

Kommunikation und verteilte Systeme – Zusammenfassung

Verteilte Systeme:

Kooperation von unabhängigen Computern

Kein Failure point

Key Characteristics: Ressource sharing, openness, concurrency, scalability, fault tolerance, transparency

Vorteile von Verteilte Systeme gegenüber Mainframes:

- Wirtschaftlichkeit (besseres Preis/Leistungs- Verhältnis von Mikroprozessoren)
- Leistung & Zuverlässigkeit (VS haben bessere Gesamtleistung und der Ausfall einer Komponente führt nicht zum Ausfall des ganzen Systems)
- Skalierbarkeit & Flexibilität (schrittweise Erhöhung der verarbeitungs- Leistung sowie funktions und Lastausgleich sind möglich)
- Inhärente Verteilung (gewisse Applikationen müssen verteilt sein)
- Gemeinsame Daten und Geräte (gemeinsame Nutzung von Daten und von teuren Peripheriegeräten)
- Software (wesentlich billiger)
- Netzwerk (Flaschenhals-Situationen, Verlust von Daten)
- Sicherheit (komplexer, schwer administrierbar, dafür angriffresistenter)
- Daten (Inkonsistenz von Daten und Systemzustände ist unvermeidbar)

Transparenz: Informatik vs. Real leben:

Informatik:

- Ort der HW/SW unbekannt/irrelevant
- Anwender nutzen Ressourcen gemeinsam, erfahren aber nichts voneinander
- Maskierung von Fehler, graceful degradation
- Aktivitäten für den Anwender könne parallel ausgeführt werden
- unabhängig von qualitativen und quantitativen Systemänderungen
- zugriff jederzeit möglich

Alltag:

- gläserne maschine
- Ablauftransparenz
- Anwenderkontrolle über Datenverarbeitung
- persistente Dokumentation der Abläufe
- einfache Einsichtnahme und Verstehbarkeit

Computernetzwerke

Direktverbindung:

Punkt zu Punkt: viele Verbindungen zwischen einzelnen paaren von maschinen. Um von der quelle ans ziel zu gelangen, muss ein Paket zuerst eine oder mehrere dazwischenlegende maschinen besuchen. Da mehrere Routen mit unterschiedlicher Länge möglich sind, spielen routing Algorithmen eine wichtige rolle.

Kleine netze: Broadcasting

Grosse verteilte netze: punkt zu punkt Verbindung

Mehrfachzugriffverbindung

Leistungsvermitteldes vs. Paketvermitteldes Netzwerk

Zusammenschluss von Netzwerken über Router (Knoten oder Vermittler in einem verbindungslosen Paketvermitteldes Netz (z.B. IP-Netz), dienen als Verbindungspunkte zwischen LAN-Segmenten, oder als Gateways zwischen LANs und WANs) und Gateways.

Unicast:

Broadcast: durch ein speziellen Code im Adressfeld wird das Paket an alle ziele adressiert. Jede maschine im netz nimmt es an und verarbeitet es.

Multicasting: Übertragung an einer Teilmenge der angeschlossenen maschinen

Gemeinsame Ressourcen:

Multiplexen: aufteilen einer Systemressource auf mehrere Benutzer, erforderlich auf der Bitübertragungsschicht, wo der gesamte verkehr aller Verbindungen über möglichst wenige Leitungen geschickt werden muss. Mehrere Verbindungen auf dem gleichen Übertragungsmedium zu bedienen.

Demultiplexen: Ressourcenfreigabe, z.B. gemultiplexten Datenstrom am ende des Kanals in Einzelströme zerlegen

Frequency Division Multiplexing (FDM): vergibt zu jeden Strom ein dediziertes Frequenzband. Z.B. bei Telefon bekommt jeder Benutzer 3000 Hz pro Sprachkanal. Die Sprachkanäle werden zuerst in die Frequenz mit je einem anderen umfang angeboten. Dann werden sie gemischt, weil jetzt nie 2 Kanäle den gleichen Spektrumsbereich nutzen. Das zu übertragende Signal moduliert ein anderes Signal, das in einer viel höheren Frequenz gesendet wird. Jede Trägerfrequenz ist eindeutig. Time Division Multiplexing (TDM): vergibt für jeder Verbindung einen dedizierten Zeitschlitz. Haben die Zeitschlitze eine feste Dauer und werden sie in regelmässigen abständen zugewiesen, spricht man von einem synchronen zeitmultiplexing.

