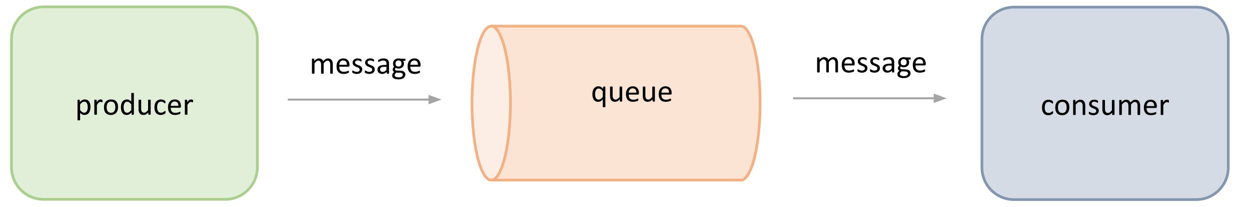
# Asynchronous messaging patterns

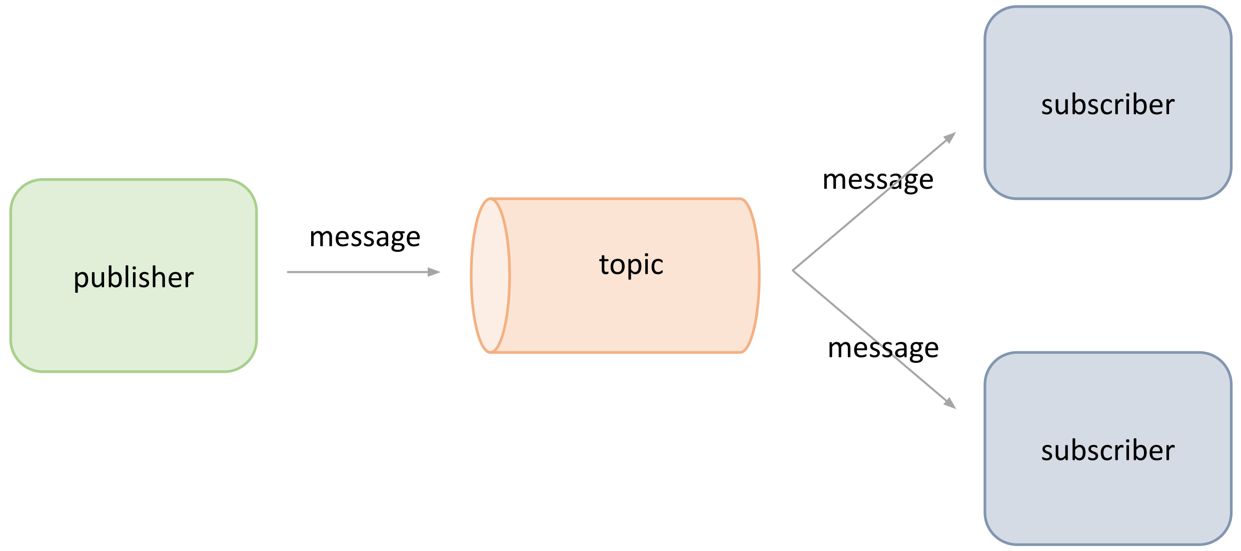
## message queuing

only a single consumer gets the message

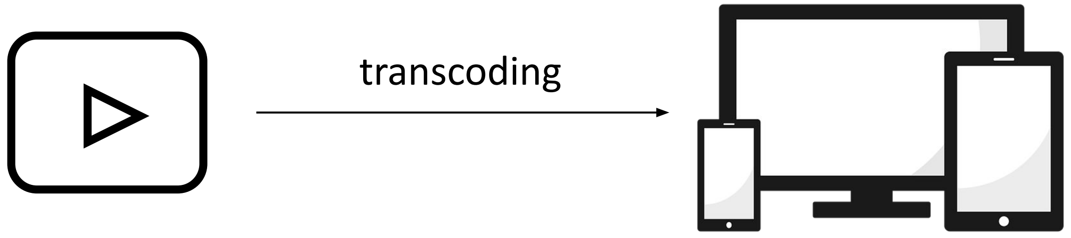


