INTRODUCTION TO UML 2

Elements of Software Construction
Week 2 & 3

Exercise 0 (12 minutes)

- Read the system requirement in the file telenetwork.pdf

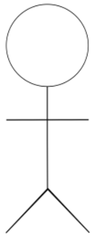
Use Case Diagrams

- A diagram that shows a set of use cases and actors and their relationships.

- Use case diagrams represent system functionality, the requirements of the system from the user’s respective.

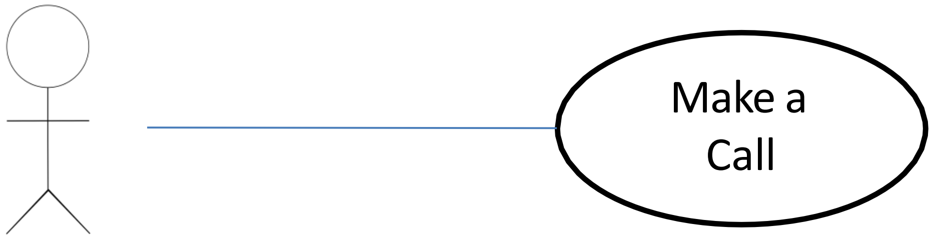
Actors

Actors are the people or systems that provide or receive information from the system; they are among the stakeholders of a system, which could be human beings, other systems, timers and clocks or hardware devices.



Actors that stimulate the system and are the initiators of events are called primary actors (active). Actors that are only receive stimuli from the system are called secondary actors (passive).

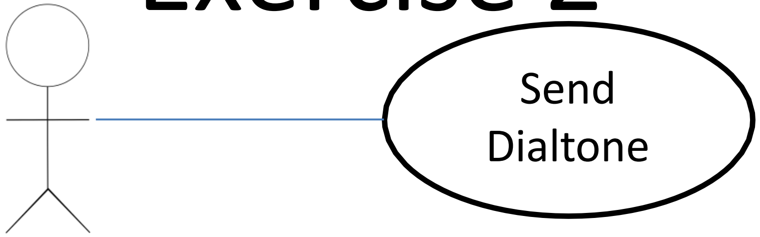
Exercise 1



Who are the primary and secondary actors?

Exercise 1:
primary actor: subscriber or caller.
secondary actor: local switch.
Exercise 2:
primary actor: local switch. secondary actor: subscriber or caller.

Exercise 2



Who are the primary and secondary actors?

Who are the Actors?

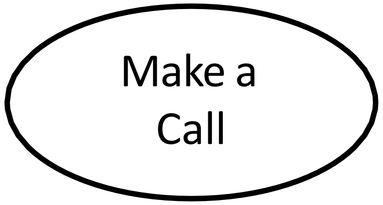
Actors

- Who/what will be interested in the system?
  - Any human interested in making telephone calls
- Who/what will want to change the data in the system?
  - The switch
- Who/what will want to interface with the system?
  - The telephones, The switch
- Who/what will want information from the system?
  - The telephone network operators

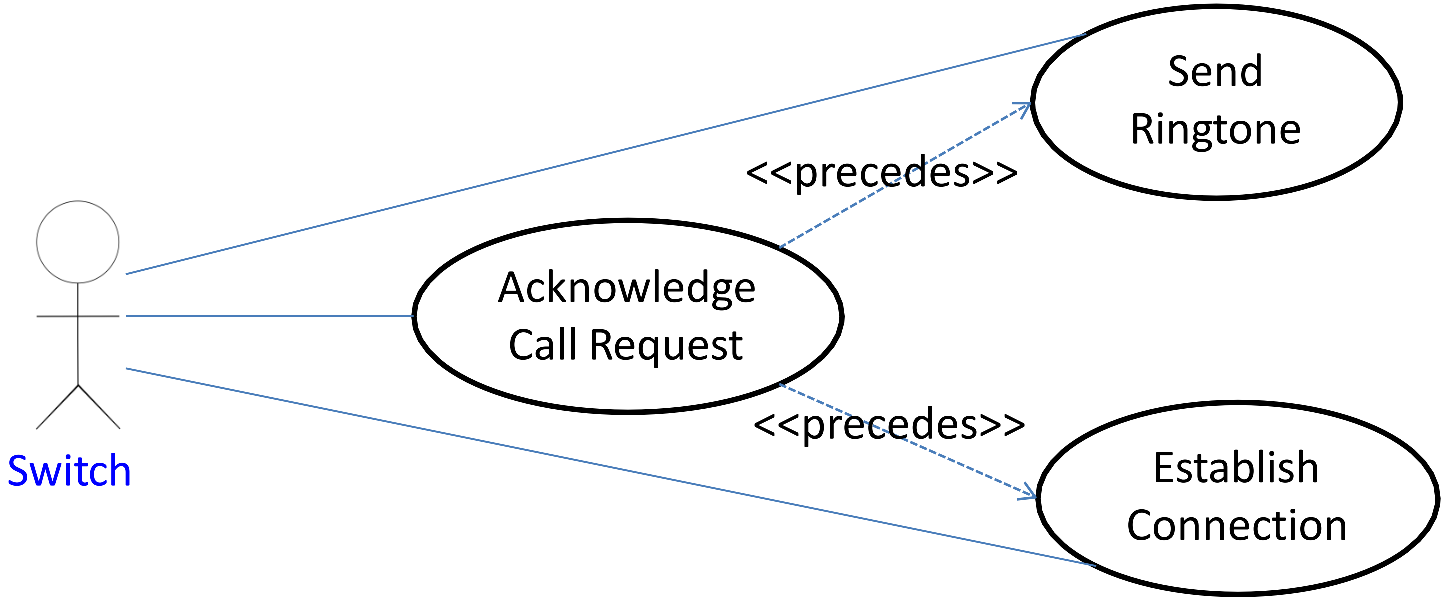
Use Case in Use Case Diagrams

Use Case

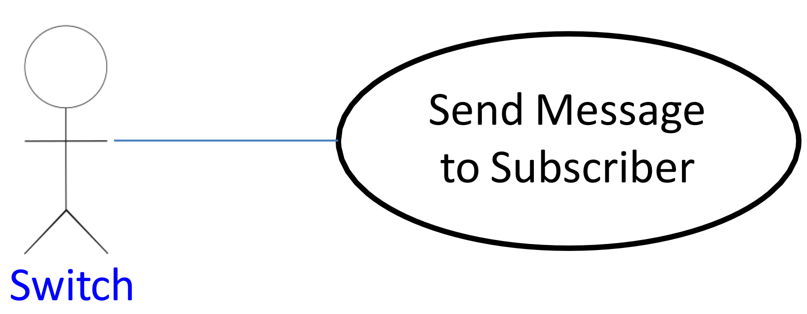
- Is a description of a set of sequences of actions, including variants, that system performs that yields an observable value to an actor.
- Should ideally begin with a verb.

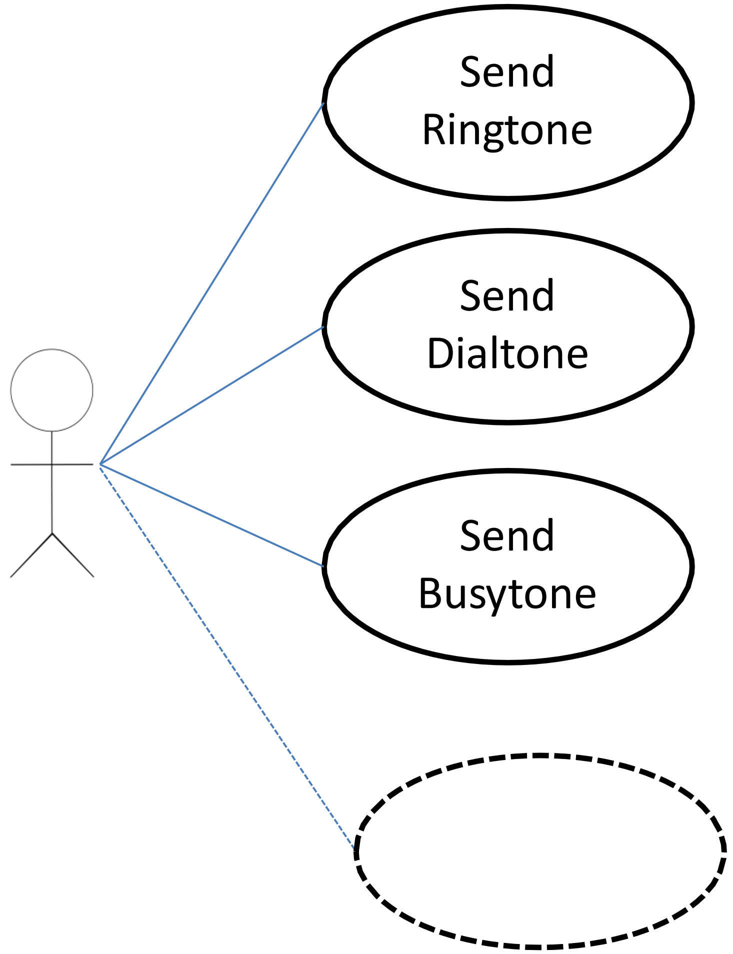


Use Case Diagram: Example



Granularity of Use Cases



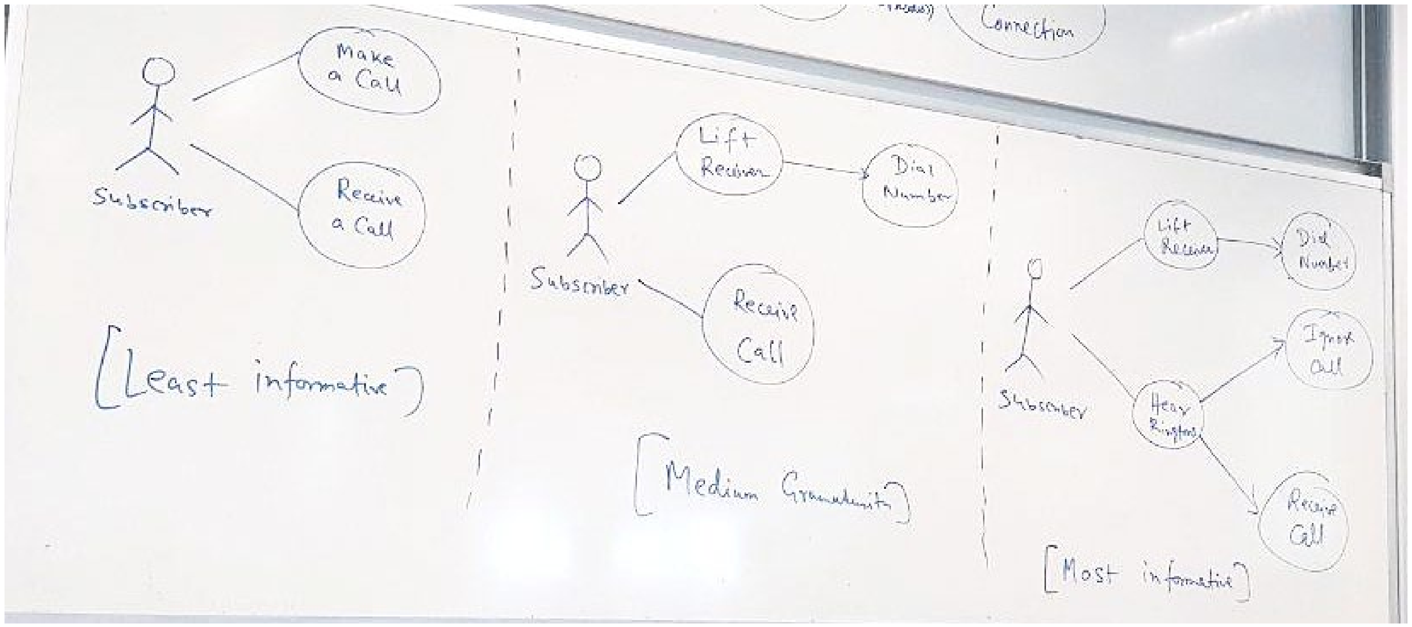


Generally, a use case should embody sufficient levels of granularity without which the use case may not be rendered as useful.

