

# TCP/IP und UDP/IP – ist da sonst gar nichts mehr?

Bernd Reuther, Dirk Henrici

TU Kaiserslautern, Fachbereich Informatik  
AG Integrierte Kommunikationssysteme  
Paul-Ehrlich-Straße, 67663 Kaiserslautern  
{reuther, henrici}@informatik.uni-kl.de

**Zusammenfassung:** TCP/IP und UDP/IP sind heute die dominierenden Transport- und Netzwerkprotokolle. Es existieren zwar alternative Protokolle, diese werden jedoch in der Praxis nur selten genutzt. Problematisch ist, dass neue oder spezialisierte Protokolle explizit von den Anwendungen unterstützt werden müssen. Hier wird ein Modell vorgestellt, das Applikationen Transportdienste anbietet, wobei die verwendeten Protokolle für die Applikation transparent sind. Geeignete Protokolle werden unter Berücksichtigung der Ausführungsumgebung zur Laufzeit ausgewählt und konfiguriert.

## 1. Motivation

Das heutige Internet basiert wesentlich auf den Protokollen der TCP/IP-Suite. Das Internetprotokoll wurde erstmals 1980 als RFC veröffentlicht und ist inzwischen allgegenwärtig geworden, andere Layer-3-Protokolle wie beispielsweise Novells IPX sind selbst in lokalen Netzen beinahe komplett verdrängt worden. Eine derartige Alleinstellung haben auch die darauf aufsetzenden Transportprotokolle UDP (User Datagram Protocol) und TCP (Transmission Control Protocol) erlangt, obwohl schon frühzeitig mehrere alternative Protokolle zur Verfügung standen, z.B. RDP, TP4, ST2 oder XTP.

Die TCP/IP-Suite selbst blieb über die Jahre natürlich nicht unverändert. Zu den wichtigsten Veränderungen gehört die Weiterentwicklung der Flusskontrollmechanismen von TCP. Innerhalb der Forschung und für spezialisierte Anwendungen wurden eine große Zahl modifizierter Verfahren entwickelt (siehe z.B. [MA05] [SG06]). Weit verbreitet haben sich die TCP-Varianten mit den Bezeichnungen Tahoe, Reno, NewReno und BIC TCP. Den Neuerungen von Tahoe lag die Erkenntnis zugrunde, dass Paketverluste im Internet hauptsächlich auf Überlastsituationen im Netz zurück zu führen sind und nicht auf Bitfehler oder der Überlastung des Empfängers. Reno führte das *additive increase multiplicative decrease* (AIMD) Prinzip in TCP ein, bei dem die Bandbreite linear zunimmt aber im Fall von Überlast exponentiell abnimmt. Da Reno die Bandbreite im Fall einzelner verlorener Segmente weniger stark zurücknimmt als Tahoe, wird die Flusskontrolle von Reno auch als *Fast Recovery* bezeichnet. NewReno schließlich verbessert das Verhalten von Reno im Fall von mehreren verlorenen Paketen.

NewReno ist die zurzeit am häufigsten genutzte TCP-Variante. BIC TCP ist eine Variante von TCP, die insbesondere für die Übertragung großer Datenmengen konzipiert worden ist. Unter Linux wird seit der Kernelversion 2.6 BIC TCP verwendet.

Andere verbreitete Erweiterungen von TCP sind zum Beispiel das *selective acknowledgment* (SACK) Verfahren und die *window scaling option*:

- SACKS ermöglicht es, über TCP-Headeroptionen mehrere nicht zusammenhängende Blöcke von erfolgreich empfangenen Daten zu spezifizieren [MM96]. Damit kann einerseits der Verlust einer Folge von Paketen beschrieben werden und andererseits die erneute Übertragung bereits empfangener Pakete vermieden werden. SACKS wird inzwischen von den meisten TCP-Implementationen genutzt.
- Die TCP *window scale option* [JB92] ermöglicht es, einen Skalierungsfaktor für das nur 16-bit lange TCP-Feld „window size“ auszuhandeln, so dass für Netze mit großem Bandbreiten-Delay-Produkt größere Fenster verwendet werden können. Ein Anwendungsprogrammierer kann diese Option indirekt über die Größe des Socket-Buffers beeinflussen<sup>1</sup>.

Bemerkenswert ist, dass diese und andere Evolutionsschritte von TCP/IP transparent für die Applikationen sind und keine bzw. nur geringe Anpassung der Netzwerkinfrastruktur bedürfen. Entwicklungen wie der immer wieder diskutierte Übergang von IP Version 4 zu Version 6 oder auch nur die Einführung der *explicit congestion notification* (ECN) [RF01] sind dagegen weitaus schwerer zu realisieren, da diese eine Anpassung der Infrastruktur erfordern. Für die Unterstützung von IP Version 6 mussten außerdem viele Anwendungen angepasst werden.

