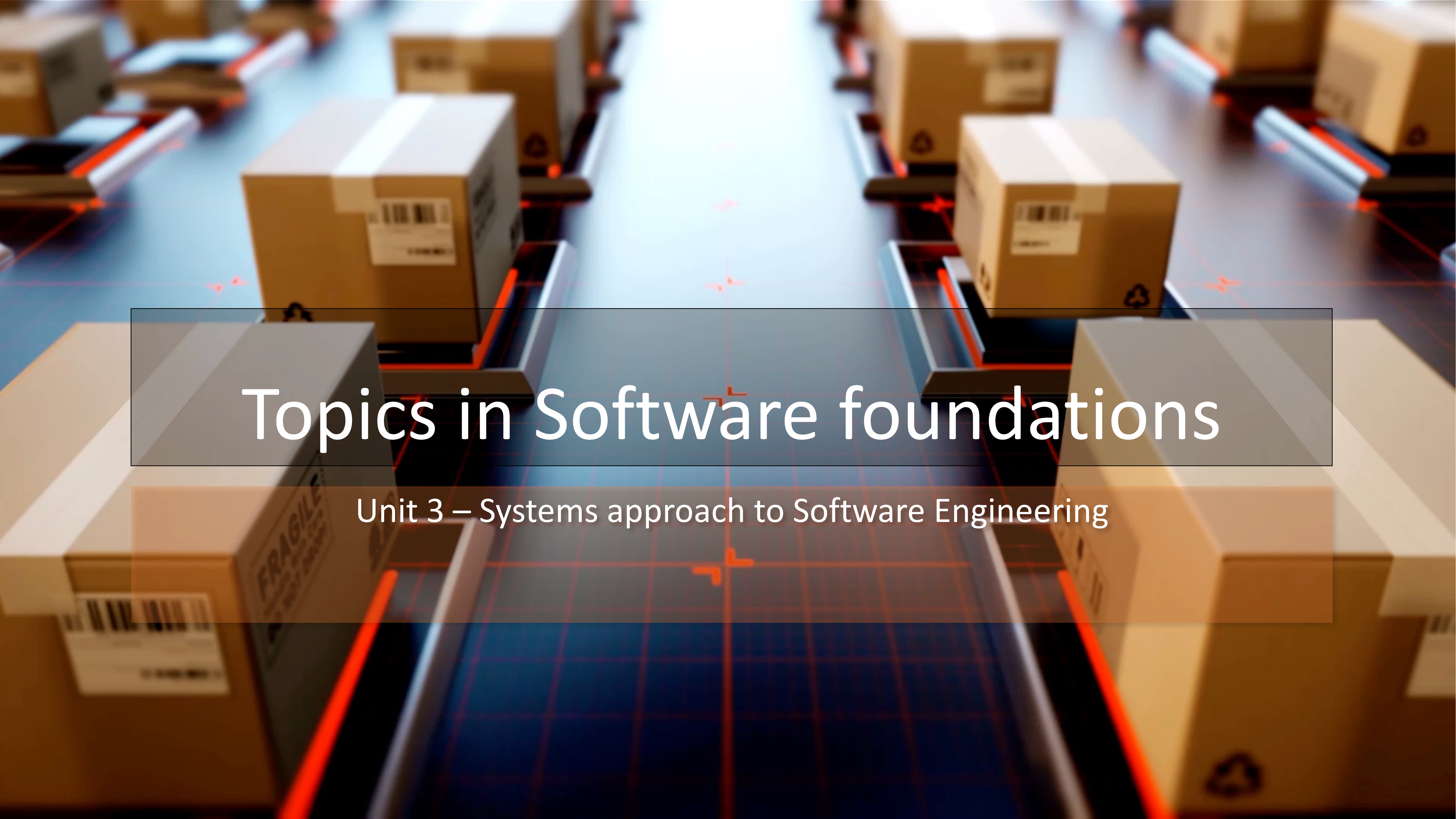


Topics in Software foundations

Unit 3 – Systems approach to Software Engineering

Topics we cover

Introduction to systems modeling

Modeling System of Systems (SoS)

