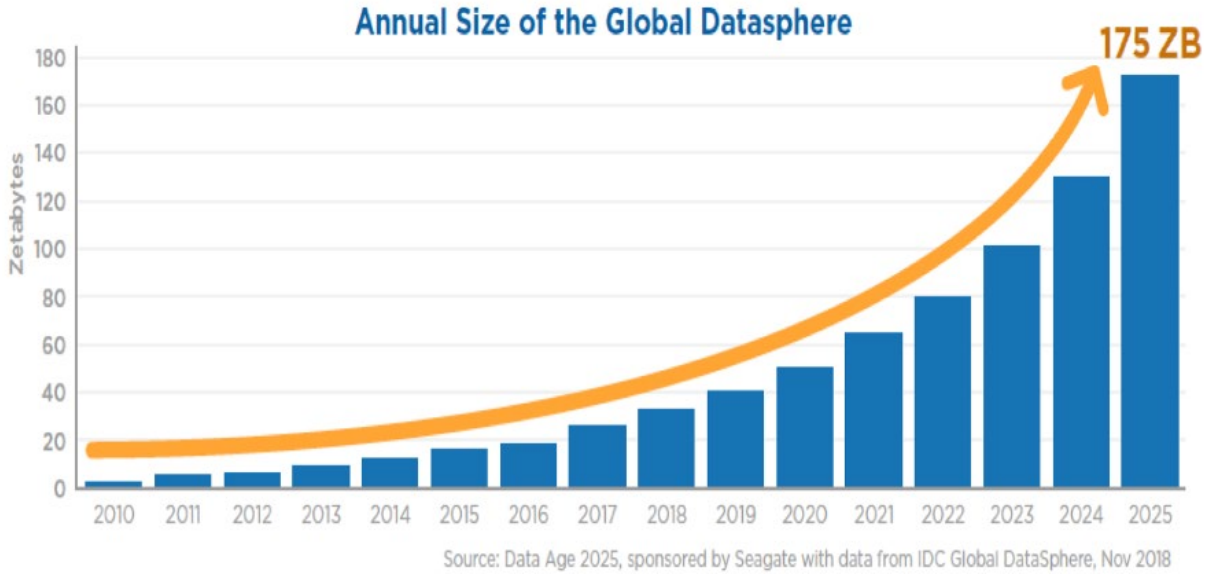
On the Limit Performance of Floating Gossip

Gianluca Rizzo, Noelia Perez Palma, Marco Ajmone Marsan, and Vincenzo Mancuso

IEEE INFOCOM 2023, NY

The push towards Edge Computing is fostering distributed learning schemes

- Data production is growing at a vertiginous pace



- Most of that data is produced at the edge of the network
- **Scalability: keep data storage and elaboration at the edge**

Distributed Learning

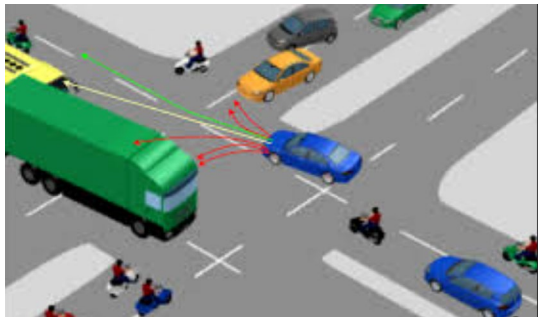
We focus on distributed continual learning

Learning is continual (in some way or another)

- Environment which keeps on changing
- Involves forgetting past, outdated info

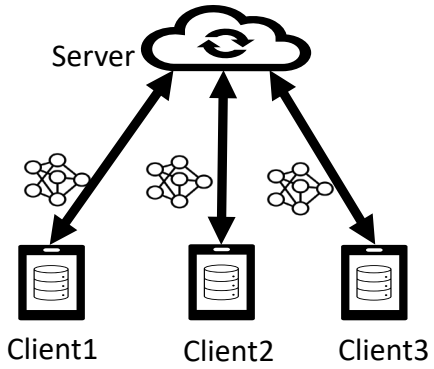
Example 1: Personalized, location-based recommendations, e.g. in amusement parks

Example 2: Vehicular trajectory prediction in urban scenarios



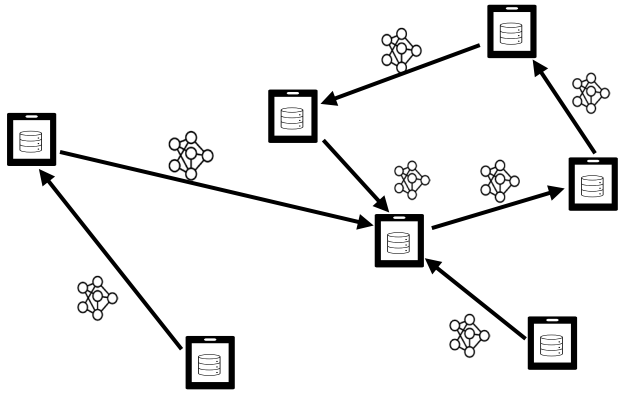
Current distributed learning schemes

Server-based architectures
(Federated Learning)



(single point of failure, poor scalability)

Server-less architectures
(Gossip Learning)



mainly on **static** settings
single connected component

What is Floating Gossip?

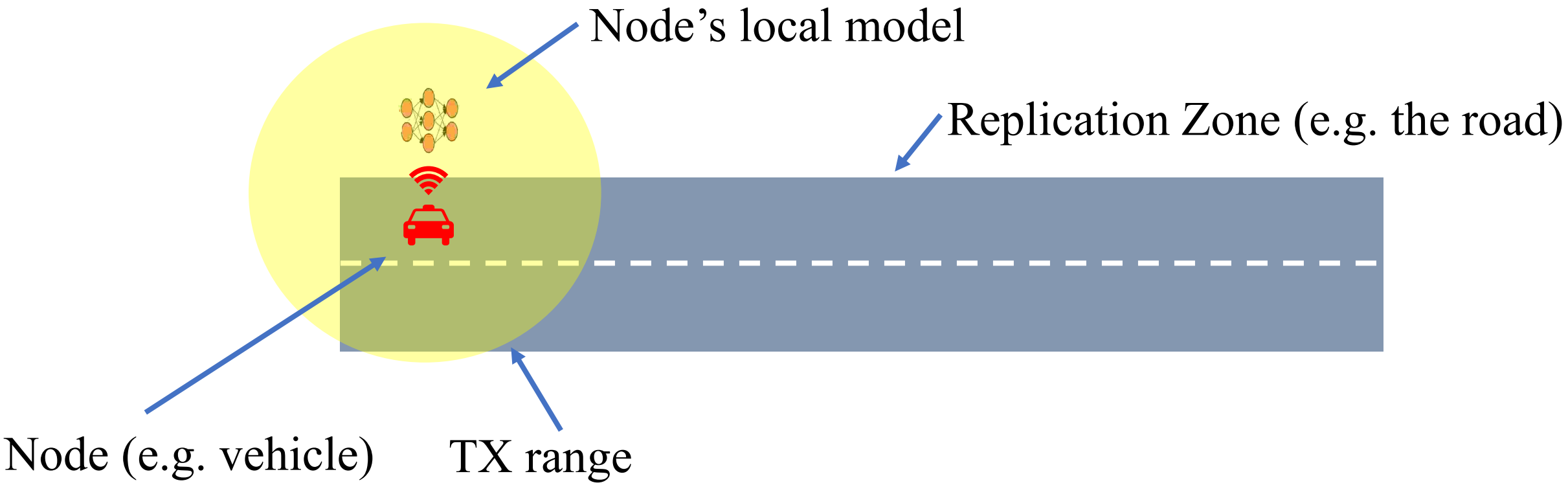
- A Gossip Learning scheme for dynamic scenarios
  - Tied to a specific region of space (Replication Zone, RZ)