Aktuell werden das *stream control transmission protocol* (SCTP) [SX00] und das *datagram congestion control protocol* (DCCP) [KH05] als Alternativen zu TCP bzw. UDP diskutiert. SCTP ist inzwischen fester Bestandteil aktueller Linux-Kerne. SCTP nutzt die gleichen Flusskontrollmechanismen wie TCP, bietet aber beispielsweise zusätzlich Multihoming, Unterscheidung mehrerer Streams oder hohe Resistenz gegen SYN-Flooding-Attacken [SA04]. DCCP ist wie UDP ein ungesichertes Protokoll und hat daher gute Verzögerungseigenschaften. Im Gegensatz zu UDP nutzt DCCP jedoch Flusskontrollmechanismen und erreicht so in Lastsituationen deutlich geringere Verlustraten.

Darüber hinaus existieren viele spezialisierte Transportprotokolle, die auf sehr unterschiedliche Einsatzzwecke abgestimmt worden sind. Beispielsweise für die Unterstützung von QoS (FPTP), wireless Netze (WTCP), interaktive Anwendungen (WebTP) oder Overlay-Netzwerke (TOP) [SE04].

Die Entwicklung neuer Protokolle ist oft durch neue Anwendungsfelder motiviert. So stellen beispielsweise Grid-Anwendungen besonderes hohe Anforderungen an den

---

<sup>1</sup> Die Abbildung der Socket-Buffergröße auf die window scaling option ist jedoch plattformabhängig.

Datendurchsatz, die zunehmende organisationsübergreifende Vernetzung von Business-Anwendungen erfordert besondere Sicherheit oder die schwankenden Bandbreiten und wechselnde Verfügbarkeit von Wireless-Netzwerken kann von spezialisierten Protokollen besser berücksichtigt werden.

Einerseits besteht also ein Bedarf für neue oder spezialisierte Protokolle und andererseits stehen auch Techniken zur Verfügung, die auf diesen Bedarf abgestimmt sind. Es stellt sich daher die Frage, warum andere Protokolle als die der TCP/IP-Suite trotzdem nur äußerst selten zum Einsatz kommen.

Wird ein Protokoll verändert oder ersetzt, so kann dies nicht isoliert an einer Stelle erfolgen, sondern alle miteinander kommunizierenden Instanzen müssen diese Veränderung nachvollziehen; andernfalls ist der Einsatz des Protokolls unmöglich. Änderungen auf der Ebene der Anwendungen (OSI-Schichten 5 bis 7) oder der Netzwerktechniken (OSI-Schichten 1 und 2) erfordern meistens nur Anpassungen in einer sehr begrenzten und damit beherrschbaren Domäne. Dementsprechend können sich Neuerungen innerhalb von Applikationen oder Netzwerktechniken erheblich leichter durchsetzen. Veränderungen an den Transport- oder Netzwerkprotokollen betreffen dagegen eine weitaus größere Infrastruktur. Zudem wird diese Infrastruktur von vielen prinzipiell unabhängigen Organisationen/Unternehmen betrieben. Veränderungen der Schicht 3 und 4 Protokolle sind daher nur mit hohem Koordinationsaufwand und mit erheblichen Kosten möglich. Ein Beispiel hierfür ist die seit über einem Jahrzehnt geführte Diskussion über die Einführung von IP Version 6.

Zusätzlich besteht ein vergleichbares Problem an der Schnittstelle zwischen den Transportprotokollen und deren Nutzern, also den Applikationen. Werden Protokolle verändert oder sogar ersetzt, so müssen auch deren Nutzer angepasst werden. Die Nutzer der TCP/IP-Suite sind jedoch eine Vielzahl von Applikationen, welche nur mit enormem Aufwand angepasst werden könnten. Obwohl die Veränderungen von IP Version 6 gegenüber Version 4 aus der Sicht der Nutzer eher gering ist, war es dennoch notwendig, viele Anwendungen anzupassen, da diese meist fest auf die 4 Byte langen Adressen von IP Version 4 ausgelegt waren. Wollte man heute TCP durch ein anderes Protokoll ersetzen, so müsste beinahe jede Anwendung, die Netzwerke nutzt, überarbeitet werden.

Es gibt jedoch Ausnahmen von dieser Problematik, nämlich wenn Veränderungen der Protokolle für andere Instanzen bzw. deren Nutzern transparent sind. So kann beispielsweise DiffServ oder MPLS auf einzelne Domänen beschränkt werden, d.h. ein Netzwerkbetreiber kann ohne Abstimmung mit anderen Betreibern entscheiden, diese Techniken einzusetzen. Gleiches gilt für die Evolution der TCP Flusskontrollmechanismen und die TCP-Erweiterungen SACK und window scale option. Auch für die Applikationen sind diese Entwicklungen von TCP transparent.