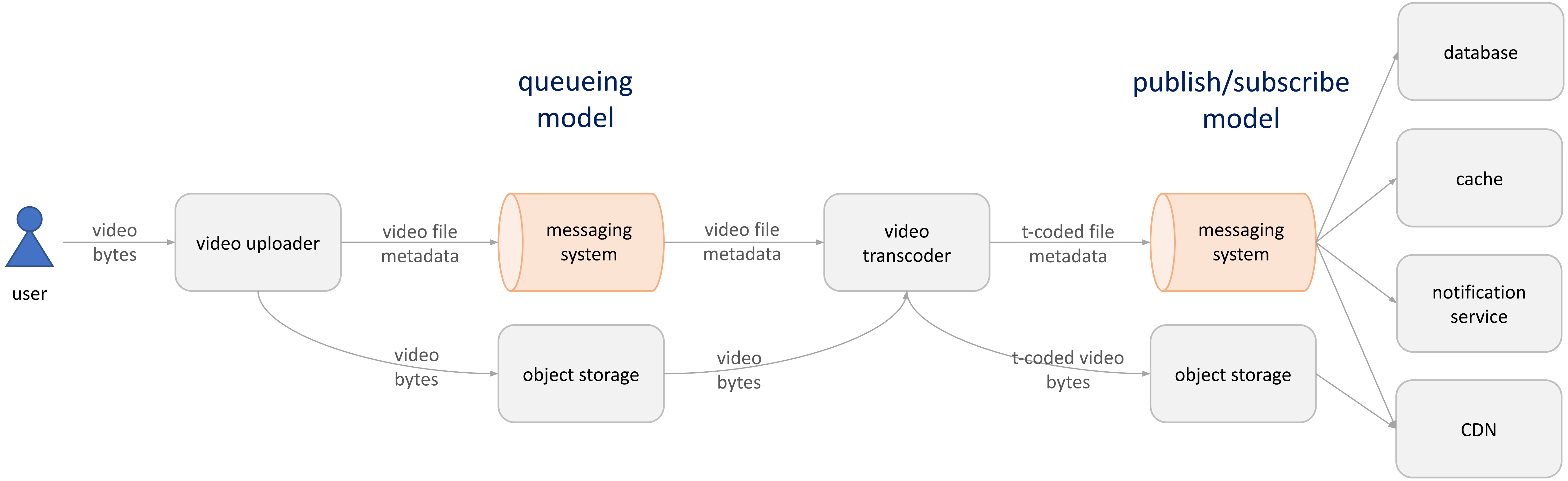
## publish/subscribe

the same message is delivered to all subscribers



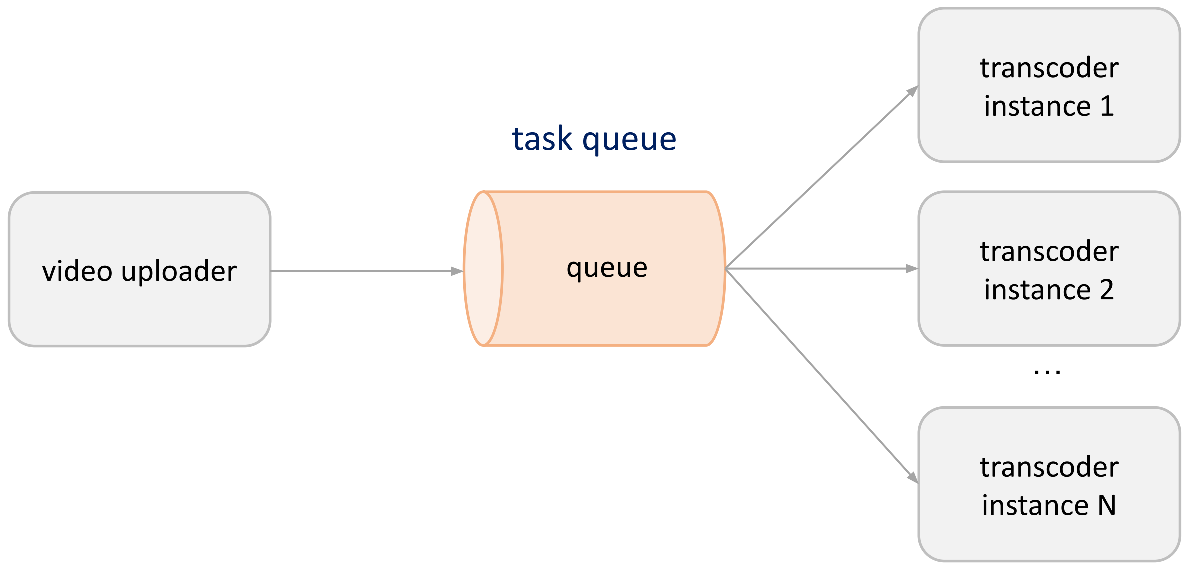
# Asynchronous messaging patterns





# Asynchronous messaging patterns