Modeling communications

Modeling UI systems

Modeling data access

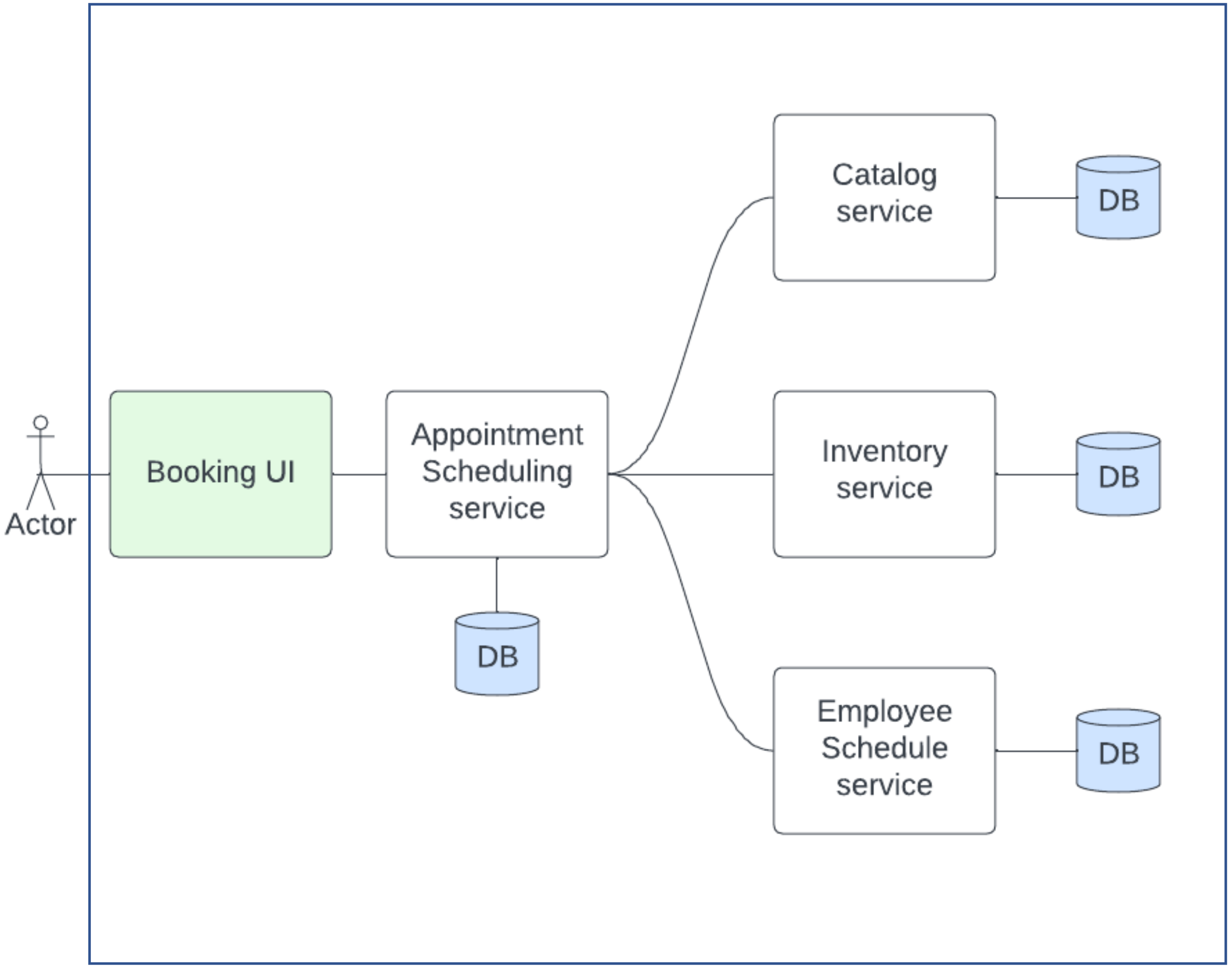
Formalizing TS as a modeling language

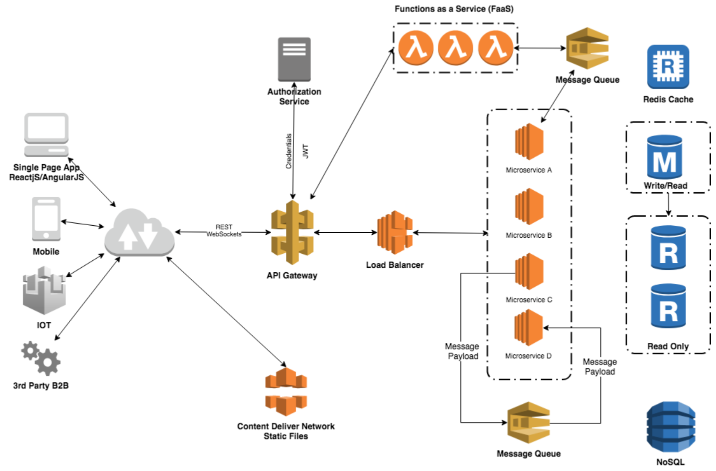
Modeling Communications

Unit 3 – TiSF S’23

Session 4 (2023-04-06)

Appointment booking SoS

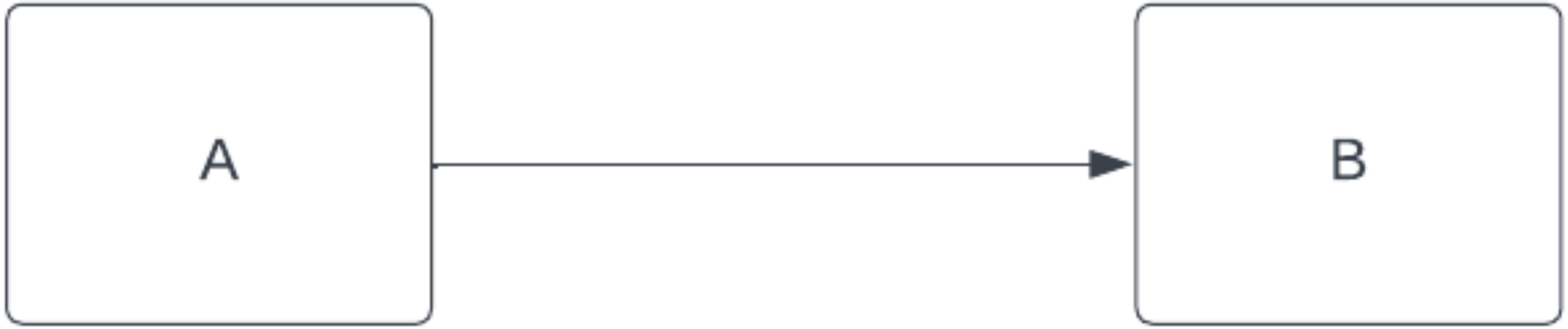




We write behavioral model of the SoS – this captures the requirements, the behavior the actor expects to see

We then break into component systems, each with its own transition system model, and interconnections between systems

