



Task 2: Large Message Proxy

Message Types

```
LargeMessage<T>{  
    T message;  
    ActorRef receiver;  
}
```

original message

receiver of original message

```
BytesMessage<T>{  
    T bytes;  
    ActorRef sender;  
    ActorRef receiver;  
    int messageLength;  
    long messageId;  
    int chunkOffset;  
}
```

serialized chunk of original message

sender of original message

receiver of original message

total length of original message

identifier for original message

index of message chunk in original message

Solution Description



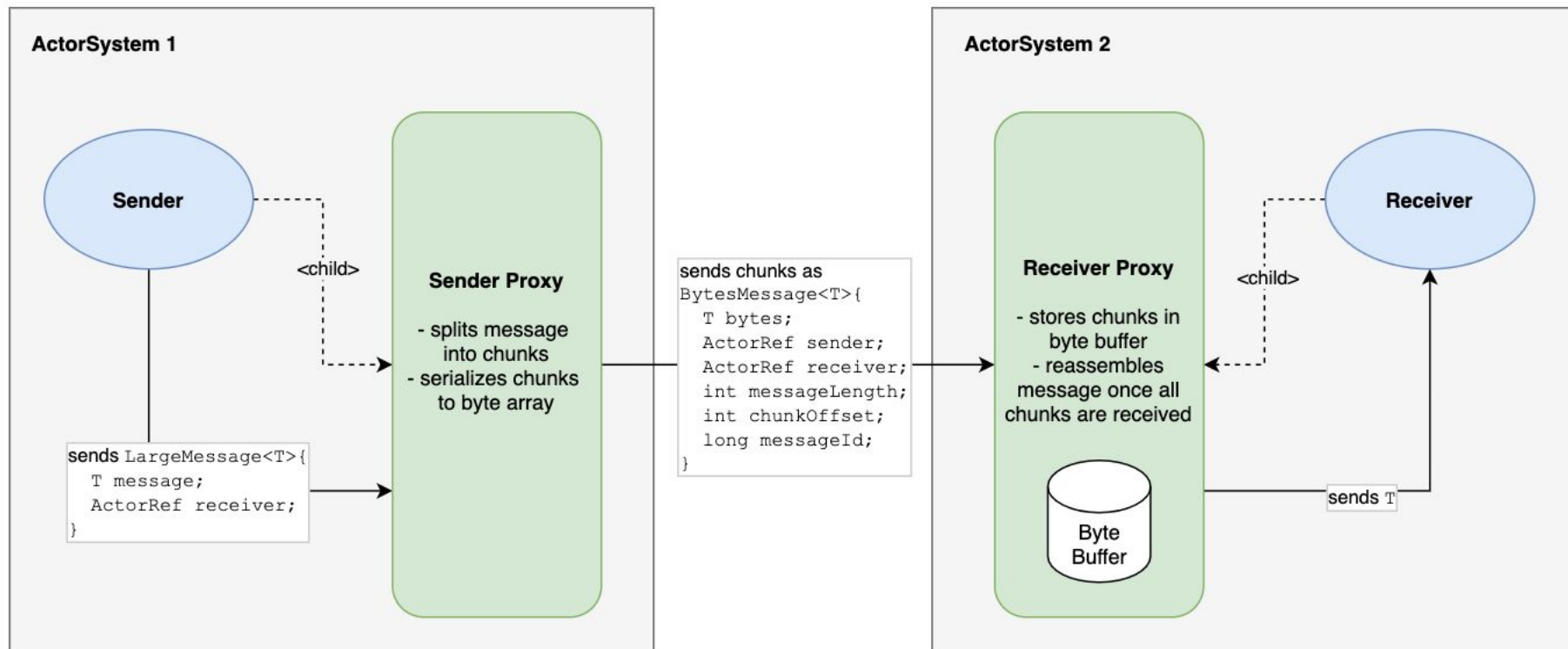
Sender Proxy

- gets `LargeMessage` from Sender
- 1) splits the message into smaller chunks of fixed size
 - `largeMessageChunkSize` configurable in Configuration
 - serialized to byte array with `KryoPool`
- 2) separately sends each chunk as `ByteMessage` to Receiver Proxy

Receiver Proxy

- gets all `ByteMessage` chunks from Sender Proxy
- 1) stores the message chunks in `ByteBuffer`
- 2) checks if all chunks are present
 - if yes:
 - reassembles message's content
 - deserializes it with `KryoPool`
 - sends it to Receiver
 - if no: waits for more chunks

Solution Visualization



Critical Reflection of the Solution



- To add more fault tolerance and reliability:
 - Store messages in sender proxy until receiver proxy acknowledges
 - Add a timeout and retries until acknowledgement is received
 - Detection of missing or duplicate messages in receiver proxy
- To ensure autonomy of the process:
 - Use randomly generated UUIDs instead of AtomicCounter for IDs