HLA part II

Architecture --> not a software
The software part is RTI

Example NAVY :
Simulator of simple ship (program In c++)

Simulator of multiple ships --> more situations to consider like collapse or COLLISIONS between navi --> all the navi on the same machine during simulation (navi rappresent federates)

Another difference is that I manage TIME

Free to define new Federate with my own attribute in my simulation, without define it in the external programme.

Execution of Federation is to let join to hla all federations and assign an identifiers to each member

issues:
Not sense connect more federations together

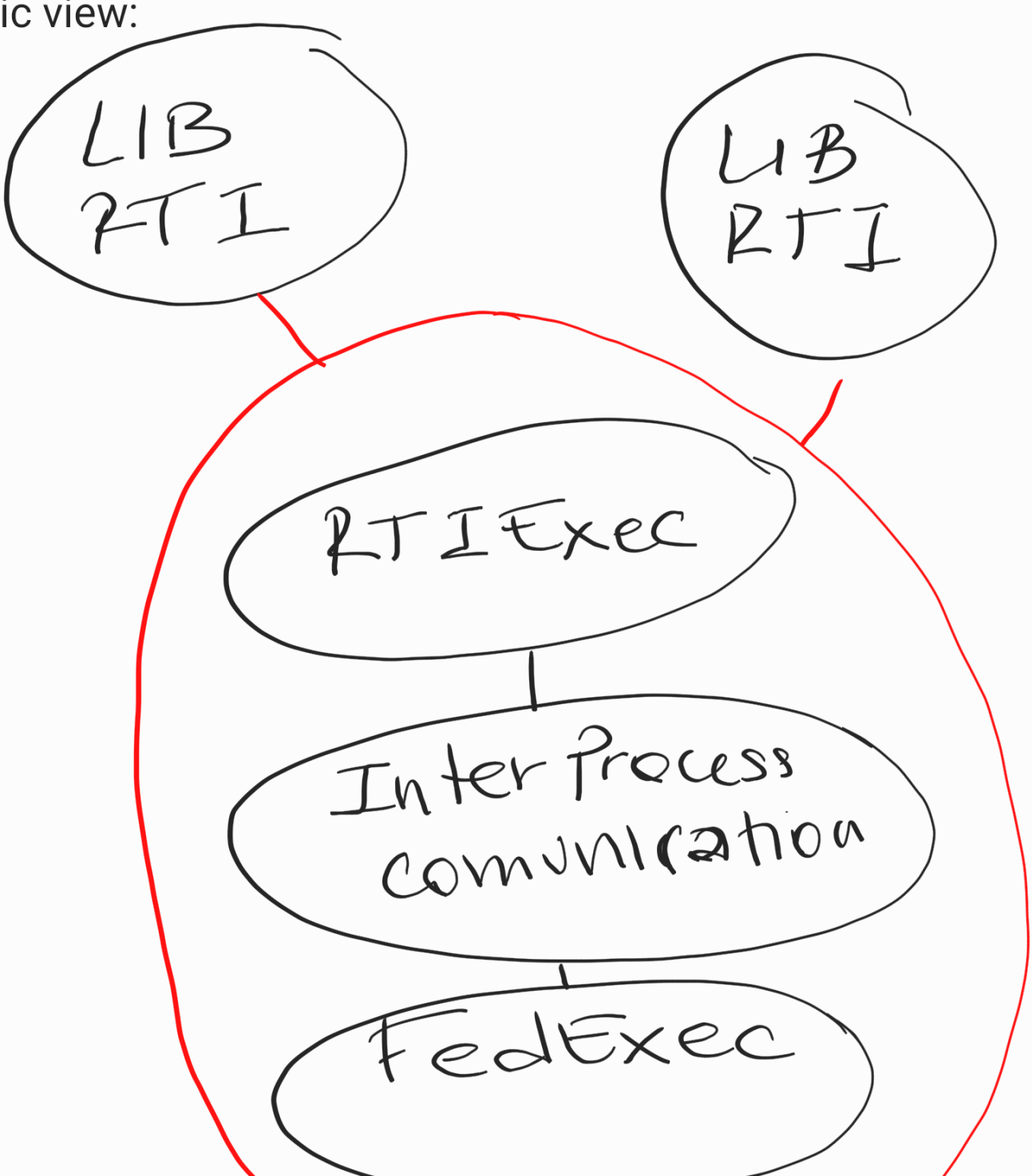
Start rti--> start yatch Federation-->

RTI LIBRARY : implemented usually in c++

Federation Code
Federation component:

- Logic
- View

Logic view:

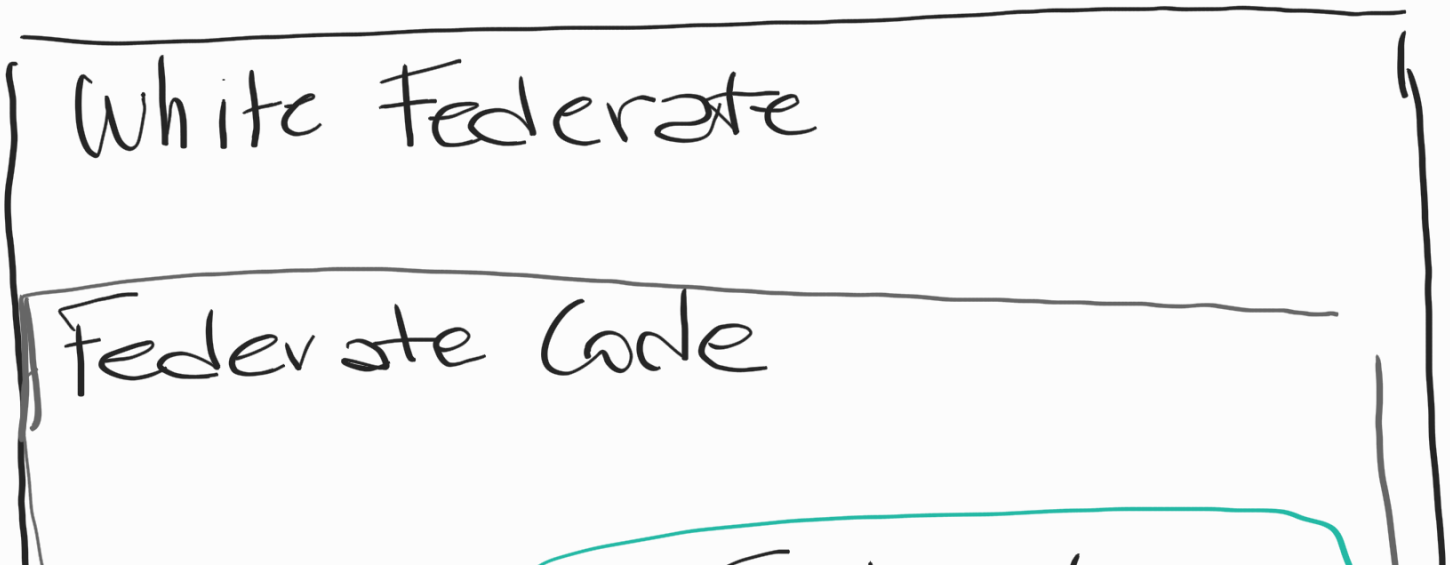


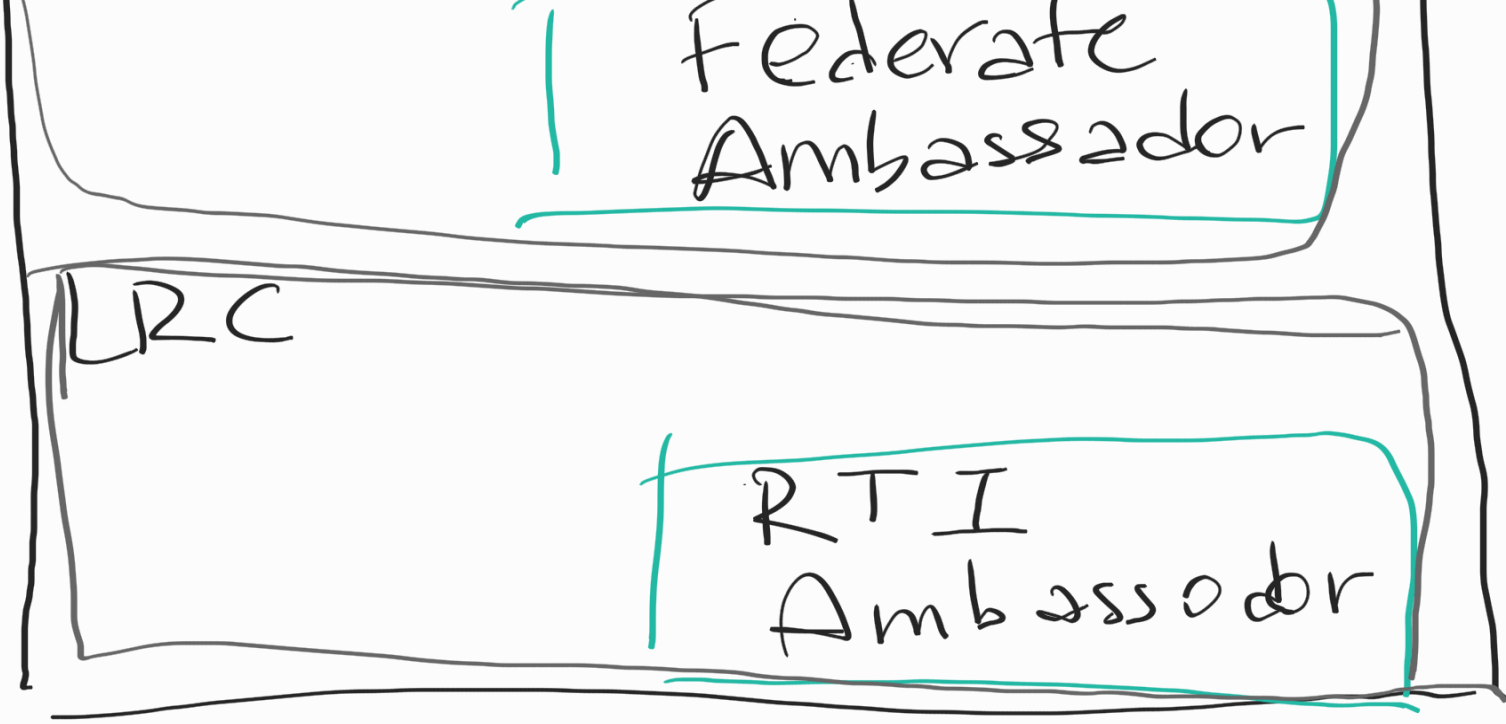
Each federate have a library RTI.
A process for each Federation.
FEDEXEC: goal is to allow user to join federation. Assign an identifiers to each attributes of an instance or to an instance of an object. When subscribe to an objects in reality give an attributes go handle the object. Facilities exchange among federates.