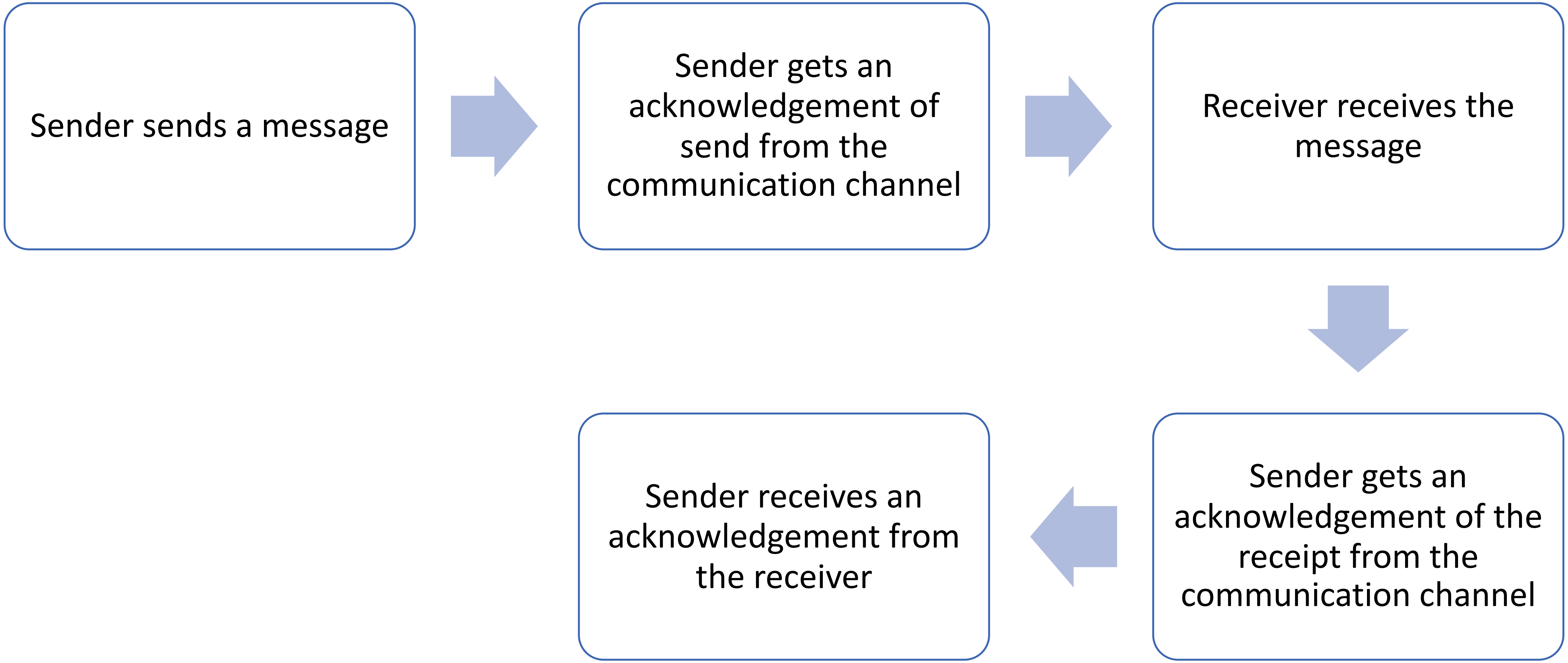
1-1 system communication



Sender

Receiver

1-1 system communication



Sender sends a message

Sender gets an acknowledgement of send from the communication channel

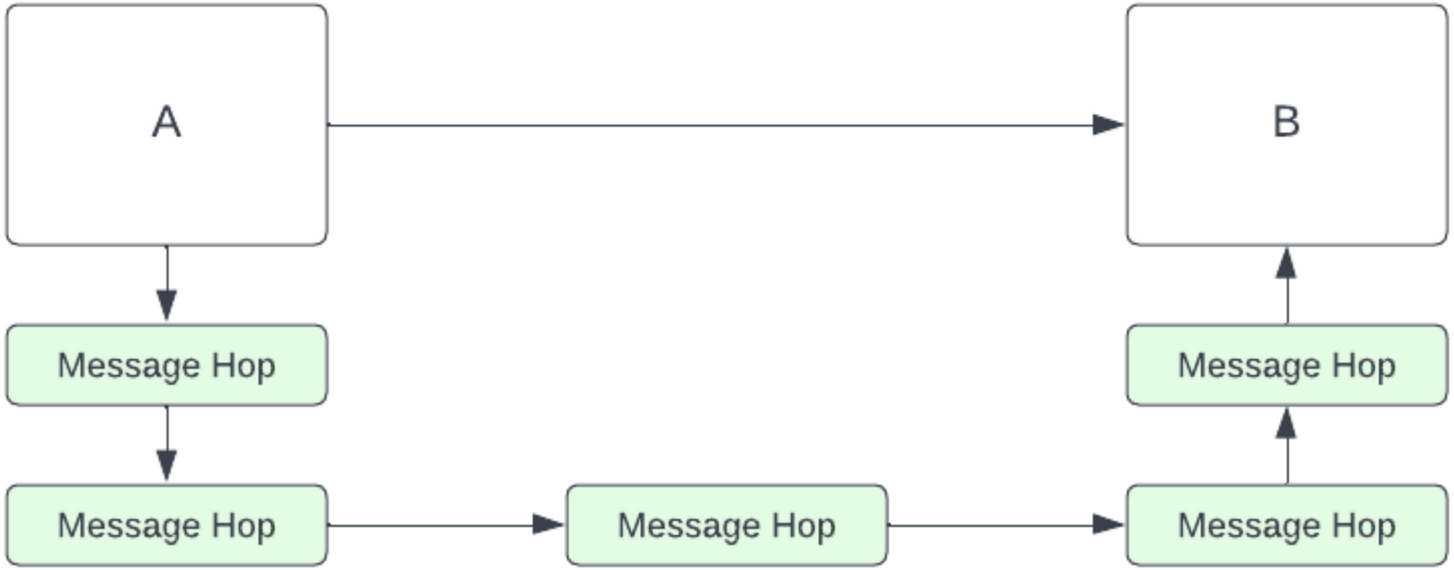
Receiver receives the message

Sender gets an acknowledgement of the receipt from the communication channel

Sender receives an acknowledgement from the receiver

Not all steps are done (or required) in a communication

When is the message assumed (by sender) to be delivered?



It is an almost arbitrary choice. The reliability of the systems determine where you draw the line.