- No infrastructure (in principle): Reliance on **gossip exchanges**
  - Data & model storage, and training are **probabilistic**

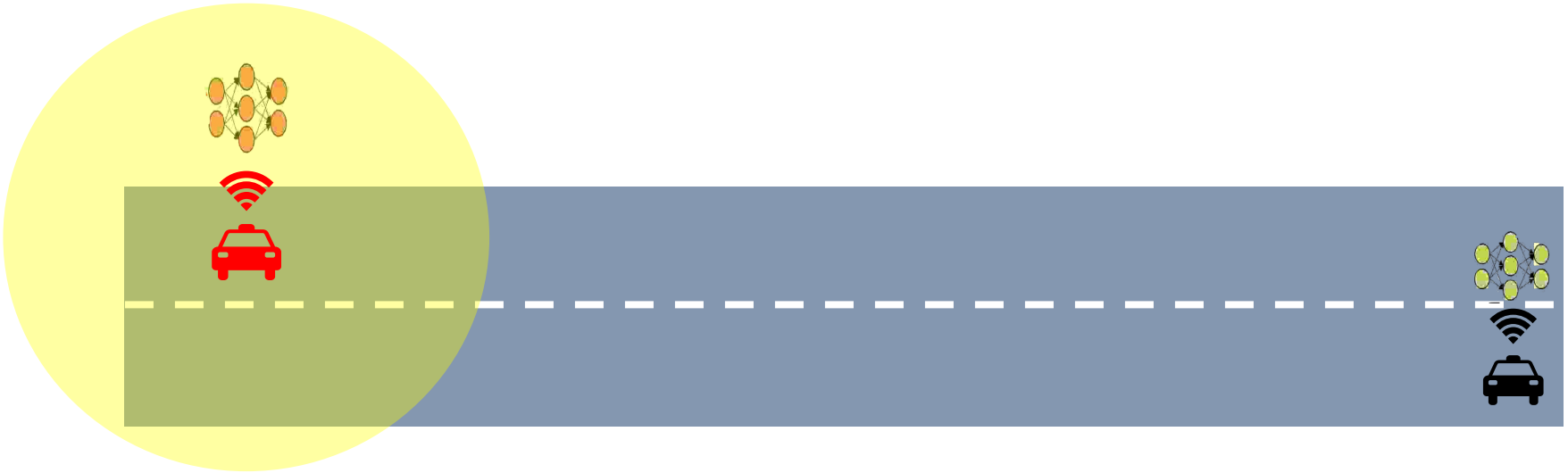
- **Goal**: Continual training of models over data from the RZ
  - Users are (typically) nodes themselves
  - Combination of static and moving nodes is ok

How does Floating Gossip work?

- When entering the RZ, each node has its own local (untrained) model
- As nodes move in the RZ, they collect data and incorporate it in their local model

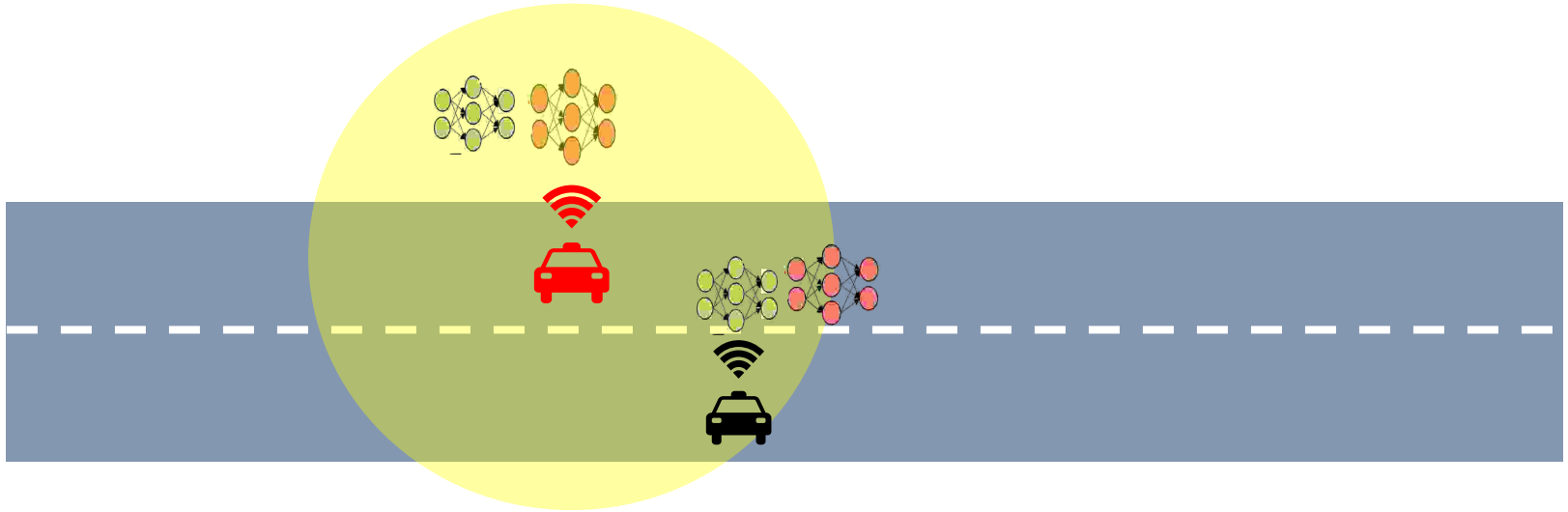


How does Floating Gossip work?