RTIEXEC: manage multiple federations creation and destruction. In this network we can run a Federation but also run another federation on a different simulation, or another federation on a different machine but same name of the first (like open 2 program on the same page, PowerPoint and word for example).
Global Process run on unique platform.

FEDERATE COMPONENTS

How to compound a federate





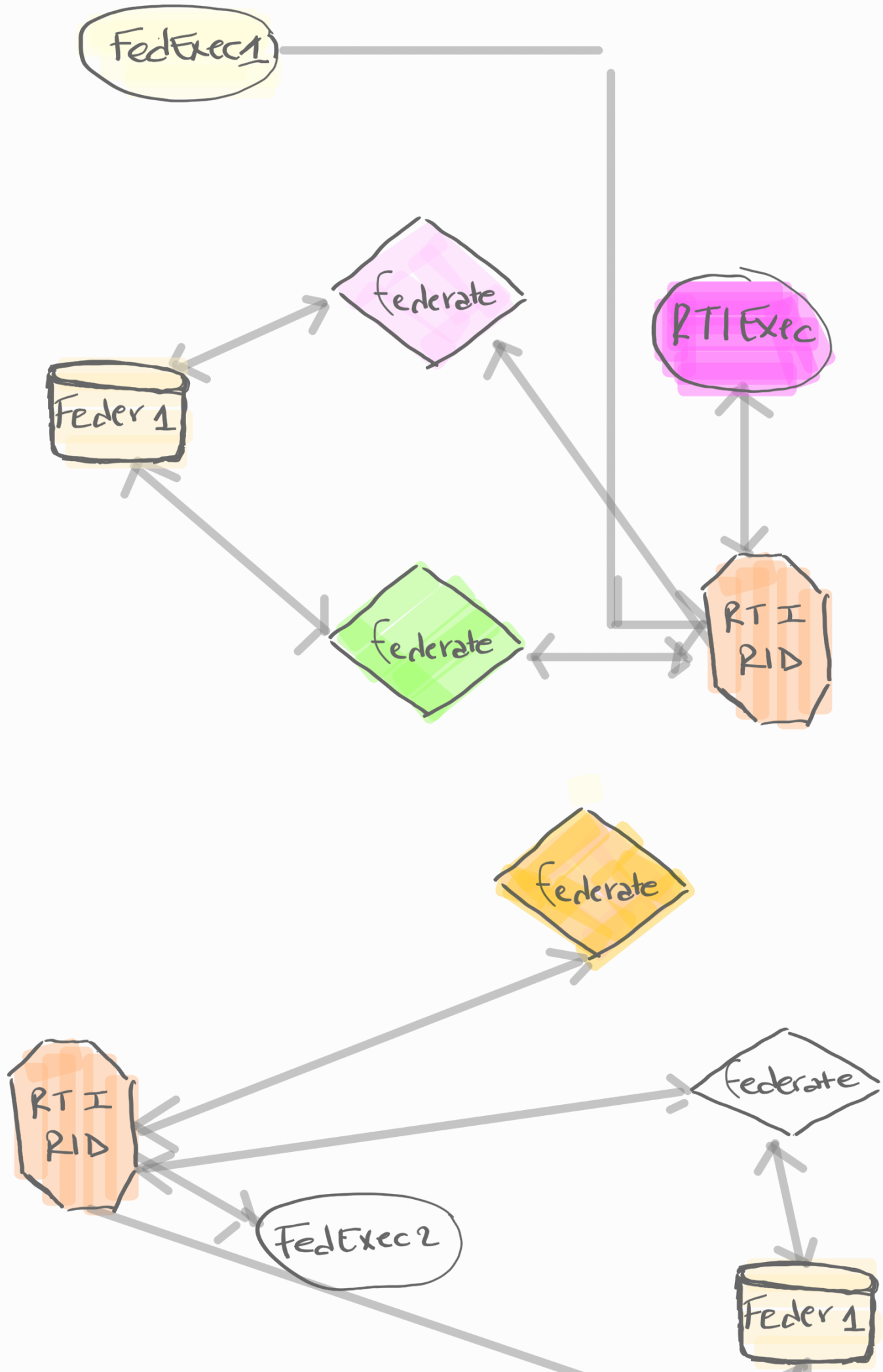
Local RTI Component: LRC will run RTI ambassador; Garantisce la chiamata tra RTI e ambassador. Garantisce is processo esterno come specificato in IFSpec.
Federate Code run the federates Ambassador and allows the internal processing.

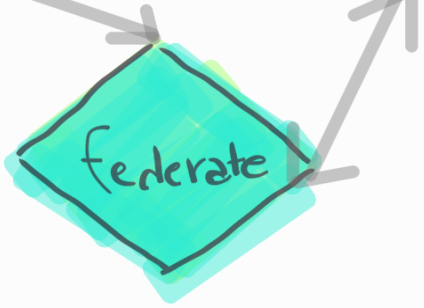
The "Ambassador" :

- Federate one: RTI communicate with federate requesting Federate ambassador function member
- RTI one: federate requires RTI services request RTIAmbassador member function.

Responsibility of federate code: federate code implement/ run the ambassador --> create the Federation Object Variation, that call RTI ambassador --> that Update RTI Object Variation --> RTI have all the object inside and guarantee federate ambassador functionality.