Types of communication (sender side)

Synchronous Communication (for ex, function call)

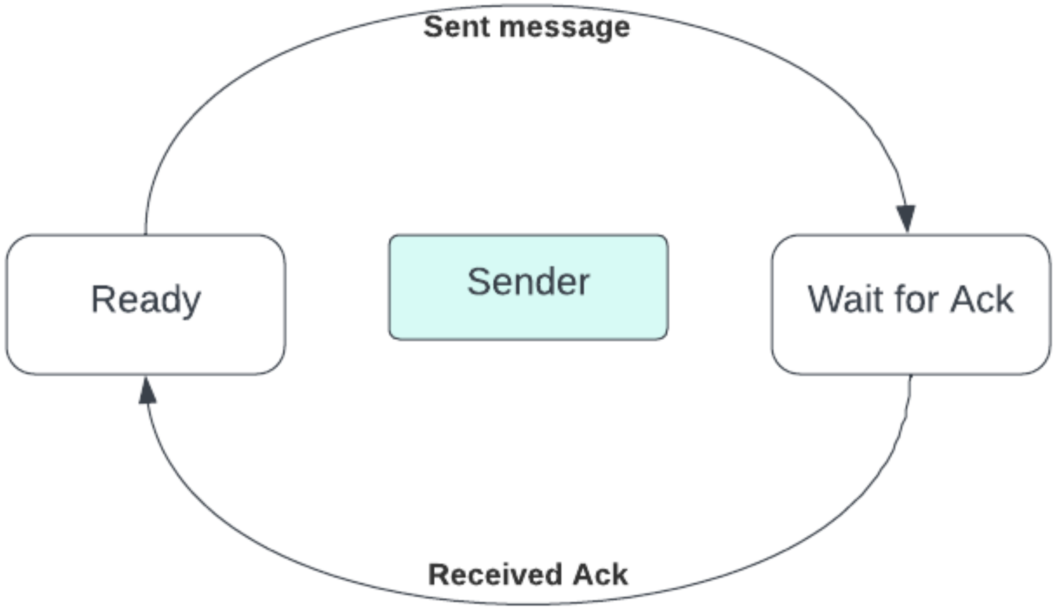
- Sender waits after sending message to get the 'result'
- Wait for what?
  - One or more responses from the receiver(s) or intermediaries
  - No other execution is performed by the sender (except waiting)
- 'result' triggers the sender to resume execution

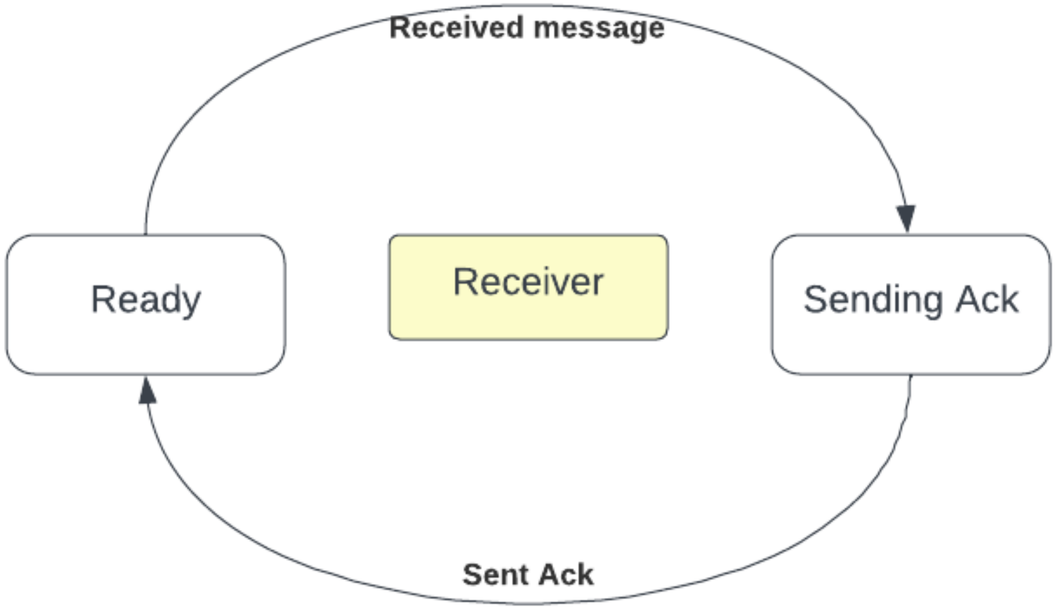
Asynchronous Communication (for ex, message passing)

- Sender sends the message and continues with the execution
- 'result' is handled as any other communication received by the sender