Cohort Exercise 3 (10 minutes)

Draw the use case diagram for the subscribers in Telephone Network System. Pay attention to the granularity of use cases. Draw at least three use cases.

Different levels of granularity



Misuse cases

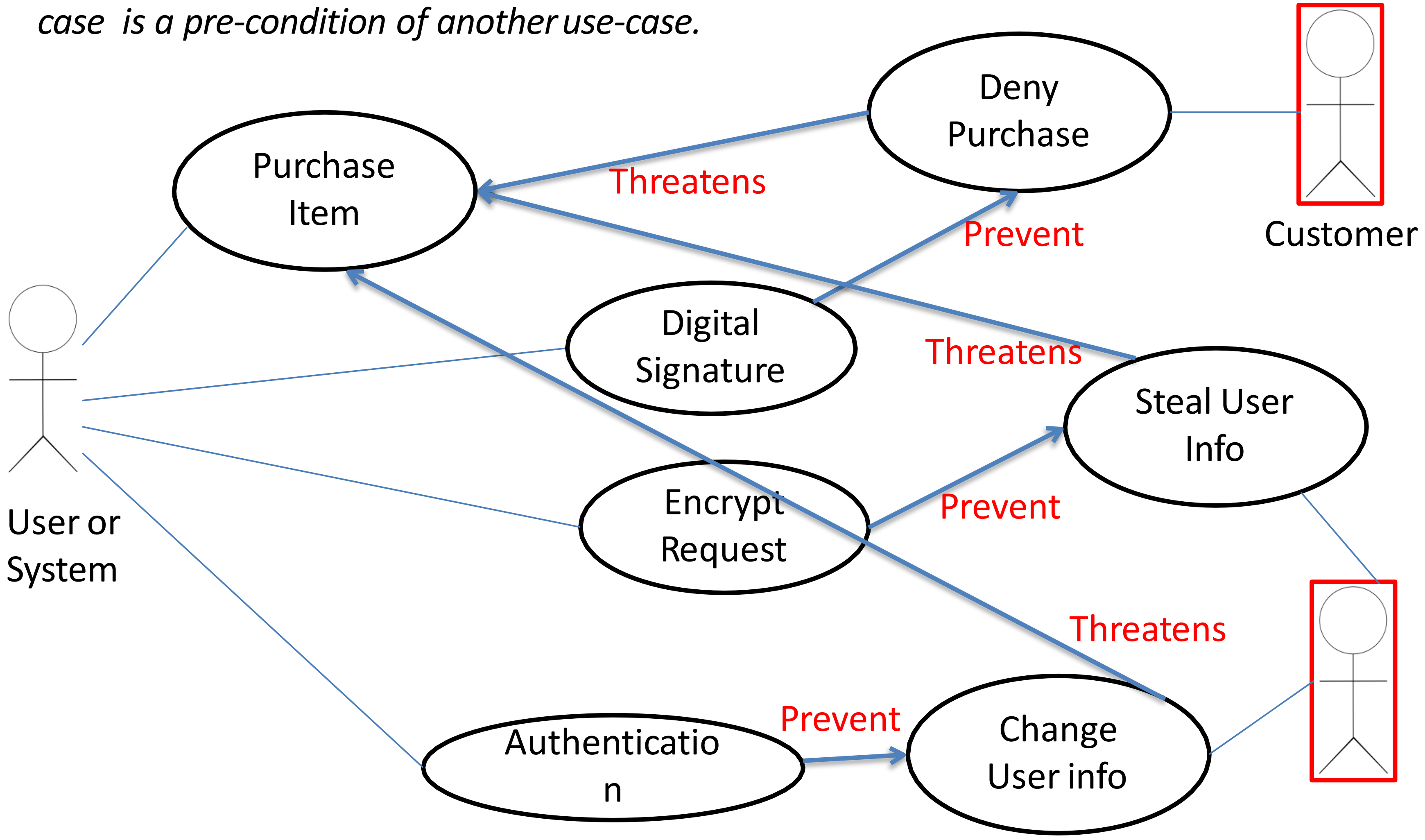
- Use cases only capture how a system is **used**
- Use cases do not consider adversarial scenarios
  - Abusing system functionality
  - Hacking and malfunctioning
  - In general, any non-functional requirement, especially security
- Clients and customers are unlikely to provide misuse cases
  - The onus is on the software engineers

Misuse cases: Example

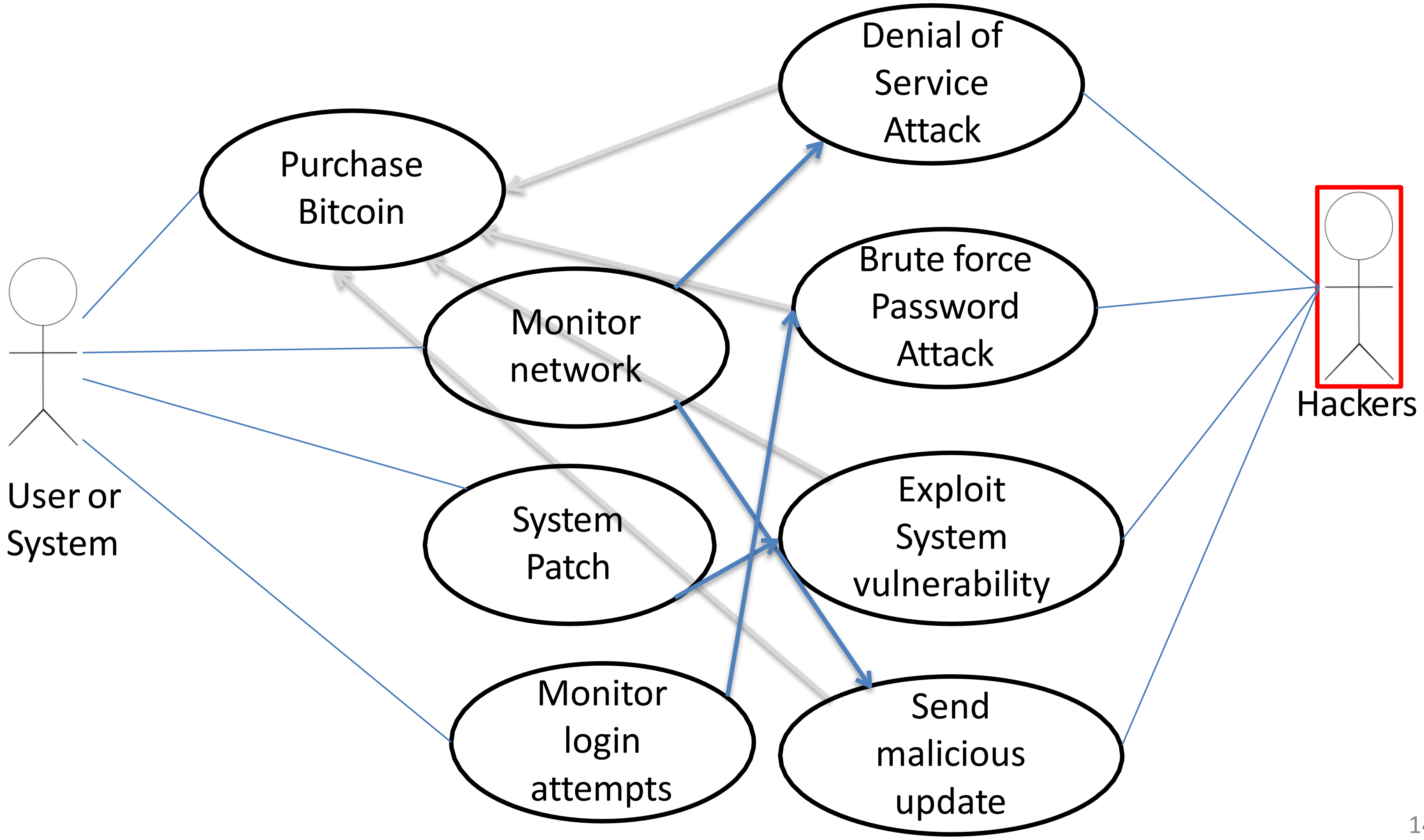
- Consider an e-commerce website where arbitrary users can purchase different items
- The actors can be
  - Customer
  - System

Misuse Case : Online Purchase

The <<precedes>> paradigm captures which use-case is a pre-condition of another use-case.



Misuse Case Diagram: Example

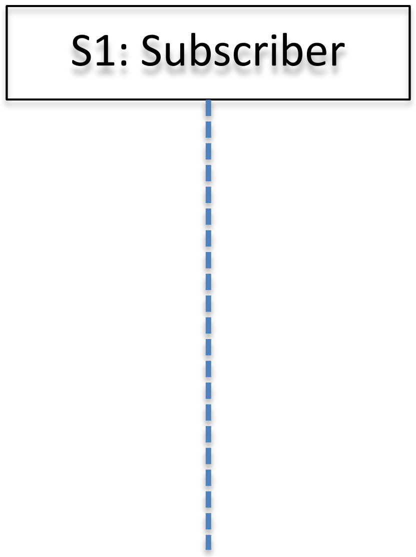


Sequence Diagram

- A **sequence diagram** is an interaction diagram that highlights the order in which the messages are exchanged.
- Sequence diagram complements a use case with details on the workflow of events.

Lifelines

lifeline notation elements are placed across the top of the diagram. Lifelines represent either roles or object instances that participate in the sequence being modeled.

