ + B_{jj} a_j^2 o_j^P(\tau - d_{M,j})(1 - o_j^P(\tau - d_{M,j})) + \\ + B_{ji} a_i a_j o_i^S(\tau - d_{M,j})(1 - o_j^P(\tau - d_{M,j})) - \frac{\alpha a_j o_j^P(\tau)}{D_j A}$$

with $d_{I,j} \leq \tau \leq \tau_l$, and with initial condition

$$o_j^P(\tau) = \begin{cases} 0 & \tau < d_{I,j} \\ \frac{1}{\lceil a_j D_j A \rceil} & \tau = d_{I,j} \end{cases} \tag{11}$$

$$\frac{do_i^S(\tau)}{d\tau} = B_{ii} a_i (1-a_i) o_i^S(\tau) + B_{ji} a_j (1-a_i) o_j^P(\tau) + \\ + B_{ii} a_i^2 o_i^S(\tau - d_{M,i})(1 - o_i^S(\tau - d_{M,i})) + \\ + B_{ji} a_i a_j o_j^P(\tau - d_{M,i})(1 - o_i^S(\tau - d_{M,i})) - \frac{\alpha a_i o_i^S(\tau)}{D_i A}$$

with $d_{I,j} \leq \tau \leq \tau_l$, and with initial condition

$$o_i^S(\tau) = \begin{cases} 0 & \tau < d_{I,j} \\ \frac{1}{\lceil a_i D_i A \rceil} & \tau = d_{I,j} \end{cases} \tag{12}$$

Where $B_{jj} = \frac{b_{jj}}{T_{S,jj}} E_j S_{jj}$, and $B_{ji} = \frac{b_{ji}}{T_{S,ji}} E_{ji} S_{ji}$.

Proof. We derive the drift, which describes the average local variation of observation availability over time. We assume that the time between two consecutive contacts between two nodes j is $\sim Exp(\eta_j)$ and between a node j and a node i is $\sim Exp(\eta_{ji})$. Thus, the time t until the next contact is $\sim Exp(\eta_j + \eta_{ji})$, and the probability that this contact is with a node j is $\frac{\eta_j}{\eta_j + \eta_{ji}}$.

Let $N_{o,j}(\tau, A)$ denote the number of class j nodes in the RZ possessing a model instance including the given observation, at a given time τ from observation generation, for a RZ area A. I.e., $N_{o,j}(\tau, A) = N(A) a_j o_j(\tau, A)$. For what concerns the initial period after the new observation arrives in the system, two cases are possible:
1) Arrived at a class j node (with probability $\frac{D_j}{D}$). We have

$N_{o,j}(\tau, A) = 0,\ 0 \leq \tau \leq d_{I,j}$. After incorporation, the number of class j nodes possessing an instance with that observation will not increase for at least $d_{M,j}$ seconds. Thus, $N_{o,j}(\tau, A) = 1,\ d_{I,j} \leq \tau \leq \tau_j$, with $\tau_j = d_{I,j} + d_{M,j}$ *2) Arrived at a class i node (with probability $\frac{D_i}{D}$).* We have $N_{o,j}(\tau, A) = 0,\ 0 \leq \tau \leq \tau_i$ where $\tau_i = d_{I,i} + d_{M,j}$. Thus, in all cases, $N_{o,j}(\tau_0, A) = 1$. For $\tau \geq \tau_j$ (resp. $\tau \geq \tau_i$), the rate at which this quantity varies over time is given by the sum of three components.

First component. It is due to contacts between, on one side, a node with a model that incorporates the given observation, and a node possessing only the default model. The mean rate at which class j nodes exit the busy state, from a j-j contact is $\frac{N_{b,jj}}{T_{S,jj}}$. The end of a contact may lead to an increase of at most one unit in the number of nodes of the j-th class that own a non-default model instance. Thus, the actual rate of increase is $0.5\frac{N_{b,jj}}{T_{S,jj}}$. Moreover, let us consider one of these terminating exchanges. The probability that the model was transferred to the node possessing only the default model during such exchange is equal to the probability that the sender node had the model (i.e., that specific model incorporating the observation we are considering) and that the receiving node did not have it, given by $\frac{2a_j(1-a_j)}{1-(1-a_j)^2 1_{M=1}} S_{jj}$. The probability that the transferred model incorporated the given observation is instead $\frac{N_{o,j}(\tau,A)}{a_j N_j} = o_j(\tau, A)$.

Second component. It accounts for the case in which the increase of the observation availability is due to the fact that a node already with a non-default instance incorporated the given observation through merging with a received instance which contained the given observation. Thus, this event is associated to an exchange of instances among two nodes, each possessing a non-default instance. For the j-j case, proceeding as already done, we get $\frac{N_{b,jj}}{T_{S,jj}} \frac{a_j^2}{1-(1-a_j)^2 1_{M=1}} S_{jj} o_j(\tau - d_{M,j}, A)(1 - o_j(\tau - d_{M,j}, A))$ and, for the j-i case, $\frac{N_{b,ji}}{T_{S,ji}} \frac{a_i a_j}{1-(1-a_i)(1-a_j)1_{M=1}} S_{ji} o_i(\tau - d_{M,j}, A)(1 - o_j(\tau - d_{M,j}, A))$. Note that the observation availability is evaluated at $\tau - d_{M,j}$. as we account for the time taken to merge the received instance.

Third component. It is given by class j nodes with the given observation exiting the RZ, and it is given by the rate at which nodes exit the RZ (i.e., α) multiplied by the fraction of those exiting nodes which has the model and incorporates the observation (i.e., $w a_j o_j(\tau, A)$). Putting all together, and normalizing by $D_j A a_j$, we get the final expression of the drift. The rest of the proof follows the standard procedure for mean field convergence, i.e., the convergence of initial conditions, and the condition on the drift of the Population Continuous Time Markov Chain (PCTMC) associated with the system. As its drift is continuous for $\tau \geq d_I$, let $\bar{o}(\tau) = \lim_{R\to\infty} o(\tau, A)$. As our sequence of PCTMC models satisfies these properties, by Theorem 1 in [32] for any finite time horizon $T \leq \infty$, the sequence of population models associated to A converges *almost surely* to the dynamics of the differential equations in (9), which proves Theorem 3. □

In the two-class scenario, this result shows that the observation availability for class j is the average of the observation availability for class j when the observation arrives (and is trained) at a class j node (denoted as $o_j^P(\tau)$) and of the one that we have when the observation arrives and is trained at a class i node (denoted as $o_j^S(\tau)$). This result states that the probability of observing a difference between any point of the trajectory of the availability of a given observation and the solution of the DDEs in Theorem 3 goes to zero as the RZ area (and hence the mean number of nodes in a RZ) grows. That is, in the mean field limit, the error made by considering a deterministic system characterized by $o_j(\tau)$, instead of the actual system goes to zero. Moreover, at any time τ from observation generation, for any A, $o_j(\tau)$ is the expected value of the observation availability at τ in a finite system.