Example of FEDEXEC Analysis (diagramma workflow)


FedExec1
federate
RTIExec
Feder 1
RTI
RID
federate
federate
RTI
RID
federate
FedExec2
Feder 1



PROCESS STEPS

User start RTI --> RTIExec --> user became federated--> federated start federation --> FedExec --> other federates join federation in FedExec phase

If there is not a syncronization timer, when other federate join their are not syncronized refer to a start point...but we don' t care.

FEDEXEC ANALYSIS

Base flow:
Federation creation --> statements manage--> management of temporal statements --> support services --> objects management--> time management --> Federation destruction

Exeception: time is expired, time is not valid, timeadvanceRequest still pending, federate is not exec member, rescue in progress, restore in progress, RTI internal error

The "Tick": some kind of refresh frequency that define how often the federation ambassador check some of the situation. By changing tick I can increase control loop and same time I can speed up simulation if I have latency issues.

MESSAGES HLA

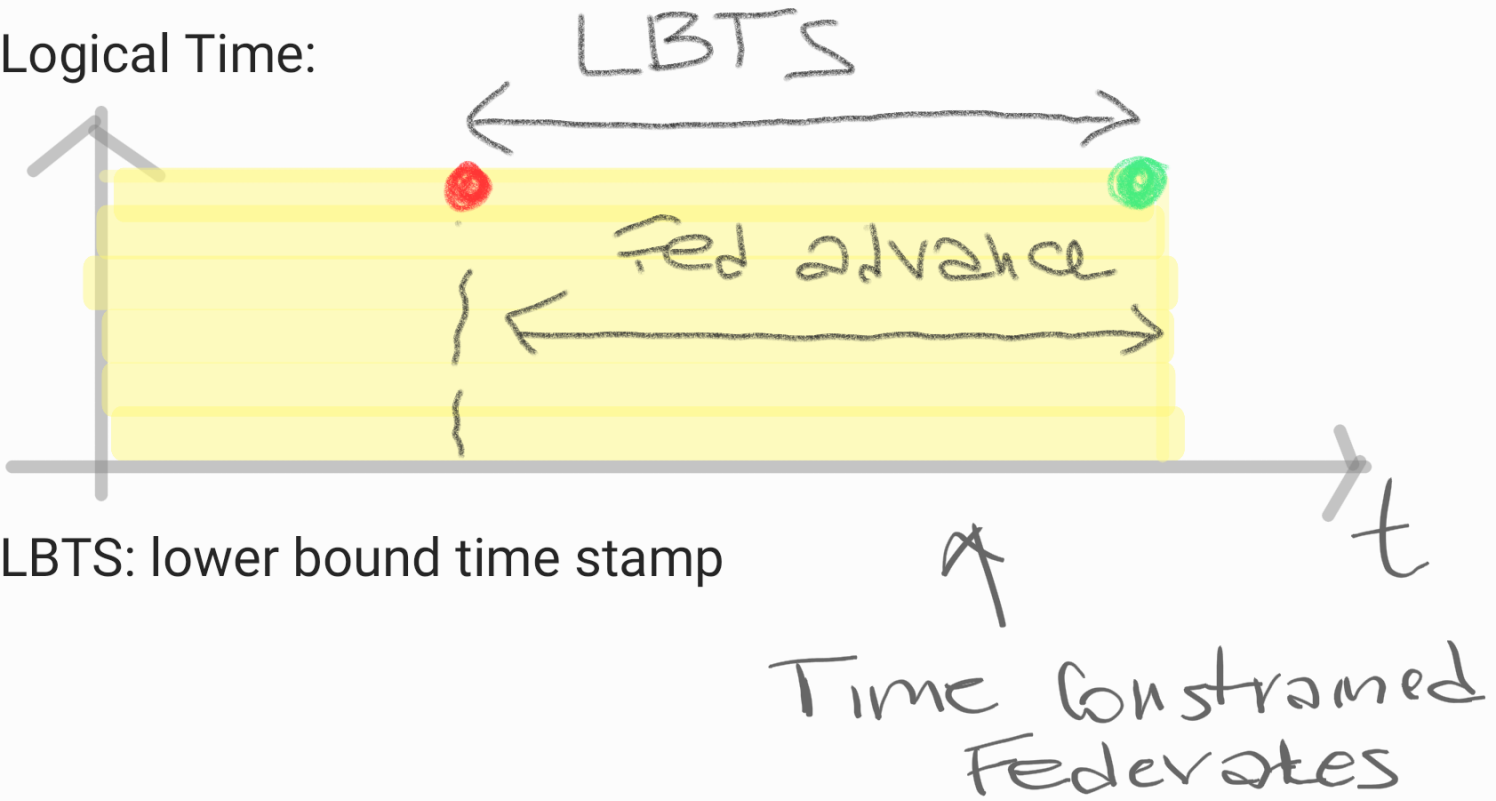
Updates : send and receive new attributes values for each time step ending
Interaction: send and receive parameters describing events when happens. Interaction are temporary.
Subscription and pubblication: I can subscribe to an object that is publication or I can publish an object. Registration is for instance of objects. For Interaction everything is temporary as Interaction itself.