## competing consumers



### scalability

add more instances as the number of messages increases over time

### availability

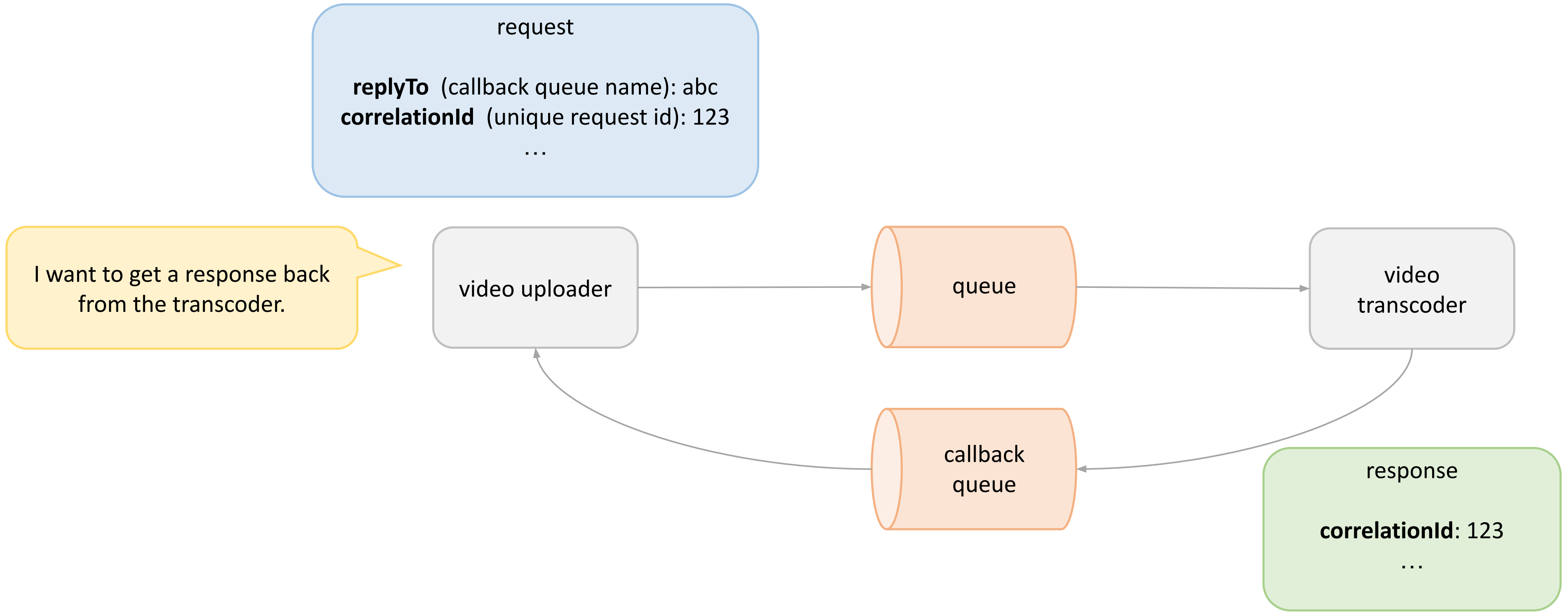
when some instances fail,
others continue to process messages