Finally, note that the evolution over time of observation availability in class j nodes is the weighed average between that which is observed when the given observation arrives in a class j node, and that which takes place when the observation first arrives at a class i node, and it is then passed to class j nodes at la later time, via gossip based instance exchanges and merging. The following results gives an expression for the return-to-default probability, as a function of the observation availability at the mean field limit.

Lemma 1. ***(Return-to-default (RTD) probability)*** *Let $o_j(\tau)$ be the solution of the system of differential equations in Theorem 3, where a_j is derived from Theorem 2 by assuming $P_{d,j}(t) = 0$. Then the RTD probability $P_{d,j}(t)$ is given by*

$$P_{d,j}(t) = \lambda o_j(\tau_l) E_{n,\tau_1,\ldots,\tau_n}\left[\prod_{q=1}^{n}(1 - o_j(\tau_q))\right] \qquad (13)$$

where: (i) $\forall q \in [1, n]$, τ_q are distributed as a $Erlang(\lambda, q)$, truncated and restricted to the interval $[T_{T,j}, \tau_l]$; (ii) n is $Pois(\lambda(\tau_l - T_{T,j}))$.[3]

Proof. We consider a given node in the RZ, and we model the evolution of the system (i.e. a single node of class j) over time by a continuous time Markov chain (CTMC), where the state of a node is the number of observations included in the model possessed by the node. We focus on the state in which only a single observation is included in the model. In this case, the RTD probability is the joint probability that (1) in the time interval $(t, t + dt)$ the node does not incorporate any observation in the model, and that (2) the only observation included in the model has been generated in the time interval $(t - \tau_l, t - \tau_l + dt)$, and thus expires during the time interval $(t, t + dt)$. The probability of event (1) is the probability that no new observation is included by training in the node's model in that time interval, and that the node does not conclude a merging process in $(t, t+dt)$. For the first point, the process of arrival of new observations at a node is Poisson with intensity $\frac{\lambda}{N}$. For $dt \to 0$, the probability that in dt no new observation arrives at the node is $1 - \frac{\lambda}{N} dt$. The rate at which models to be merged arrive at the node is given by the product of the node

[3]Recall that the empty product is 1, i.e. $\prod_{q=1}^{0} = 1$.

contact rate, the probability that the other node in contact with the given node has the model too, and the probability that the two models have to be merged. We denote with P_{in} such rate, and with $P_{in}dt$ the probability that such an event takes place in a time interval of duration dt. We now compute the probability that the *only* observation included expires in $(t, t+dt)$. This is equivalent to the product of the probability that an observation is generated in a time interval of duration dt (equal to λdt) and that such an observation is still included in the model $\tau_l - dt$ seconds after its generation (probability $o_j(\tau_l, t, A)$). At the same time, for the above mentioned observation to be the only one included, all the other observations generated in the time interval $[t-\tau_l, t-T_{T,j}]$ must not be incorporated in the model instance at the node. If $n \geq 0$ is the total number of these observations, by the definition of observation availability we have that this probability is given by $\prod_{q=1}^{n}(1-o_j(\tau_q, t, A))$ with $n \sim Pois(\lambda(\tau_l - T_{T,j}))$.

As interarrival times are $Exp(\lambda)$, $\tau_q \sim Erlang(q, \lambda)$, truncated and restricted to the interval $[T_{T,j}, \tau_l]$. Putting all together, the probability of a RTD event for the given node during an interval dt is thus $o_j(\tau_l, t, A)(1 - dt\frac{\lambda}{N}\frac{D_j}{D})(1 - dtP_{in})\lambda dt \prod_{q=1}^{n}(1 - o_j(\tau_q, t, A))$. As $dt \to 0$, this expression becomes $N_j a_j \lambda dt o_j(\tau_l, t, A) \prod_{q=1}^{n}(1 - o_j(\tau_q, t, A))$. By computing the expectation of the above expression with respect to all the random variables involved, we have Equation (13). □

A measure of the ability of the FG scheme to incorporate the most recent measured data is *model staleness*.

Definition 3. (**Model staleness**) The *staleness* of a model instance at time t is the age of the most recent observation in the training set of that instance at t. The *model staleness* at t is the average staleness over all instances of that model type present in the RZ.

The mean model staleness is a measure of how much "up to date" on average an instance of that model is, in a given scenario. Note however, that a low model staleness is not necessarily an indicator of a high efficiency in incorporating observations generated in the RZ, as it may also be due to a high observation arrival rate.

Theorem 4. *(**Mean model staleness**) When the stability condition holds, for an observation lifetime τ_l, the mean staleness of a model is $F = \sum_j \frac{D_j}{D} F_j$, where*

$$F_j = \frac{\sum_{n=1}^{\infty} Pois(\lambda\tau_l, n)}{1 - Pois(\lambda\tau_l, 0)} \frac{\sum_{x=1}^{n} E[\tau_x] P_j(x)}{\sum_{x=1}^{n} P_j(x)}$$

with

$$P_j(x) = E\left[o_j(\tau_x) \prod_{q=1}^{x-1}(1 - o_j(\tau_q))\right] \tag{14}$$

where $\forall x \in \mathcal{N}$, $\tau_x = \sum_{k=1}^{x} \xi_k$, with ξ_k i.i.d. $\sim Exp(\lambda)$. τ_x follows thus a truncated $gamma(x, \lambda)$ distribution over the interval $[T_{T,j}, \tau_l]$.

Proof. We aim at deriving $E[\tau|\tau_l]$. This corresponds to considering only the staleness of those models which are non-default. Let n be the number of observations that have been generated in a time interval of duration τ_l. Clearly $n \sim Pois(\lambda\tau_l)$. Thus $E[\tau|\tau_l] = \frac{\sum_{n=1}^{\infty} E[\tau|\tau_l, n] Pois(\lambda\tau_l, n)}{1 - Pois(\lambda\tau_l, 0)}$. The normalization is required as we assume that at least one observation is generated in τ_l.

For a given time t, let $x \in \mathcal{N}$ be the label of the x-th last observation generated in the system, counting backward in time from t. With $E[\tau|\tau_l, n, x]$ we denote the expectation of τ conditioned on having observation 1 to $x-1$ not included in the model, and observation $x \leq n$ included. With $P_j(x)$ we denote the probability of such an event. We have, by the law of total probability, $E[\tau|\tau_l, n] = \frac{\sum_{x=1}^{n} E[\tau|\tau_l, n, x] P_j(x)}{\sum_{x=1}^{n} P_j(x)}$, $\forall n$. Again, normalization is required as $\sum_{x=1}^{n} P_j(x)$ does not necessarily give 1, despite we assume that at least one observation is included. As observation arrivals are Poisson with intensity λ, the distribution of τ conditional to τ_l, n, x is $gamma(x, \lambda)$, upper truncated at τ_l (conditioning to n has no effect).