Two mesage order type:

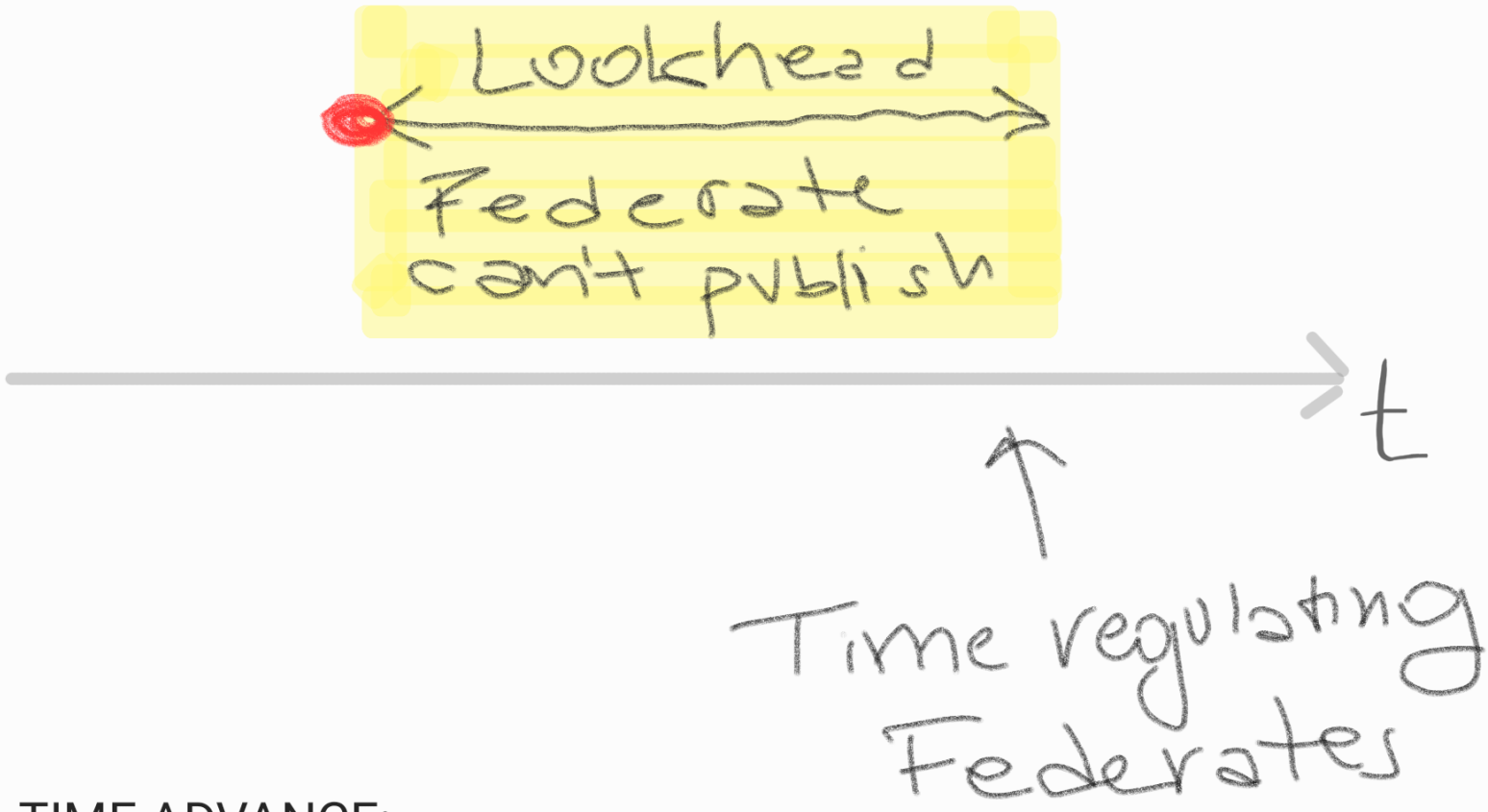
- TSO: time stamp order, message delivered to federate in the indicated order
- RO: receive order, message delivered to federate in the sent order

Time constraints and time regulating influenza l' ordine di ricezione dei messaggi. La federate deve dichiarare questi due parametri per ricevere o inviare TSO.

Logical Time:



