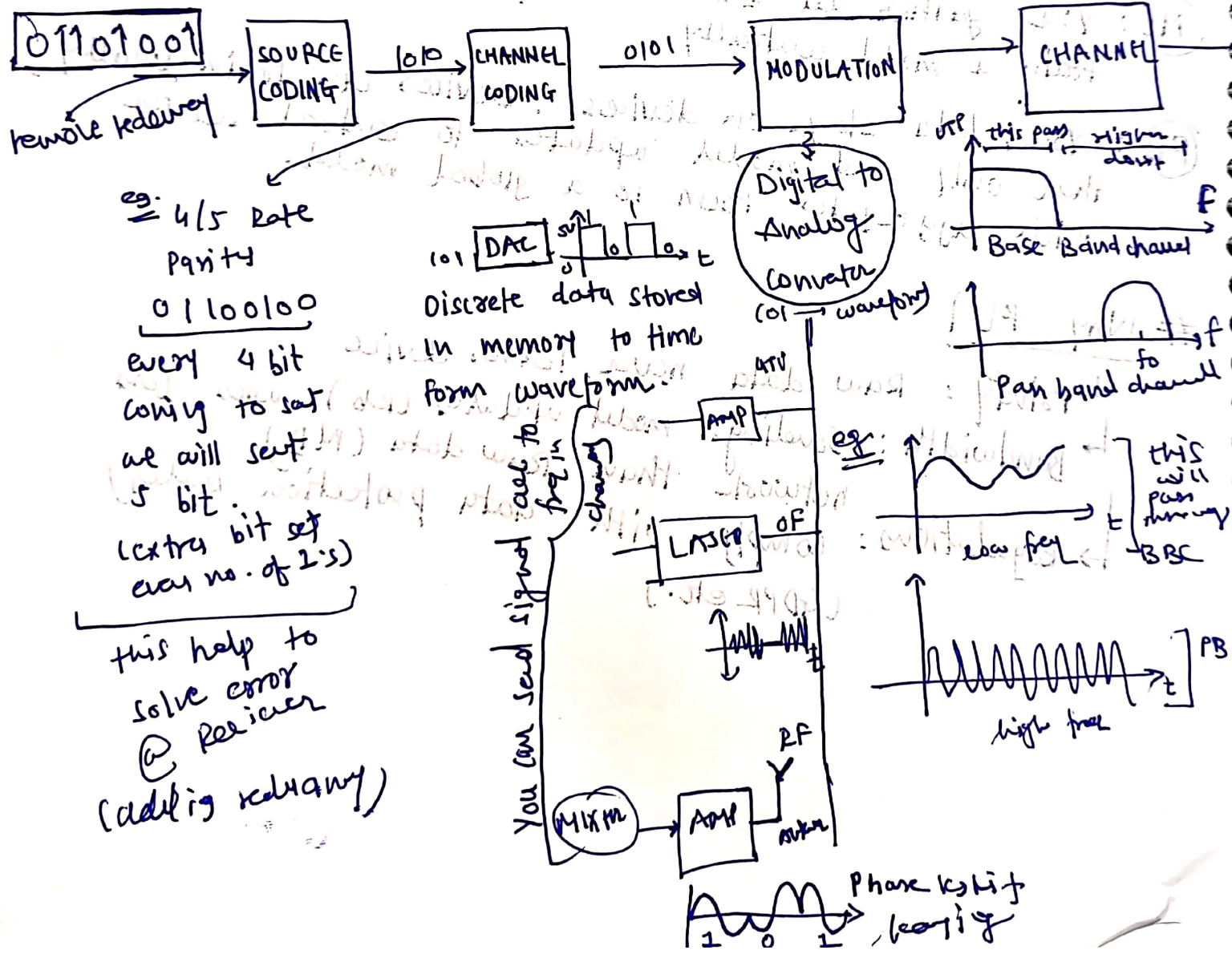
OFDMA: Orthogonal freq. Division Multiple

→ A technology in WIFI-6, improves wireless network performance by establishing independently modulating subcarriers within frequencies.

→ This approch allows simultaneous transmissions to and from multiple clients.

→ In dense depolyments, OFDMA enables multiple users to transmit data simutaneouly by dividing freq. spectrum into smaller subchannels & assigning them to different users. (Resource Units (RU))

How is Data Sent? AN Overview Of Digital Communications



MODULAION:

- The process of varying one or more props (Amp., phase or freq.) of a periodic carrier waveform in order to encode & transmit information. the information is a sequence of bits & each distinct bit pattern is mapped to a specific change in the carriers parameter(s).

Discrete (Digital) Modulation:

- Any modulation scheme in which carrier's parameters take on a finite set of distinct val, each representing a symbol (a grp of bits).

⊛ Federated learning:

(ML): you gather all data in one place (a server) then train a model centrally.

(FL): Your data stays on devices. Devices can train locally then only send model updates to central server which aggregates them to a global model.

Why FL?

↳ Privay: Raw data never leaves device

↳ Bandwidth: Sending model updates (1Kb) uses less network than Raw data (MB)

↳ Regulations: Comply with data protection Rules (GDPR etc.)

⊛ FedAvg Algorithm:

1. **server initialization:**

   server picks initial model parameter ω^0

2. **For each global Round $T=1,2,\cdots$:**

   a. server → clients : Broadcast ω^{T-1} to all selected clients.

   b. local Trainning : (each client m) :

   $\omega_m^0 = \omega^{T-1}$

   For each $i = 1 \to A$ (local epochs):

   $$\omega_m^i = \omega_m^{i-1} - \eta \nabla F_m(\omega_m^{i-1})$$

   Result : ω_m^A :

   c. clients ⇒ server : Each client sends update

   $$\Delta_m = \omega_m^A - \omega^{T-1}$$

   d. Aggregation: server completes weighted average:

   $$\omega^T = \omega^{T-1} + \sum_{m=1}^{M} \frac{|D_m|}{\sum_{m'} |D_{m'}|} \Delta_m$$

   ↳ weight according to data size.