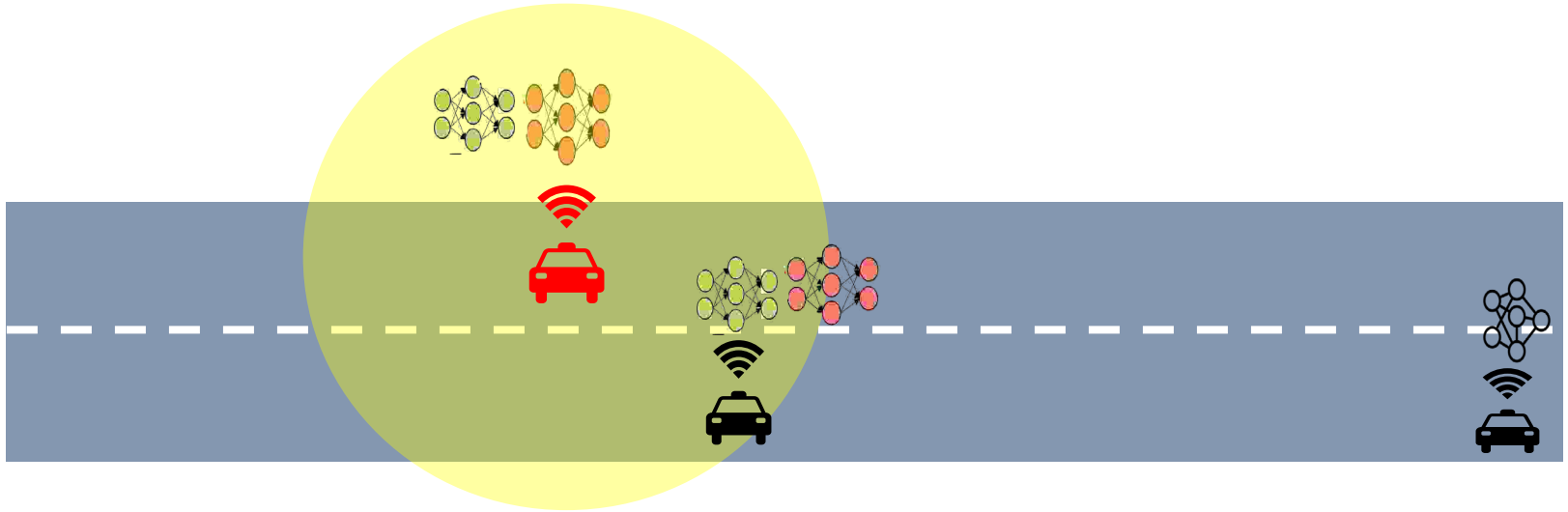


How does Floating Gossip work?

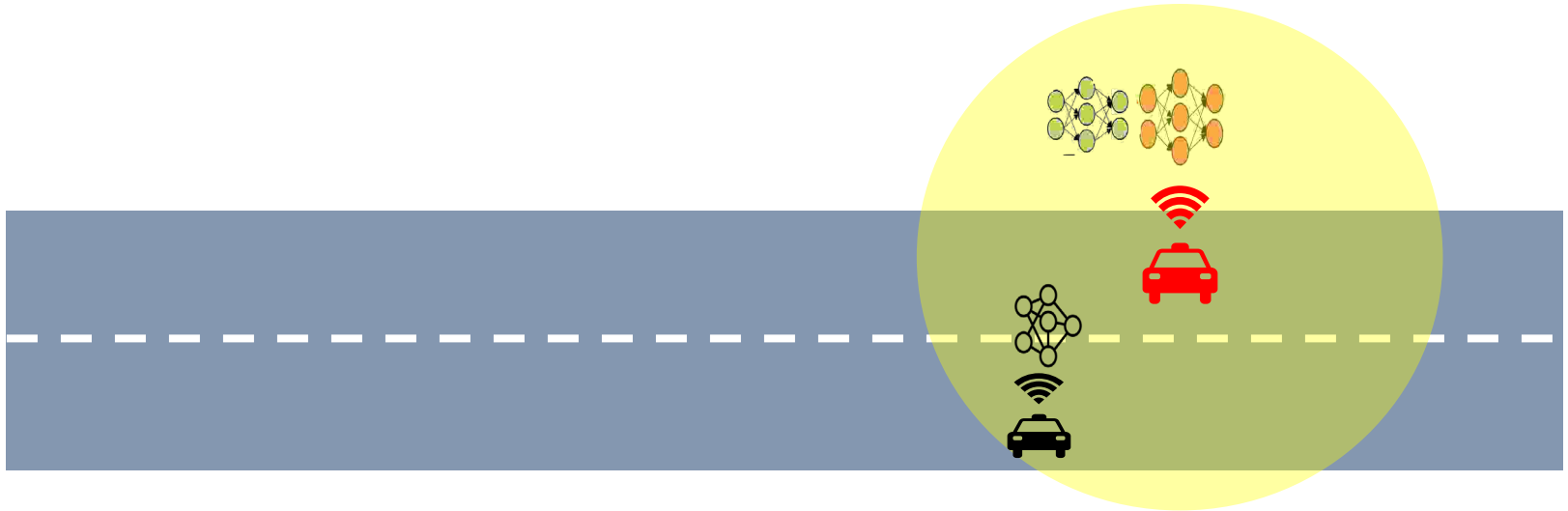
- At every contact **within the RZ**, nodes exchange opportunistically their local models
- Thus, models (and the information they incorporate) persist probabilistically in the RZ, even in presence of churn



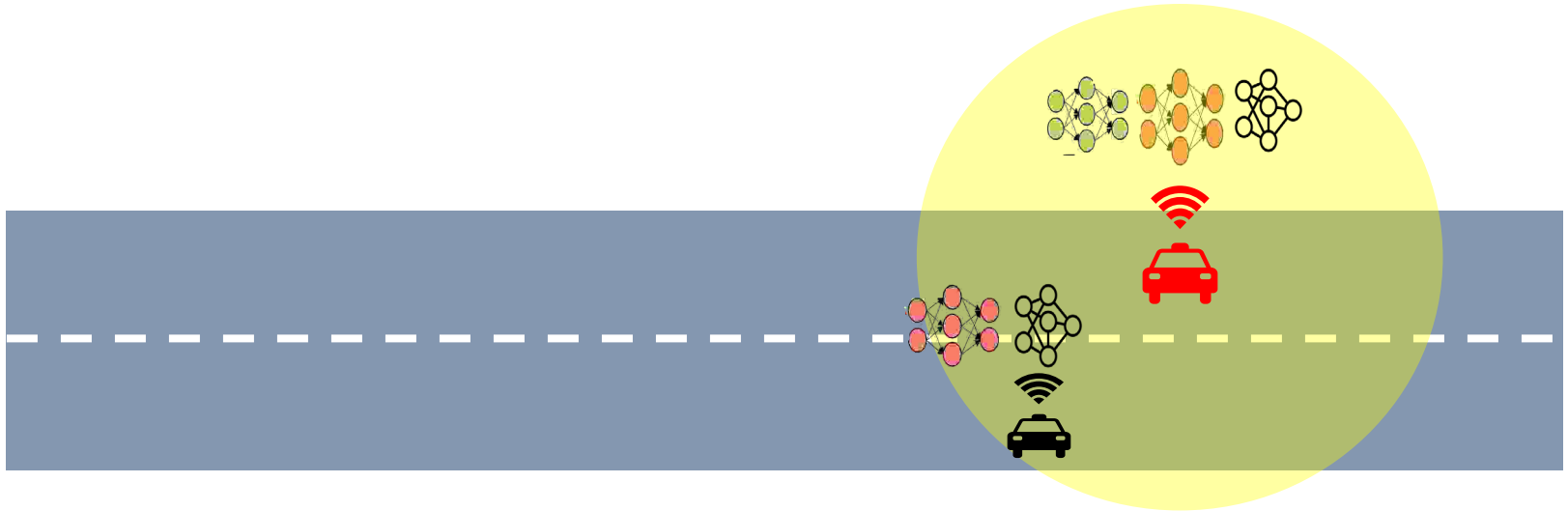
How does Floating Gossip work?



How does Floating Gossip work?