We now turn to the derivation of $P(x)$. Let $\tau_j = \sum_{k=1}^{j} \xi_k$ denote the time since the generation of the j-th observation, with ξ_k i.i.d. $\sim Exp(\lambda)$ and upper truncated at τ_l, and $j \leq x$.

From the properties of the system at the mean field limit, and the definition of observation availability, for an observation generated x seconds ago, the probability to be included in a given model is $o_j(x)$. This gives Equation (14). □

V. Learning Capacity of an FG system

In this section, we derive a set of results that allow evaluating the key performance indicators of our decentralized FG framework. In what follows, we mostly refer to the case in which models are neural networks, but considerations may be extended to a large class of deep and shallow learning. In what follows, we refer to the concept of *model capacity*, intended as the maximum number of observations it can store. This idea stems from the fact that, when the total number of coefficients of an ML model instance is larger than the cardinality of its training set, then labels can be perfectly recovered (see [34] and references therein). One of the interesting consequences of this result is that it establishes a linear relationship between model size L, i.e., the total amount of bits required to represent the model coefficients, and the maximum cardinality of the training set associated with a model instance, related to the maximum amount of *information* that the model can store. Note that data in a training set may not be independent and identically distributed (i.i.d.), and data points might be redundant and correlated to varying degrees. Thus, the relationship between the size of a given dataset and the amount of information stored by an ML model instance trained on that dataset generally varies according to the specific set of data points considered. In what follows, we simplify by assuming that, if a model has length L, and k coefficients are needed on average to encode *independently from other coefficients* the mean amount of information contained in a data point, with a given ML algorithm, then the ratio L/k is the largest amount of data points that a model can store and recreate with arbitrarily small error [34]. k is thus the *memory efficiency* of the model.

One of the measures of effectiveness of FG training in a dynamic setting is the *node stored information*.

Definition 4. (**Node stored information**) *The node stored information at a given time t is the total number of non-expired observations incorporated in all of the model instances a node possesses at that time.*

Lemma 2. ***(Mean node stored information at steady state)*** *Consider an FG system at steady state for the process of model training, for which the stability condition* (1) *holds. Then the mean node stored information in a class j node is $Ma_j \min\left(\frac{L}{k}, \lambda \int_0^{\tau_l} o_j(\tau) d\tau\right)$.*

Proof. The probability for a class j node to possess an instance of a given model is a_j, and thus Ma_j is the mean number of model instances it possesses, at the mean field limit. For each model, the mean amount of observations incorporated is the minimum between the maximum amount that the model can store, and the mean of the total number of observations incorporated in an instance of that model with an age between 0 and τ_l, given by the integral of the observation incorporation rate within $[0, \tau_l]$. □

In what follows, we assume that, in a model already storing L/k observations, the incorporation of new data points implies discarding the oldest data point already stored in the model. This assumption corresponds to an ideal case, as in practical cases, many models simply degrade their accuracy when the number of points incorporated approaches the model's capacity. Indeed, in realistic ML architectures, this condition may give rise to *catastrophic forgetting*, i.e., to the erasure of a large share of the information stored, if appropriate techniques for model retraining are not adopted [35]. On the other side, a model capacity much larger than the amount of information it incorporates is a waste of storage, computing, and communication resources, and it exposes it to the risk of overfitting and thus to a worsening of generalization error [36].

Another key performance indicator of the FG scheme is the *learning capacity*.

Definition 5. (**FG learning capacity**) *In an FG system at steady state, the learning capacity (LC) is the maximum achievable ratio between the node stored information and the mean amount of non-expired observations present in the RZ at any time instant, over any feasible value of number of models and model size.*

The learning capacity gives an idea of up to which point our FG scheme can effectively incorporate in the models possessed by nodes in the region those observations which are continuously generated within it, and thus to keep the models up to date.

From its definition, the learning capacity of an FG scheme is the solution to the following problem:

$$\underset{M\geq 1, L\geq 1}{\text{maximize}} \quad \frac{\sum_j D_j a_j(L,M) \min\left(\frac{L}{k}, \lambda \int_0^{\tau_l} o_j(\tau, L, M) d\tau\right)}{D\lambda\tau_l}$$

subject to Eq.(1), (4), (9), (15)

The objective function is obtained by dividing the node stored information by the quantity $M\lambda\tau_l$, i.e. by the mean amount of information which is present in the FG system at any time instant, derived through Little's result. It can be easily seen that model and observation availability decrease with increasing communication load (and thus with model size L and number of models M), and that those constraints which are not satisfied for a given M_0 L_0 are not satisfied for any other $M > M_0$ and $L > L_0$. Thus, the solution of the above optimization problem is for $M = 1$.

Let us consider the cases of a single class and a two-class scenario. In the single class case, let L_{sat} be the value of L for which $a_j(L, M)$ is equal to 80% of its value at $L = 1$. Thus, for the optimization over L, let $L = L^*$ be the unique solution of the following fixed point problem:

$$\frac{L^*}{k} = \lambda \int_0^{\tau_l} o_j(\tau, L^*, M) d\tau \tag{15}$$

Then if $L^* >> L_{sat}$ the stored info is maximized for $L = L^*$. Else, it is maximized for $L = L_{sat}$ or around that value.

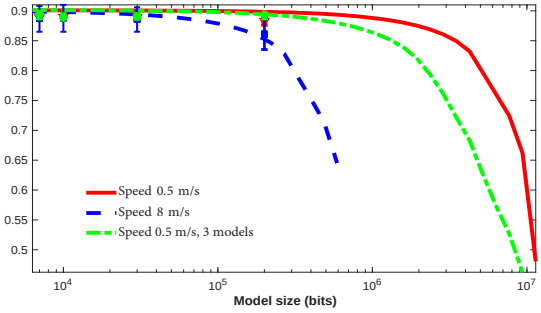
In the two classes case, for every class j, let L_j^* be the solution of the fixed point problem for that class. For each class j, the stored info for that class decreases monotonically for $L > L_j^*$, until saturation kicks in. The optimal L is within those values of L which are between the smaller and the larger L_j^*, and at the same time larger than $L = L_{sat}$.

VI. Numerical Evaluation

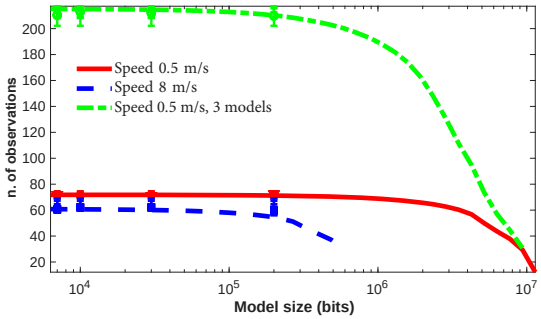
In this section, we use the mean-field model presented in this paper to characterize the limit performance (in the mean-field sense) of an FG system as a function of the main system parameters, and we assess the model's accuracy through detailed simulations.