$F_m(\omega)$: loss func. of m

$|D_m|$: No. of data samples on client m

η : Learning Rate

⊛ Differential Privacy (DP) in FL:

- A randomized mechanism M (eg: the noise adding step) is (ε, δ)-DP if any two adjecent datasets D & D' and for any measurable output set S:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr(M(D') \in S) + \delta$$

ε : privacy loss ⇒ smaller : strong privay

δ : small prob. of faliure

Gaussian Mechanism For model of FL updates:

For each clients model update Δm, we add noise:

$$\tilde{\Delta m} = \Delta m + \mathcal{N}(0, \sigma^2 I)$$

σ is calibrated by l_2-sensitivity ... Δ_s of Δm:

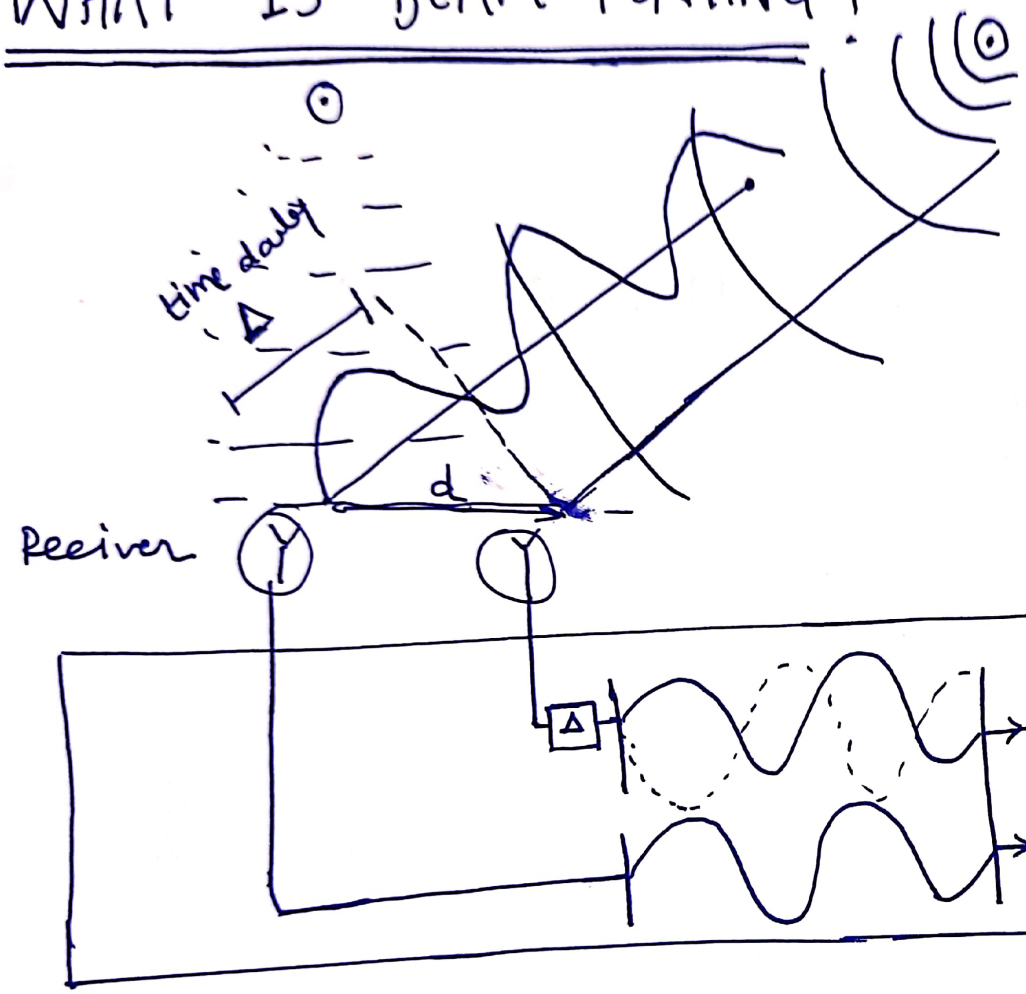
$$\Delta_s = \max_{D, D'} \| \Delta m(D) - \Delta m(D') \|_2,$$

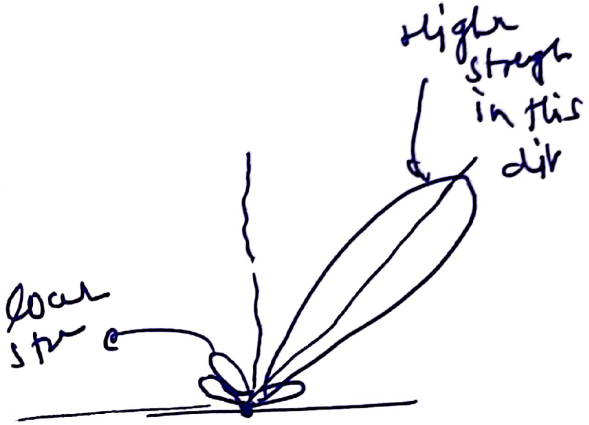
then; $\sigma = \frac{\Delta_s}{\varepsilon} \sqrt{2 \ln \frac{1.25}{\delta}}$

for G Round:

$$\varepsilon_{total} = \sqrt{2G \ln \frac{1}{\delta_{total}}}\, \varepsilon \quad \text{(approx. via adv. comp}$$

WHAT IS BEAM-FORMING?





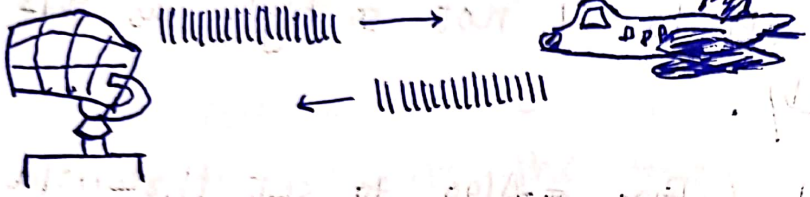
→ A technique that ~~discrete~~ directs wireless signals towards specific receiving devices rather than broadcasting in all directions

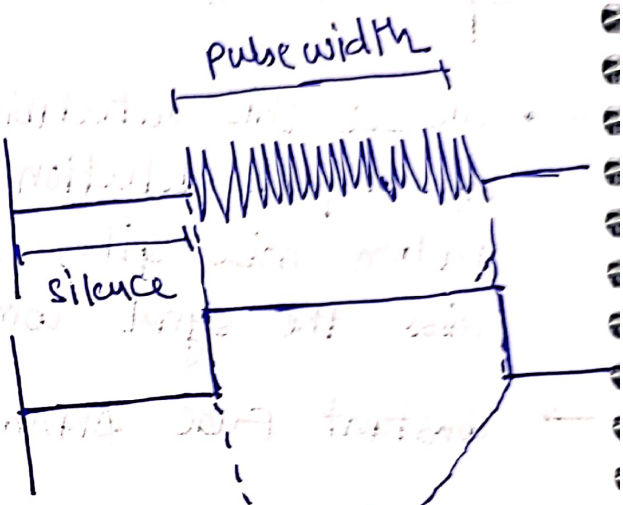
Enhance signal strength

Pulse-Doppler Radar | Understanding Radar Principles:

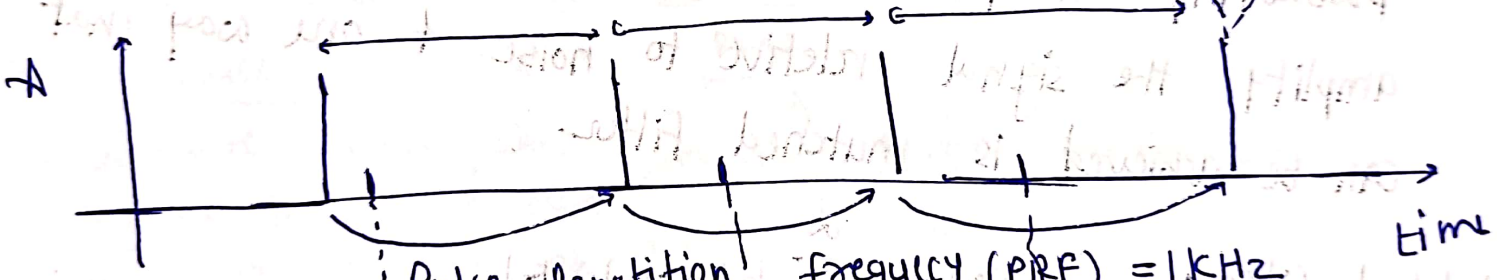
→ pulse Radar sends out short bursts or pulses of high energy waves followed by long periods of silence where receiver is listining for Reflected signal.