Asynchronous communication (message passing) is a key communication model

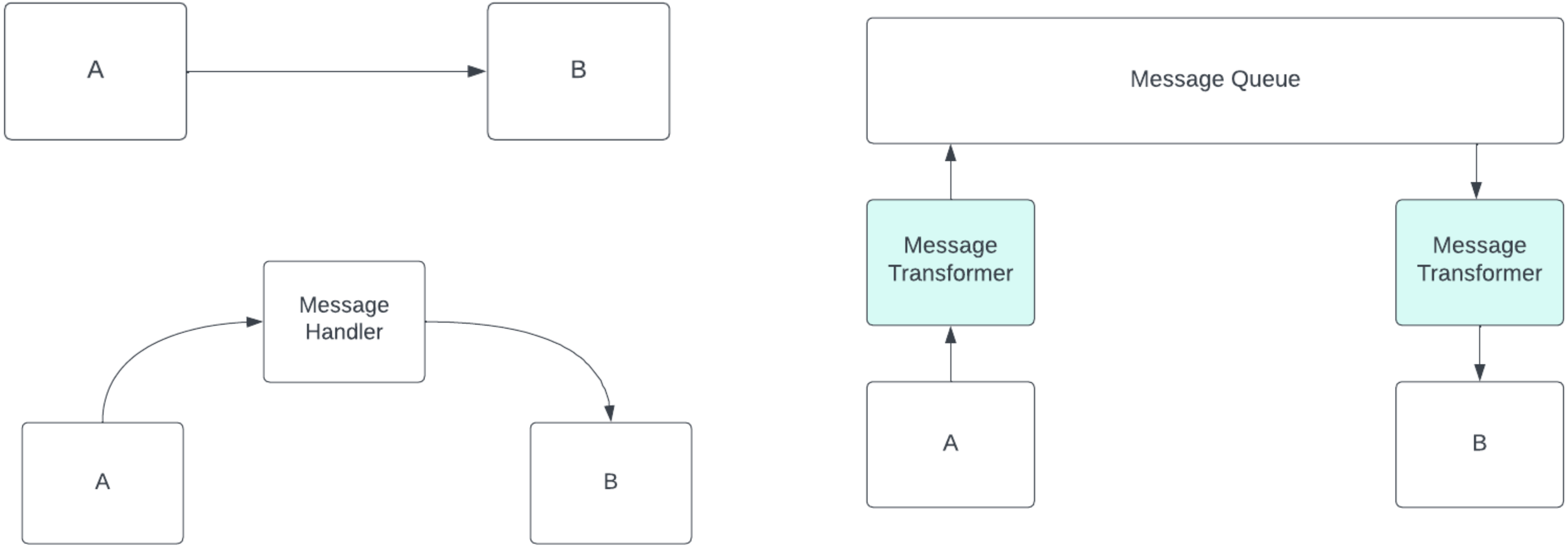
Typical sender and receiver systems





Modeling communications

Multiple ways of modeling message passing



Communication needs a protocol

A series of requests and responses with unambiguous semantic

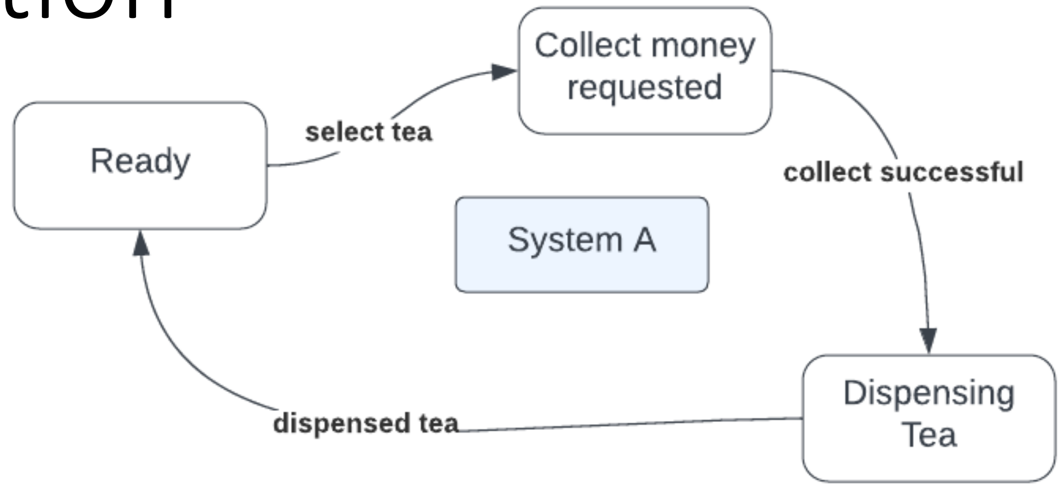
- TCP/ UDP (everything!)
- HTTP (Web)
- XMPP (Chat)
- SIP/H.323 (Call control)

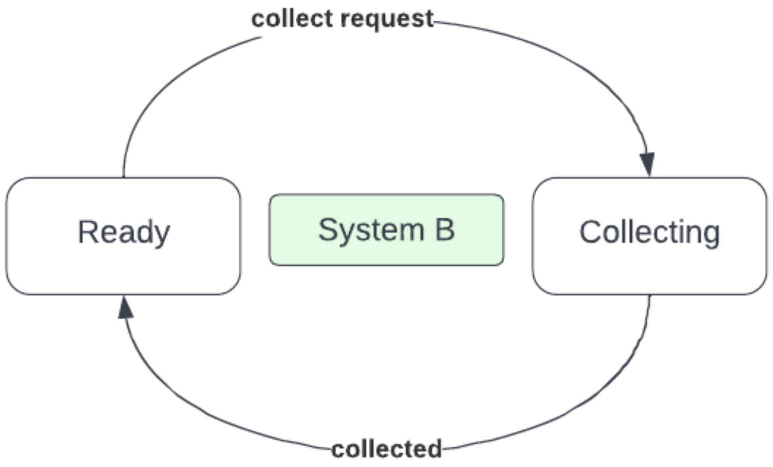
Fire and forget (one send)

- Logging
- Event

Communicating systems must agree to a communication protocol, it is an important design concern