### performance

process more messages in parallel by adding more instances

# Asynchronous messaging patterns

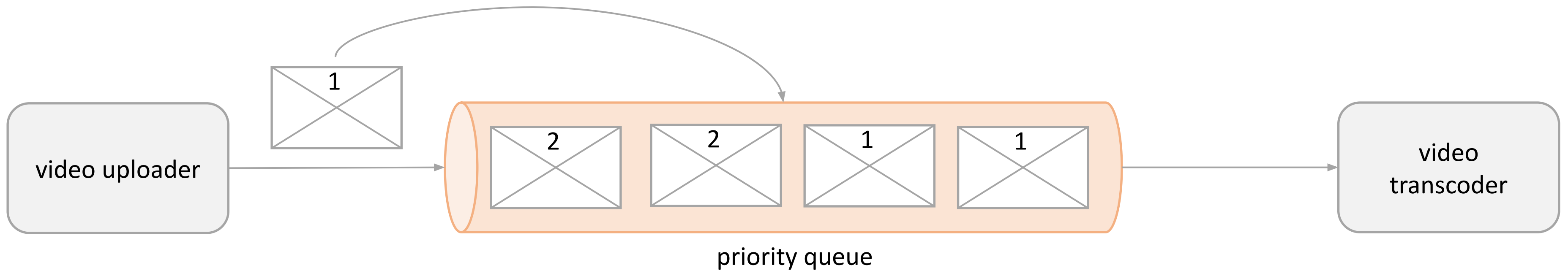
## request/response messaging


request
**replyTo** (callback queue name): abc
**correlationId** (unique request id): 123
…
I want to get a response back from the transcoder.
video uploader
queue
video transcoder
callback queue
response
**correlationId**: 123
…