A. Setup-default configuration

Unless otherwise specified, in our experiments we consider a square area with a side of 200 m, with a circular RZ at the center of it, whose radius is 100 m. Nodes have a transmission range of 5 m (typical of, e.g., Bluetooth devices, but we will also consider longer ranges), and enough storage to accommodate an arbitrary number of models. Nodes are assumed to move at a constant speed of 0.5 m/s (modeling average pedestrian mobility in crowded areas) or 8 m/s (about 30 km/h, i.e., slow urban vehicular mobility). We assume a fixed number of nodes (204 is the default value) move within the area according to the Random Direction Mobility Model (RDMM [37]), with reflections at the boundary of the simulated area, generating at any point in time a uniform spatial node distribution. Such a choice of user density and RZ size ensures that, for the given transmission radius, at any point in time, the wireless D2D connectivity among nodes does not give rise to a single cluster, while the area size and shape have been chosen to minimize border effects on the spatial distribution of users. When two nodes are in contact, the channel data rate is constant and equal to 10 Mb/s. The default duration of training and merging tasks is 5 s and 2.5 s, respectively, as when training simple (non-deep)

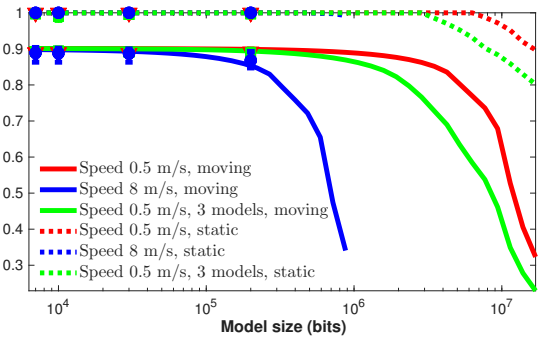


(a) Mean model availability

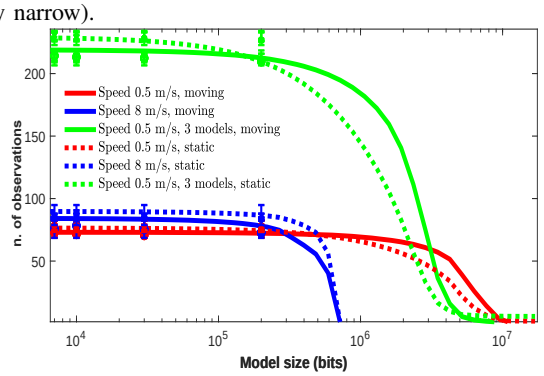


(b) Node stored information

Fig. 2: Mean model availability and node stored information in a single-class FG system versus model size L, for different values of node speed and number of models M. Simulation results (represented by markers) are averages with 95% confidence intervals (reported as vertical bars, sometimes too short to be visible because confidence intervals are very narrow).



(a) Mean model availability.



(b) Node stored information.

Fig. 3: Mean model availability and node stored information in a two-class FG system with 5% static nodes versus model size L, for different values of node speed and number of models M. Simulation results (represented by markers) are averages with 95% confidence intervals.

ML models on small datasets and on devices with modest computational resources, such as smartphones and IoT devices. The default maximum age for an observation is 20 minutes, to model dynamic settings such as theme parks or large open-air festivals, in which data gets stale relatively quickly. The default model size is 10 kb (hence, model exchanges require 2 ms), compatible with e.g., simple mobile-based applications, running on devices that limit computing power and storage consumed by a single app, not to jeopardize system resources. Unless otherwise specified, the memory efficiency parameter k has been set to its highest value (i.e., 1), though we also explore the impact of lower values of efficiency on FG performance.

At the beginning of each simulation run, nodes are randomly positioned within the simulated area, and they only possess default model instances. Simulation time is divided into equally sized slots, whose duration is chosen in such a way as to minimize quantization effects on the accuracy of results. Simulations are run until a steady state is reached, and the effects of transients are filtered out from the final results.

B. Analysis validation

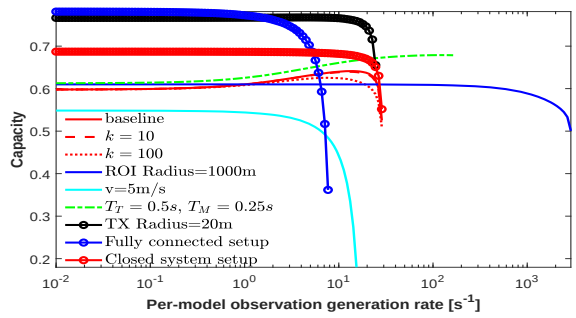
In a first set of experiments, we assessed the accuracy of our analytical model by comparing analytical and simulation results for the mean model availability and the mean number of observations stored by each node, versus model size.

Fig. 2 refers to an FG system with just one node class, while in Fig. 3 we consider an FG system with two node classes. In the latter figure, static nodes are 5% of the total, and in both figures, the speed of moving nodes can be either 0.5 or 8 m/s, and the number of models can be either 1 or 3.

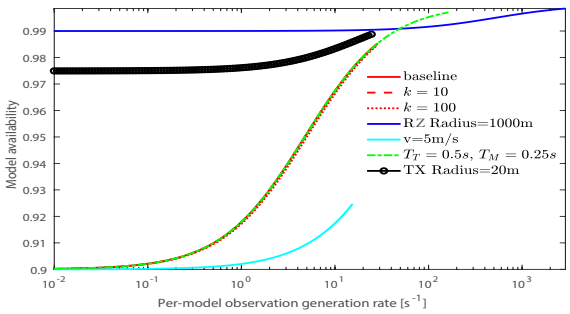
As Fig. 2 and 3 show, the estimates derived with our mean-field approach well match the simulation results across different values of node speed, model size, number of models, number of node classes, and training and merging times, despite the numerous approximations introduced in the analysis. We also see that when model size gets close to the maximum that can be sustained by the system (due to limitations in contact duration and in channel throughput), our analytical results yield slightly optimistic results, as the mean-field approach is less accurate when the population is sparse.