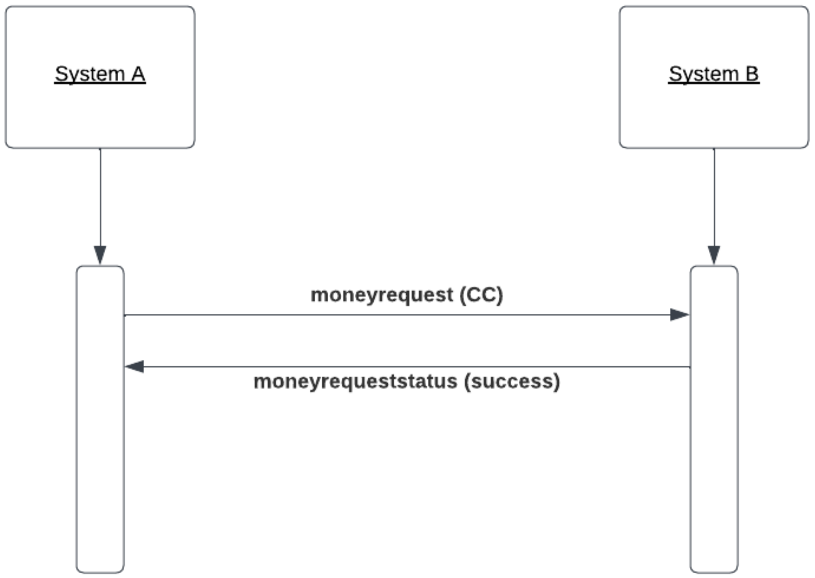
1-1 system communication



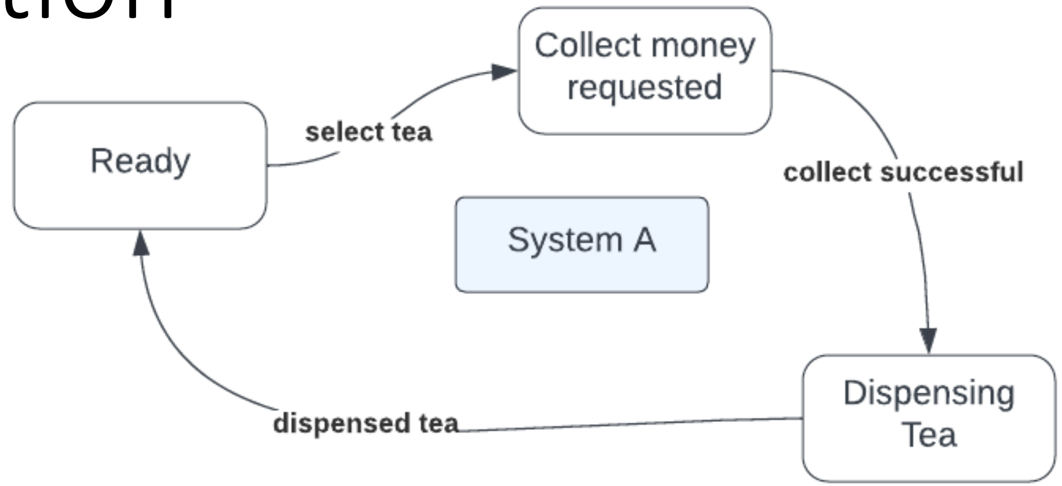


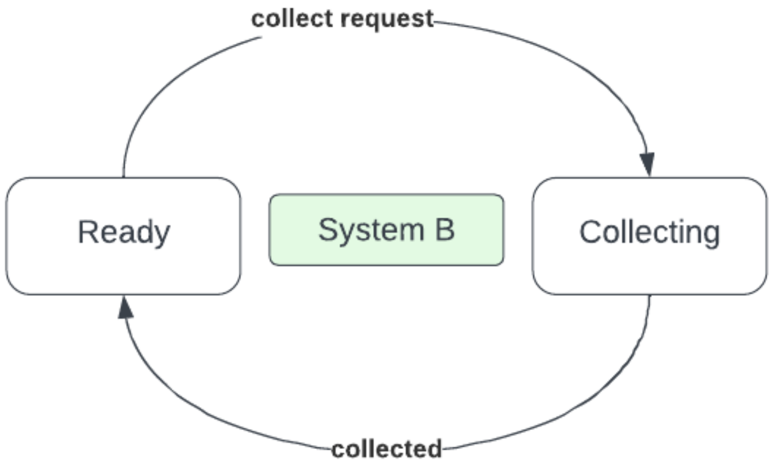
1-1 system communication

Communication protocol is request-response pair



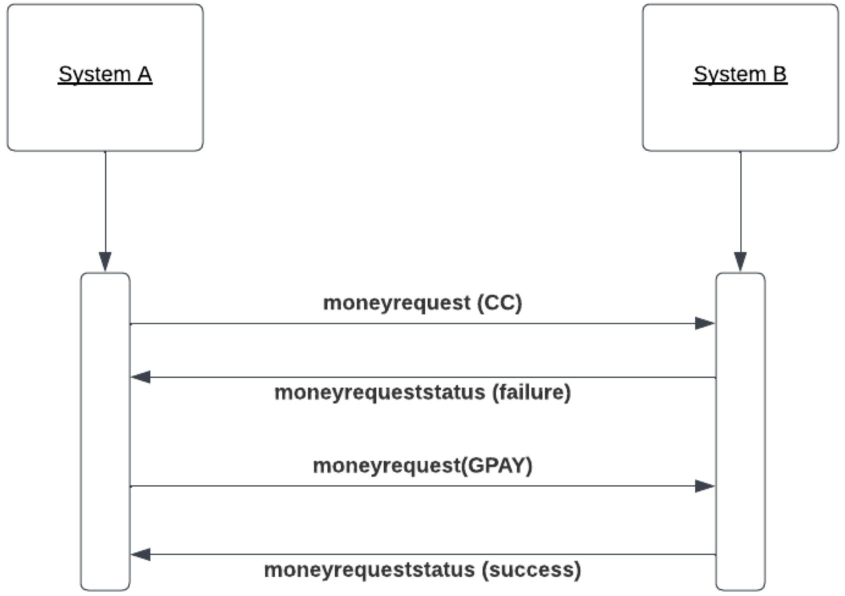
msg(A,B,moneyrequest), msg(B,A,moneyrequeststatus)