→ In this system, pulse has to Radiate out, reflect off of an object & return to the radar before the next pulse is sent.



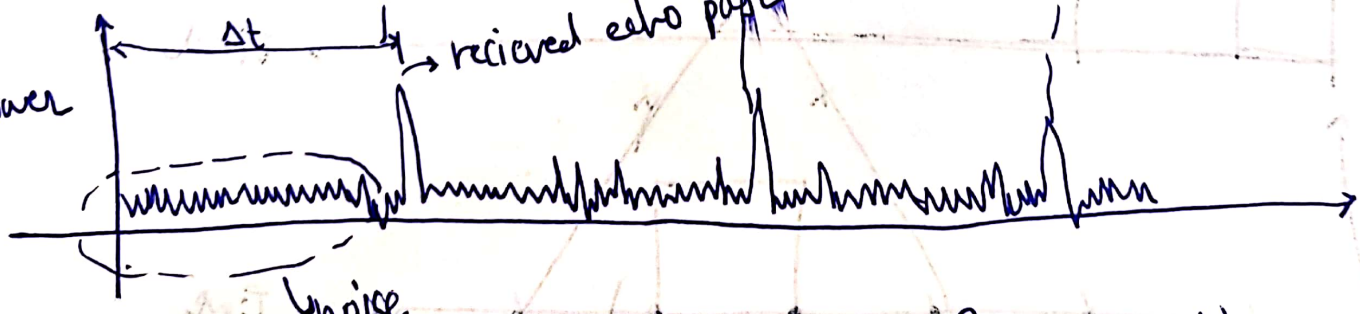


So, for so many such Rangular pulses.—



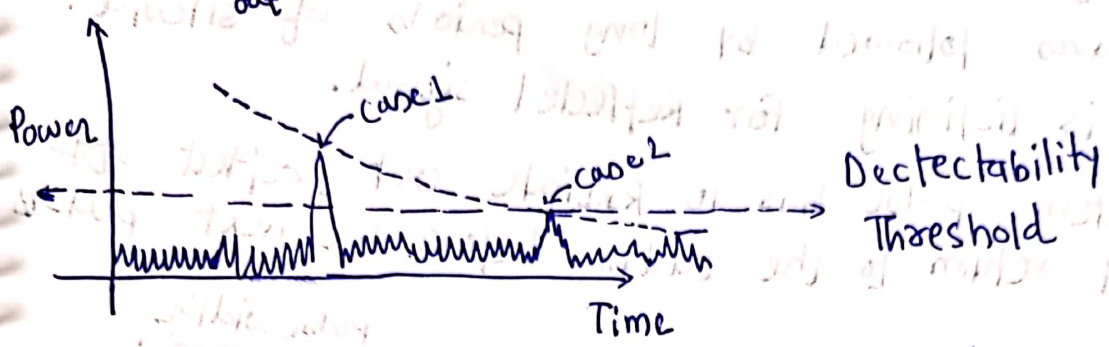
Pulse Repetition frequency (PRF) = 1 KHz.

Max range = $\frac{c}{2}$ · dead time between pulses ≈ 150 km



Range = $\frac{c}{2}\Delta t$.

→ signal power drops by the distance to the object raised to the power 4. As the object further away; it becomes difficult to pull it out of the noise.

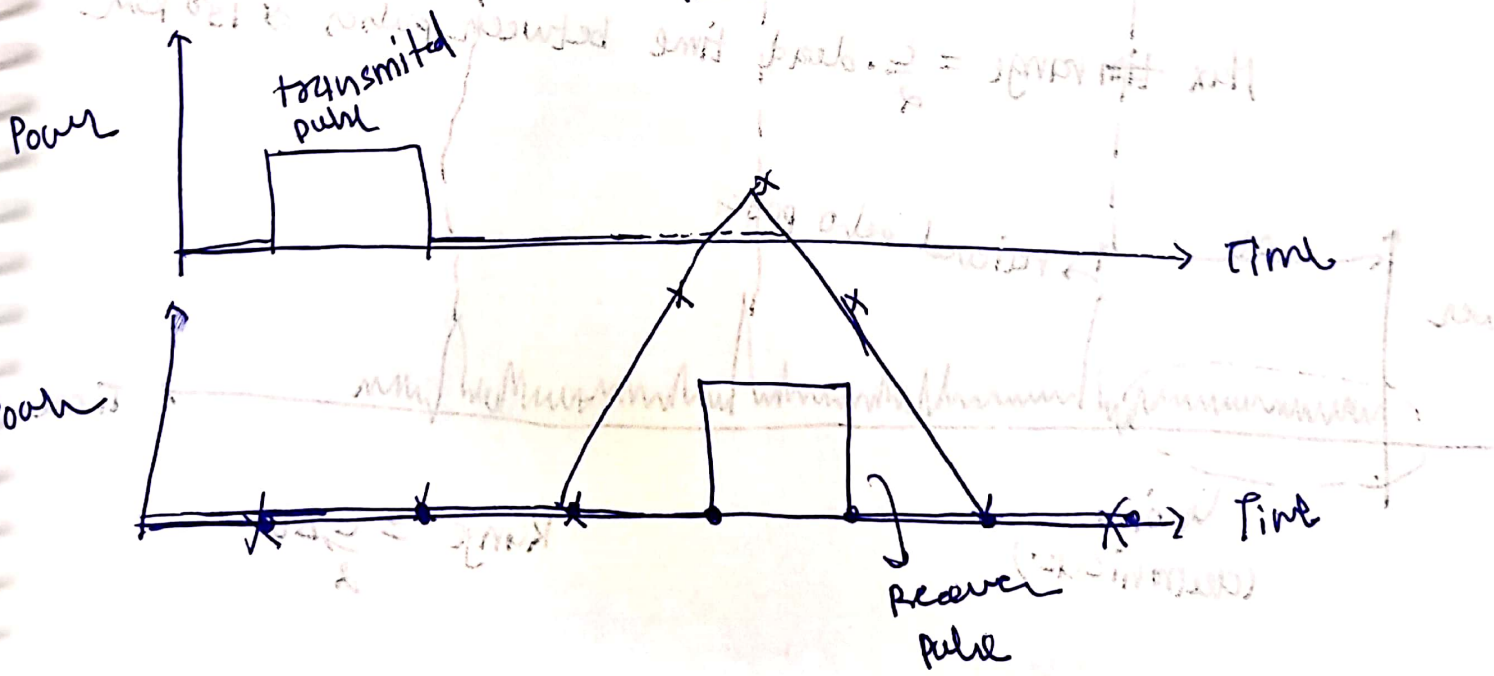


→ we set this dectectability threshold such that Probability of a false detection is low enough to not constantly claim radom noise spikes are objects & not so high as this miss the signal completely.