C. One class of nodes

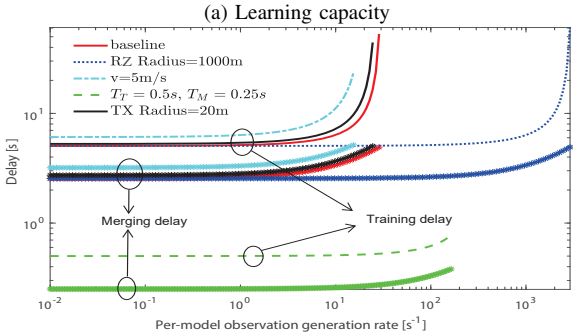
In a second set of experiments, we aimed at a first characterization of the main performance patterns of the FG system in a single-class scenario. Fig. 4a shows the FG learning capacity, derived from the solution of Problem 1, as a function of per-model observation generation rate λ. Fig. 4b instead shows how model availability at capacity evolves as a function of λ. From the plots on learning capacity, we can see that its value is closer to one (i.e. to its maximum theoretical value) when the time taken by an observation to spread among the node population in the RZ is small compared to the observation lifetime, and when contact frequency among nodes is high enough for the observation to spread rapidly to a large fraction of the nodes in the RZ. Thus, as visible in the plots, at low



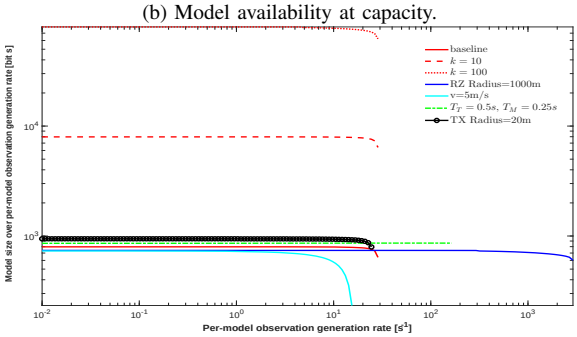
(a) Learning capacity



(b) Model availability at capacity.



(c) Mean training and merging delay at capacity



(d) Ratio of model size at capacity over learning load.

Fig. 4: Learning capacity, mean model availability, mean training and merging delay, and model size at capacity in an FG system versus per-model observation generation rate, as a function of system parameters. Baseline configuration: Tx radius 5 m; RZ Radius 100 m; Training time $T_T = 5$ s, Merging time $T_M = 2.5$ s; Node speed $v = 0.5$ m/s; Memory efficiency $k = 1$.

learning loads (low values of λ), capacity is larger when the contact rate among nodes is increased, e.g., when increasing the transmission range or the speed of nodes. Indeed, this also increases the model availability at low loads. Capacity at low loads is also increased when decreasing training and merging times (green curve in Fig. 4c), as faster training and merging accelerate the overall process of diffusion of observations.

As the plots show, the smaller is model availability at low λ, the more learning capacity (and thus the efficiency with which the FG system can incorporate information) increases when the observation arrival rate (and thus the amount of information that needs to be captured) grows. This feature, which we call *recruiting effect*, is present because a higher λ leads to a larger probability that, by arriving at a node with only a default version of the model, observations turn it into a non-default model. This effectively increases model availability (as visible in Fig. 4b), thus facilitating the spreading of observations. Note that this effect is less pronounced when nodes have a small sojourn time in the RZ (light blue curve in Fig. 4b with 5 m/s node speed), and thus have little time to be "recruited", or when model size is such that the time available for exchanges among nodes becomes the main performance bottleneck of the system (red dotted curve in Fig. 4b, with $k = 100$).

These experiments also show that in all configurations, there are values of learning load beyond which the system is no longer stable, i.e., it can no longer sustain the demand for computing (due to training and merging) that the arrival rate of observations induces. When this happens, a fraction of the training and merging tasks at each node are never performed. This is also evident from Fig. 4c, which shows the training/merging delay as a function of the learning load. As expected, since merging tasks have priority over training tasks, the queues that become unstable are those for training tasks. Note that faster node speeds lead to a higher queue buildup for training and merging tasks, and thus to an earlier onset of instability. The plots in Fig. 4c also show that decreasing training and merging times by a factor of ten relative to their default values increases the maximum value of the observation arrival rate beyond which the system is unstable. Not surprisingly, decreasing those computing times also increases learning capacity, since it accelerates the diffusion of models and observations. In addition, these results indicate that the number of nodes in the RZ plays a crucial role in determining the maximum learning load that the system can support. Specifically, the plots show that, keeping node density and speed constant, a larger RZ area (100 times larger than the default, as indicated by the dark blue curve in Fig. 4c) does not lead to a significant increase in capacity at low loads. This means that the number of nodes in the system does not affect the effectiveness with which the information is incorporated into models. However, it increases (by roughly the same factor) the maximum learning load at which the system is stable. This means that larger systems have a larger service capacity, as they can process a larger amount of information per unit time than smaller systems. Indeed, the arrival rate of training tasks at a node is proportional to the ratio between the learning load and the number of nodes in the RZ.

To better understand the interplay between node churn, communication constraints, and capacity, we examined two boundary cases that relax specific assumptions of the general FG model, illuminating key performance drivers. In a *closed system setup*, there is no node churn in the RZ. This eliminates

a source of random fluctuation in the population of nodes with a model. Consequently, model availability equals one and observations achieve full dissemination. As expected, this amounts to a higher learning capacity of the system even at low learning loads, as a closed system does not lose the observations it incorporates in models by having nodes leave the RZ. However, these results show that the closed system has the same maximum learning load as open systems, as the lack of fluctuations does not impact significantly the overall computing load for nodes in the RZ. Bottom line, a closed system is more efficient, as it does not need to compensate for the consequences of node churn on the learning process.

Complementing this, a *fully connected setup* maintains node churn but assumes all-to-all connectivity within the RZ. As a consequence, model availability is one (or very close to it, as model transfer takes time). This scenario isolates the impact of opportunistic communications on system performance. By eliminating store-carry-and-forward delays and locality constraints on model diffusion, this case quantifies the performance gap attributable to D2D opportunistic exchanges alone. Indeed, model (and observation) diffusion are very fast compared to the case in which opportunistic communications are used. Still, at low learning loads, capacity is around 0.8, which is greater than 0.6 achieved with a tx radius of 5 m, but still significantly less than one. This highlights the contribution to model diffusion of the training and merging delays (and of the fact that exchanges are still in pairs). Moreover, the higher rates at which models are exchanged increases the computing load of nodes, and thus the maximum learning load which the system can support. Thus, this limit case highlights a key tradeoff in the FG system: when communications are not a bottleneck, a higher capacity is achieved, but at the expense of a lower maximum learning load.

In Fig. 4d we plot the ratio between the model size at capacity over the learning load λ, as a function of learning load. As the plot shows, as the learning load increases, capacity is achieved by increasing proportionally the size of models, by a factor that does not vary while varying learning load (except when the system is close to saturation, as it may be expected). That is, when learning load increases, to achieve capacity, the system adjusts by increasing proportionally the amount of observations that can be stored in each model. The constant of proportionality is essentially a function of memory efficiency k. Indeed, the plot shows that an increase of one order of magnitude in k brings to an increase of the constant of proportionality of the same amount.

