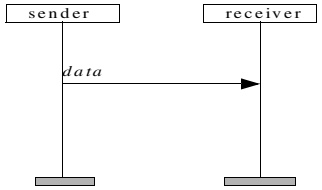
Modellierung der Basisdienste UDP und TCP

Die Ausgangssituation ist die folgende: Die Signale (z.B. *data*) werden zwischen den Protokollinstanzen *(Sender* und *Receiver)* über einen SDL-Kanal ausgetauscht. Die Ausgetauschten Signale werden auch PDU´s (Protocol Data Unit) genannt.



Diese in der Vorlesung Rechnernetze als "***virtuelle Kommunikation***" bezeichnete Architektur soll in diesem Entwicklungsschritt durch die ***"Dienstarchitektur***" ausgetauscht werden, in der die Protokollautomaten nicht direkt, sondern über einen Dienst (hier TCP oder UDP) kommunizieren. Die Signale, die zwischen den Protokollinstanzen und dem Basisdienst ausgetauscht werden (z.b. UDP_data.req), werden SDU´s (Service Data Unit) genannt.

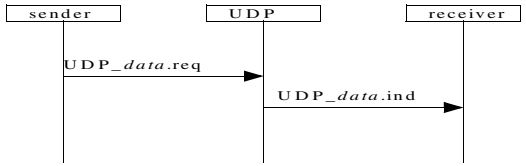
UDP:

Da UDP einen Datagrammdienst anbietet ist der Übergang nicht besonders schwierig: Eine PDU wird auf eine SDU (Datagramm) abgebildet. Das Versenden eines Diagramms wird mit *.req* und das Empfange mit *.ind* abgeschlossen. Um die namensbezogene Signalverarbeitung in SDL Prozessen zu behalten und auch keinen Dekodierer spezifizieren zu müssen, werden PDU-Signalname in dem SDU-Signalnahmen mitkodiert. So werden für PDU´s *data1* und *data2* vier unterschiedliche SDU´s : UDP_*data1*.req(ind) und UDP_*data2*.req(ind) definiert.

Signale:

- UDP_*data*.req
- UDP_*data*.ind

Verhalten



Fehlermodell

Unzuverlässiger Datagrammdienst: Datagramme können verloren und vertauscht werden.

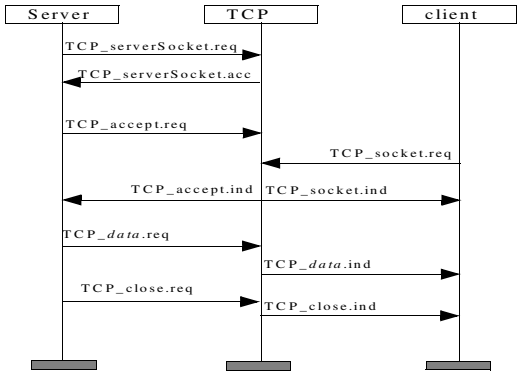
TCP

Die Abbildung auf den TCP-Basisdienst ist schwieriger, da man sich zusätzlich um die Verbindungsaufbau und Verbindungsabbau kümmern muß. Die Verbindungsaufbau ist nicht symmetrisch. Man unterscheidet zwischen dem Server (Wartet auf eine Verbindung) und dem Client (bemüht sich aktiv um eine Verbindung). Da TCP-Anbindung in JAVA implementiert werden soll, entsprechen auch die Signale der JAVA Socket-Schnittstelle.

Signale

	Signal	***Bedeutung***
Server	**TCP_serverSocket.req**	*Öffnen eines Server-Sockets, auf dem der Server auf die Verbindungen warten wird.*
	TCP_serverSocket.acc	
	TCP_serverSocket.ref	
	TCP_accept.req	*Blockierendes Warten auf eine Verbindung*
	TCP_accept.ind	
Client	**TCP_socket.req**	*Öffnen eines Sockets mit gleichzeitiger Verbindungsaufbau.*
	TCP_socket.acc	
	TCP_socket.ref	
Beide	**TCP_*data*.req**	*Verschicken von Daten*
	TCP_*data*.ind	
	TCP_close.req	*Schließen einer TCP-Verbindung*
	TCP_close.ind	

Verhalten:



Fehlermodell: TCP bietet einen zuverlässigen Transportdienst.