→ Constant false Alarm rate / cfar = Algo to set threshold.

→ To increse the Prob. of the Detection & decrease probability of false alarm we need to find a way to amplify the signal reletive to noise & one way that can be achieved is matched filter.

Matched filter: $y[n] = \sum_{k=-\infty}^{\infty} h[n-k]\, x[k]$



→ we know the rectangular shape of transmitted pulse will have same shape when Received. (x in eq) (h in eq.)

→ By multiplying x by h that has been reversed in time & then summing up the result we are amplifing areas of strong corelation between h & x.

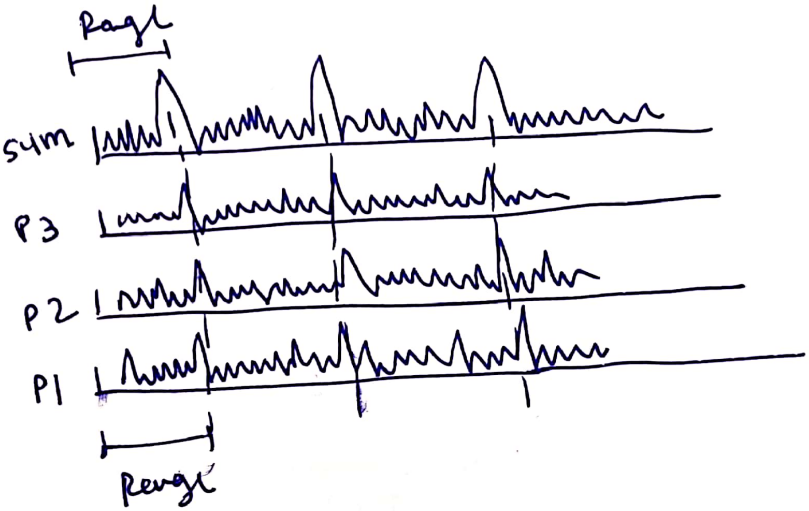
→ The effect of amplyfing the signal & making signal more narrow by turning wide rectangle pulse into a single peaked triangle & thats why matched filter is called "compresions pulse".

→ This type of filter will amplify the signal more than that will amplify the ~~signal~~ noise since there is a perfect corelation between pulse & received signal.

How about boost signal to noise even more?

↳ "By pulse integration."

- Integrate several pulses together in a row & sum up all of energy in each of them until the signal rises above detectability - threshold.



INTRODUCTON TO FL :- (CH:1)

→ FL is an approch to ML in which trainning data is not managed centrally.

→ Data is retained by data parties that participate in FL processes & isn't shared with any other entity.

IN ML
- Problems :
  - → Bringing data together in centralized repo is problemstic.
  - → privacy issues.

→ The application of ML depends on availability of high quality trainning data.

→ sometimes, privacy considerations prevent trainning data to be brought to central repository to be proceed for ML.

"Federated learning is an approach to train ML models on trainning data in disparate locations, not requiring the collection of data."

#Regulations → [on use of consumer data] → GDPR, HIPAA, CCPA

Problem → Poor Communication connections and the sheer amount of data collected by sensors or in telecommunication devices can make central data collection infeasible.

USE → FL also enables different companies to work together creating models for mutual benefit without revealing their trade secrets.

*. How Does FL Works?

(In FL Approach) → (a set of distinct parties) → (who controlls their respective trainning data) ↓ (collobrate to train a ML model) ← [without sharing their trainning data with other PARTIES.]

⇝ PARTIES = CLIENTS = DEVICES

- consumer devices such as smart phones or cars
- cloud services of different providers
- data centers processing enterprise data in different countries
- Application silos within company
- Embaded System such as Automative plant, Robots.

⇝ "AGGREGATOR" = "SERVER" = "COORIDINATOR"
↳ do collobrations

[Parties Perform a local trainning data process based on their private training data] ⇝ (local training complete) → [model updates] → Aggregator

↳ local parameters

eg. for neural network, model parameter might be the weight of network.

⇝ "MODEL FUSION"

[once aggregator receives model updates from parties] ⇝ [they can be merged into the common model.] ⇝ (called Model fusion)

eg. In neural network the can be as simple as averaging weights.

→ the resulting merged model is then distributed again to parties as model updates to form the basis of next "ROUND" of trainning.

* Distributed learning V/S FL:-

(i) **Distributive learning** :

↳ uses clusters (groups) of computers (nodes) to share & speedup ML tasks.
↳ Managed centrally where data is split & given out to nodes.
↳ Often uses "parameter server" to gather & aggregate results from each node.

Key differences :

1] **Data Distribution & Privacy** :

↳ data is managed centrally
↳ All nodes recieve controlled & unknown data splits

2] **Federated Learning** :

↳ data is remains local
↳ The system can't assume that each device has data ~~limit~~ that is both independent & identically distributed (non-IID)
↳ some parties have much more data than others causing imbalance.

+ Implications of Training :

↳ FL algorithms must be designed to handle data imbalance & the lack of central control over stastistical props. of data.

CONCEPT OF FEDRATED LEARNING:

"Fedrated learning is a distributive machine learnir approach where a global model M (representing a predictive function f) is collaboratively trained usir data partitioned accross multiple parties. Each Party P_k holds its own private data D_k & no party (inc ding aggregator) have direct access to another's d This paradigm preserves data privacy while still allowing the training of global model."

Vanilla FedAvg Mathematical Model: | Phase : 1

Overview Fed learning: INTRODUCTION

FL is a decentralized ML paradigm in which multiple clients collaboratively train a shared global model under the coordination of a central server, while keeping their data locally.

Instead sending raw data to the server, clients compute local updates based on their private datasets & send only model updates [e.g. gradients or parameters] to server.

This design promotes
- → data privacy
- → low communication cost
- → scallibity

across distributed systems
↳ collection of independented comp's. that works together to achieve common goal.

TYPICAL FL ROUND:

1) BROADCAST:
server sends current global model to clients.

2) LOACAL TRAINING:
Each client performs local optimisation using its private data.

3) AGGREGATION:
clients send updated models to server, which averages them to update global model.