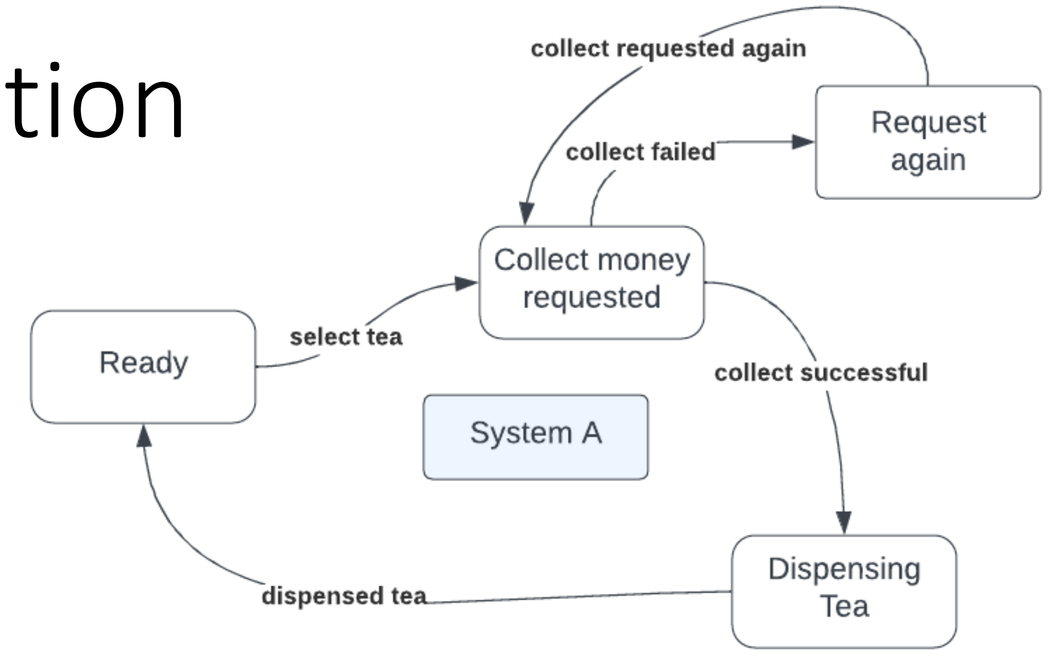


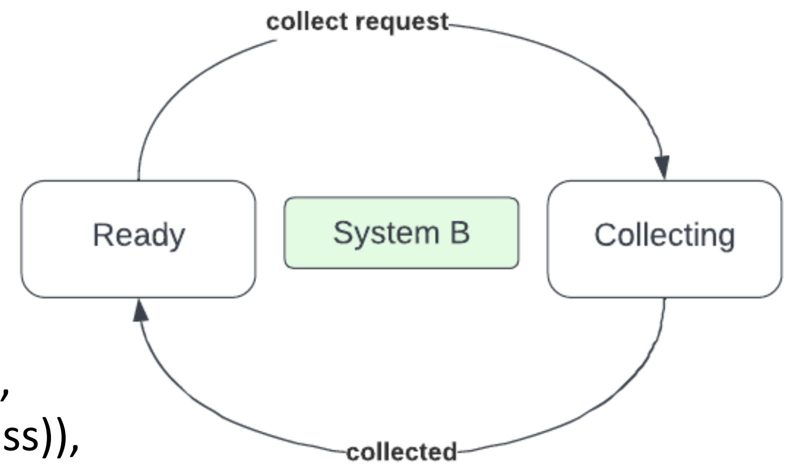


1-1 system communication

Communication protocol sequences can be arbitrarily long.







msg(A,B,moneyrequest(CC)), msg(B,A,moneyrequeststatus(failure)),
msg(A,B,moneyrequest(GPAY)), msg(B,A,moneyrequeststatus(success)),

Other types of communication

- 1-n system communication (multicast or broadcast)

A reminder on data

An action is associated with data, so a better way to represent action is $U = name_{event} \times data_{event}, u = (n_u, d_u)$. There may not be an event name.

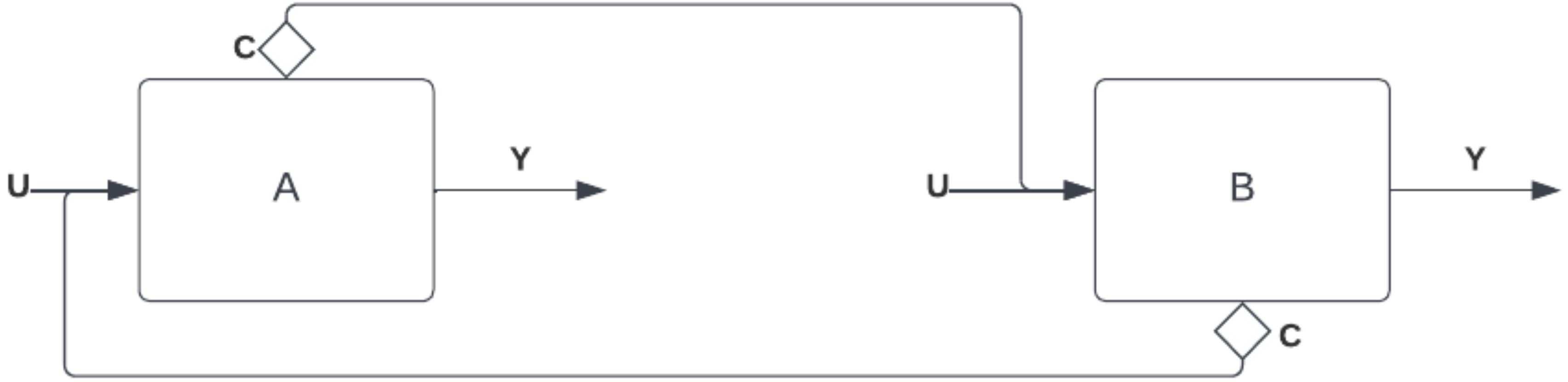
A state is the same way, so a better way to represent state is $X = name_{state} \times data_{state}, x = (n_x, d_x)$. There may not be a state name.

Since transition is determined by action and state $x^{'} = f\big((n_x, d_x), (n_u, d_u)\big)$, event data potentially influences the behavior of the transition function and the state variables for the new state

Sender uses event data to distinguish between continuing messages of an ongoing communication and a new sequence

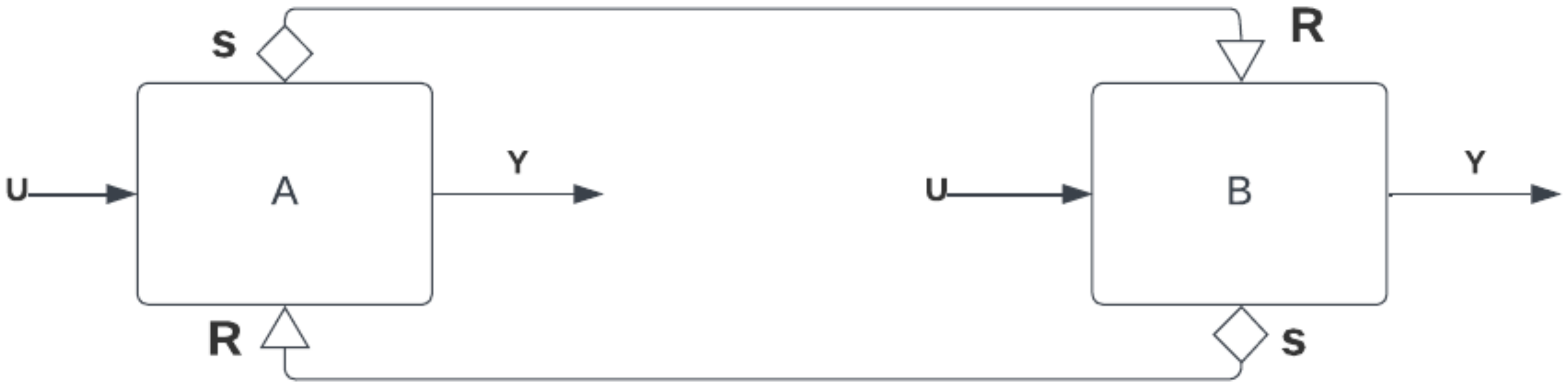
Wiring the components for communication

Assume a sendmsg() primitive method that uses the available communication channel to deliver the message to B as an action. C is a communication (send) port.