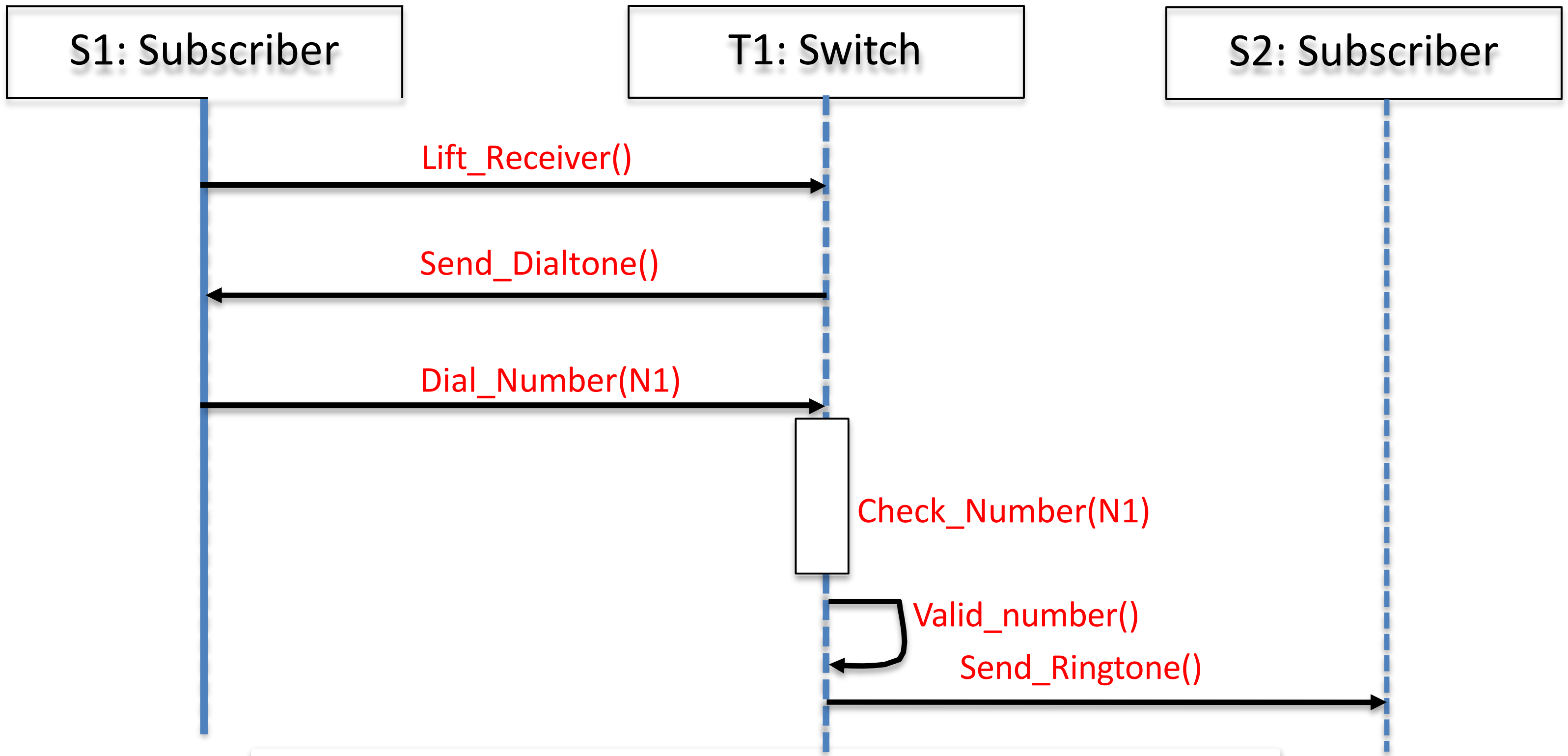


An object named S1 of type Subscriber.

box -> doing some internal computation

Messages

To show an object (i.e., lifeline) sending a message to another object, you draw an arrow to the receiving object.

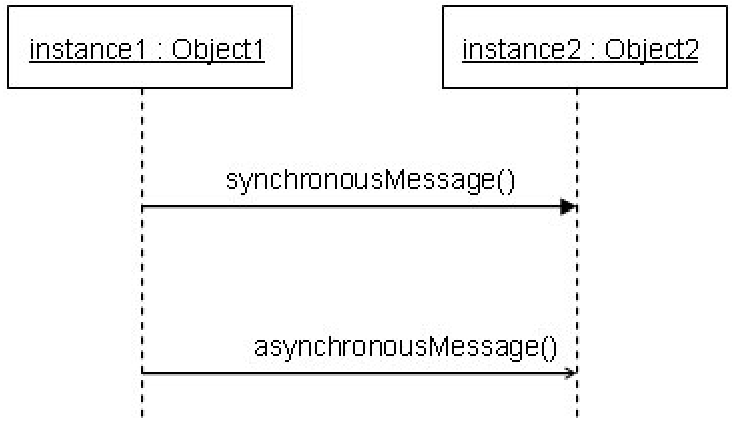


Optional Messages are put dotted.

Ask the difference between synchronous and asynchronous call? Say that caller has to wait when a synchronous signal is sent.

Arrows and Orders

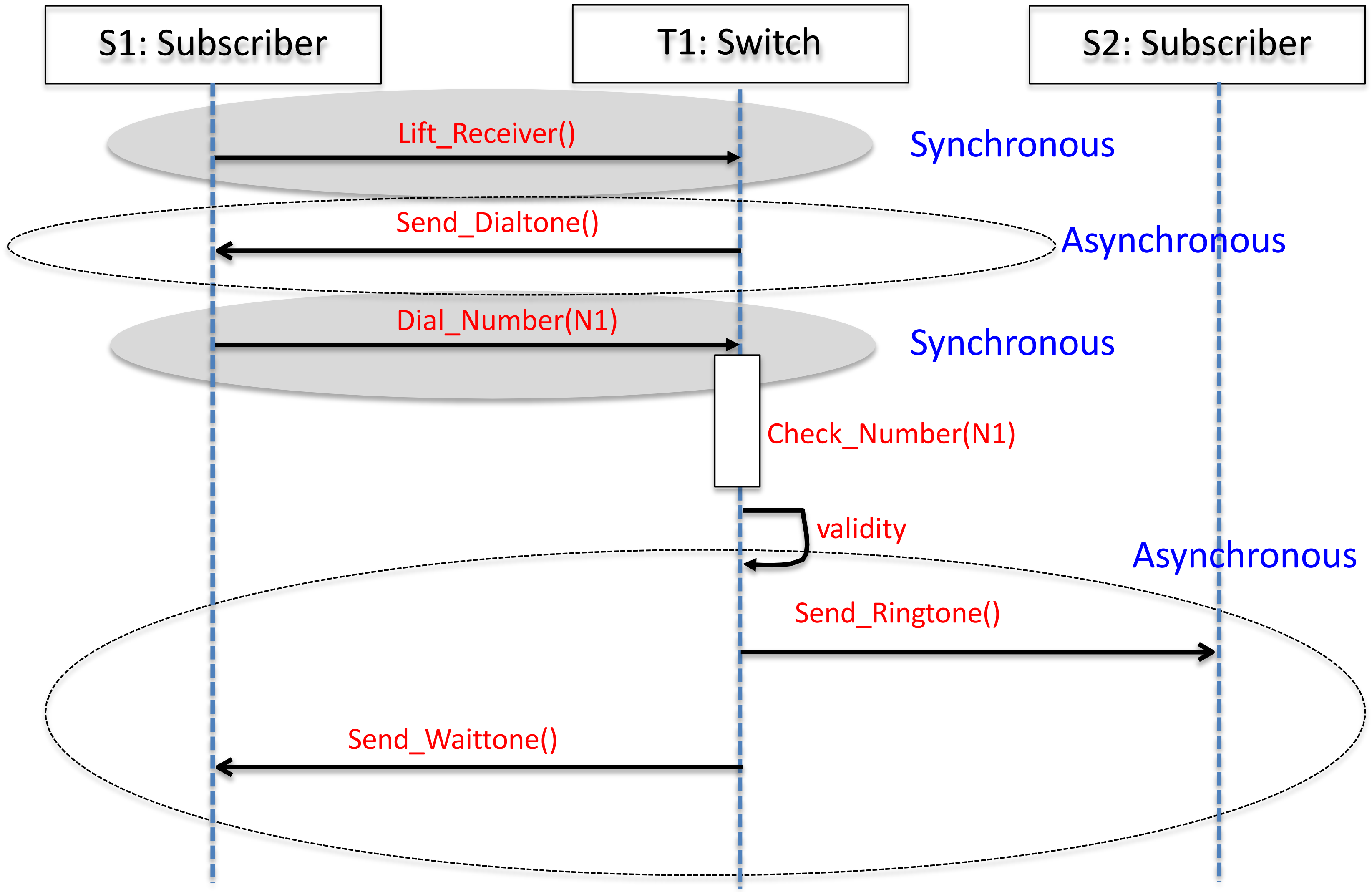
- A solid arrowhead if a synchronous call operation
- A stick arrowhead if an asynchronous signal
- The order of the messages is defined two rules:
  - On the same lifeline, a higher message precedes a lower message
  - Message sending precedes message receiving



synchronous: system is blocked until message is received (cannot have another message until message is received)

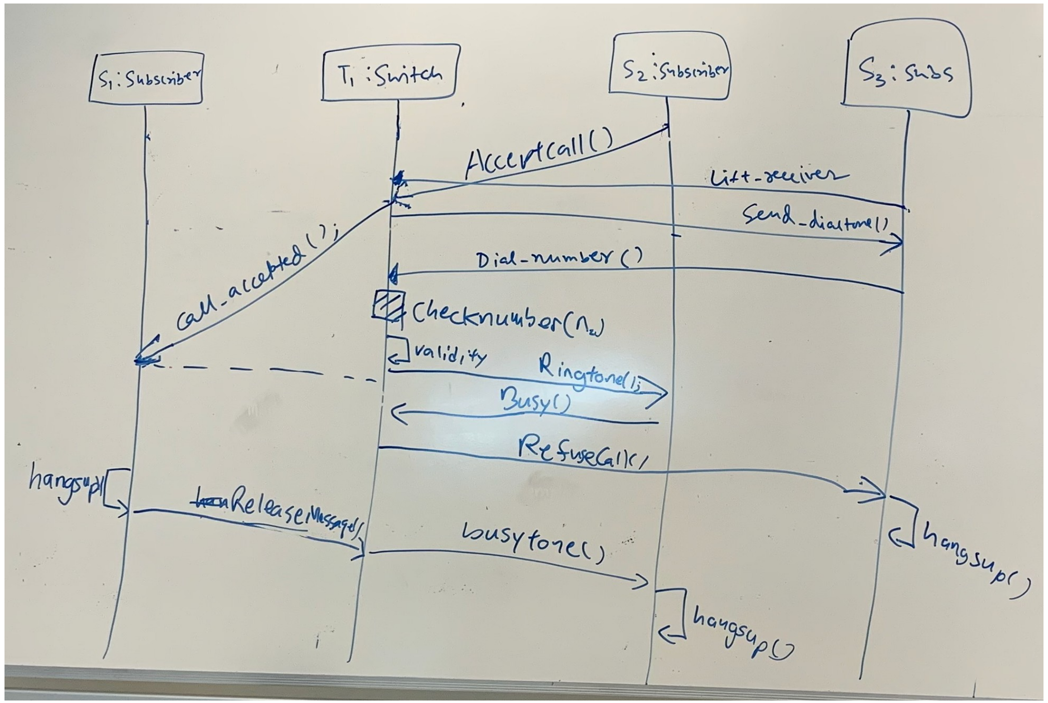
function calls are example of sync async example: multi-threaded programming