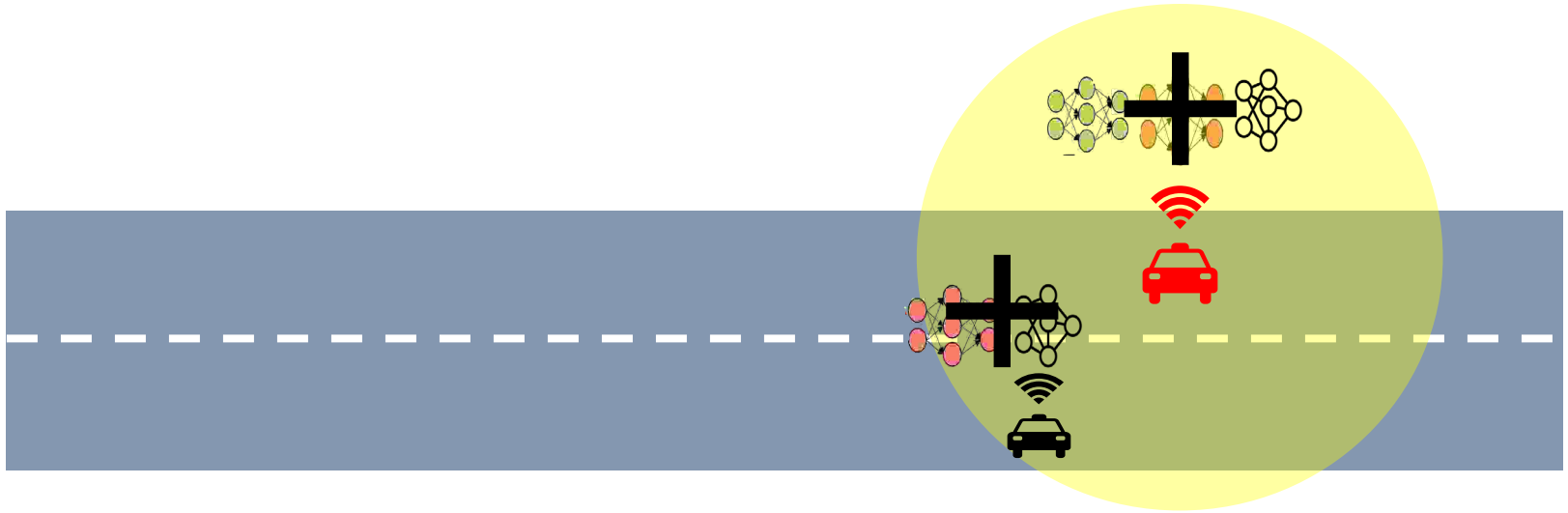


How does Floating Gossip work?



How does Floating Gossip work?

- Each node periodically **merges** its local model as well as all the received models
  - It creates a new local model (e.g. as an average of all models, as in FedAvg)
  - **Goal**: Improve performance of local model, via some form of knowledge transfer



When is Floating Gossip viable in dynamic scenarios?

- **Existing results**: Ad-hoc for specific learning architectures, task, and datasets

  ➡ **When is Floating Gossip viable?**

- Is it a **relevant** problem?

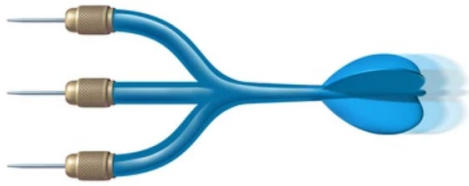
  - Edge is expanding towards user devices, and including moving devices

    ➡ **mobility matters!**

  - Floating Gossip is robust against churn and volatility

- **How does its performance compare to server-based approaches?**

Modeling FG performance: A three-pronged approach



- **Mean Field Theory**: Model ML model persistence and effects of mobility, and of communications efficiency
  - FC parameters: model availability
- **Classical Queuing Theory**: Model impact of computing load and communication dynamics on system stability and performance
- **Information theory**: Bounds on capacity of ML models
  - Maximum information which can be stored in a model, vs model size

System model

- Nodes moving on the plane
  - Homogeneous distribution in space
  - Same speed, communication technology
  - Perfect knowledge of position in space
  - **On entering the RZ**: each node possesses M **default models**

- Poisson arrivals of observations (data points) at nodes in the RZ
  - Intensity λ
  - Observations have a **finite** lifetime τ_l
  - They are incorporated in models via **local training**
  - They spread within the RZ via opportunistic model replication and **merging**

Model availability and observation availability

- **a:** Mean fraction of nodes who possess a model (**model availability**)
- **b**: Mean fraction of nodes who are busy exchanging models

$$\begin{cases} \dfrac{da}{dt} = \dfrac{b}{T_s(t)} a(1-a)S(t) + \lambda(1-a) - \alpha a \\ \dfrac{db}{dt} = 2g(1-b)^2 - \dfrac{b}{T_S(t)} - 2\alpha b \end{cases}$$

- DE obtained with mean field approach

Model availability and observation availability

- **a:** Mean fraction of nodes who possess a model (**model availability**)
- **b**: Mean fraction of nodes who are busy exchanging models

$$\begin{cases} \dfrac{da}{dt} = \dfrac{b}{T_s(t)} a(1-a)S(t) + \lambda(1-a) - \alpha a \\ \dfrac{db}{dt} = 2g(1-b)^2 - \dfrac{b}{T_S(t)} - 2\alpha b \end{cases}$$

New node contacts

Model availability and observation availability

- **a:** Mean fraction of nodes who possess a model (**model availability**)
- **b**: Mean fraction of nodes who are busy exchanging models

$$\begin{cases} \dfrac{da}{dt} = \dfrac{b}{T_s(t)} a(1-a)S(t) + \lambda(1-a) - \alpha a \\ \dfrac{db}{dt} = 2g(1-b)^2 - \dfrac{b}{T_S(t)} - 2\alpha b \end{cases}$$

New node contacts

End of model exchanges

Model availability and observation availability

- **a:** Mean fraction of nodes who possess a model (**model availability**)
- **b**: Mean fraction of nodes who are busy exchanging models

$$\begin{cases} \dfrac{da}{dt} = \dfrac{b}{T_s(t)} a(1-a)S(t) + \lambda(1-a) - \alpha a \\ \dfrac{db}{dt} = 2g(1-b)^2 - \dfrac{b}{T_S(t)} - 2\alpha b \end{cases}$$

New node contacts

End of model exchanges

Nodes moving out of the RZ

Model availability and observation availability

- **a:** Mean fraction of nodes who possess a model (**model availability**)
- **b**: Mean fraction of nodes who are busy exchanging models

$$\begin{cases} \dfrac{da}{dt} = \dfrac{b}{T_s(t)} a(1-a)S(t) + \lambda(1-a) - \alpha a \\ \dfrac{db}{dt} = 2g(1-b)^2 - \dfrac{b}{T_S(t)} - 2\alpha b \end{cases}$$

New node contacts

End of model exchanges

Nodes moving out of the RZ

Model availability and observation availability

- **a:** Mean fraction of nodes who possess a model (**model availability**)
- **b**: Mean fraction of nodes who are busy exchanging models

$$\begin{cases} \frac{da}{dt} = \frac{b}{T_s(t)} a(1-a)S(t) + \lambda(1-a) - \alpha a \\ \frac{db}{dt} = 2g(1-b)^2 - \frac{b}{T_S(t)} - 2\alpha b \end{cases}$$