Vanilla FedAvg Mathematical Model: | Phase : 1

Overview Fed learning: INTRODUCTION

FL is a decentralized ML paradigm in which multiple clients collaboratively train a shared global model under the coordination of a central server, while keeping their data locally.

Instead sending raw data to the server, clients compute local updates based on their private datasets & send only model updates [e.g. gradients or parameters] to server.

This design promotes
- data privacy
- low communication cost
- scalibity

across distributed systems
↳ collection of idependented comp's. that works together to achieve common goal.

TYPICAL FL ROUND:

1) BROADCAST:
server sends current global model to clients.

2) LOACAL TRAINING:
Each client performs local optimisation using its private data.

3) AGGREGATION:
clients send updated models to server, which averages them to update global model.

Key Notations :

M = no. of clients
m = Index of clients
K = no. of classes
(e.g. 10 Fo MNIST)
d = dimension of Input Vec.
D = Global dataset $D \subset \mathbb{R}^d \times \{1,\dots,K\}$
D_m = local Dataset @ client m s.t. $D = \bigcup_{m=1}^{M} D_m$
$|D_m|$ = size of local Dataset @ client m
D_{tot} = total no. of samples accross all clients. i.e $D_{tot} = \sum_{m=1}^{M} |D_m|$
ω = Global model parameter $\omega \in \mathbb{R}^n$ (n = total parameters)
ω_m = local copy of global model that client m trains on its own dataset
$\ell(\omega; x, y)$ = loss incured when predicting label y from input x using model parameter as
(For classification loss → cross entropy)
(MSE For Regression)

$F_m(\omega)$ = local empherical loss @ client m
↳ avg. loss over client m's local dataset

$$\therefore F_m(\omega) = \frac{1}{|D_m|} \sum_{(x,y) \in D_m} \ell(\omega; x, y)$$

$F(\omega)$ = Global loss function
↳ weighted avg. of local losses (weighted by dataset size)

$$F(\omega) = \sum_{m=1}^{M} \frac{|D_m|}{D_{tot}} F_m(\omega)$$

$\frac{|D_m|}{D_{total}}$ = weight

↳ approx. the loss over entire dataset D while not requiring access to full data locally.

#Assumptions on Data Distributions:

For vanila FedAvg setup...

- Global Dataset D is IID (independely & identically Distributed) i.e each sample (x,y) in D is drawn from the ~~~~ underlying distribution.
- Data partitoned uniformly at random accross clients.
  $D_1, D_2, \ldots, D_M$ are (IID) subsets of (D)
- Each clint hold approx. equal-sized partitions.
  $|D_m| \approx \frac{D_{tot}}{M}$ for each $m = 1, \ldots, M$

System Model::

clients & Server:

system consist of a [central server] & [M clients] (eg. mobile devices) (edged N)

server → [Coordinates training process]
- → Maintainning global model.
- → Broadcasting model params to clints
- → local updats Aggregate

clients → Performs local trainning based on their private datasets & return updated model parmets to server.

ata Partitionning:

$D \subset \mathbb{R}^d \times \{1, \ldots, K\}$] each (x,y) in D consist of input feature vector $x_i \in \mathbb{R}^d$ & lable $y \in \{1, \ldots, K\}$

ta is partitioned into M disjoint local subsets.

$D = \bigcup_{m=1}^{M} D_m$; For $i \neq j$ $D_i \cap D_j = \emptyset$ (IID participation)

& each clint m have $|D_m|$ size data.

Model Architecture:

- shared model is parameterized by a vector $\vec{w} \in \mathbb{R}^n$ (e.g. the weights & biases for neural networks)

- Simple Multiplier Perceptron (MLP):
  - one hidden layer of size 32
  - ReLU activation
  - softmax output for K-classification (e.g. MNIST)

- Each client maintains a local copy of the global model denoted by w_m which updates during local training.

Local Objective & Global Objective:

Local objective on client m:

- each client minimizes a local loss function $F_m(w)$ based on its private dataset D_m.

$$f_m(w) = \frac{1}{|D_m|} \sum_{(x,y) \in D_m} \ell(w; x, y)$$

$\ell(w; x, y)$ = loss incurred on a single training example (x, y) with parameters $\vec{w}$.

- for classification typical choice is cross-entropy-loss.

Global (Aggregated) objective :

- server seeks to minimize global loss func., defined by **weighted Avg.** of local losses.

$$\therefore F_m(w) = \left[\sum_{(x,y)\in D_m} \ell(w; x, y)\right] \frac{1}{|D_m|}$$

$$\therefore F(w) = \sum_{m=1}^{M} \frac{|D_m|}{\sum_{j=1}^{M} |D_j|} F_m(w)$$

global loss function

$$\therefore F(w) = \sum_{m=1}^{M} \frac{|D_m|}{\sum_{j=1}^{M} |D_j|} \cdot \left[\sum_{(x,y)\in D_m} \ell(w; x, y)\right] \frac{1}{|D_m|} \quad \textcircled{*}$$

Optimality Target :

the global goal of trainning is to find an optimal model w^* that mimize the global objective:

$$\therefore w^* = \arg\min_{w\in\mathbb{R}^n} F_m(w) = \arg\min_{w} \sum_{m=1}^{M} \frac{D_m}{D_{tot}} F_m(w)$$

$$\therefore w^* = \arg\min_{w\in\mathbb{R}^n} \sum_{m=1}^{M} \frac{|D_m|}{\sum_{j=1}^{M} |D_j|} \cdot \left[\sum_{(x,y)\in D_m} \ell(w; x, y)\right] \frac{1}{|D_m|} \quad \textcircled{*}$$

Now ; Here comes Interesting thing :

As; Global loss :

$$\therefore F(w) = \sum_{m=1}^{M} \frac{|D_m|}{D_{total}} f_m(w) \qquad \text{here; } D_{total} = \sum_{j=1}^{M} |D_j|$$

$$\therefore F(w) = \sum_{m=1}^{M} \frac{D_m}{D_T} \left(\frac{1}{D_m} \sum_{(x,y) \in D_m} \ell(w; x, y) \right)$$

$$\therefore F(w) = \sum_{m=1}^{M} \frac{1}{D_T} \sum_{(x,y) \in D_m} \ell(w; x, y)$$

$$\therefore F(w) = \frac{1}{D_{Total}} \cdot \sum_{m=1}^{M} \sum_{(x,y) \in D_m} \ell(w; x, y)$$

$$\boxed{\therefore F(w) = \frac{1}{D_{Total}} \sum_{(x,y) \in D} \ell(w; x, y)}$$

(00) !! - Just avg of entire dataset D.

- What's Role of $|D_m|$?

Why do we still Weight Clients by D_m ?

- we don't send gradients or losses to the server. we send parameter updates, which are influenced by how many data points the clients used to compute them.

when server aggregates updates :

$$\boxed{w_{t+1} = \sum_{m=1}^{M} \frac{D_m}{D_{Total}} w_m}$$

- its doing to ensure that each client's contribution is proportional to the size of its dataset - not because $f_m(w)$ numerically includes D_m, but because the update (w_m ~~itself~~ was trainned on D_m samples.