Synch and Asynch Messages

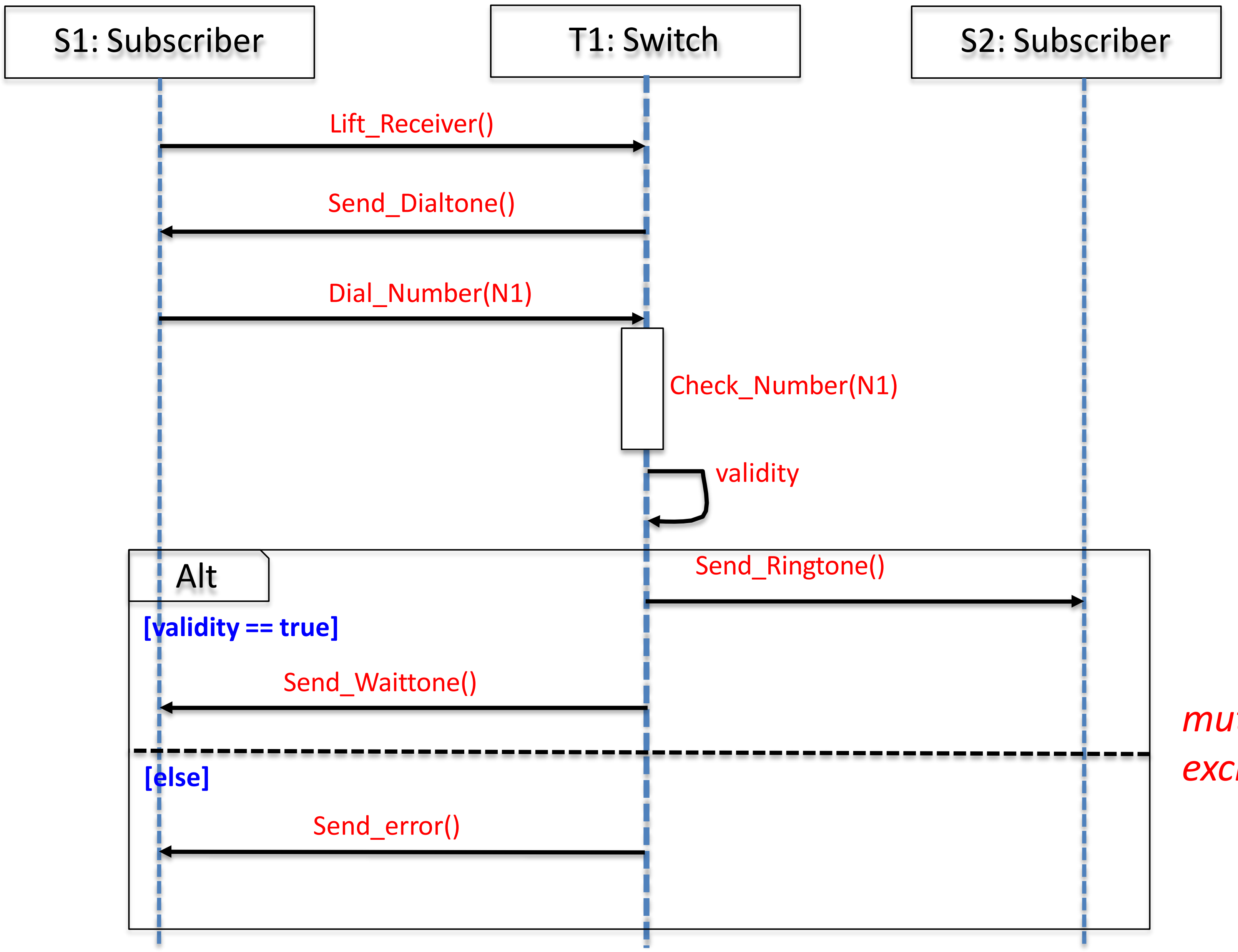


Alternatives

- Assume that a subscriber may dial a valid or invalid phone number. The messages exchanged for a valid or invalid phone number might be different.



Alternatives

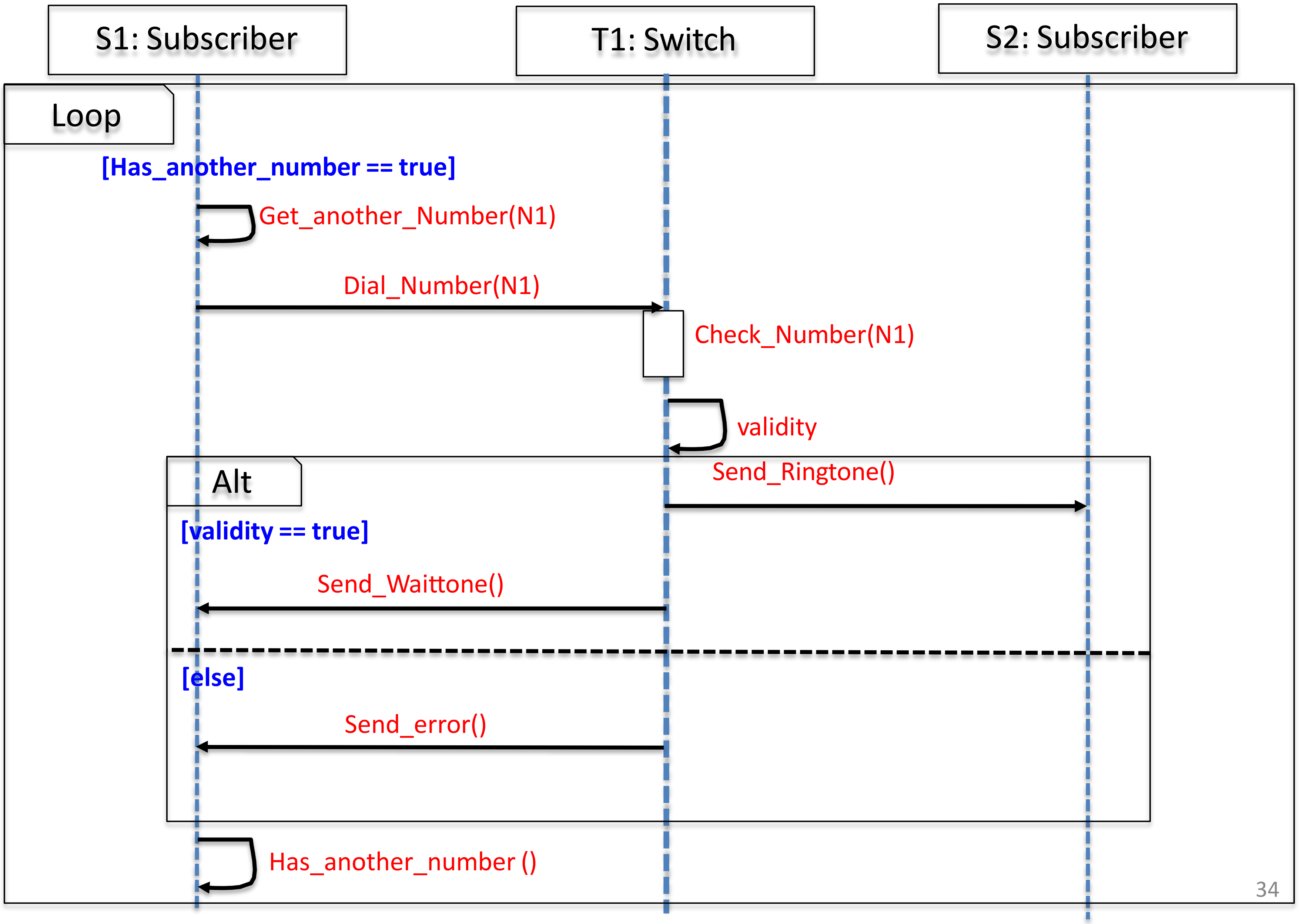


mutually exclusive

Loops

- Assume that a subscriber has a record of phone numbers (each phone number can be valid or invalid). We need a sequence diagram where the subscriber make a call to each phone number in her record.

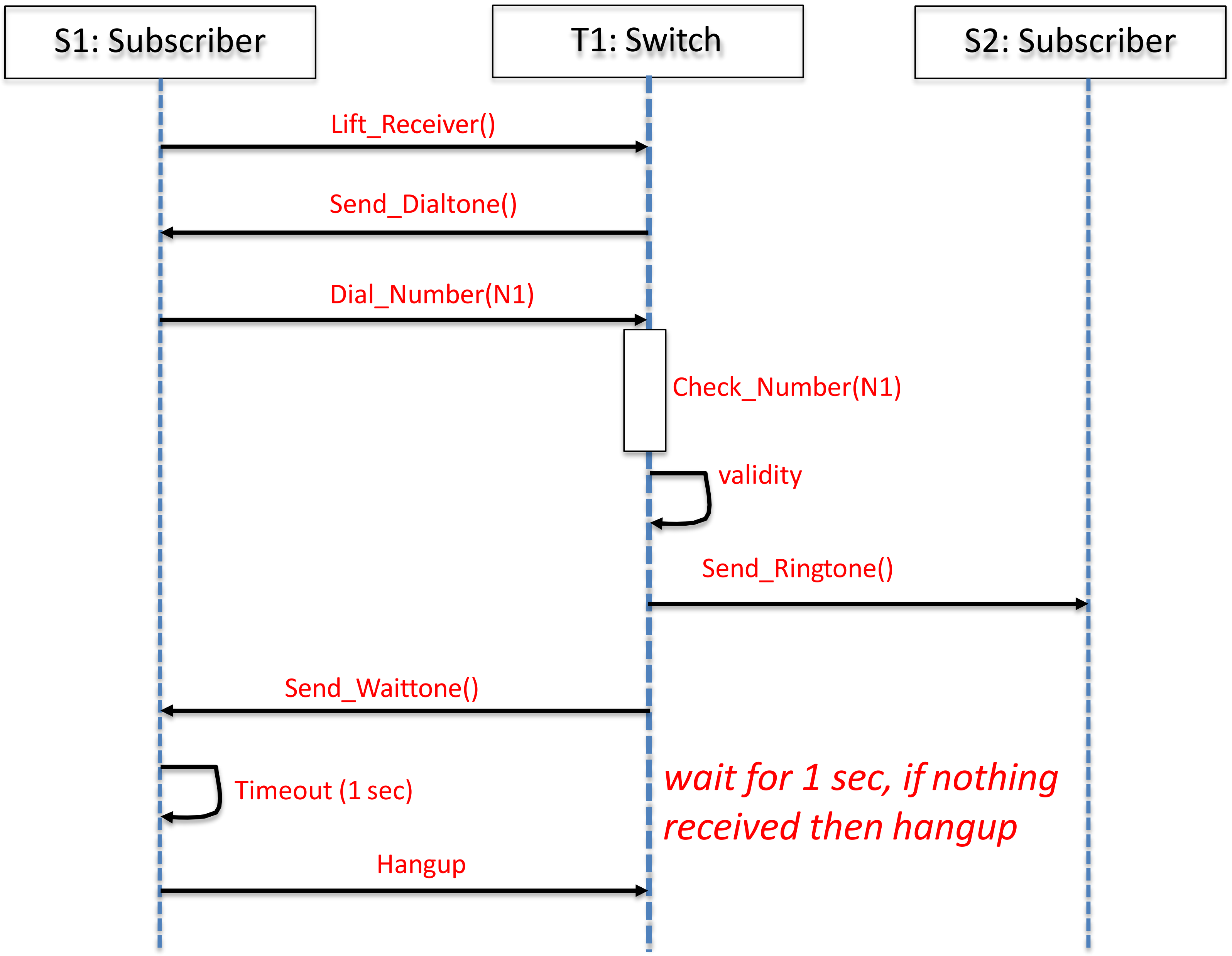
Loops


S1: Subscriber
T1: Switch
S2: Subscriber
Loop
[Has_another_number == true]
Get_another_Number(N1)
Dial_Number(N1)
Check_Number(N1)
validity
Alt
Send_Ringtone()
[validity == true]
Send_Waittone()
[else]
Send_error()
Has_another_number ()

Timeout

- Timeout messages might be incorporated if certain messages occur only after a specific time interval is elapsed.