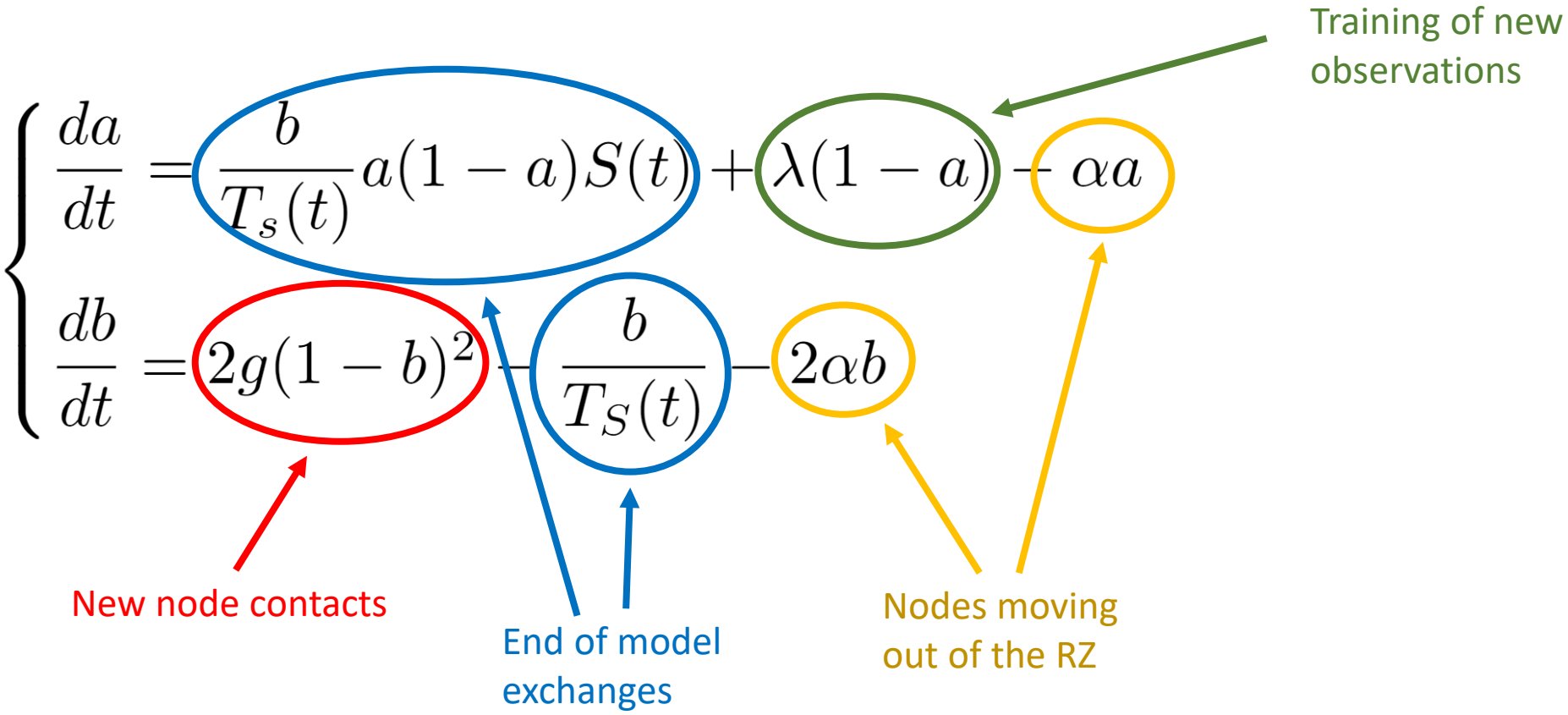
New node contacts

End of model exchanges

Nodes moving out of the RZ

Model availability and observation availability

- **a:** Mean fraction of nodes who possess a model (**model availability**)
- **b**: Mean fraction of nodes who are busy exchanging models



Model availability and observation availability

- **Observation availability o(t)**: Mean fraction of nodes with a model incorporating a given observation at time t

$$\frac{do(\tau)}{d\tau} = \frac{bS}{T_s}\left[(1-a)o(\tau) + ao(\tau - d_M)(1 - o(\tau - d_M))\right] - \alpha o(\tau)$$

$$o(\tau) = \begin{cases} 0 & \tau < d_I \\ \frac{1}{\lceil aN \rceil} & d_I \leq \tau \leq d_I + d_M \end{cases}$$

Model availability and observation availability

- **Observation availability o**: Mean fraction of nodes with a model incorporating a given observation

$$\frac{do(\tau)}{d\tau} = \frac{bS}{T_s}\left[(1-a)o(\tau) + ao(\tau - d_M)(1 - o(\tau - d_M))\right] - \alpha o(\tau)$$

$$o(\tau) = \begin{cases} 0 & \tau < d_I \\ \frac{1}{\lceil aN \rceil} & d_I \leq \tau \leq d_I + d_M \end{cases}$$

New node contacts

Model availability and observation availability

- **Observation availability o**: Mean fraction of nodes with a model incorporating a given observation

$$\frac{do(\tau)}{d\tau} = \frac{bS}{T_s}\left[(1-a)o(\tau) + ao(\tau - d_M)(1 - o(\tau - d_M))\right] - \alpha o(\tau)$$

$$o(\tau) = \begin{cases} 0 & \tau < d_I \\ \frac{1}{\lceil aN \rceil} & d_I \leq \tau \leq d_I + d_M \end{cases}$$

New node contacts

Model merging

Model availability and observation availability

- **Observation availability o**: Mean fraction of nodes with a model incorporating a given observation

$$\frac{do(\tau)}{d\tau} = \frac{bS}{T_s}\left[(1-a)o(\tau) + ao(\tau - d_M)(1 - o(\tau - d_M))\right] - \alpha o(\tau)$$

$$o(\tau) = \begin{cases} 0 & \tau < d_I \\ \frac{1}{\lceil aN \rceil} & d_I \leq \tau \leq d_I + d_M \end{cases}$$

New node contacts

Model merging

Nodes moving out of the RZ

The learning capacity measures Floating Gossip learning effectiveness

- **Floating Gossip Learning Capacity**: Maximum of the mean amount of information “stored” *in ML models*

$$\max_{M,L} \; a \min\left(\frac{L}{\lambda k}, \int_0^{\tau_l} o(\tau) d\tau\right)$$

The learning capacity measures Floating Gossip learning effectiveness

- **Floating Gossip Learning Capacity**: Maximum of the mean amount of information "stored" *in ML models*

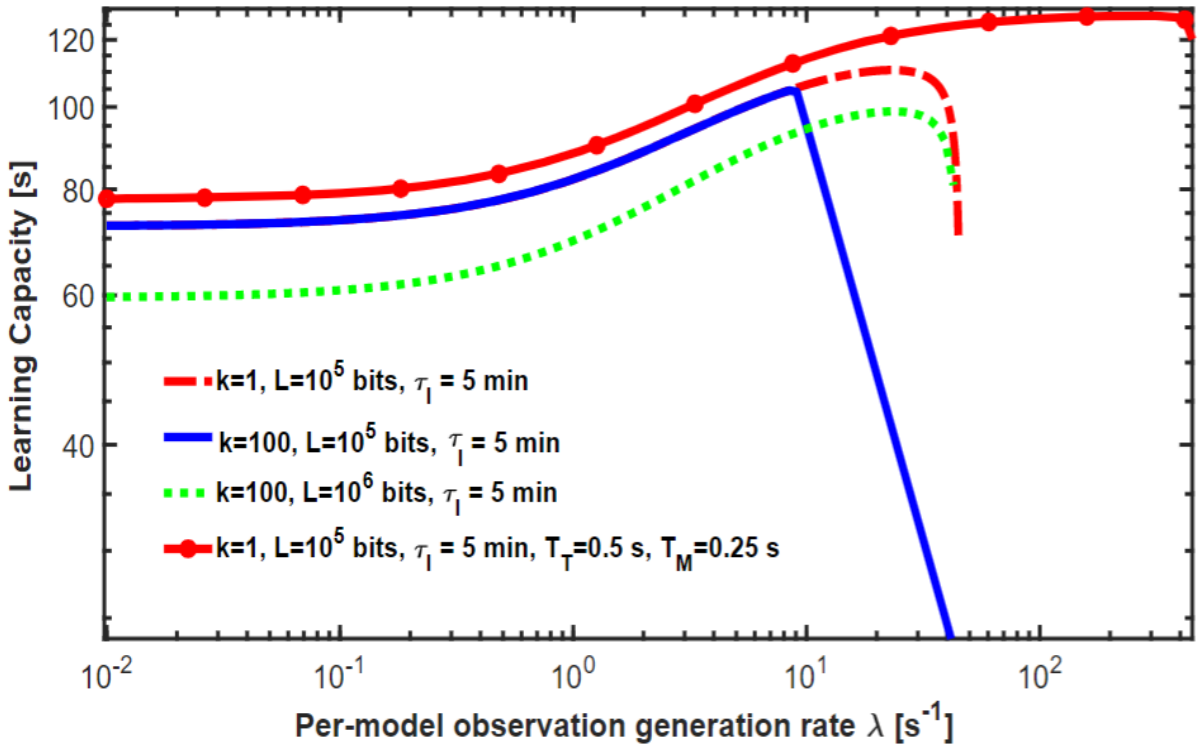
$$\max_{M,L} \; a \min\left(\frac{L}{\lambda k}, \int_0^{\tau_l} o(\tau) d\tau\right)$$