Die Schlussfolgerung hieraus ist, dass sich veränderte oder neue Transport- und Netzwerkprotokolle wesentlich leichter durchsetzen können oder zumindest leichter genutzt werden können, wenn diese:

1. transparent für die Applikationen sind
2. unabhängig von anderen Netzbetreibern eingesetzt werden können

Das Ziel der hier skizzierten Arbeit ist es, diese beiden Arten der Transparenz für beliebige Transport- und Netzwerkprotokolle zu ermöglichen. Ersteres wird durch eine Service-orientierte Sicht auf die Kommunikation erreicht, bei der die verwendeten Protokolle für die Applikationen generell transparent sind. Die Auswahl geeigneter Protokolle erst zur Laufzeit erlaubt es, spezialisierte Protokolle optional dort einzusetzen, wo es sinnvoll und möglich ist, aber alternativ Standardprotokolle zu verwenden, falls dies notwendig ist.

## **2. Lösungsansatz: Transparenz**

### **2.1. Transparenz der Protokolle**

Eine Service-orientierte Sichtweise auf Kommunikationssysteme ermöglicht es, Transport- und Netzwerkprotokolle für Applikationen transparent zu gestalten. Eine solche Sichtweise kann verwendet werden, da es für die Applikationen ausreichend ist, den Service zu kennen und nicht die zur Realisierung des Services verwendeten Mechanismen bzw. Protokolle. Für die Umsetzung der Service-orientierten Sicht auf Transportdienste ist folgendes notwendig:

- Transportdienste müssen beschrieben werden können, sowohl die von den Nutzern angeforderten als auch die von Dienstleistern angebotenen Dienste. Eine solche Beschreibung muss einerseits frei sein von protokollspezifischen Details, aber gleichzeitig müssen Dienste detailliert genug beschrieben werden können, um den am besten geeigneten Dienst für eine gegebene Anforderung zu bestimmen.
- Die Schnittstelle zwischen den Applikationen und den Transportdienstleistern darf ebenfalls keine protokollspezifischen Details an die Applikationen weitergeben oder von den Applikationen verlangen. Zwar ist die BSD-Socketschnittstelle selbst protokollunabhängig, allerdings leitet sie protokollspezifische Parameter von bzw. an die Applikationen weiter, beispielsweise das TCP\_NODELAY Flag oder IP-Nummern. Damit sind die verwendeten Protokolle nicht mehr transparent für die Applikationen und ein Austausch der Protokolle würde in der Regel die Anpassung der Applikationen erfordern.

Für die Netzbetreiber sind die von den Endsystemen verwendeten Transportprotokolle theoretisch transparent. In der Praxis ist es jedoch üblich, auch Daten der

Transportschicht innerhalb des Netzwerks auszuwerten, zum Beispiel innerhalb einer Firewall, in IDS oder für Accounting. Netzbetreiber bestimmen als nicht nur, welche Netzwerkprotokolle verwendet werden, sondern können auch die erlaubten Transportprotokolle einschränken. Daraus folgt, dass Veränderungen sowohl der Netzwerk- als auch der Transportprotokolle mit anderen Netzbetreibern abgesprochen werden müssen.

Bei der Größe des heutigen Internets ist der Prozess, ein Protokoll durch ein anderes zu ersetzen, jedoch äußerst aufwändig und langwierig. Der Übergang von IP Version 4 zu IP Version 6 ist ein gutes Beispiel dafür. Es kann daher davon ausgegangen werden, dass jedes neue Protokoll zunächst nur zusätzlich zu bestehenden Protokollen eingeführt wird und dies auch nur von wenigen Netzbetreibern. Neue Protokolle können unabhängig von anderen Netzbetreibern angeboten werden, wenn diese lediglich als Option zur Verfügung stehen. Damit solche optionalen Protokolle auch genutzt werden, muss seitens der Nutzer eine Auswahl zwischen verschiedenen alternativen Protokollen stattfinden. Dies geschieht heute bereits vereinzelt, z.B. können DNS-Clients oder die VoIP-Anwendung Skype TCP anstelle von UDP verwenden, falls eine Kommunikation mittels UDP nicht möglich ist [BS06]. Allerdings ist die Unterstützung mehrerer Protokolle für Applikationen aufwändig, da überprüft werden muss, welches Protokoll eingesetzt werden kann/soll. Des Weiteren können durch die Unterstützung weniger bestimmter Protokolle zukünftige Entwicklungen der Protokolle nicht berücksichtigt werden, so dass erneut die Anpassung von Applikationen notwendig wird.