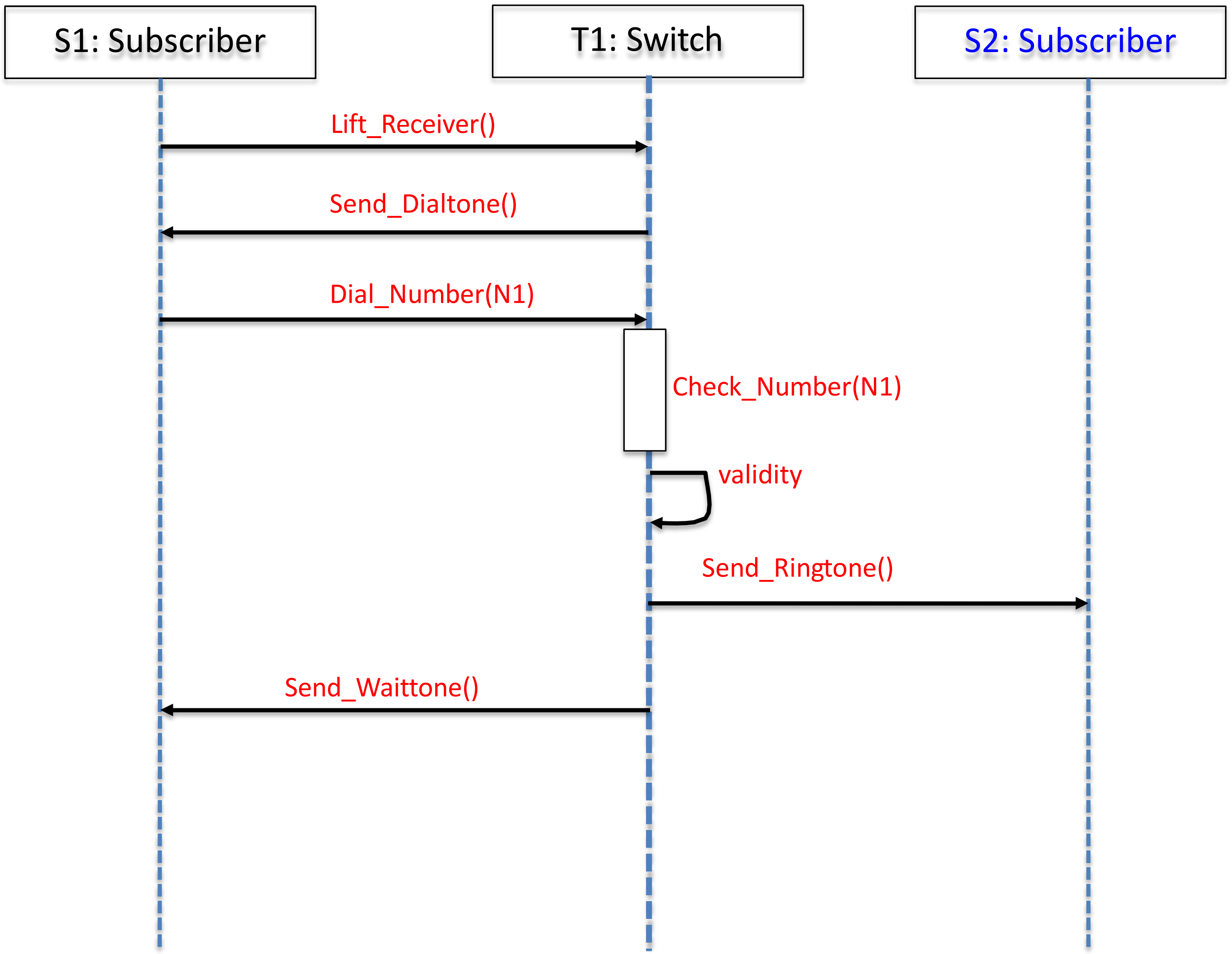
Timeout


S1: Subscriber
T1: Switch
S2: Subscriber
Lift_Receiver()
Send_Dialtone()
Dial_Number(N1)
Check_Number(N1)
validity
Send_Ringtone()
Send_Waittone()
Timeout (1 sec)
wait for 1 sec, if nothing received then hangup
Hangup

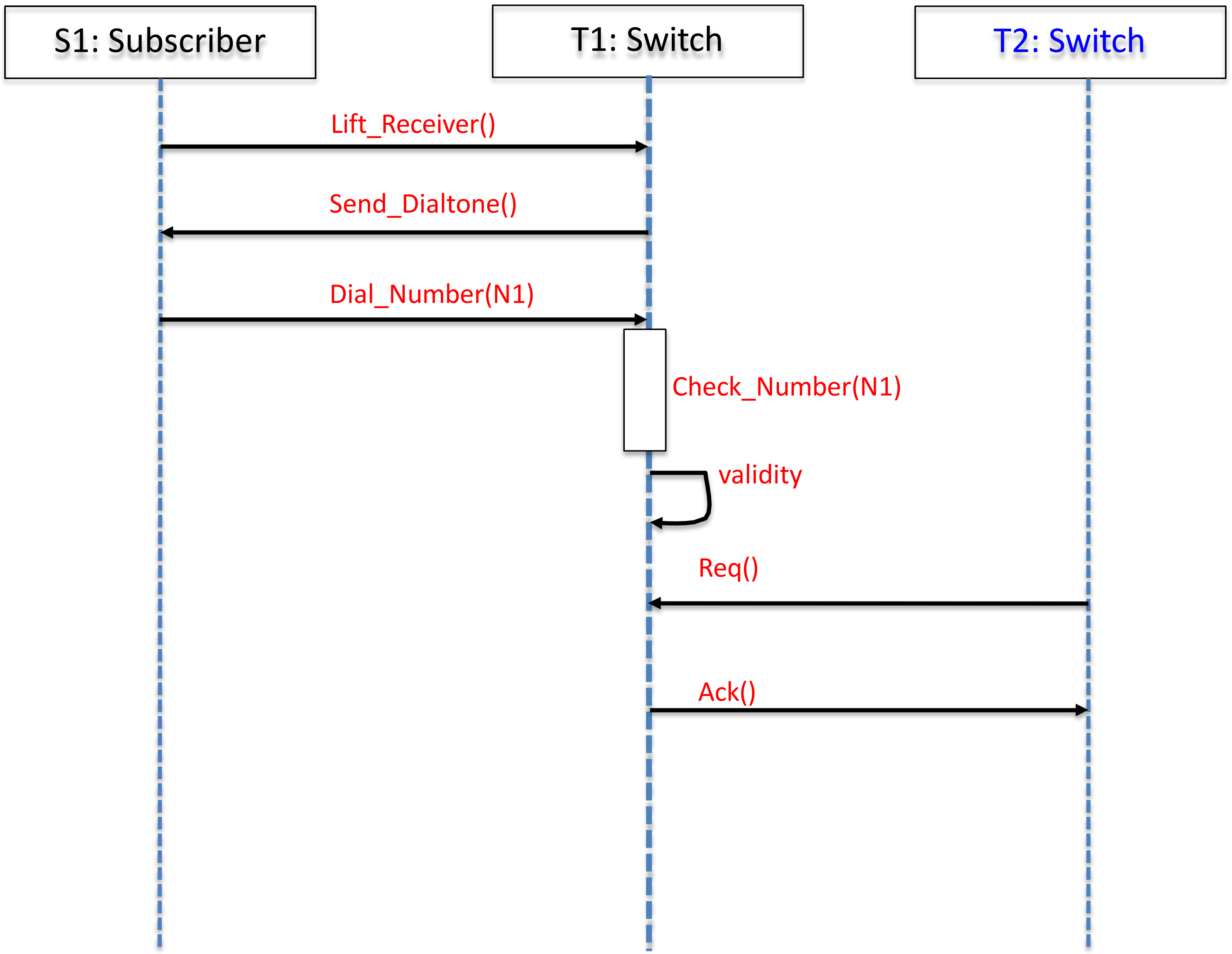
Parallel

- A telephone switch needs to handle multiple heterogeneous messages at the same time. For instance, consider the case when it receives messages from a subscriber and from a remote switch at the same time.

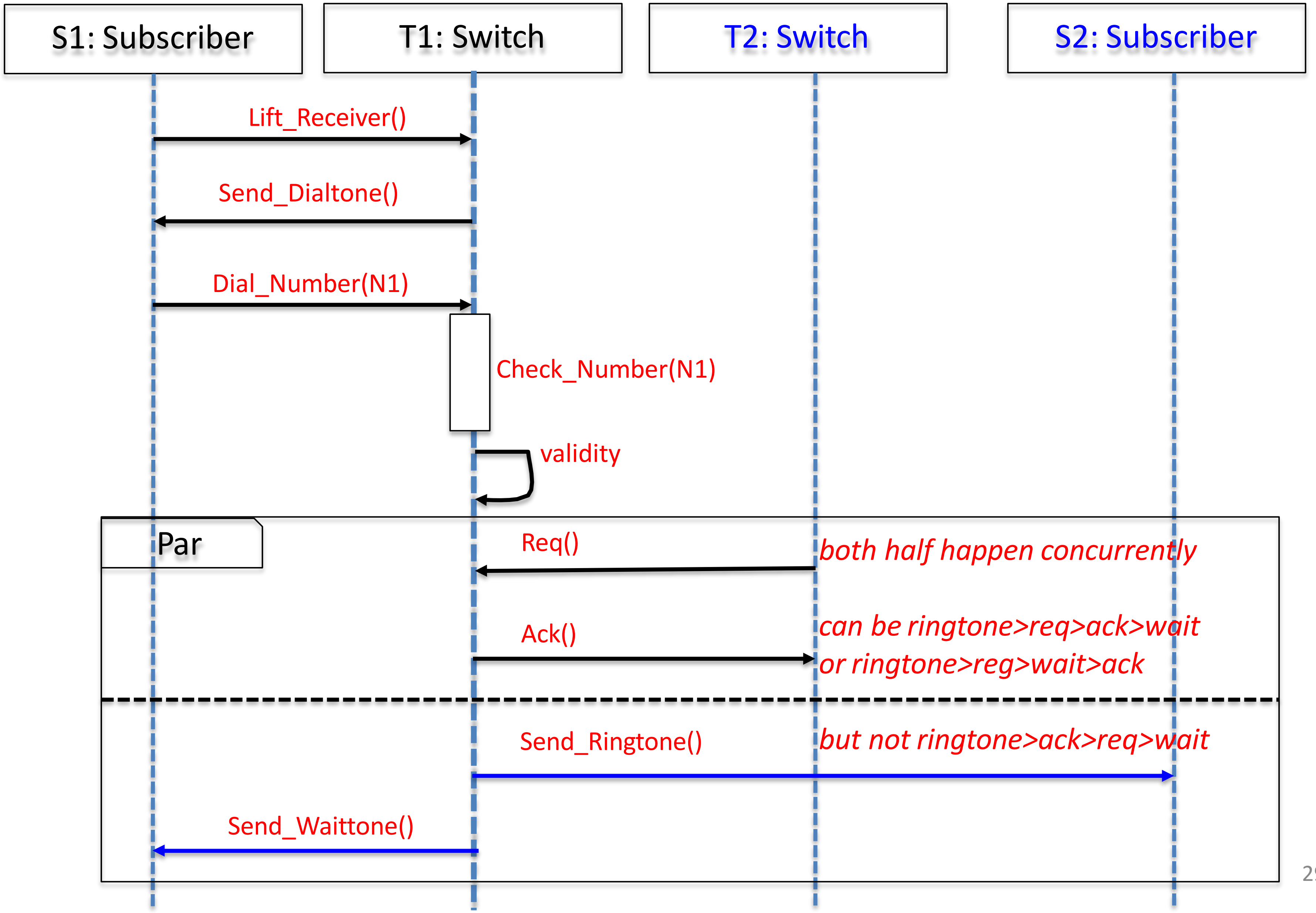
Parallel (Scenario 1)



Parallel (Scenario 2)