To characterize the stability region of the system, in Fig. 5 we plot the range of values of load and of number of models for which the stability condition (1) holds, i.e, for which the FG system possesses enough service capacity to incorporate new observations in a timely manner. In the figure, the color scale corresponds to the value of the argument of (1). Hence, the yellow area above the isotonic line at value 1 represents unstable points. The same line also illustrates the tradeoff between the number of models and the observation generation rate. In this specific case, the system is stable with at most

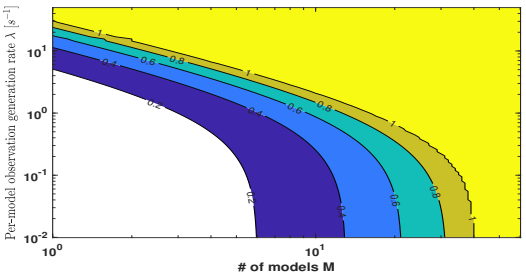


Fig. 5: Stability region boundary (from equation (1)) versus number of models M, for different values of per-model observation generation rate λ, for a model size $L = 10$ kb.

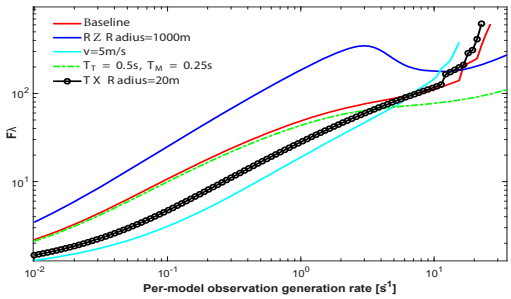


Fig. 6: Ratio of model staleness F over per-model mean observation interarrival time $1/\lambda$, versus per-model observation generation rate λ. Baseline configuration: Tx radius 5 m; RZ Radius 100 m; Training time $T_T = 5$ s, Merging time $T_M = 2.5$ s; Node speed $v = 0.5$ m/s; Memory efficiency $k = 1$, model size $L = 10$ kb.

about 40 models with a per-model observation generation rate of one every 100 seconds, but also with just one model whose observations can be generated as frequently as about 25 times per second, which is a rate suitable to capture a video. From Fig. 5, it emerges that for values of per-model observation generation rate which are small with respect to observation lifetime, the queue of merging tasks is essentially determined by the rate at which nodes come in contact.

When the observation generation rate increases, the system performance is progressively more dependent on the maximum computing load that each node can support. Clearly, in this regime, an increase in the number of models entails not only an increase in the overall observation arrival rate but also in the training load. Thus, the higher the λ, the lower the maximum number of models the system can support. This is coherent with the results for training and merging delay (Fig. 4c) that show that instability is driven by the queue for training tasks. Note that discontinuities in the plot are due to quantization in the number of models.

Another key indicator of the performance of an FG system is mean model staleness, which is related to the speed at which observations arriving in the system are incorporated into a model. As it is expected that model staleness decreases with increasing learning load, in Fig. 6 we plotted the ratio between model staleness and mean observation interarrival time, as a function of per-model observation generation rate. As the plot shows, the two quantities (respectively, staleness and mean observation interarrival time) are not directly proportional. Rather, model staleness decreases more slowly than

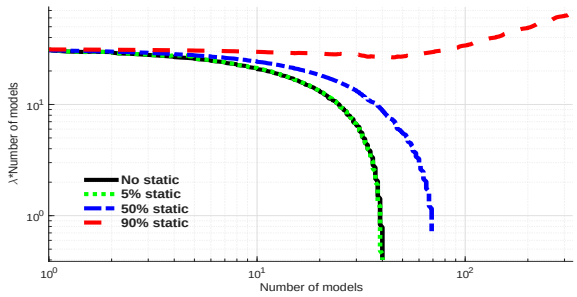


Fig. 7: Maximum total learning load for which the FG system is stable (condition (1)) as a function of the percentage of static nodes, versus number of models M, in the baseline configuration. Model size is 10 Kb.

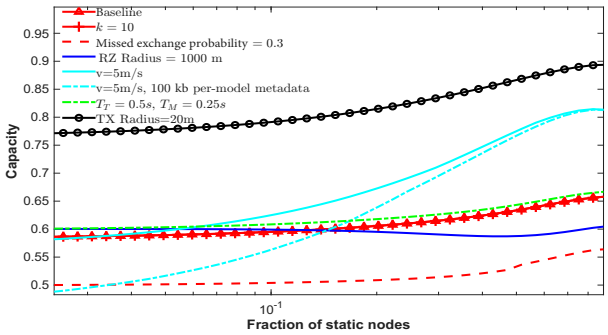


Fig. 8: Capacity vs fraction of static nodes, for different system configurations, for a model size $L = 10$ kb, and a per-model observation generation rate $\lambda = 10\ s^{-1}$.

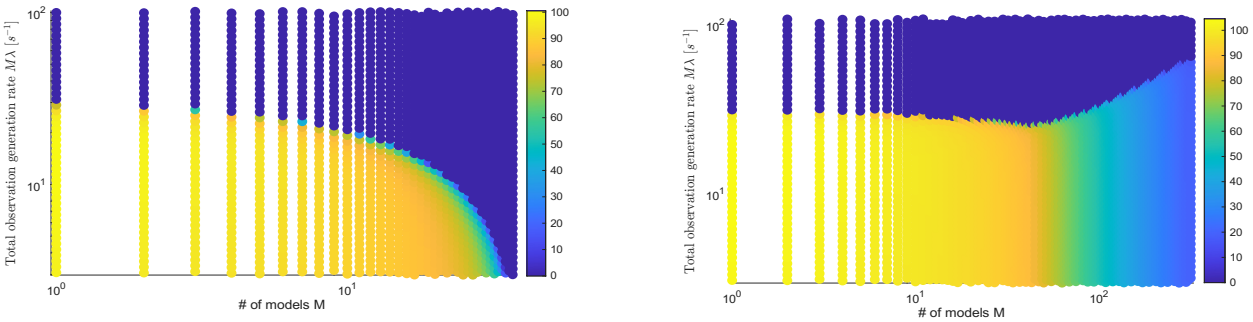
model interarrival time. This is because, by increasing the learning load, the time taken to process and incorporate a new observation increases, due to longer training and merging delays, which also affect the time taken by each observation to spread. As expected, when the size of the RZ is larger, model staleness is larger too, as the curve for a RZ radius of 1000 m suggests. In all configurations, we also observe that the slope of the staleness-interarrival ratio varies with λ, with this effect being more pronounced for the configuration with RZ radius equal to 1000 m. Such a behavior is because, at each node, the mean model staleness is the minimum between two components. The first "exogenous" component is the minimum age of observations derived from other nodes (and incorporated through merging). The second component is "endogenous", i.e., due to observations arriving at the node (and incorporated through training). Thus, the varying slope is due to the interplay between these two components, whose value decreases with λ at different speeds. In particular, if N denotes the mean number of users in the RZ, the mean age of the most recent observation incorporated by training at a node is N/λ. Instead, the exogenous component depends on how effectively model instances spread. For large values of RZ radius, the exogenous component is very large, due to the longer time taken by the observation to diffuse. Thus, the peak happens because the endogenous component becomes dominant and drastically lower than the exogenous one, at least for a given interval of values of λ. Then, by further increasing λ, the exogenous component becomes again smaller than the endogenous component.