## **2.2. Auswahl von Protokollen zur Laufzeit**

Die abstrakte Service-orientierte Sichtweise erfordert, dass vor der Kommunikation mit anderen Teilnehmern ein konkreter Dienstleister (Transport Service Provider, TSP) und dessen Konfiguration aufgrund einer Dienstbeschreibung ausgewählt wird. Die Auswahl basiert dabei auf den verfügbaren Informationen über die angebotenen Transportdienste. Um einen Dienst möglichst genau und vollständig beschreiben zu können, muss klar sein, wodurch ein angebotener Dienst beeinflusst werden kann (siehe Abbildung 1). Der Dienst wird natürlich wesentlich von den verwendeten Protokollen bestimmt. Zusätzlich hängt die Anwendbarkeit und Qualität eines Dienstes von den zur Verfügung stehenden Ressourcen in den Endsystemen und innerhalb der Netzwerk-Infrastruktur ab. Beispiele für Ressourcen sind die verfügbare CPU-Zeit, Bandbreite und Speicherplatz, die Anwendbarkeit kann dagegen durch Firewalls und Authentifizierungssysteme geregelt werden. Zusätzlich kann ein Transportdienst auch von Informationen über seine Nutzer abhängig sein. Applikation sind die direkten Nutzer von Transportdiensten; ist zum Beispiel deren Verkehrscharakteristik bekannt, so könnte ein Dienstleister a priori Ressourcen reservieren und so einen qualitativ verbesserten Dienst anbieten. Ebenso können auch Informationen über den Anwender die angebotenen Dienste beeinflussen. Ein Anwender mit der Rolle „Mitarbeiter“ darf eventuell andere Dienste nutzen als ein Anwender mit der Rolle „Gast“.

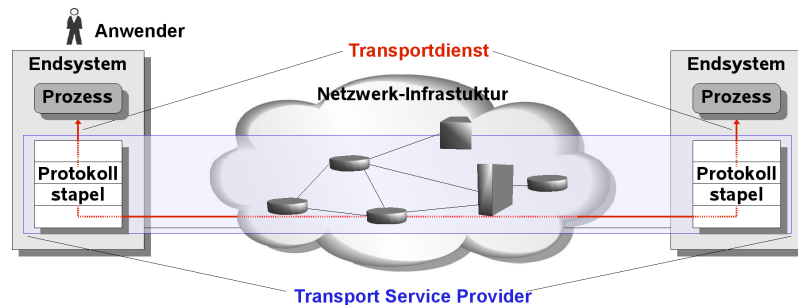


Abbildung 1: Transportdienste und Transport Service Provider

Informationen über die verfügbaren Ressourcen oder über den Nutzer sind jedoch nicht oder nur eingeschränkt vor der Laufzeit einer Applikation bekannt. Daher soll die Auswahl und Bindung an einen spezifischen TSP erst zur Laufzeit erfolgen. Eine solche dynamische Auswahl der TSP hat zusätzlich den Vorteil, dass Fallweise entschieden werden kann, ob neue Protokolle zum Einsatz kommen können (z.B. innerhalb eines lokalen Netzwerkes) oder ob weit verbreitete Protokolle zum Einsatz kommen sollen.

### 3. Modell eines Service-orientierten Kommunikationssystems

Die Abbildung 2 zeigt das vorgeschlagene Modell eines Service-orientierten Kommunikationssystems. Die zentrale Komponente ist der Broker; dieser vermittelt Dienstangebote (1.) auf der Basis von Dienstanforderungen seitens der Nutzer (2.). Das Ergebnis der Vermittlung wird an den oder die ausgewählten Protokollstack(s) weiter gereicht (3.). Nachdem eine Kommunikationsbeziehung initialisiert wurde, kann die Applikation Daten mit anderen Systemen austauschen (4.).

Die Aufgabe der API ist es, die Applikationen von den Protokollen zu entkoppeln. Sie leitet Aufrufe der Applikation entweder an den Broker oder an die Protokolle weiter. Dafür ist es notwendig, dass jeder Protokollstack über ein einheitliches Service Provider Interface (SPI) angesprochen werden kann. Ein Adapter bildet die Funktionalität des SPI auf die jeweiligen protokollspezifischen Funktionen ab. Dies ermöglicht es, viele existierende Protokollstacks einzubinden, ohne die Protokolle selbst an das SPI anzupassen. Der Adapter kann neben den Protokollstacks auch spezifische Funktionalitäten der Netzwerkumgebung nutzen, z.B. Freischalten von Ports in einer Firewall über ein Midcom-Protokoll [SK02].

Die Spezifikation der Dienstangebote bestehen aus einem statischen Teil, der die Protokollstacks beschreibt und einem dynamischen Anteil, der – soweit verfügbar – Zustandsdaten über die Endsysteme oder das Netzwerk liefert. Anforderungen werden von den Applikationen über die API and den Broker weitergereicht. Eine Applikation kann es dem Anwender ermöglichen, auf die Dienstanforderung Einfluss zu nehmen. Alternativ können Daten über den Nutzer auch separat über ein User-Frontend zur Verfügung gestellt werden, z.B. Daten für eine Authentifizierung.