If we don't weight by D_m, client with 10 samples could have same influence as one with 10,000 which would be unfair & degrade convergence.

Convergence Analysis: ???

Goal of optimisation:

- In FL (or in any iterative process), we repeatedly update our model parameter w so as to (hopefully) drive the loss $F(w)$ down toward its minimum $F(w^*)$.
- Convergence analysis quantifies "HOW FAST" & "UNDER WHAT CONDITIONS" these updates will actually get you close to minimum.

Guarantees v/s. Practice:

In practice you run FedAvg for some fixed no. of G & pick hyper-parameters (learning rate η, local epochs E etc.) by trial & error.

Convergence analysis gives you guidance:

- How should η scale with E, G & problem's smoothness L?
- How does data Heterogeneity (clients seeing diff. data) slow you down?
- Will you even converge if η is too large or too small?

Comparing Methods:

- By deriving upper bound on $F(w_G) - F(w^*)$, you can compare FedAvg to other Algorithms under same assumptions.

When Do We Do It?

→ Designing a new algorithm:
e.g. adding compression, noise you need to show that it still converges.

→ Tuning hyper-parameters:
The theory often prescribes safe regions

→ Understanding trade-offs:
more local work (A) reduces communication but may degrade convergence if data is heterogeneous.

Convergence Analysis for FedAvg (Convex Case):

- classic (convex) analysis of vanilla synchronous FedAvg, without any DP noise. We'll see the bound depends on three key factors:
  - ↳ Number of Global Rounds G
  - ↳ Number of Local steps A
  - ↳ Data heterogeneity.

Intutive Anatomy of the One Round Bound:

After one round of FedAvg (each client doing ηA local steps of size η), we can Roughly show:

steepness of curve — Heterogenity penalty

$$\underbrace{E[F(w_{t+1})]}_{\text{Expected global loss after aggreg.}} \leq \underbrace{F(w_t)}_{\text{Loss before this Round}} - \underbrace{\eta A}_{\text{Efforts per Round}} \|\nabla F(w_t)\|^2 + \frac{\eta^2 LA}{2M} \sum_m \|\nabla F_m(w_t) - \nabla F(w_t)\|^2 + \underbrace{\eta^2 A L \sigma^2}_{\text{Variance panelt}}$$

1) $F(w_t) \rightarrow E[F(w_{t+1})]$

- When you start round loss = $F(w_t)$
- when you end up next Round, expected Loss = $E[F(w_{t+1})]$

2) $\eta A \|\nabla F(w_t)\|^2$

- $\|\nabla F(w_t)\|^2$: how steep the loss landscape is @ w_t
  ηA : Total "step-size budget" per Round
- The bigger the gradient norm & more local steps you take, the more you expect to decrease the loss that Round.

3) Heterogenetiy penalty

- $\sum \|F_m - \nabla F\|^2$: If client sees diff data, their local gradients points in slight diff dirc.
  So you don't make as pure a descent step. The penalty quantifies that misalignment.

④ Variance Penalty:

- $\propto \sigma^2$: variance of (mini batch) stochastic gradients
- Noise can undo your progress

So:

If data is IID (no heterogenity) + you use full gradients ($\sigma^2 = 0$)....

$$\therefore E[F(w_{t+1})] \leq F(w_t) - \eta A \|\nabla F(w_t)\|^2$$

but when data is heterogeneous ; the extra terms reduce your net gain in each round.

#Inferences:

1] More local steps A

2] larger learning Rate η
(but both penalty will scale like η^2 so too large η will actully increase loss)

3] Data Homogeity Helps

4] Larger Batches → smaller σ^2 → less Variance penalty

Big Picture:

Inequality is a tuner for AvgFed.

- Balance ηA so descent dominates the penalties
- Mitigate penalties by increasing batch size or reducing heterogeneity.

$$\therefore \underbrace{\text{Gain}_t}_{\eta A \|\nabla F\|^2} \geq \underbrace{\text{Cost}_{\text{net}}}_{\propto \eta^2 A} + \underbrace{\text{Cost}_{\text{var}}}_{\propto \eta^2 A}.$$

Synchronous FedAvg Algorithm:

Params : G, A, η

Initalizeon:

server sets ω^0 (random/pretrained):

For Each Global Rond $t = 0 \rightarrow G-1$:

a. Broadcast ωt to all clients

b. On each client m:

i) $\omega_m^{t,0} = \omega t$

ii) For $j = 1, \dots A$:

$$\omega_m^{t,j} = \omega_m^{t,j-1} - \eta \nabla F_m(\omega_m^{t,j-1})$$

c. Client m returns $\omega_m^{t,A}$

d. server Aggregates:

$$\omega^{t+1} = \sum_{m=1}^{M} \frac{D_m}{D_{tot}} \omega_m^{tA}$$

output ω_G:

Differentially Private Wireless Federated Learning With Integrated Sensing & Communication :-

↳ novel framework for Differentially private (DP) wireless FL with ISAC.

Framework : DP - ISAC - FL ⇒ wireless device → Sense data → upload train local models using ISAC technique

→ @ each device local trainning take place concurrently with sensing!

INTRODUCTION :

→ IN wireless world, devices (phones, cars) can do something with radio waves. They send out radio "ping" and listen for the echo. That tells them where the objects are. This is called "Echo" "sensing".

→ devices use those same radio waves to send data like mesg, pictures or model updates to server or to each other. This is called "communication".

Most of them have two seperate devices
- ↳ one for sensing (radar like)
- ↳ one for communication (wifi/cell phone)

If we could make a single device that both do sensing & comm. @ same time [ISAC]

ISAC-FL

- ↳ Senses (sense env data)
- ↳ Trains (trains own model on data)
- ↳ Communicate (Model exchange)
- ↳ Repeats

all using same device (same radio circuitry & spectrum)

Time-Division Multiplexing:

↳ as both are happening through same device, both can't happen @ same instant.

Sensing – comm. – sens. – com – – –
1sec 1sec 1sec 1sec

This is Time Divison Multiplexing that you share your time btw sensing & commu.

ISAC-FL optimisation:

↳ selection of smart devices for sensing, training & model updating
↳ time allocation (train, sense, update, share)
↳ beam forming designs for radar sensing
↳ allocation of freq channels & transmit power for devices selected to deliver their local models.

Differential Privacy (DP):

You are doing sensing (like radar) & training a model on that radar data to identify things...

But] Radar data can reveal your location:

eg- car radar sees object 5 m ahead
↳ that radar pattern basically shows where car is.
↳ here your sensing data leaks location info.

What DP does!

↳ adds radom noise to model befor uploding

↓

This makes mathematically hard to infer things.