D. Two classes of nodes

To further the characterization of the main performance patterns of FG, in another set of experiments, we considered a scenario with two different classes of nodes, each characterized by a different mobility pattern. Specifically, we investigated the case in which a fraction of the nodes in the system is static, and deployed uniformly at random within the RZ. This configuration models settings in which static devices play the role of dedicated internal infrastructure, aiming at integrating the service capacity of the system in a way that does not depend on mobility patterns (as is the case with moving nodes) and which is thus expected to be more reliable.

In Fig. 3 we have plotted the mean model availability and mean node stored information for both static and moving nodes, in three different configurations, for a setting where 5% of the nodes are static. These results show that our analysis achieves good accuracy in a variety of scenarios and configurations. As the plots show, while the parameters of moving nodes (in terms of model availability and node stored information) are substantially unaltered by the presence of static nodes, those of static nodes exhibit a model availability equal to one, and a node stored information generally larger than that of moving nodes. These properties hold because there are no "clean-slate" static nodes (i.e., nodes with only the default model) in the system, except (possibly) at FG bootstrap time. Instead, due to churn in the population of moving nodes inside the RZ, at any time instant, a fraction of the moving node population in the RZ is constituted by nodes that just entered, and thus possess only the default version of each model.

Another key impact that the presence of static nodes has on an FG system is relative to the stability region of the system. Fig. 7 shows the *maximum total learning load*, i.e., the values of the product of the per-model observation arrival rate and the number of models trained, beyond which the FG system is unstable. These plots show that in all configurations, an increase in the number of models leads to a decrease in the maximum total learning load. This is expected, as splitting the same learning load among many models increases proportionally both the number of model instances exchanged at each contact and the number of merging tasks at each node. These results also show that when there is a single model in the system, the maximum total learning load is not impacted by the fraction of static nodes. However, when increasing the number of models, the effect of the increase in the mean amount of merging tasks at each node becomes progressively more evident, and it is different according to the fraction of static nodes in the system. Specifically, a higher fraction of static nodes makes the system able to support a larger amount of total learning load, because of the lower contact rate that static nodes induce, thus lowering the overall arrival rate of merging tasks at static nodes. The configurations with a fraction of static nodes lower than 50% have a maximum number of models they can support, determined by the load on the queue of merging tasks. However, when the fraction



(a) Only moving nodes.

(b) 90% static nodes.

Fig. 9: Ratio of the per-model stored information versus the mean amount of information which can be incorporated by training in a model instance, as a function of the total learning load and of the number of models, for the case with only moving nodes and with 90% static nodes, for a model size $L = 10$ kb.

of static nodes is very high, at high values of the number of models, the system increases its maximum training load when the number of models increases. This happens because the queue for merging tasks hits an upper bound due to the duration of contacts among nodes, which limits the number of instances exchanged and thus merged. For high values of the number of models and of the fraction of static nodes, models are not exchanged effectively anymore, the few models received contain no new observations, and the computing load at each node is almost exclusively due to training tasks. This is visible in Fig. 9, which shows, for both the setup with no static nodes and that with 90% static nodes, the ratio between the mean amount of observations stored in a model instance and the mean amount of observations which can be incorporated by training at that model instance. This latter quantity is the mean amount of information that is incorporated in a model instance when nodes train it in isolation. Thus, the figure shows that despite increasing the fraction of static nodes expands the stability region of the FG system by allowing the training in parallel of a larger number of models, when the number of models is large, the FG cooperative training approach gradually worsens its performance. That is, despite in these setups the system is stable and somehow managing to incorporate almost every arriving observation, every observation is incorporated in at most a single model instance. As model instances get exchanged less and less effectively, the more models are in the system, when the fraction of static nodes takes high values, the system progressively shifts towards a regime in which each node trains its model based mainly on its data, in isolation from other nodes. As Fig. 8 suggests, in addition to expanding the stability region of an FG system, the presence of static nodes in the RZ increases the system's capacity in all of the considered setups. This is a direct consequence of the fact that static nodes have on average longer contact durations, and model availability equal to one. As both these factors, as seen in Fig. 3, bring to a larger node stored information, they also contribute to a larger capacity. The marginal increase in capacity is larger for scenarios with larger node speed, as static nodes mitigate the volatility in the population of nodes with models within the RZ due to mobility. The marginal increase is also significant for the configuration with a 20 m transmission radius, as static nodes tend to form stable node clusters, enhancing the system's ability to incorporate and share observations effectively.

The light blue broken line reports the impact of a very large overhead for the exchange of metadata (100 kb of metadata for models whose size is 10 kb). This curve proves that even with such an extreme amount of overhead, the loss in performance is not very large. With small fractions of static nodes, capacity decreases by about 20%, and with large fractions of static nodes (above 10%) the performance loss is below 10%. To evaluate the impact of the use of lossy data structures (e.g., hashes, Bloom filters) to encode the metadata, we have run a set of experiments assuming a probability of missed exchanges due to these losses. The red dashed line in Fig. 8 shows that by assuming a very high missed exchange probability (equal to 0.3), the capacity loss does not grow larger than about 20%, for all fractions of static nodes. This is an indication of the fact that the impact of lossy data structures and overhead in data transmission is low, and it can be neglected when metadata overhead is not larger than model size, and loss probabilities are of the order of a few percentage points.

Finally, we have analyzed the impact on staleness of the percentage of static nodes, for the same settings considered in Fig. 8. In all cases, increasing the fraction of static nodes from 10^{-2} to 0.9 brings to a steady but slight decrease in model staleness. This suggests that the presence of a population of static nodes does not accelerate significantly the process of diffusion of trained models. These results highlight key trade-offs: higher node speeds accelerate model diffusion (reducing staleness) but increase training/merging delays and reduce stability margins due to more frequent churn. Similarly, larger RZ sizes scale capacity proportionally to node count but worsen staleness as diffusion times grow. Static nodes expand stability regions and capacity (especially at a high number of models) yet yield diminishing returns on staleness reduction (about 5% improvement).