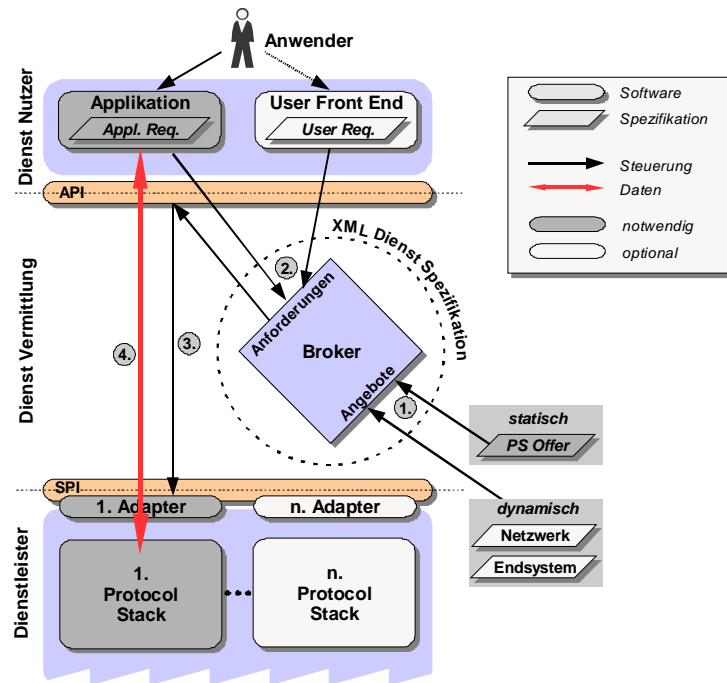


Abbildung 2: Modell eines Service-orientierten Kommunikationssystems

### 3.1. Service-orientierte API

Die API muss alle protokollspezifischen Details vor den Applikationen verstecken. Nach dem vorgestellten Modell reduziert sich die direkte Interaktion zwischen den Applikationen und den Protokollen auf den Datentransfer, d.h. auf Lese- und Schreiboperationen. Welche Datentypen dabei übergeben werden, wird zuvor mit dem Broker ausgehandelt. Alle anderen Funktionalitäten werden auf Interaktionen mit dem Broker abgebildet und sind daher nicht protokollspezifisch. Im Fall einer Server-Applikation melden die TSP den Applikationen zusätzlich noch neue Kommunikationsbeziehungen. Ebenso müssen natürlich Fehlermeldungen der Protokolle an die Applikationen weitergegeben werden.

Jede Kommunikationsbeziehung wird von der API durch ein eigenes Objekt (*flow*) repräsentiert. Falls ein verbindungsorientiertes Transportprotokoll verwendet wird, so entspricht ein *flow* genau einer Verbindung. Ein *flow* kann ebenso Kommunikationsbeziehungen verbindungsloser Protokolle repräsentieren. In diesem Fall wird ein *flow*-Objekt automatisch erzeugt, sobald Daten von einem Kommunikationspartner eintreffen. Ob ein *flow* Daten von nur einem oder von verschiedenen Kommunikationspartnern entgegen nehmen oder senden kann, hängt davon ab, ob der *flow* als unicast oder multicast eingerichtet wurde [HR04].

### 3.2. Spezifikation von Transportdiensten

Die Spezifikation der Transportdienste soll flexibel bezüglich der Angaben sein, die über Transportdienste gemacht werden. So sollten Dienstangebote so genau wie möglich beschrieben werden können; Dienstanforderungen dagegen sollten nicht mehr Angaben enthalten als notwendig. Um jederzeit auch neue Protokolle unterstützen zu können, muss die Spezifikationsform erweiterbar sein, ohne dass dazu Modifikationen des Brokers notwendig werden.

Hier wird daher vorgeschlagen, Dienste anhand von Mengen von Eigenschaften zu beschreiben. Festgelegt wird nur die Syntax, mit der eine einzelne Eigenschaft beschrieben wird; die Semantik der Eigenschaften wird durch Referenzen (URI) beschrieben, so dass jederzeit neue Eigenschaften definiert werden können. Ein Dienst kann dann wahlweise durch viele Eigenschaften detailliert oder durch wenige Eigenschaften grob beschrieben werden.

Für die Auswahl von Diensten ist es essentiell, entscheiden zu können, ob ein Dienstangebot alle notwendigen Eigenschaften einer Dienstanforderung erfüllt. Es werden daher zwei Typen von Eigenschaften unterschieden: notwendige und wünschenswerte Eigenschaften (siehe auch [RH04]). Die notwendigen Eigenschaften stellen keine Bewertung der Dienste dar, sie sind entweder erfüllt oder nicht erfüllt. Die wünschenswerten Eigenschaften dagegen dienen explizit der Bewertung von Diensten.