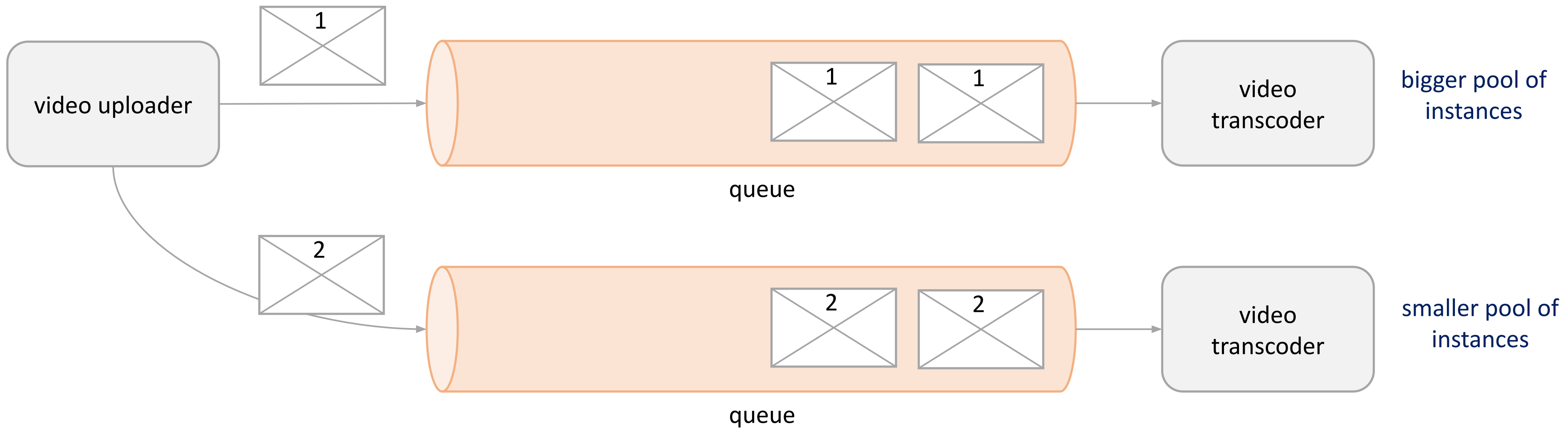
# Asynchronous messaging patterns

## priority queue

native support

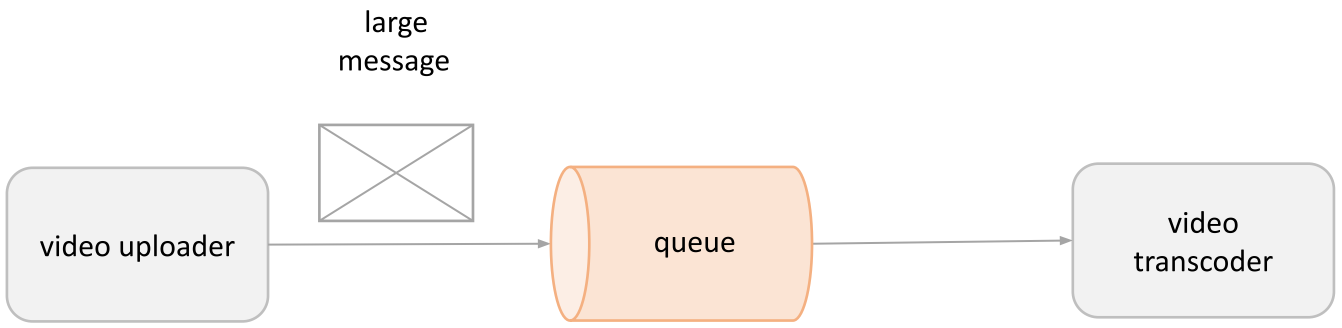


separate queue for each priority



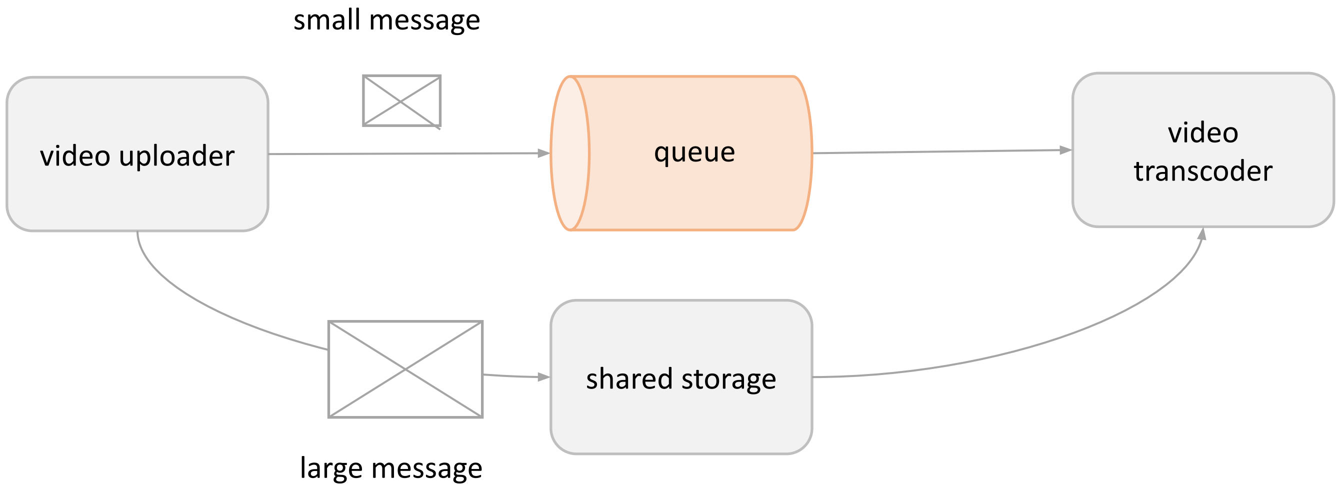
# Asynchronous messaging patterns

## claim check



this is bad because

- messaging systems have limits on message size
- large messages may cause memory and performance issues


small message
video uploader
queue
video transcoder
shared storage
large message