But adding = loosing
noise accuracy

global model will train a bit slower

∴ Effect on Converg

∴ If your dataset is big (good sensing) → even if you add noise → not much effect on accuracy

So Here; **Balance** [Privacy (more noise)] + [Performance (less noise)]

IN GIVEN PAPER: **DP-ISAC-FL framework**:

DP-ISAC-FL: FL with DP using ISAC

OFDMA for sensing: . Uplink communication (model updoads) uses "orthogonal frequency Division Multiple Access"; so multiple device can uplod simutaneously on separate frequency solts (subchannels).

Discrete Modulations: Devices can't use "any" single modulation - they pick from real-world sets like QPSK, 16-QAM That reflects practicle hardware limits.

What is analyzed?

↳ CONVERGENCE UPPER LIMIT of FL process.

i.e. "given all delays, noise, limits in radio & computing, what is best accuracy the global model can hope to reach."

→ **limits** → which devices are selected
→ How time is split between sensing & comm.
↳ which subchannels or modulation types are assigned
→ How much power each device use

What is optimized?

↳ "Built a new algorithm that minimize that convergence upper limit."

i.e. It helps system learn faster while keeping privacy and resource limits in check.

Algo Jointly Decides:

- Sensing beamforming = which dir. & strgn. to use for radar sensing.
- Time allocation = How much time to spent on sensing & uplo ding
- Subchannel assignment = Which device get which freq. band for uploading
- Modulation choice = What data rate / modulation scheme each device should use
- Power controll = How much transmission power each device should use for effective, low error uploads.

→ This paper builds & analyzes a smart, private FL system where car / devices share sensing and comms hardware & It designs a joint scheduling algo. to make them learn as fast as possible - despite real world limits like noisy channels, fixed modulations, limited power & privacy rules.

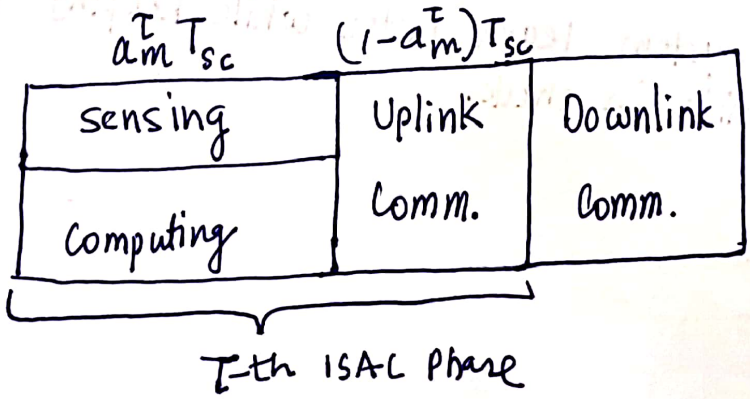
[+] Architecture of DP-ISAC-FL

A. System Model

Proposed DP-ISAC-FL has OFDMA interface. → where FL server serves M clients via K subchannels → Server has M_s antennas.

→ trains global model relying on individual datasets of M clients by periodically aggregating the local models trained at clients.

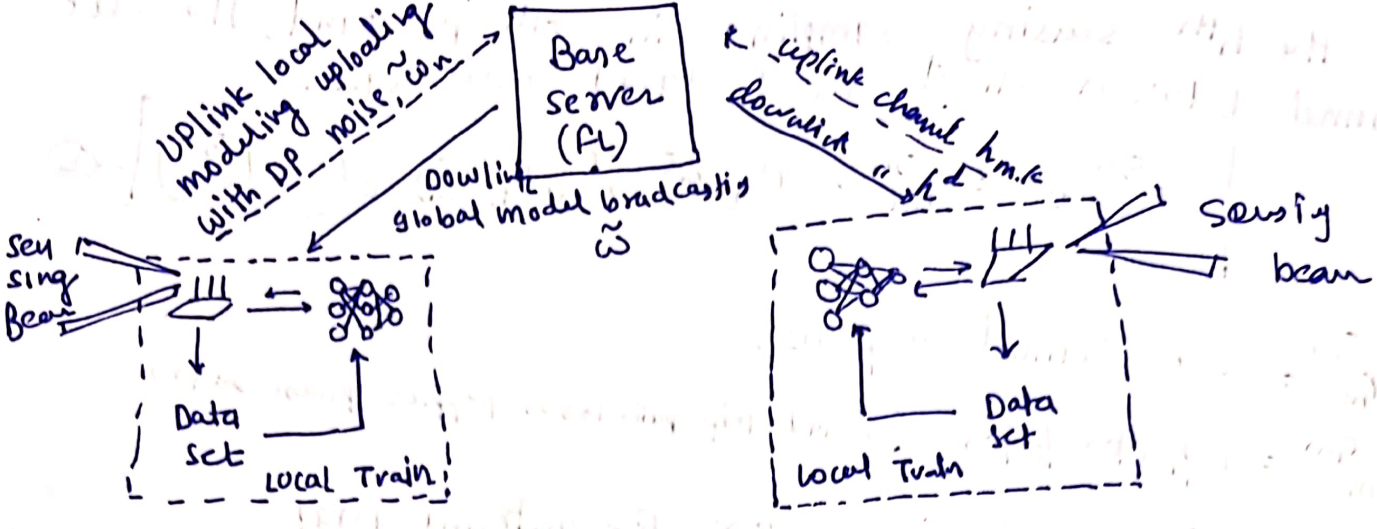
→ Each client $m \in \mathcal{M} = \{1, \dots, M\}$ has M_a antenas & can perform sensing and communication in a "time division multiplexing (TDM)" manner in each FL round.



a_m^I = Time allocation btw sensing & comm. at each client in I-th FL round

T_{sc} = Duration of an ISAC phase

Time allocation in DP ISAC-FL aggregation round



sensing + trainning parallelly + uploading afterwards (communication)

[B] Radar Sensing Model

a_m^τ = fraction of time client m spends on sensing in round τ (between 0,1)

C_s = sampling interval $\Rightarrow 1/C_s$ samplin rate (Hz)

D_τ^m = Total no. of Data samples M have @ t end of round τ

T_{sc} = Total duration of ISAC Round

θ_m^τ = client m detects a target θ_m^τ relative to clients position.

∴ local Dataset @ end of τ Round of client m. :

$$D_M^\tau = D_M^{\tau-1} + \frac{a_\tau^m T_{sc}}{C_s} \quad \text{-----①}$$

∴

$$a_m^\tau = \frac{C_s \left(D_M^\tau - D_m^{\tau-1}\right)}{T_{sc}} \quad \text{----①} \; ; \forall m,\tau$$

↳ measer or adjust sensing time based on dataset growth and Sampling speed.