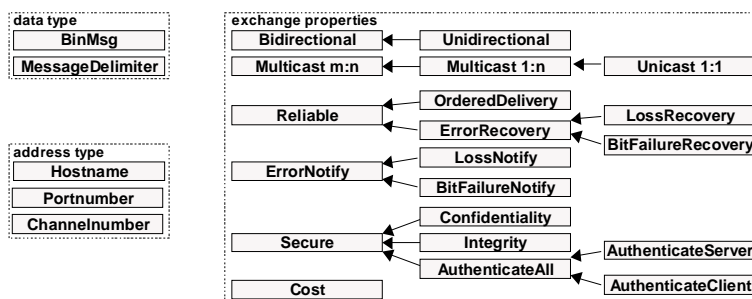


Abbildung 3: Beispiele für notwendige Eigenschaften

Die Abbildung 3 listet Beispiele für notwendige Eigenschaften auf. Diese beschreiben die unterstützten Daten- und Adresstypen sowie verschiedene Merkmale des Datenaustausches. Notwendige Eigenschaften können Ober- und Untergrenzen spezifizieren, die eingehalten werden müssen, z.B. kann für den Datentyp „binäre Nachricht“ eine minimale MTU in Anforderungen bzw. eine maximale MTU in Angeboten spezifiziert werden. Eine notwendige Eigenschaft wird dann durch das Tripel  $\langle URI, lb, ub \rangle$  spezifiziert. Der URI ist eine Referenz auf die Semantik der Eigenschaft. Untere und obere Grenzwerte werden durch  $lb, ub \in \mathfrak{X}$  beschrieben. In Anforderungen wird das Intervall interpretiert als  $\exists i \in [lb, ub]$  und in Angeboten als  $\forall i \in [lb, ub]$ . Damit eine Eigenschaft eines Angebotes eine geforderte notwendige Eigenschaft erfüllt, müssen deren URI übereinstimmen und die Intervalle in mindestens einem Punkt überlappen.

Die Pfeile in der Abbildung 3 deuten Verallgemeinerungen an. Eine Protokoll, welches bidirektionale Kommunikation unterstützt, kann auch für unidirektionale Kommunikation genutzt werden, oder die Eigenschaft „Sicherheit“ wird als Oberbegriff für die Eigenschaften Vertraulichkeit, Integrität und Authentifizierung verwendet.

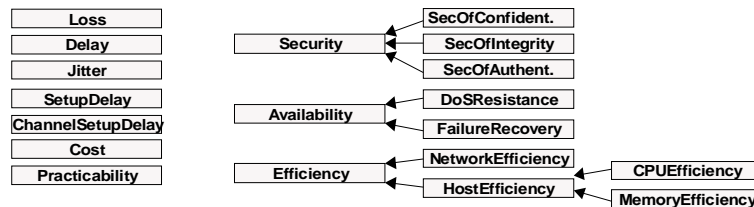


Abbildung 4: Beispiele für wünschenswerte Eigenschaften

Die Abbildung 4 zeigt Beispiele für wünschenswerte Eigenschaften. Neben den klassischen Qualitätsmerkmalen: Verlustrate, Verzögerung und Verzögerungsschwankung, können auch die Dauer der Initialisierung (SetupDelay), Kosten oder Bewertungen für Sicherheit oder Effizienz genannt werden. Mit der Eigenschaft „Practicability“ wird beschrieben, mit welcher Wahrscheinlichkeit ein Dienstleister genutzt werden kann. Sehr spezifische Protokolle können nicht in jeder Umgebung angewendet werden, so dass deren Einsatz fehlschlagen kann und ein anderer Dienstleister gewählt werden muss. In Anforderungen drückt die Eigenschaft „Practicability“ die Bereitschaft zur Verwendung spezieller oder neuer Protokolle aus.

In Dienstangeboten wird die Bewertung aller wünschenswerten Eigenschaften als abstrakte Qualität  $q \in [0,1]$  angegeben. Ein solches einheitliches Bewertungsmaß ermöglicht es, unterschiedliche Eigenschaften wie Verzögerung und Sicherheit miteinander zu vergleichen. Dabei gilt, dass  $q=0$  die „schlechteste“ und  $q=1$  die „beste“ Ausprägung einer Eigenschaft beschreibt. Für jede wünschenswerte Eigenschaft wird separat festgelegt, wie die abstrakte Qualität  $q$  bestimmt wird. Idealerweise werden dazu messbare eigenschaftsspezifische Maße, wie Millisekunden für Delay und Jitter oder Prozent für Lossrate, auf das Intervall  $[0,1]$  abgebildet. Optional kann in Anforderungen und Angeboten angegeben werden, welche Werte der eigenschaftsspezifischen Einheit einem  $q=0$  bzw  $q=1$  entsprechen, so dass die Interpretationen von „bester“ bzw. „schlechtester“ Ausprägung einer Eigenschaft zwischen den Anforderungen und Angeboten abgeglichen werden können (siehe auch [RH06]).

In der Spezifikation einer Dienstanforderung legt ein Nutzer Gewichte für einzelne Eigenschaften fest und bestimmt damit, nach welchen Kriterien ein Dienst ausgewählt bzw. optimiert wird.

### 3.3. Spezifikation der Dienste von Transport Service Providern

Theoretisch können nach der in Kapitel 3.2 vorgestellten Weise alle angebotenen Transportdienste beschrieben werden. Allerdings kann jeder Transport Service Provider (TSP) abhängig von seiner Konfiguration, von Daten über die Applikation, dem Anwender und den Zuständen der Netzwerk-Infrastruktur sehr viele verschiedene

Dienste anbieten. Es ist daher wenig praktikabel, die Menge aller Dienste eines TSP einzeln aufzulisten. Stattdessen werden die Einflussfaktoren auf den Dienst eines TSP explizit beschrieben und zur Laufzeit ausgewertet.