Parallel (Combined)



both half happen concurrently

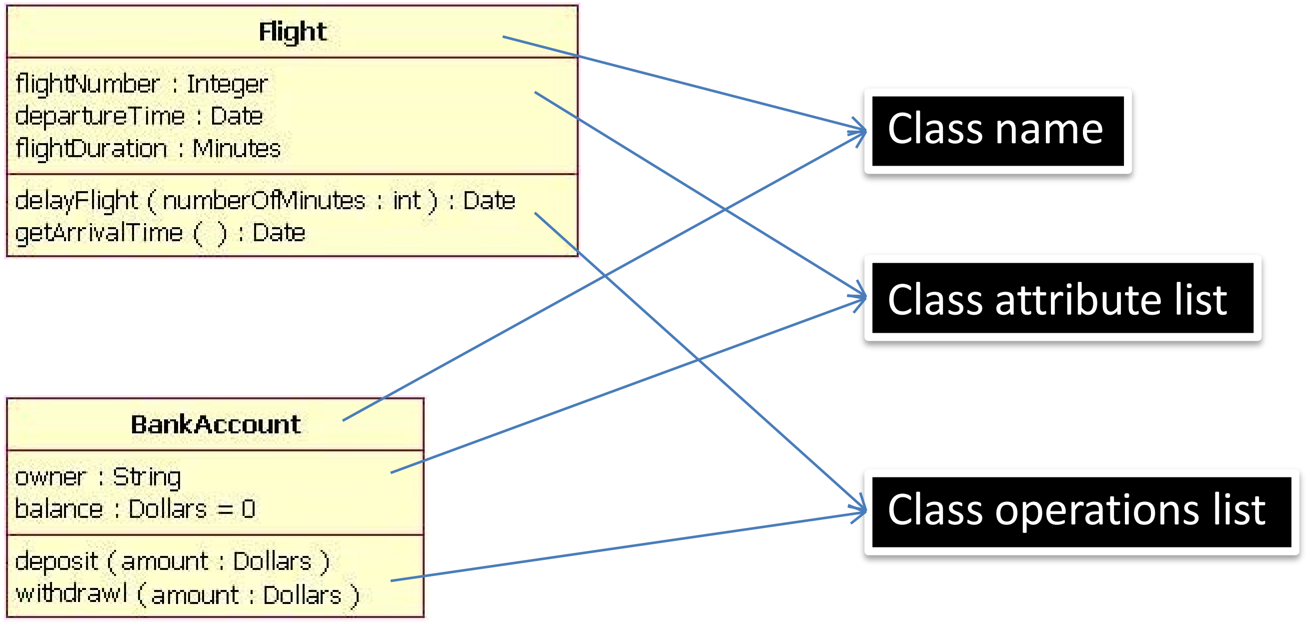
can be ringtone>req>ack>wait
or ringtone>reg>wait>ack

but not ringtone>ack>req>wait

Class diagrams

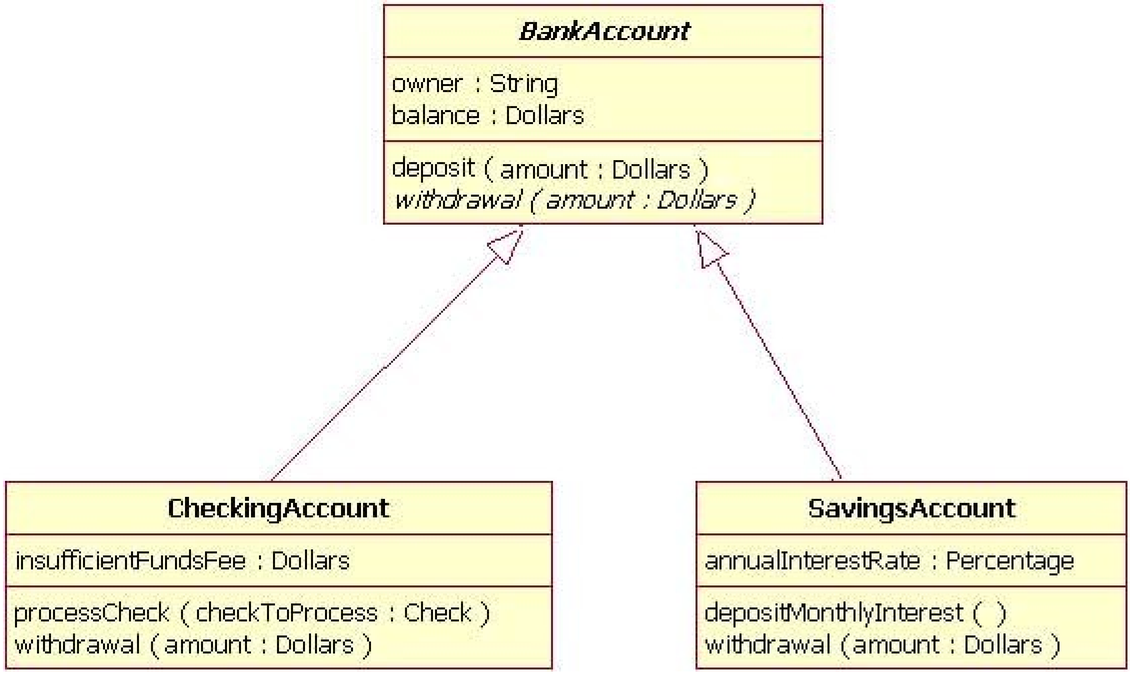
- A **class diagram** depicts a set of classes, interfaces, and collaborations and their relationships. Class diagrams are used most frequently while modelling object-oriented systems.

Basics



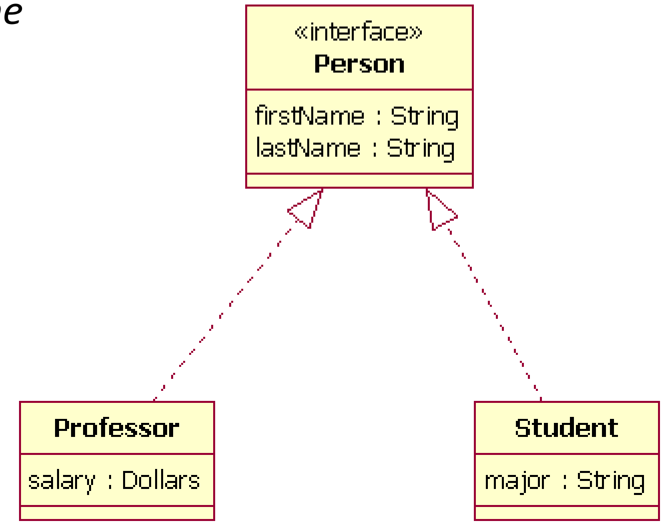
Flight
flightNumber : Integer
departureTime : Date
flightDuration : Minutes
delayFlight (numberOfMinutes : int) : Date
getArrivalTime ( ) : Date
BankAccount
owner : String
balance : Dollars = 0
deposit (amount : Dollars)
withdrawl (amount : Dollars)
Class name
Class attribute list
Class operations list

Inheritance


BankAccount
owner : String
balance : Dollars
deposit (amount : Dollars)
withdrawal (amount : Dollars)
CheckingAccount
insufficientFundsFee : Dollars
processCheck (checkToProcess : Check)
withdrawal (amount : Dollars)
SavingsAccount
annualInterestRate : Percentage
depositMonthlyInterest ( )
withdrawal (amount : Dollars)

Interfaces

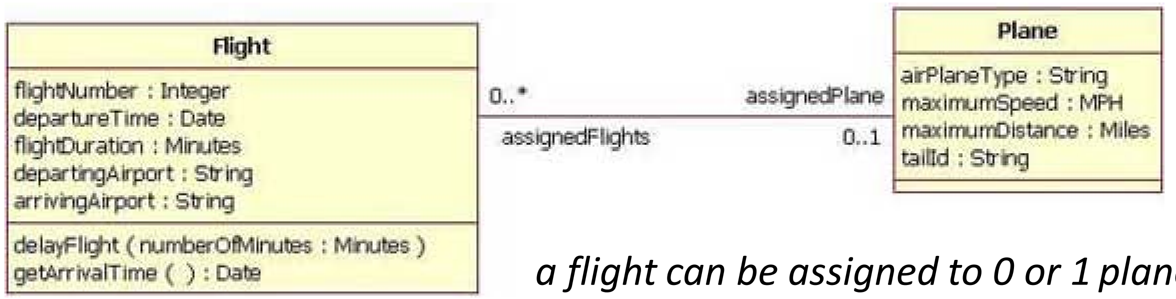
Interface is an abstract class. Does not contain any implementation. The class implementing the interface must define all methods of the interface.



Dotted line indicates it's NOT inheritance.

Associations

- When you model a system, certain classes will be related to each other.
- Bi-directional (standard) association: both classes are aware of each other and their relationship



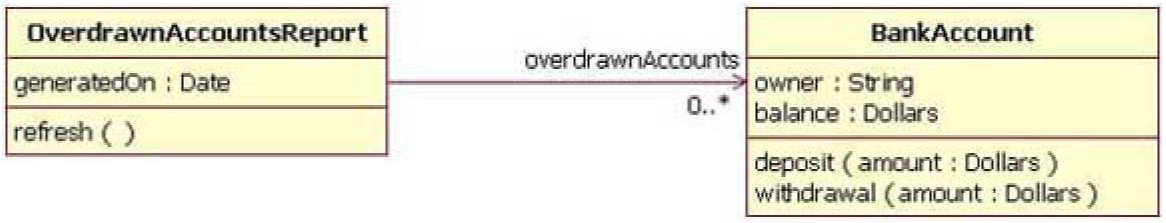
a flight can be assigned to 0 or 1 plane
a plane can be assigned from 0 to ... flights

Multiplicity

Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
*	Zero or more
1..*	One or more
3	Three only
0..5	Zero to Five
5..15	Five to Fifteen

Uni-Directional Association

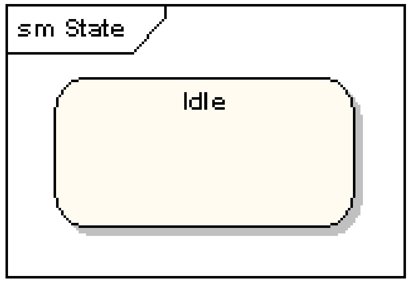
- In a uni-directional association, two classes are related, but only one class knows that the relationship exists.



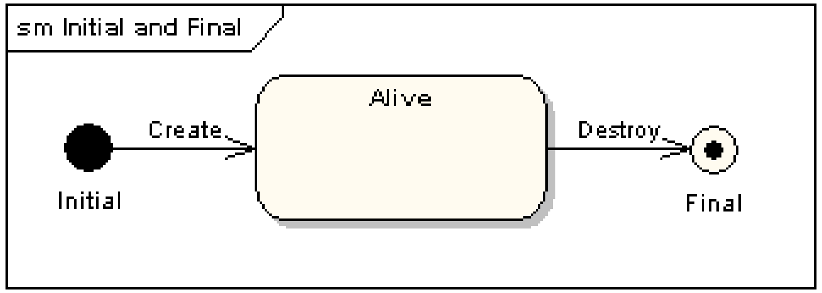
Why do you think it is unidirectional? Can overdrawnbankaccount report be part of an account?