At the n^{th} sensing sampling in τ^{th} PRI round, the echo channel between client m & target given by...

$$G_m^{\tau,n}(t) = g_m^{\tau,n}\, \alpha(M_a, \theta_m^\tau)\, \alpha^T(M_a, \theta_m^\tau)\, \delta(t - d_m^{\tau,n}) \quad -- (1)$$

$G_m^{\tau,n}$ = echo channel response

$g_m^{\tau,m}$ = reflector factor (round trip path loss, Doppler phase shift)

$\alpha(M_a, \theta_m^\tau)$ = steering vector for antenna array.

$$\alpha(\overset{\text{no. of antennas}}{M_a}, \theta_m^\tau) = \left[1, e^{j2\pi d \sin\theta_m^\tau/\lambda}, \dots, e^{(M_a-1)j\,2\pi d \sin\theta_m^\tau/\lambda}\right]^T$$

↳ how to focus its beams in direction θ_m^τ (directional vector)

$\delta(t - d_m^{\tau,n})$ = direct delta function = ping @ time $d_m^{\tau,n}$

↳ how far target is
↳ how long the echo took to return

$G_m^{\tau,n}(t)$ = echo = reflecting strength × beamforming direction vectors × delayed impulse return

---

For clutters (unwanted echos)

Clutter Signal

$$C_m^\tau(t) = \sum_{q=1}^{Q \to \text{Total clutter}} \underbrace{C_m^{\tau,q}}_{\text{reflection factor}}\, \alpha(M_a, \underbrace{\psi_m^{\tau,q}}_{\text{AOD}})\, \alpha^T(M_a, \underbrace{\phi_m^{\tau,q}}_{\text{AOA}})\, \delta(t - d_m^{\tau,q}) \quad -- (2)$$

- Client m sends a radar sensing signal (often an FMCW (Freq-Modulated Continuous Wave) pulse.
- The signal is beamformed using w_m^T - a vector that focuses the radar wave in direction θ_m^T.

The Received signal @ client m....

∴ Received echo is...

$$y_m^{T,n}(t) = \left[\left(G_m^{T,n}(t) + C_m^T(t) \right) * w_m^T \right] * s_m^{T,n}(t)$$

$$\therefore y_m^{T,n}(t) = \left[g_m^{T,n} \hat{G}_m^T w_m^T + \sum_{q=1}^{Q} c_m^{T,q} \alpha(M_q, \psi_m^{T,q}) \alpha^T(M_q, \phi_m^{T,q}) w_m^T \right] \times s_m^{T,n}(t - d_m^{T,n}) + n_m \quad \text{---- (4)}$$

$\hat{G}_m^T \triangleq \alpha(M_q \theta_m^T)\, \alpha(M_q, \theta_m^T)$

$*$ = convolution op.

$n_m \sim CN(0, \sigma^2 I_{M_q})$ = additive white Gaussian noise. (AWGN)

Signal-to-Interference-plus-Noise Ratio (SINR):

SINR tells us how strong the useful echo is compared to junk + noise.

$$SINR_m^T = \frac{\| g_m^T \hat{G}_m^T w_m^T \|^2}{\| C_m^T w_m^T \|^2 + \sigma^2}$$

→ signal power from desired target

→ noise + clutter power

High SINR = Better sensing quality

$\Gamma_{min} \leq SINR_m^T$ → threshold for effective sensing

$\| W_m^T \| \leq P_{max}$

emmison power of each client must not exceed the max. transmit power.

[C]. FL Model

* T = total iterations of local updates
* Every A iteration you do one global aggregation.

$$\therefore G = T/A$$

→ we want to collaboratively train a single model w that minimizes the weighted sum of all client losses:

$$F(w) = \sum_{T=1}^{G} \sum_{m=1}^{M} \frac{|D_m^T|}{\sum_{m'} |D_{m'}^T|} F_m(w) \quad \text{——— (5)}$$

$F_m(w)$ = client m's local loss function on that clients data.

The optimal model w^* solves...

$$w^* = \arg\min_w F(w) \quad \text{——— (6)}$$

@ Local Model Trainning..

- At each local iteration i, client m updates its local parameter w_m^i by one step of gradient descent:

$$w_m^{i+1} = w_m^i - \eta \nabla F_m(w_m^i) \quad \text{——— (7)}$$

[η = learning rate
$\nabla F_m(w_m^i)$ = the gradient loss on client m at w_m^i]

= Each client does such A local updates before the next global aggregation.

Once client m finishes its A local updates in round τ, it has a local model ω_m^τ To protect privacy, it adds Gaussian noise:

$$\tilde{\omega}_m^\tau = \omega_m^\tau + n_m^\tau \quad ; \quad n_m^\tau \sim \mathcal{N}(0, (\sigma_{DP}^\tau)^2 I) \quad \text{--- (8)}$$

- $\tilde{\omega}_m^\tau$: noisey model which will be sent to server
- n_m^τ: random noise vector

Global Aggregation:

At the end of each round τ, server has received $\tilde{\omega}_m^\tau$ from each selected client m. It forms the new global model $\tilde{\omega}^{\tau+1}$ by weighted average:

$$\tilde{\omega}^{\tau+1} = \tilde{\omega}^\tau + \sum_{m=1}^{M} \frac{|D_m^\tau|}{\sum_{m'} |D_{m'}^\tau|} \cdot (\tilde{\omega}^{\tau+1} - \tilde{\omega}_m^\tau) \quad \text{--- (99)}$$

since each clients local update $\tilde{\omega}_m^{\tau+1} - \hat{\omega}_m^\tau$ is roughly $-\eta \sum_{j=1}^{A} \nabla F_m(\tilde{\omega}_m^{\tau+j})$

$$\therefore \quad \tilde{\omega}^{\tau+1} = \tilde{\omega}^\tau - \sum_{m=1}^{M} \frac{|D_m^\tau|}{\sum_{m'} |D_{m'}^\tau|} \left(\eta \sum_{j=1}^{A} \nabla F_m(\tilde{\omega}_m^{\tau+j}) \right) \quad \text{--- (9b)}$$

↳ Each new global model is the old model minus a weighted sum of all clients gradients.

$\tilde{w}^{\tau+1}$ is used to initialize the $(\tau+1)$ FL round. Where weighted averaging helps capture the local data sizes @ clients & guarantee unbiased aggregation.

• Model Update : The FL server informs all clients of $\tilde{a}^{\tau+1}$ there, clients resume model training.

[D] Computation Model

computing power of a client = number of FLOPs (Floating point ops).

let u = no. of FLOPs req. to train mini batch of data.

∴ for client m;

$$u_m = uABm$$

A = no. of local training iterations between two systematic aggregation

Bm = size of mini-Batch

∴ The local trainning delay .... of client m is...

$$T_{m,c} = \frac{u_m}{f_m \beta_m} = \frac{uABm}{f_m \beta_m} \quad \text{——— (10)}$$

f_m = CPU freq ; β_m = no. of FLOPs that a CPU cycle can execute at a client.

→ Since, trainning is performed within sensing period of time ...

$$T_{m,c} \leq a_m^{\tau} T_{sl}$$

$$\therefore \frac{uABm}{f_m \beta_m} \leq C_s (D_m^{\tau} - D_m^{\tau-1}) \quad \text{——— (11)}$$