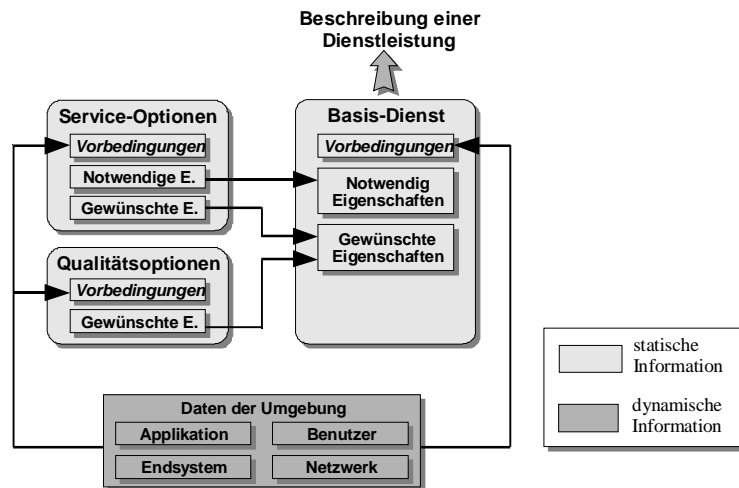


Abbildung 5: Modell zur Beschreibung der Dienste eines TSP

Die Abbildung 5 zeigt das Modell zur Beschreibung der Dienste von TSP. Jeder TSP bietet einen Basisdienst an, welcher der Standard-Konfiguration entspricht und keine Informationen über die Ausführungsumgebung voraussetzt. *Service-Optionen* beschreiben Veränderungen des Basisdienstes durch wählbare Konfigurationen. Durch die Auswahl von Service-Optionen kann ein Dienst optimiert werden. Je zwei Service-Optionen können dabei unabhängig voneinander sein oder sich gegenseitig ausschließen. Die Abhängigkeit von Gegebenheit der Ausführungsumgebung wird durch *Qualitäts-Optionen* beschrieben. Die Ausführungsumgebung kann die Eigenschaften des Dienstes beeinflussen oder sogar die Verwendung eines Dienstes oder einer Service-Option verhindern. Solche Abhängigkeiten werden durch Vorbedingungen in Form von booleschen Ausdrücken beschrieben. Der Broker hat die Aufgabe, zur Laufzeit solche Beschreibungen der TSP auszuwerten und durch die Wahl geeigneter Service-Optionen eine optimale Konfiguration der TSP zu finden. Der Basisdienst und die angewendeten Optionen liefern schließlich eine Dienstbeschreibung, wie sie im vorangegangenen Kapitel vorgestellt worden ist.

### 3.4. Auswahl von Diensten

Die Auswahl von Diensten geschieht in drei Schritten: Auswertung der Vorbedingungen, Optimierung der Dienstleistung der TSP und Auswahl eines oder mehrerer geeigneter Dienste. Durch die Auswertung der Vorbedingungen in den Beschreibungen der TSP, werden die Spezifikationen an die jeweilige Umgebung angepasst, soweit hierfür die notwendigen Informationen vorliegen. Zum einen können die verfügbaren TSP und Service-Optionen eingeschränkt werden aber auch Eigenschaften durch Qualitäts-

Optionen verändert werden. Abhängig von den gegebenen Anforderungen wird durch die Wahl geeigneter Service-Optionen der Dienst eines TSP optimiert. TSP, die nach diesem Schritt nicht alle geforderten notwendigen Eigenschaften erfüllen, werden nicht weiter betrachtet.

Falls eine Kommunikationsbeziehung aktiv initialisiert wird (Rolle eines Clients), dann wird der Beste der verbleibenden Dienste ausgewählt und genutzt. Falls dessen Anwendung fehlschlägt, wird auf den nächsten Dienst zurückgegriffen. Dies bedeutet aber auch, dass unzureichende Information über die Umgebung vorlagen. Im Fall einer passiven Initialisierung von Kommunikationsbeziehungen (Rolle eines Servers) werden alle verbleibenden TSP für die Kommunikation initialisiert. Sollten viele TSP zur Verfügung stehen, so empfiehlt es sich, nur eine begrenzte Anzahl auszuwählen und Standard-TSP (z.B. TCP/IP) zu markieren, damit diese immer genutzt werden, sobald sie alle notwendigen Anforderungen erfüllen.