State Machine Diagram

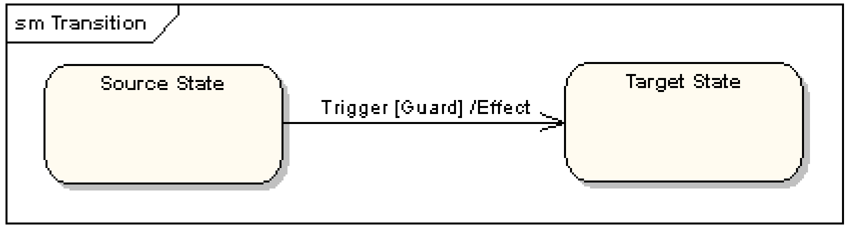
- States



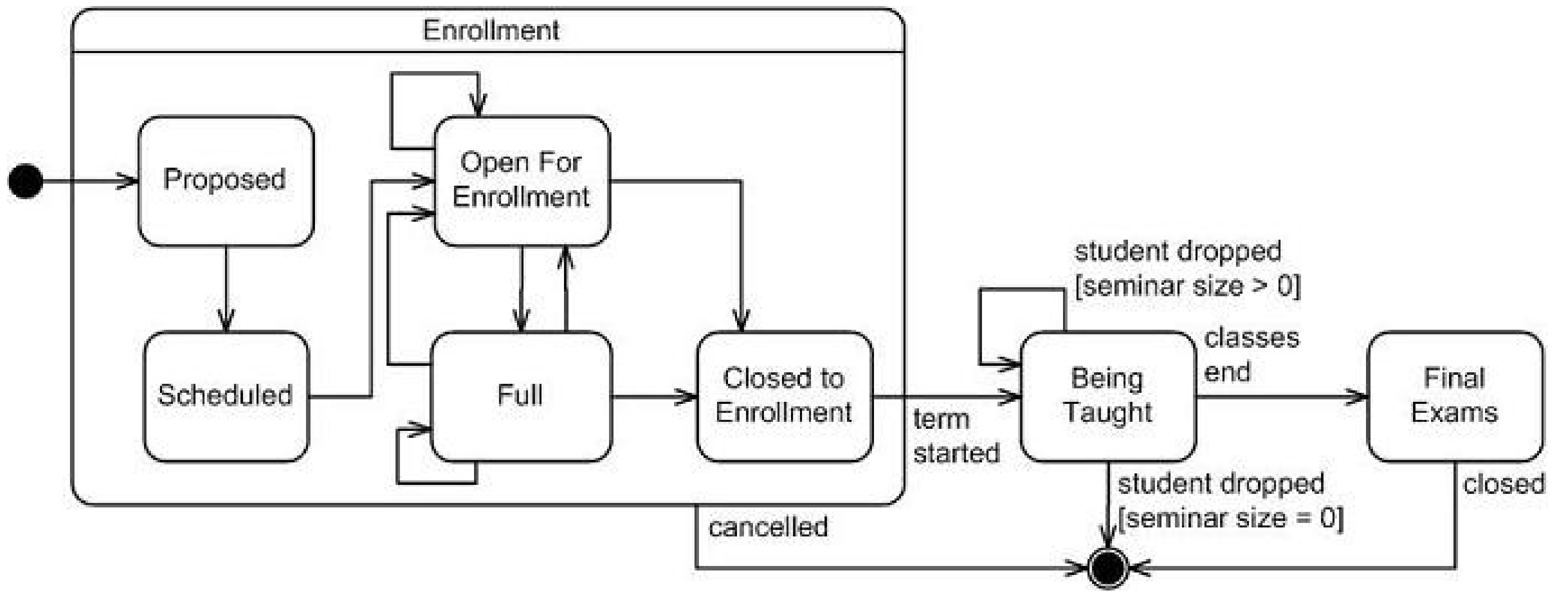
- Initial and final states



- Transitions



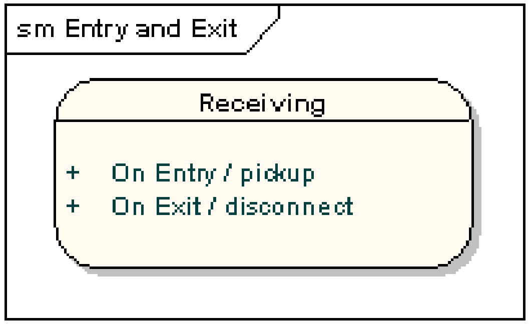
State Machine Example



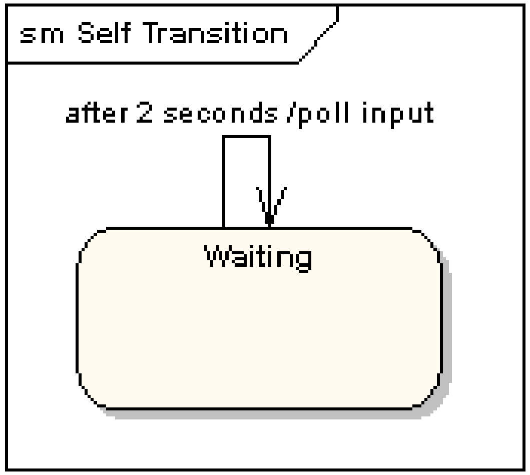
Talk about the big state. Talk about transition label – note they are conditional as well as unconditional, give examples.

State Machine Diagrams: More

- State actions



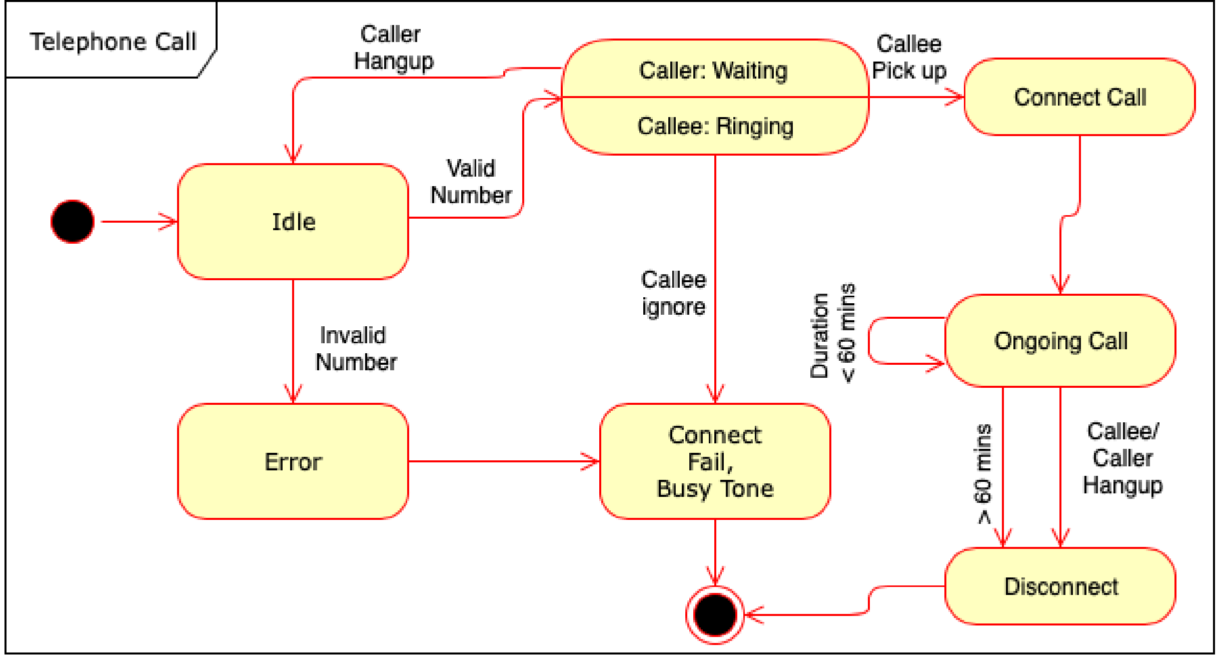
- Self-looping actions

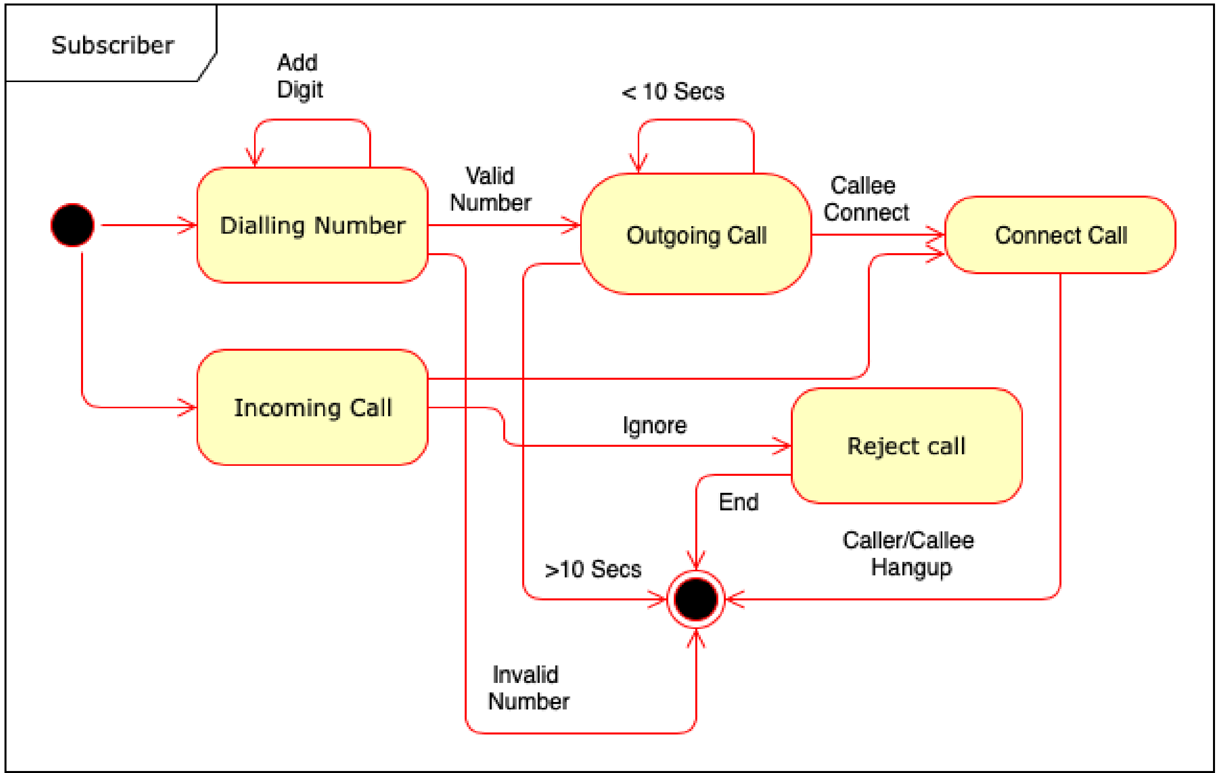


Cohort Exercise 13 (10 minutes)

Augment the state machine diagrams of Telephone call and subscriber to include the following features (Pay attention to your class diagram and transitions in the state diagram):

- A call is disconnected if someone talks over 1 hour
- After dialling a number, the caller waits at most 10 seconds for the connection to be established. Otherwise, she hangs up.