- ML model **capacity**: Amount of information it can "store"
  - Information theoretical upper bound
  - Key for accurate dimensioning, and for managing overfitting and forgetting

Numerical assessment

- Scenario and default values:
  - Random direction mobility model,
    - 0.5 m/s node speed
  - Circular RZ of radius 100 m
  - Training time: 5 s
  - Merging time: 2.5 s
- Assessment: both numerical and simulation (event-based)

FG thrives when learning load increases (but up to a point)



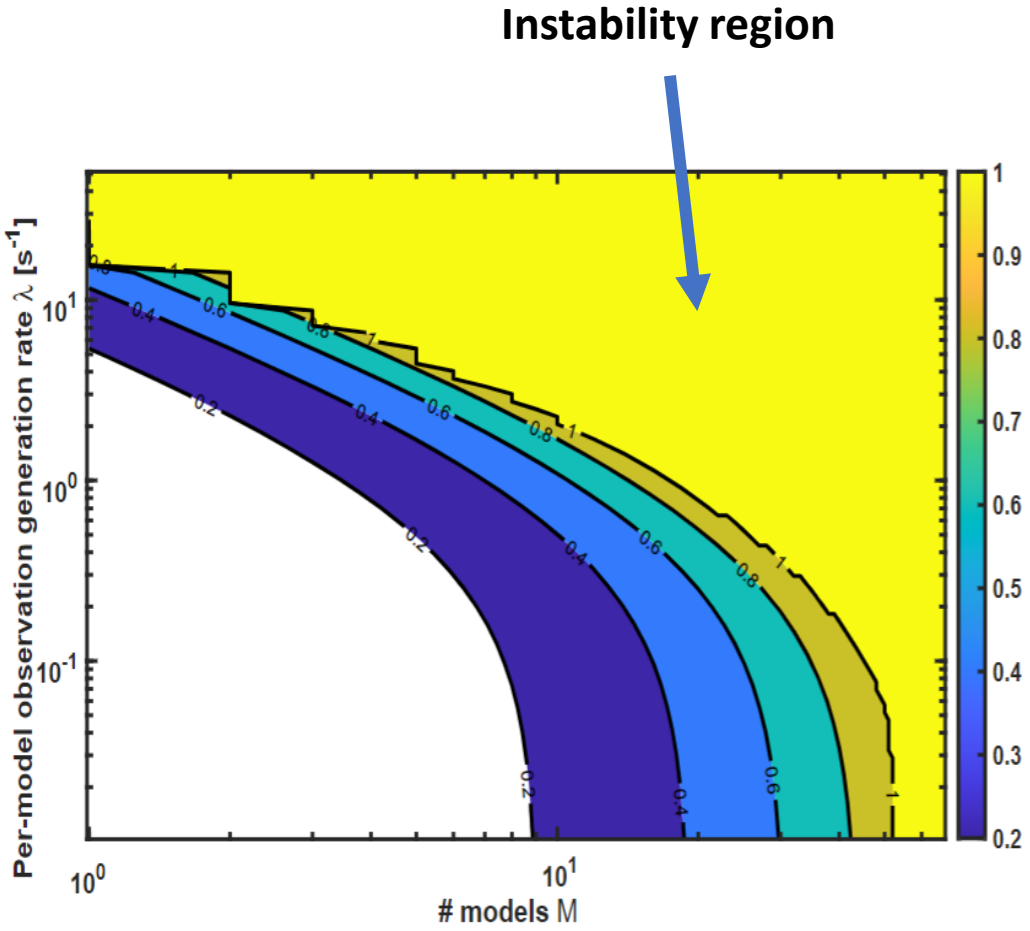
Increasing learning load has a **double effect:**

- It **increases learning capacity…**
  - Higher load means recruiting more nodes, expanding (probabilistically) service capacity
- …but it **brings the system closer to instability**

(system equivalent to catastrophic forgetting)

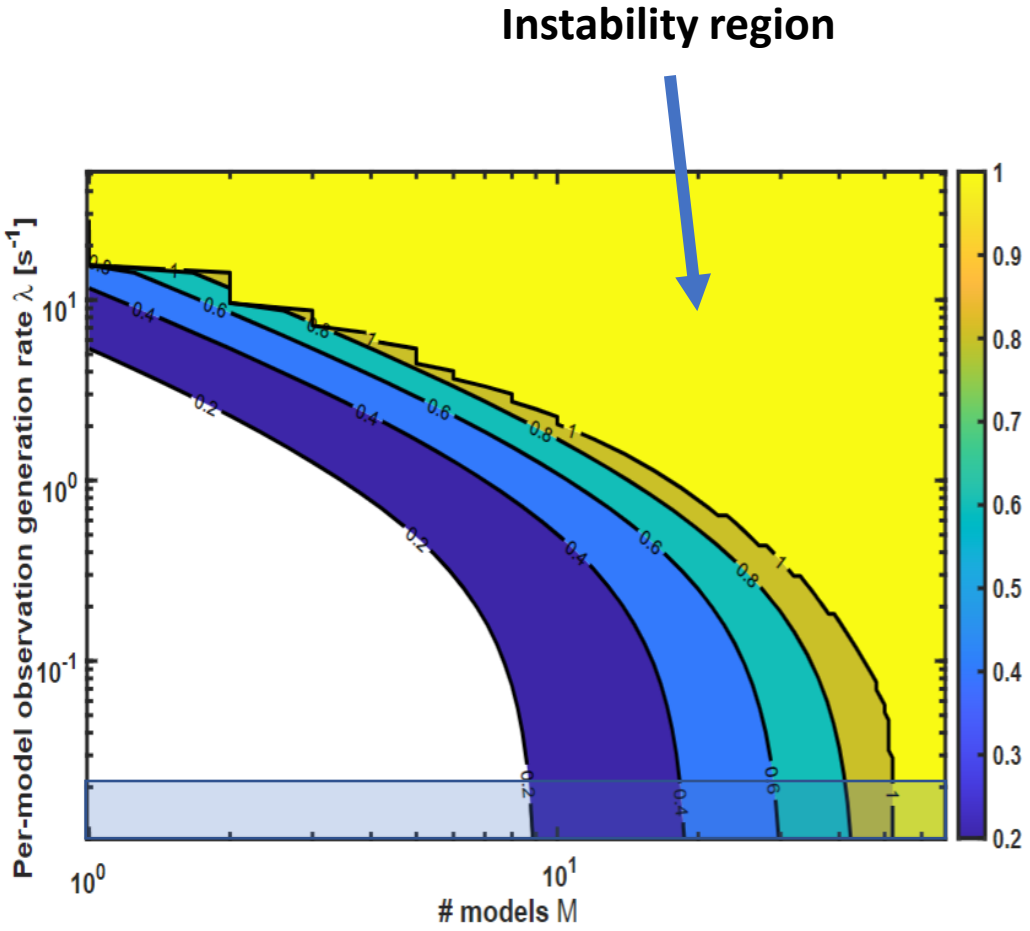
- Need for accurate system dimensioning to maximize efficiency as a function of the learning load