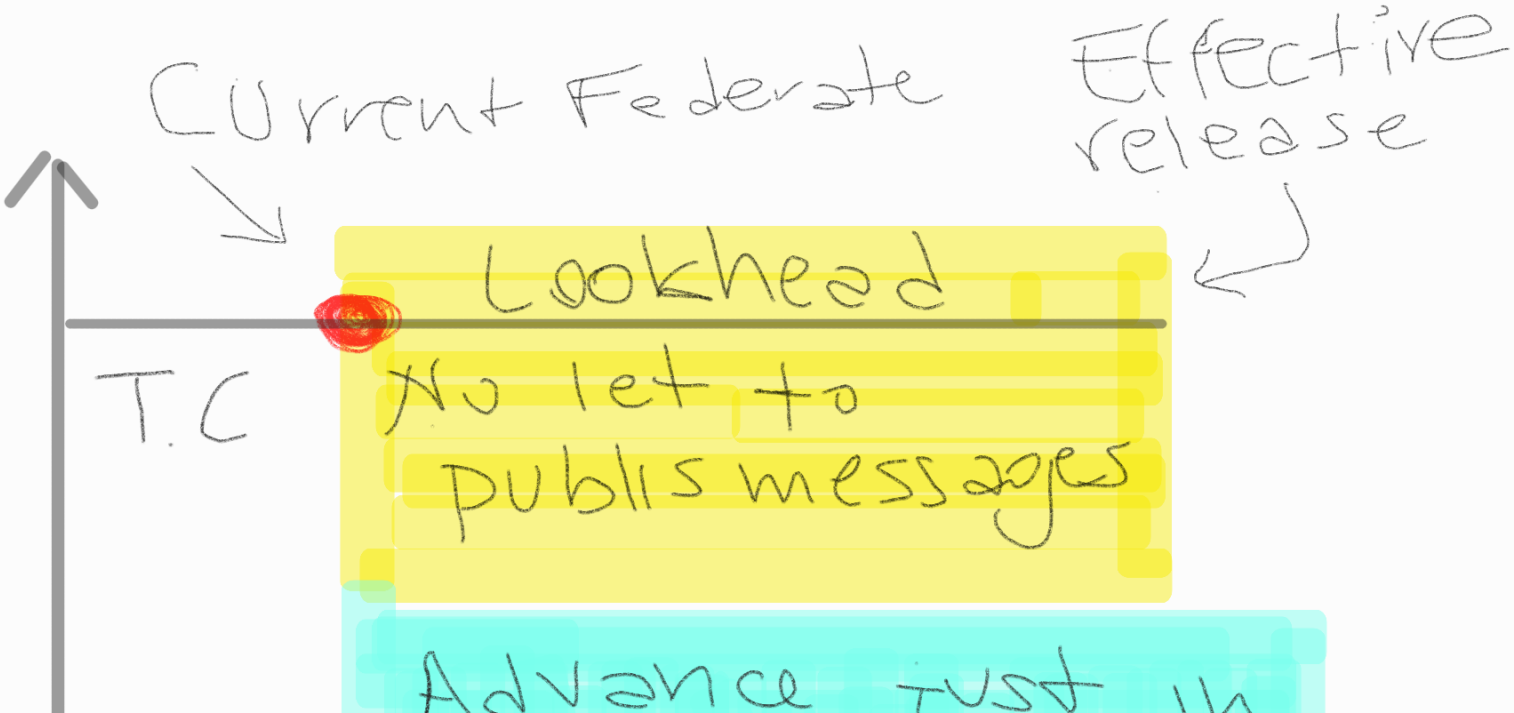
LBTS: lower bound time stamp

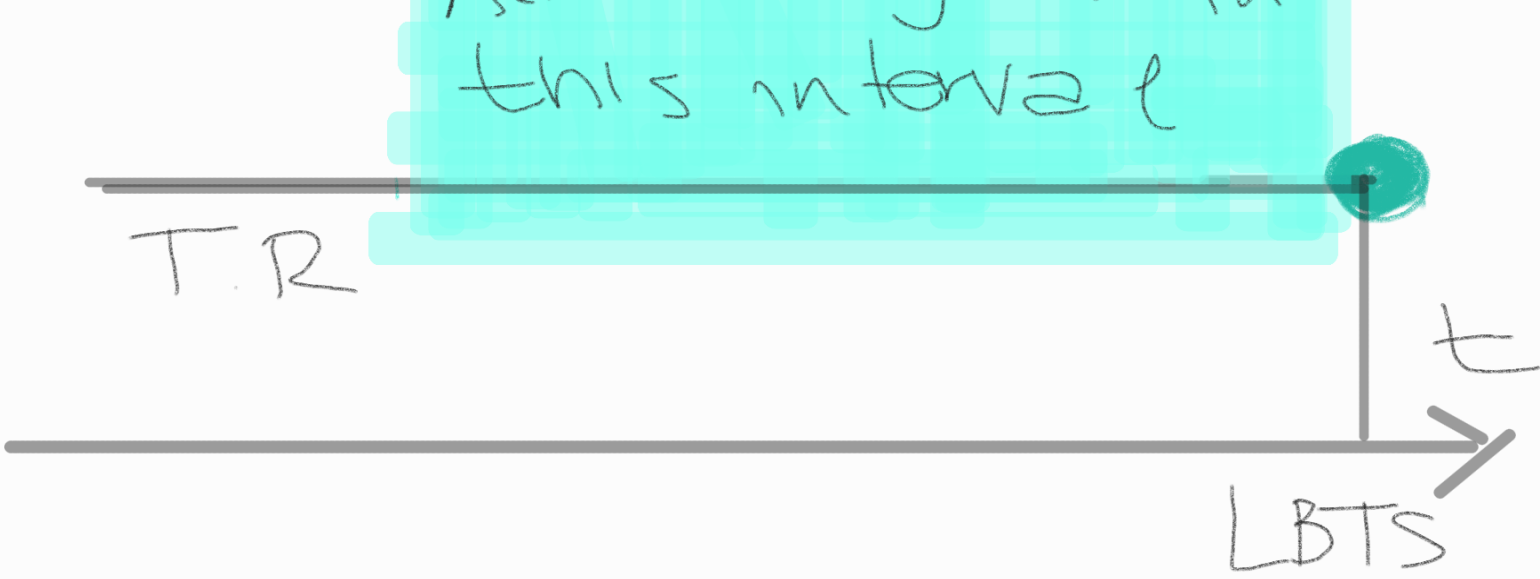


TIME ADVANCE:
In continuos simulation we use Time advance request using RTI service "request time advance" --> time advance Grant is notified

In discrete event simulation --> we have "next event request" and is notified the "next event grant " to go forward.

Federates send and receive TSO data, must be TR and TC





MANAGEMENT OF OBJECTS DECLARTIONS:
Definition what data send and what receive.
Object management refresh and updates--> RTI receive interaction and so the parameters to refresh.
Object class: attributes of federate that are maintained in the time.Venn diagram to define and vidualize what object you want to describe.
P&S of objects: one federates publish attributes another subscribes federates --> all visualized with Venn diagram.
P&S interaction refresh and updates like object management. Interactions are produced as "all or nothing" and it's not possible specify which parameters will be publish in an interaction.

