LE 5 Message Queing

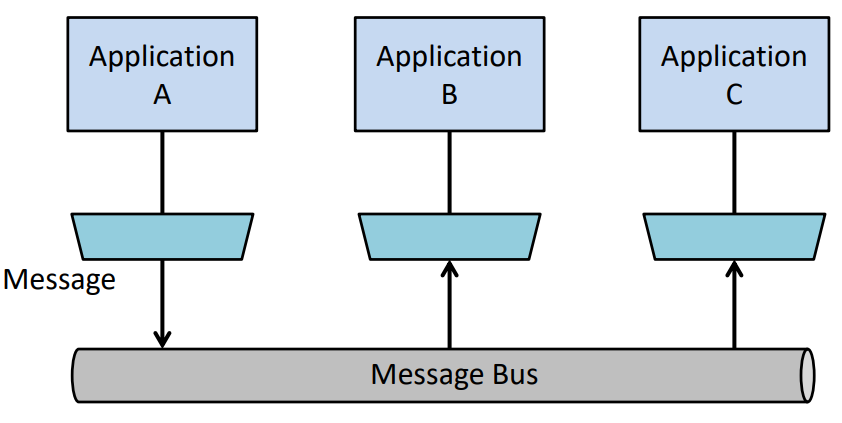
**Message Queuing Pattern**

**Problem** addressed: Enterprise has multiple applications, running on different platforms, developed in different languages:

* Share data across these applications
* Data updates need to propagate from one application to the other

Use **message (queueing) pattern** to transfer packets of data (messages) frequently, immediately, reliably, exactly once, and asynchronously, using customizable formats

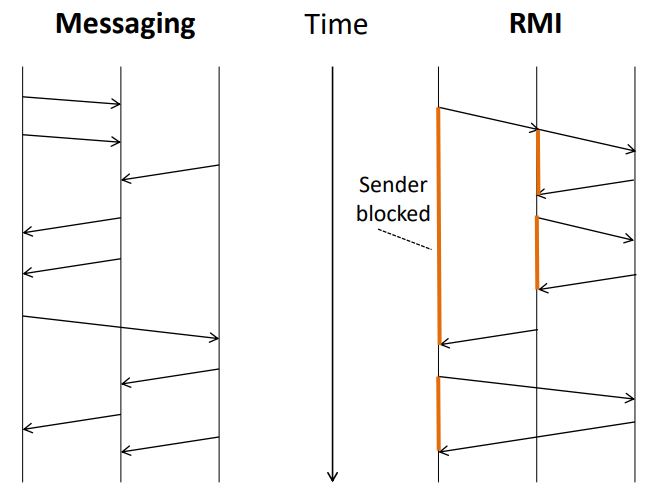
Enterprise Application Integration (EAI)



**Message Passing**

* Message besteht aus receiver, sender, type und payload
* Message passing is message queuing without message buffering
* Programming abstractions
  + send(message)
  + receive(message)
  + callback(…)
* Realization through middleware

**Messaging vs. Remote Method Invocation**



**RMI**

* Interface required and known, but differs across applications
* Programming model resembles non-remote calls
* Realizes request/reply pattern
* Sender blocks until reply arrives, unless there is an error
* As compared to non-remote call, more potential for failures

**Messaging**

* One interface (per messaging product) with generic operations
* Lower-level of abstraction than remote invocations
* Flexible in that messaging allows arbitrary interaction patterns between sender and receiver
* Sender is not blocked after sending
* Can emulate request/reply pattern
* Asynchronous behaviour more difficult to use and debug

**Message Passing Properties**

* Reliability, ordering, etc.
* Synchronous vs. asynchronous sending
* Synchronous vs. asynchronous receiving
* Buffering of messaging ⇒ Queuing

**Reliability: Message Loss**

* Unknown
* Identified
* Corrected
* Middleware mechanisms to prevent message loss
  + Sequence numbers
  + Acknowledgements (positive, negative)
  + Time-outs and re-transmission
* S12