FG Learning capacity limited by the aggregate computing capacity of the collaborative system



- Tradeoff between **number of models** to train, **model size**, and **learning load**

FG Learning capacity limited by the aggregate computing capacity of the collaborative system

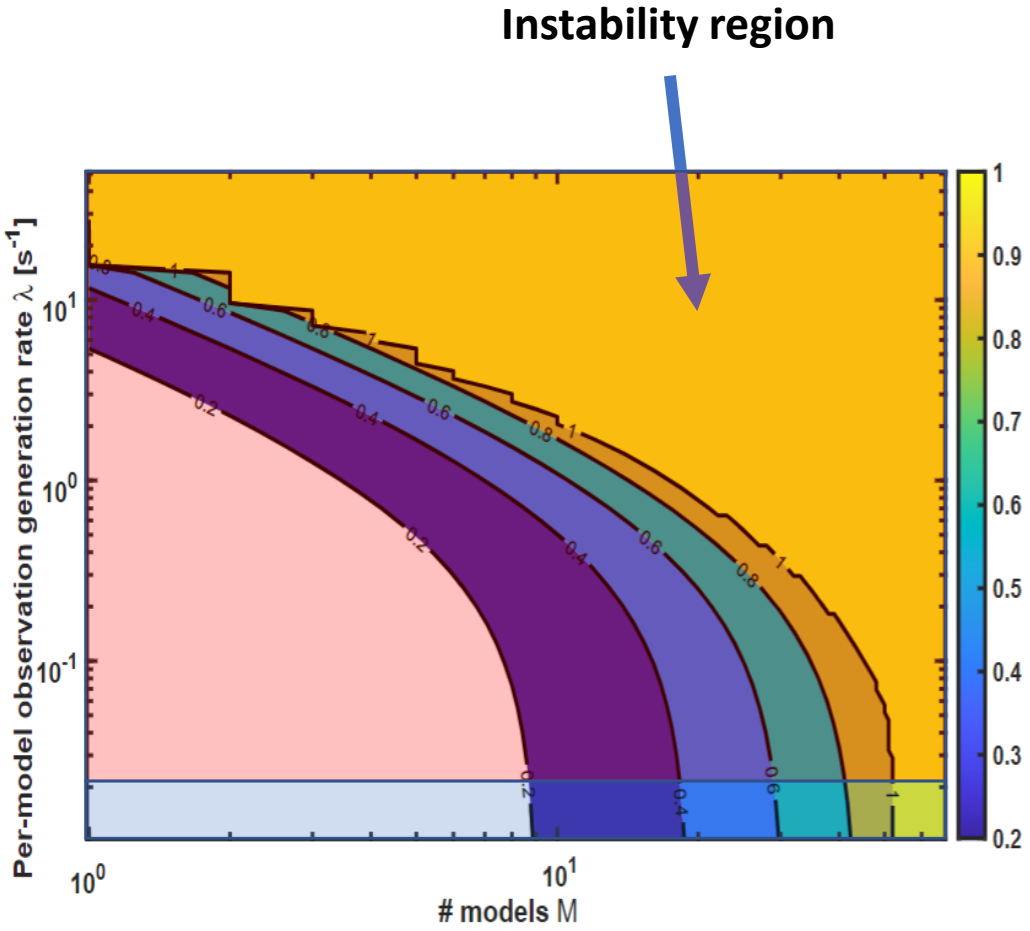


- Tradeoff between **number of models** to train, **model size**, and **learning load**

- **Model-limited regime**:

Limit performance depends on the effectiveness of gossip-based model exchange

FG Learning capacity limited by the aggregate computing capacity of the collaborative system



- Tradeoff between **number of models** to train, **model size**, and **learning load**

- **Model-limited regime**:

  Limit performance depends on the effectiveness of gossip-based model exchange

- **Computing-limited regime:** Limit performance depends on node's computing capacity

How fresh is the information incorporated in ML models via Floating Gossip?

Three regimes:

- **Low learning load**: Staleness increases with load (but it remains low)

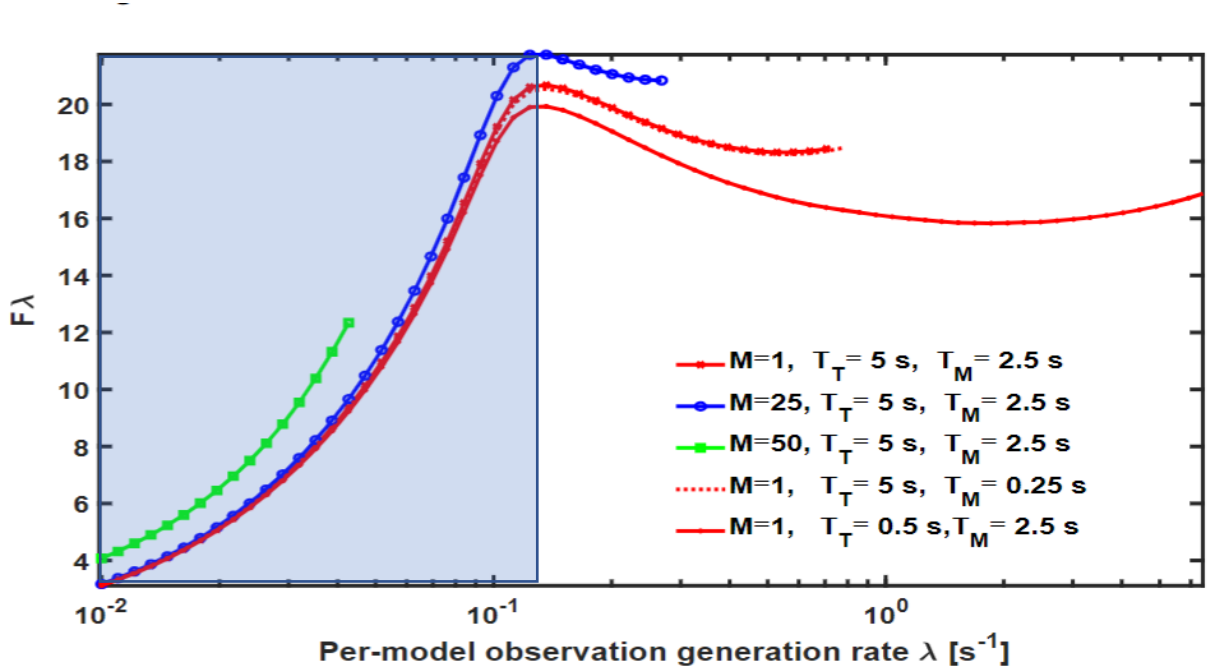


Fig. 4: Ratio of model staleness F over per-model mean observation interarrival time $1/\lambda$, versus per-model observation generation rate λ. Observation lifetime is $\tau_l = 300$ s, $k = 1$, with model size $L = 10$ kb.