Refresh of an object: to create an object the federate must before publish its class. The federate registers the new instance of the object. To discover this object, other federates must have subscribed to that object class. Those federates then will discover the object instance.

TIME MANAGEMENT
types of Simulator : continuos and discrete

Types of simulation implementations:

- Event driven implementation
- Time stepped implementation
- Real Time implementation
- Scaled real time implementation
- Non real time implementation

Time stepped example: continous model of a discrete process, uses events to communicate existence of countries. Original model built, it's assumed that each federate start at time 0.

Non existence of scheduling del time management. Each federate advances time at its own pace.
Conservative:

- syncronization: Federates advance time only when guaranteed that no past data will be received
- Optimistic synchronization : free to advance logical time, may have a Rollback.
- Active scan : advance time by mutual agreement with other federates.

Logical time restriction:

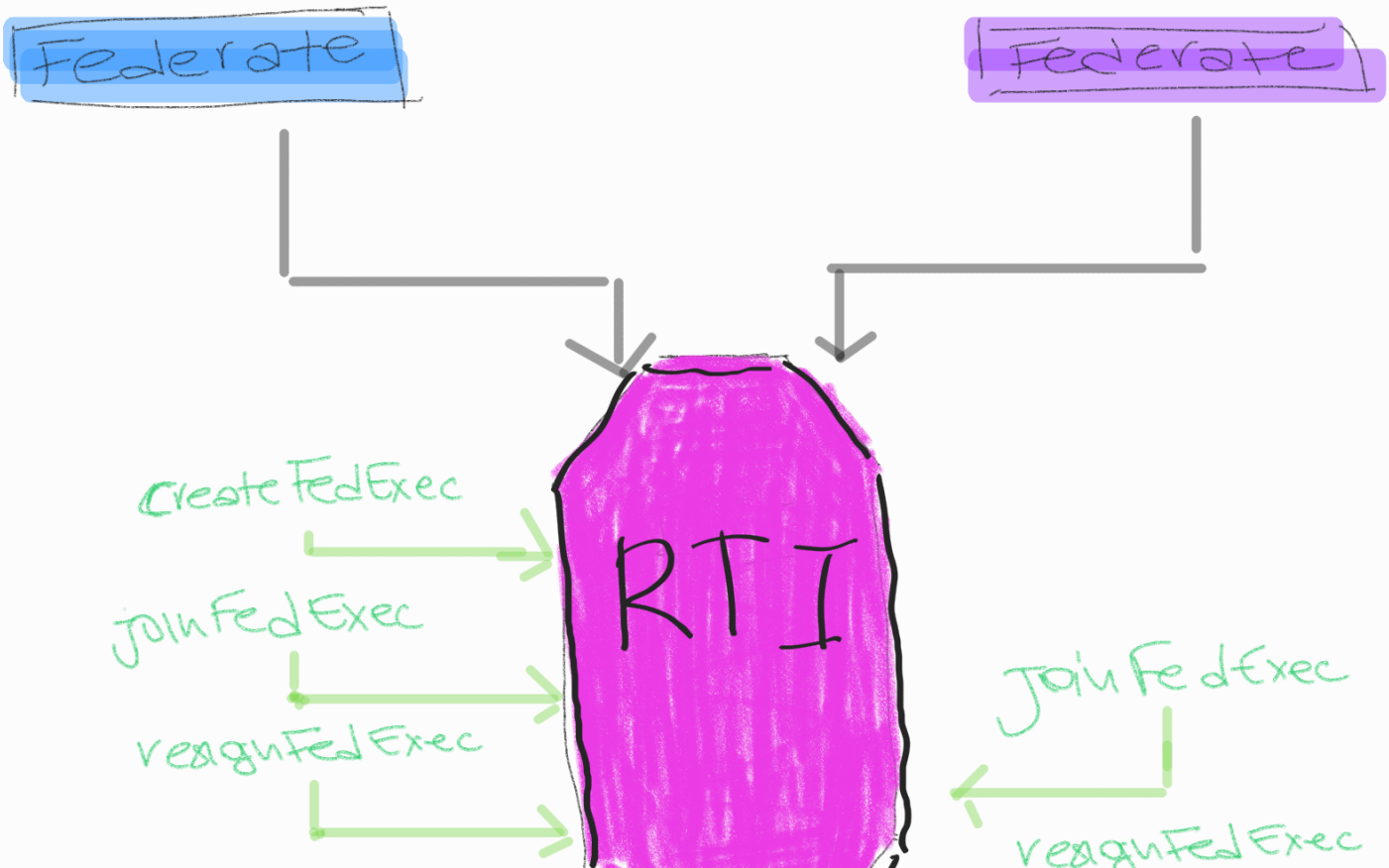
- Initial value
- Value not tied to any Measure units system
- Well ordered
- Always greater or equal than initial time
- The time is effectively discrete
- A special value exists, it' s called Positive Inifinte and it's greate than any other value

Logical time synchronization:
Case1
Update delayed until clock advances to end of time step