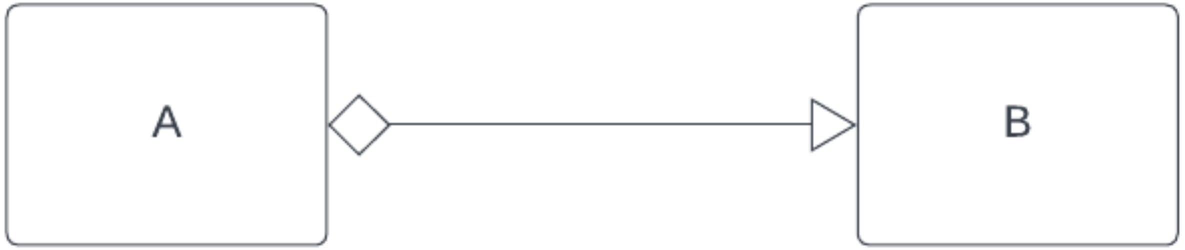


Wiring the components for communication

Two ports: a send and a receive. sendmsg() and recvmsg() primitives exist



Abstract model for communication

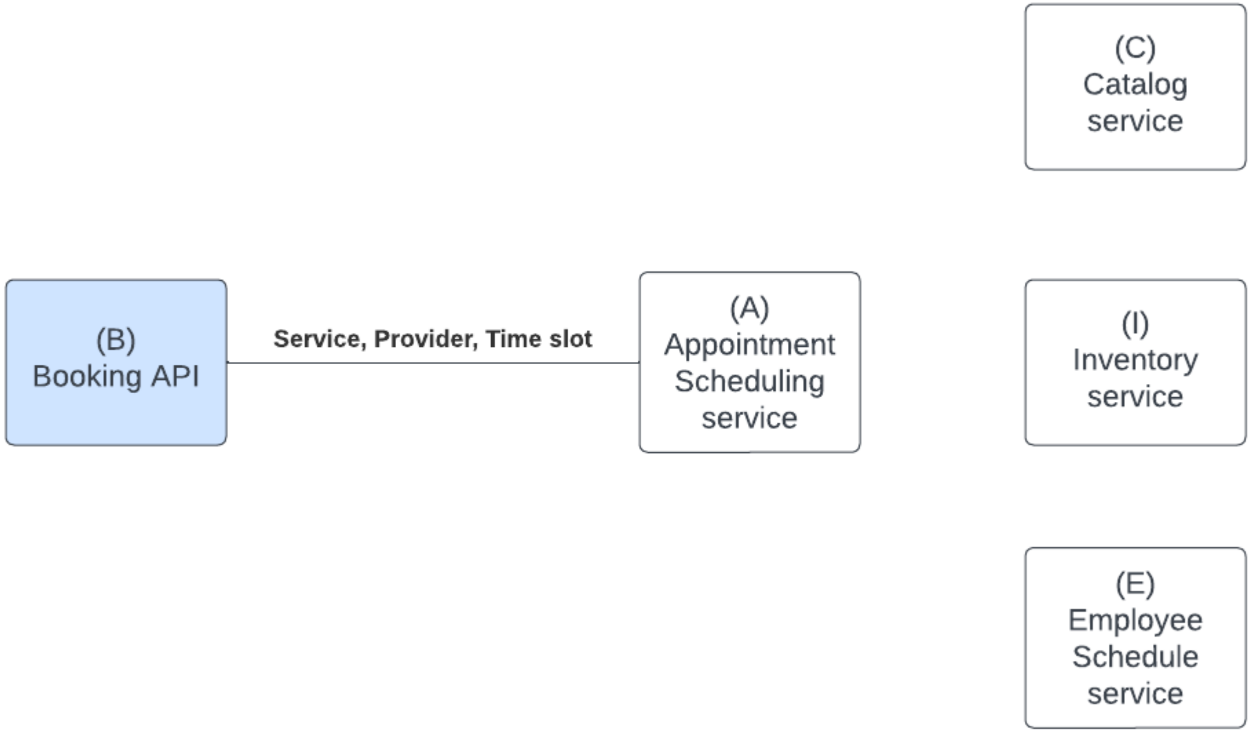


diamond is sendmsg semantic, arrow-head is recvmsg semantic

1. sendmsg*(<destination system>, <payload>*)
2. recvmsg(*<source system>, <payload>)*
3. List of (destination, payload) tuples for each system describes the communication channels used by the system
4. List of (source, destination, payload) tuples describes all the communication channels of SoS

Classwork

Booking problem (modified)


(C)
Catalog service
(B)
Booking API
Service, Provider, Time slot
(A)
Appointment Scheduling service
(I)
Inventory service
(E)
Employee Schedule service

Specification

Catalog provides the information about what products are required and in what quantity

Inventory provides two services:

- The availability of products
- Hold the product for a future service

Employee provides three services:

- Availability in a particular time slot
- Can they perform a particular service
- Hold a particular time slot for a future service

Using these service capabilities, Appointment service should book a service given the service, provider and timeslot details.

Classwork

- Draw the wiring diagram
- Specify two versions of communication protocol that can be followed by Appointment system to achieve the goal of booking an appointment
    - Sequential version: Appointment system has only one ongoing communication at a time
    - Parallel version: Appointment system can have more than one ongoing communication at a time
- Why would you choose one version over the other?

Questions?