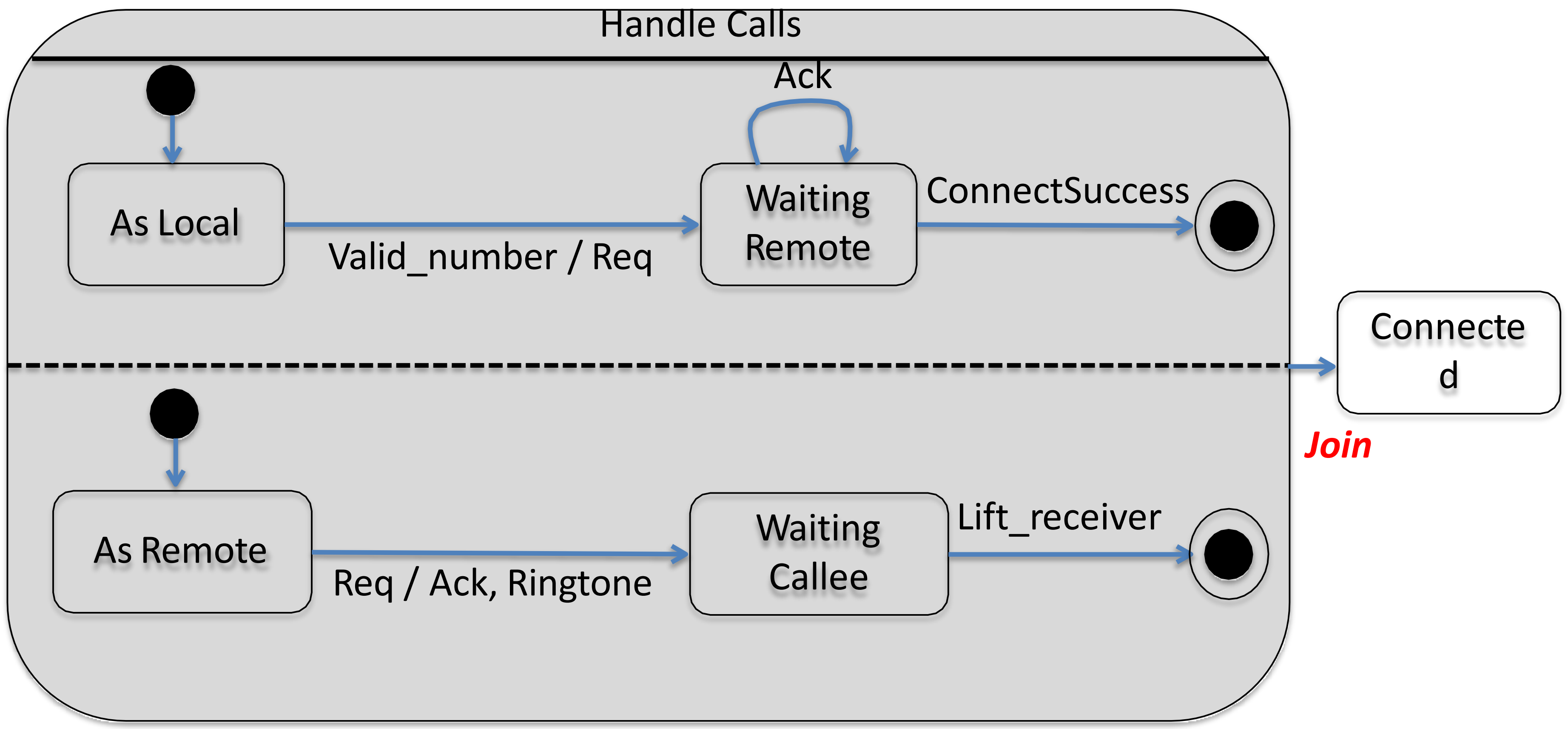

Telephone Call
Caller Hangup
Caller: Waiting
Callee: Ringing
Callee Pick up
Connect Call
Idle
Valid Number
Callee ignore
Duration < 60 mins
Ongoing Call
Invalid Number
Error
Connect Fail, Busy Tone
> 60 mins
Callee/ Caller Hangup
Disconnect


Subscriber
Add Digit
< 10 Secs
Dialling Number
Valid Number
Outgoing Call
Callee Connect
Connect Call
Incoming Call
Ignore
Reject call
End
Caller/Callee Hangup
>10 Secs
Invalid Number

Concurrent states

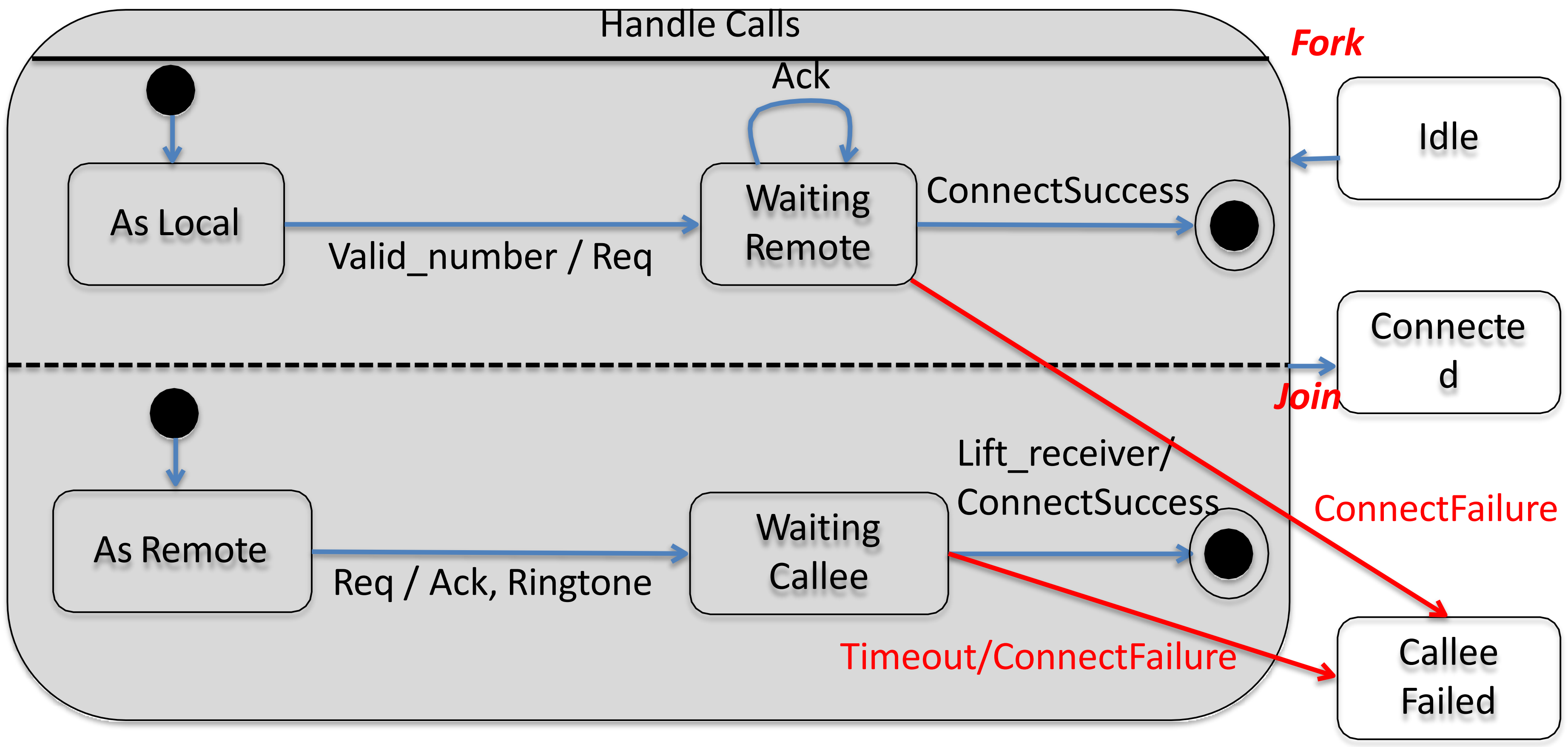
Consider the case again where a telephone switch has to perform many activities at the same time. Each separate activity can be modeled as a concurrent state.

Concurrent states



A switch may act as a local and remote switch at the same time

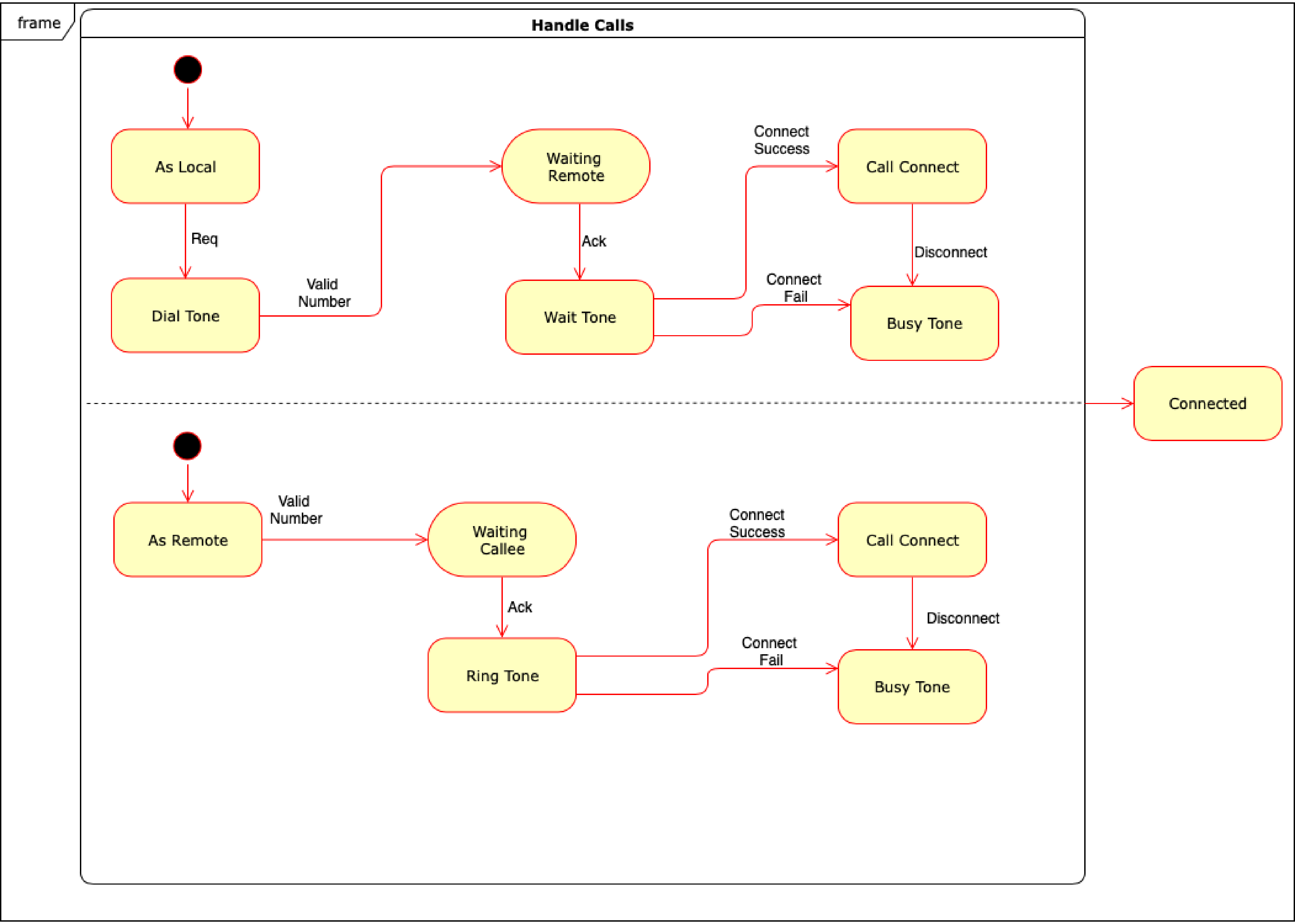
Concurrent states



A switch may act as a local and remote switch at the same time

Cohort Exercise 14 (5 minutes)

The state diagram in the previous slide does not include messages from the switch to the subscriber. Augment the state diagram to include these messages.


frame
Handle Calls
As Local
Req
Dial Tone
Valid Number
Waiting Remote
Ack
Wait Tone
Connect Success
Call Connect
Disconnect
Connect Fail
Busy Tone
Connected
As Remote
Valid Number
Waiting Callee
Ack
Ring Tone
Connect Success
Call Connect
Disconnect
Connect Fail
Busy Tone