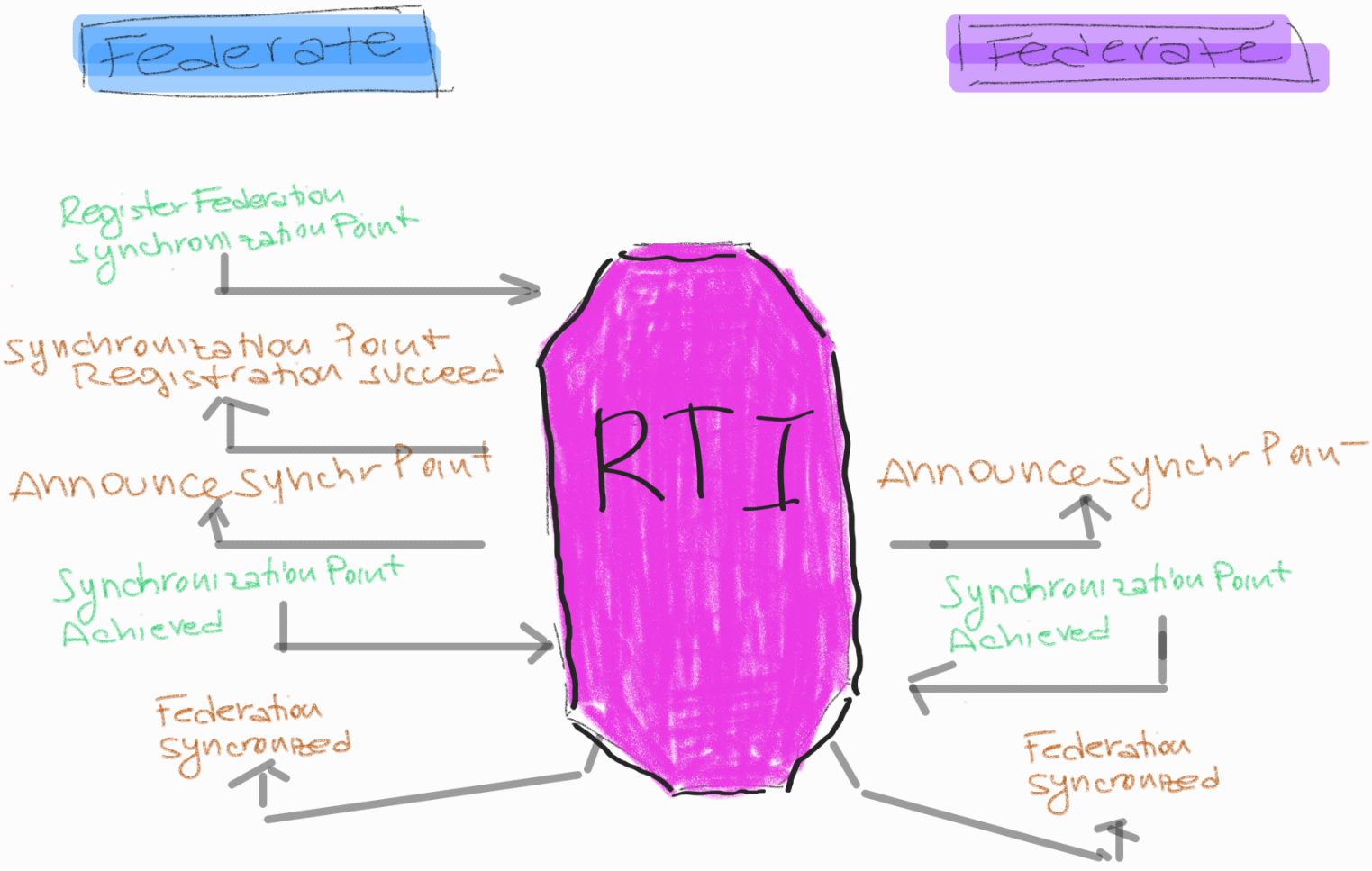
Case2
Update delayed until clock advances to end of time step

Events:
Sending events: Update attributes value, send interaction, delete object instance
Receiving events: reflect attribute values, interaction and object instance.

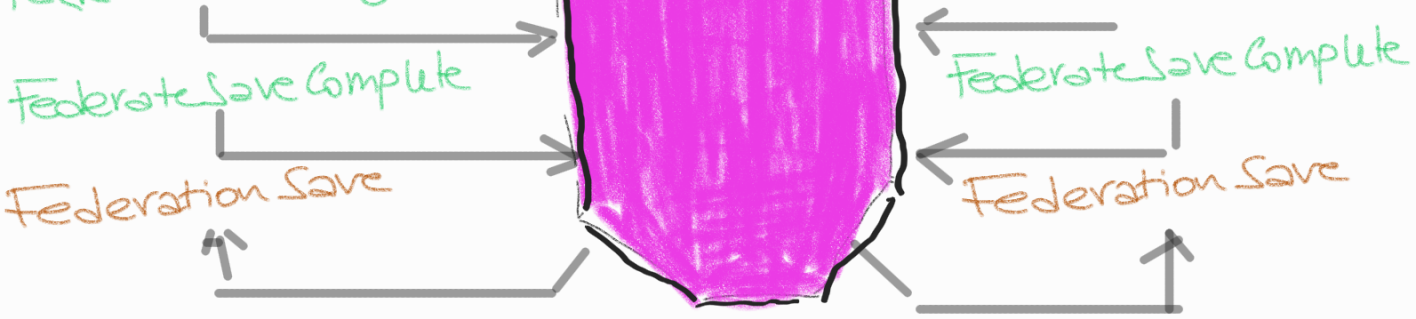
FRDERATION SERVICES MANAGEMENT
Life cycle management



Synchronization management



Save management


FederateSaveComplete
FederationSave
FederateSaveComplete
FederationSave