How fresh is the information incorporated in ML models?

Three regimes:

- **Low learning load**: Staleness increases with load (but it remains low)
- **High learning load:** (close to system saturation): The system incorporates fresher information

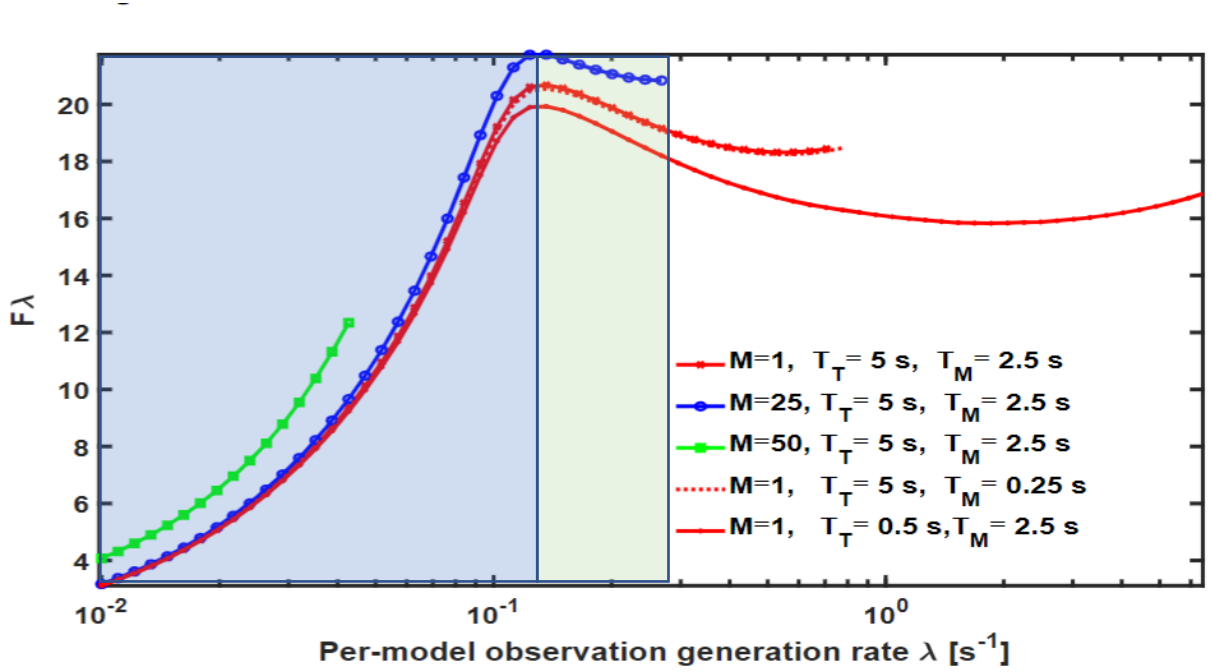


Fig. 4: Ratio of model staleness F over per-model mean observation interarrival time $1/\lambda$, versus per-model observation generation rate λ. Observation lifetime is $\tau_l = 300$ s, $k = 1$, with model size $L = 10$ kb.

How fresh is the information incorporated in ML models?

Three regimes:

- **Low learning load**: Staleness increases with load (but it remains low)
- **High learning load:** (close to system saturation): The system incorporates **fresher information**
- **Instability**

Increasing the n. of models to train brings to an earlier onset of instability, but with marginal impact on staleness (**scalability**)

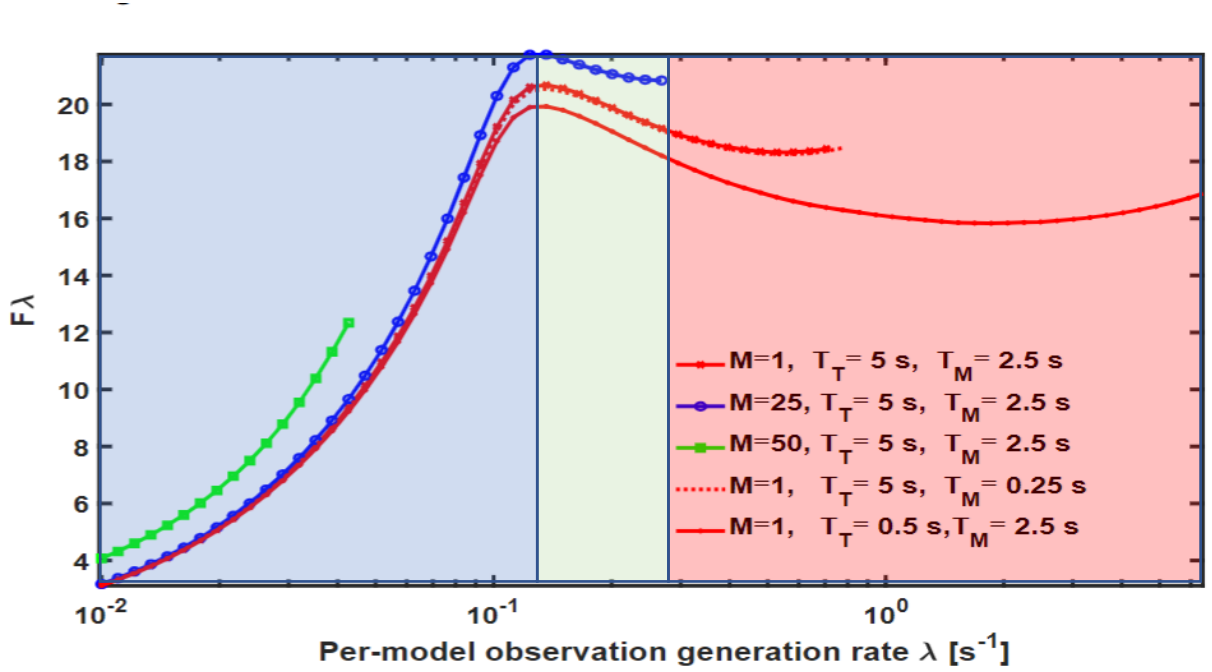


Fig. 4: Ratio of model staleness F over per-model mean observation interarrival time $1/\lambda$, versus per-model observation generation rate λ. Observation lifetime is $\tau_l = 300$ s, $k = 1$, with model size $L = 10$ kb.

Conclusions and future work

We proposed a framework for a first-order characterization of the performance of Floating Gossip

- A fully distributed, gossip-based learning scheme
- Performance patterns vary significantly from centralized, non collaborative learning schemes

Floating Gossip can be very effective in implementing fully distributed, cooperative continual learning schemes

- Very robust to node churn and mobility (and thriving on it)
- Scalable
- Performing better on high loads
- Limited by the aggregate of the computing resources of all nodes, and by gossip-based information diffusion

Future work: Heterogeneous scenarios

Thanks!

Questions: gianluca.rizzo@hevs.ch