VII. Conclusions

In this paper, we propose an analytical framework to characterize the limit performance of Floating Gossip, a fully distributed learning scheme for the collaborative continual training of ML models stored by mobile users. Our study

shows that Floating Gossip can be very effective in implementing opportunistic, fully distributed, privacy-preserving, training, and updating of ML models in a cooperative manner. Unlike centralized training schemes, whose performance is heavily degraded by node churn and mobility, gossip-based schemes have proven to be robust against those factors and to depend heavily on nodes' ability to harvest data about the environment in which they operate. We have investigated the impact of infrastructure support to FG in the form of static nodes, and we have shown that they help improve performance both in terms of the number of data points incorporated in the trained models and in the maximum total number of models that an FG system can train in parallel.

ACKNOWLEDGMENTS

This work was partially supported by COST INTERACT, SNF Dymonet project. It was also supported by the SNS JU UNITY-6G project, (101192650), and the State Secretariat for Education, Research and Innovation (SERI), Switzerland, and by the TUCAN6-CM (TEC-2024/COM-460), funded by CM (the Region of Madrid), Spain (ORDEN 5696/2024).

REFERENCES

[1] M. Woolley, "Bluetooth Core Specification Version 5.2 Feature Overview," *Bluetooth SIG: Kirkland, WA, USA*, 2020.

[2] IEEE Computer Society LAN MAN Standards Committee and others, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *ANSI/IEEE Std. 802.11-1999*, 1999.

[3] R. Molina-Masegosa and J. Gozalvez, "LTE-V for sidelink 5G V2X vehicular communications: A new 5G technology for short-range vehicle-to-everything communications," *IEEE Vehicular Technology Magazine*, vol. 12, no. 4, pp. 30–39, 2017.

[4] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A tutorial on 5g nr v2x communications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1972–2026, 2021.

[5] R. I. Ansari, C. Chrysostomou, S. A. Hassan, M. Guizani, S. Mumtaz, J. Rodriguez, and J. J. Rodrigues, "5G D2D networks: Techniques, challenges, and future prospects," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3970–3984, 2017.

[6] "Gigwalk: We've got your brand's back - Gigwalk," https://www.gigwalk.com/, (Accessed on 07/31/2022).

[7] "Driving directions, live traffic & road conditions updates - Waze," https://www.waze.com/en/live-map/, (Accessed on 07/31/2022).

[8] "Millionagents - marketing research for business on the basis of modern technologies," https://www.millionagents.com/en/, (Accessed on 07/31/2022).

[9] D. Borsetti, M. Fiore, C. Casetti, and C. F. Chiasserini, "When mobile services go local," in *MSWiM '09*. ACM, 2009, p. 235–244.

[10] S. Ali, G. Rizzo, V. Mancuso, V. Cozzolino, and M. A. Marsan, "Experimenting with floating content in an office setting," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 49–54, 2014.

[11] M. Chen, H. V. Poor, and W. Saad, "Federated learning for on-device ai: A survey of privacy and efficiency trade-offs," *Proceedings of the IEEE*, vol. 111, no. 4, pp. 512–530, 2023.

[12] F. Liu, Y. Zhang, and L. Wang, "Edge intelligence: Challenges and opportunities in deploying machine learning on resource-constrained devices," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 987–1001, 2024.

[13] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 4, pp. 556–571, 2013.

[14] J. Orozco, I. Hegedüs, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *DAIS*, ser. Lecture Notes in Computer Science, vol. 11534. Springer, 2019, pp. 68–83.

[15] L. Wulfert, N. Asadi, W.-Y. Chung, C. Wiede, and A. Grabmaier, "Adaptive decentralized federated gossip learning for resource-constrained iot devices," in *IPSN*, 2023.

[16] E. Hyytia, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott, "When does content float? Characterizing availability of anchored information in opportunistic content sharing," *IEEE INFOCOM*, April 2011.

[17] S. Ali, G. Rizzo, B. Rengarajan, and M. A. Marsan, "A simple approximate analysis of floating content for context-aware applications," *technical report, TR-IMDEA-Networks-2013-1*, 2013.

[18] G. Manzo, M. Ajmone Marsan, and G. Rizzo, "Performance Modeling of Vehicular Floating Content in Urban Settings," in *ITC 29*, vol. 1. IEEE, 2017, pp. 99–107.

[19] S. Lee, X. Zheng, J. Hua, H. Vikalo, and C. Julien, "Opportunistic Federated Learning: An Exploration of Egocentric Collaboration for Pervasive Computing Applications," in *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2021, pp. 1–8.

[20] L. Giaretta and S. Girdzijauskas, "Gossip Learning: Off the Beaten Path," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1117–1124.

[21] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 109–124, 2021.

[22] J. Daily, A. Vishnu, C. Siegel, T. Warfel, and V. Amatya, "GossipGraD: Scalable Deep Learning using Gossip Communication based Asynchronous Gradient Descent," *CoRR*, vol. abs/1803.05880, 2018. [Online]. Available: http://arxiv.org/abs/1803.05880

[23] M. Blot, D. Picard, N. Thome, and M. Cord, "Distributed optimization for deep learning with gossip exchange," *Neurocomputing*, vol. 330, pp. 287–296, 2019.

[24] Y. Chen, K. Yuan, Y. Zhang, P. Pan, Y. Xu, and W. Yin, "Accelerating Gossip SGD with Periodic Global Averaging," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 1791–1802.

[25] J. Guo, Y. Zuo, C.-K. Wen, and S. Jin, "User-Centric Online Gossip Training for Autoencoder-Based CSI Feedback," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 559–572, 2022.

[26] M. A. Dinani, A. Holzer, H. Nguyen, M. A. Marsan, and G. Rizzo, "Gossip Learning of Personalized Models for Vehicle Trajectory Prediction," in *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2021, pp. 1–7.

[27] A. Chaintreau, J.-Y. Le Boudec, and N. Ristanovic, "The age of gossip: Spatial mean field regime," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 109–120.

[28] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning: Concepts, advances, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3615–3634, 2024.

[29] S. Ali, G. Rizzo, B. Rengarajan, and M. Ajmone Marsan, "A simple approximate analysis of floating content for context-aware applications," in *2013 Proceedings IEEE INFOCOM*, April 2013, pp. 21–22.

[30] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends in cognitive sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.

[31] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of queueing theory*. John wiley & sons, 2011, vol. 627.

[32] A. Kolesnichenko, V. Senni, A. Pourranjabar, and A. Remke, "Applying mean-field approximation to continuous time markov chains," in *ROCKS*, 2012.

[33] L. Bortolussi, J. Hillston, D. Latella, and M. Massink, "Continuous approximation of collective system behaviour: A tutorial," *Performance Evaluation*, vol. 70, no. 5, pp. 317–349, 2013.

[34] G. Friedland and M. Krell, "A capacity scaling law for artificial neural networks," *arXiv preprint arXiv:1708.06019*, 2017.

[35] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.

[36] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM computing surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.

[37] P. Nain, D. Towsley, B. Liu, and Z. Liu, "Properties of random direction models," in *IEEE ICC*, vol. 3. IEEE, 2005, pp. 1897–1907.