#### **4. Ausblick**

Auf der Transport- und Netzwerkebene werden heute nahezu ausschließlich die Protokolle der TCP/IP-Suite verwendet. Es findet zwar eine Evolution der Protokolle statt, jedoch ist diese im Vergleich zu denen der Netzwerktechniken oder höheren Protokollen sehr langsam. Alternative Protokolle stehen zwar zur Verfügung, doch deren Verwendung ist für Applikationen mit erheblichem Aufwand verbunden, denn die Verfügbarkeit alternativer Protokolle kann im Allgemeinen nicht vorausgesetzt werden, so dass Applikationen explizit für die Unterstützung mehrerer Protokolle ausgelegt werden müssten.

Es hat sich in der Vergangenheit gezeigt, dass technische Neuerungen der Transport- und Netzwerkprotokolle umso leichter durchsetzbar waren, als sie transparent für Applikationen und unabhängig von anderen Netzwerkbetreibern eingeführt werden konnten. Als Konsequenz wurde hier ein Konzept vorgestellt, um Applikationen von den Protokollen zu entkoppeln, so dass Veränderungen oder gar ein Austausch der Protokolle keine Anpassung der Applikationen mehr erfordert. Ein Broker wählt aufgrund von Dienstbeschreibungen Protokolle für die Applikation aus. Die Auswahl geschieht zur Laufzeit, da erst zu diesem Zeitpunkt Informationen über die Ausführungsumgebung vorliegen können. In der Praxis liegen in heutigen Systemen jedoch nur wenige Informationen über die Verkehrscharakteristik von Applikationen oder über den Zustand der verwendeten Netzwerkinfrastruktur vor. Letzteres beschränkt sich meist auf Informationen über die Anbindung eines Endsystems oder es kann unterschieden werden, ob Kommunikation mit Systemen innerhalb oder außerhalb eines LANs stattfindet. Gegebenenfalls können frühere Kommunikationsbeziehungen ausgewertet werden, um Anhaltspunkte über zu erwartenden Netzwerkeigenschaften zu liefern.

Die Beschreibung wünschenswerter Eigenschaften beinhaltet die vergleichende Bewertung von Protokollen in Bezug auf jeweils eine Eigenschaft. Solche Bewertungen sollten nach möglichst objektiven Kriterien erfolgen. Die Bewertung von Protokollen ist im Allgemeinen jedoch nicht trivial und wird oft von bestimmten Szenarien abhängig

gemacht (siehe auch [FL]). Die separate Bewertung einzelner Eigenschaften kann die Bewertung vereinfachen, allerdings existieren auch hierfür bislang keine anerkannten Benchmarks oder gar Standards.

## 5. Literaturverzeichnis

- [BS06] S. A. Baset, H. Schulzrinne: „An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol“ INFOCOM 2006.
- [FL] S. Floyd et.al. Web-Site: „The Transport Modeling Research Group“ (<http://www.icir.org/tmrg/>) The ICSI Center for Internet Research.
- [JB92] V. Jacobson, R. Braden, D. Borman: „TCP Extensions for High Performance“ IETF RFC 1323 Mai 1992.
- [HR04] D. Henrici, B. Reuther: „A Unified, Protocol Independent API for Connection-Oriented and Connection-Less Protocols“, Invited Session hold at the 8th World Multi-Conference on Systemics, Cybernetics and Informatics 2004.
- [KH05] E. Kohler, M. Handley, S. Floyd: „Datagram Congestion Control Protocol (DCCP)“, IETF draft-ietf-dccp-spec-13 Dezember 2005.
- [MA05] A. Medina, M. Allman, S. Floyd: „Measuring the evolution of transport protocols in the internet“ ACM SIGCOMM Computer Communication Review archive, Volume 35, Issue 2, April 2005.
- [MM96] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow: „TCP Selective Acknowledgement Options“, IETF RFC 2018, October 1996.
- [RF01] K. Ramakrishnan, S. Floyd, D. Black.: „The Addition of Explicit Congestion Notification (ECN) to IP“ IETF RFC 3168 September 2001.
- [RH04] B. Reuther, D. Henrici, M. Hillenbrand: „DANCE: Dynamic Application Oriented Network Services“, 30th EUROMICRO 2004.
- [RH06] B. Reuther, D. Henrici: „A Model for Service-Oriented Communication Systems“, to appear, 32th EUROMICRO 2006.
- [SA04] R. Stewart, P. D. Amer: „Why is SCTP needed given TCP and UDP are widely available?“ ISOC member briefing #17, Juni 2004 (<http://www.isoc.org/briefings>).
- [SE04] P. Sénac, E. Exposito, A. Seneviratne, S. Ardon, T. Rakotoarivelo „New ambitions for the transport layer“, Proceedings of Interworking 2004.
- [SG06] H. Schwier, C. Grimm: „Analyse von TCP-Varianten“, D-Grid Integrationsprojekt, Fachgebiet 3 (<http://www.d-grid.de>).
- [SK02] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, A. Rayhan: „Middlebox communication architecture and framework“. IETF RFC 3303 August 2002.
- [SX00] R. Stewart, Q. Xie, et.al.: „Stream Control Transmission Protocol“, IETF RFC2960, Oktober 2000.