Code Division multiple Access (CDMA): alle Stationen könne gleichzeitig senden, aber in verschiedenen unterräume. Ermöglicht jeder Station die ganze zeit, über das gesamte Frequenzspektrum zu übertragen. Mehrere gleichzeitige Übertragungen werden mittels Kodierschema getrennt. CDMA beseitigt auch die Annahme, dass kollidierende rahmen völlig verstümmelt werden. Es wird davon ausgegangen, dass mehrere Signale linear hinzugefügt werden. Z.B. an einer party: CDMA trifft zu, wenn jedes paar eine andere Sprache spricht, sodass sich schlussendlich niemand stört.. durch CDMA ist es also möglich, das gewünschte Signal herauszuziehen und alles andere als Nebengeräusch abzutun.

Gemeinsame Ressourcen

Mehrfachzugriffprotokolle

Datenübertragung nach RoundRobin: zeitliche statische Aufteilung eines Broadcast Netzes in einzelne Intervalle und die Gewährung eines zeitgesteuerten Algorithmus, bei dem jede maschine erst senden darf, wenn ihr Zeitschlitz an die reihe kommt. Es wird Kanalkapazität verschwendet, wenn eine maschine an die reihe kommt, und diese nichts zu senden hat.

Datenübertragung auf anfrage:

Übertragung von Paketen statt von ganzen nachrichten

Paketweise Kanalzuordnung und mögliche Garantie einer Dienstgüte für spezielle Datenströme (QoS)

Zwischenspeichern in Switches wenn mehr Pakete ankommen als abgehen → Überlastungsgefahr → führt zum wegwerfen von Paketen.

Grundmodell zum Leistungsvergleich

Stationen-Modell: N unabhängige Stationen, die Pakete zur Übertragung mit Wahrscheinlichkeit $\lambda\Delta t$ erzeugen.

Einzelkanalannahme: gleichzeitige Übertragung führt zu Kollisionen und verlangt eine Neuübertragung beteiligter Pakete, alle Stationen können Kollisionen erkennen.

Zeitunterteilung: nicht unterteilte Zeit (jederzeit Übertragungsbeginn möglich) vs. in Zeitschlitz unterteilte Zeit (Übertragungsbeginn am Anfang eines Schlitzes)

Träger: Trägererkennung (carrier sense, z.B. 1-Persistent CSMA, Stationen können feststellen ob der Kanal in Gebrauch ist) vs. keine Trägererkennung.

Gemeinsame Dienste:

Anwendungsprogramme benötigen:

Netzwerkdienste um die Komplexität des Netzwerkes zu verbergen

Maskierung von Ausfällen:

Bitfehler und Burstfehler

Paketverlust (durch Bitfehler, Überlast, Software Fehler)

Ausfall von Knoten

Anforderung → hohe Leistung:

Kleine Latenz: grosse Bandbreite, kleine Ausbreitungsverzögerung, kurze Wartezeiten

Kleine Jitter: geringe Latenzschwankungen

Bandbreite: in Hz = Frequenzbereich,

Bandbreite einer Kommunikationsleitung: Gesamtkapazität des Netzes die in bps ausgedrückt wird, wie viel Bits pro Sekunde können übertragen werden?

Durchsatz: gemessene Übertragungsleistung zwischen 2 Hosts, gemessen in Bits/sec

Latenz einer Verbindung: Dauer, die eine Nachricht von einem Ende einer Leitung zum anderen benötigt = Ausbreitungsverzögerung + Übertragungsverzögerung + Wartezeit

Ausbreitungsverzögerung: Entfernung / (mediumspezifische) Lichtgeschwindigkeit

Übertragungsverzögerung: Paketgröße / Bandbreite

RTT: Round Trip time, von A nach B und zurück

Verzögerungs-Bandbreite-Produkt: wie viel Bits passen in ein Kanal ohne dass am Ende des Kanals ein Bit rauskommt.

Netzwerk Architektur

→ Leitfaden für Entwicklung, Implementierung von Netzwerken, Eigenschaften:

kostengünstig, fair, robuste Konnektivität für eine grosse Anzahl von Computern.

→ Strategie: Einführung von Abstraktionen um Komplexität zu verringern. → Definition von Modellen, die die wichtigsten Aspekte des Systems erfassen → Kapselung der Modelle in Objekte, die eine Schnittstelle zur Manipulation durch andere Systemkomponenten offerieren (die Implementierung bleibt aber verborgen)

Schichtenmodell

Um die Komplexität zu verringern, werden netze als Reihe von übereinandergestapelten schichten aufgebaut. Die Schichten haben den Zweck, den jeweils höheren schichten bestimmte dienste anzubieten (diese schichten aber mit den Einzelheiten, wie die dienste implementiert werden, zu verschonen) und greifen auf die dienste der darunterliegenden schicht über deren Schnittstelle zu.

Vorteil: Zerlegung in leichter handhabbare, potentiell austauschbare Komponenten.

Schicht n auf einer maschine führt ein Gespräch mit schicht n der anderen maschine. Regeln und Konventionen werden Protokoll genannt.

Es werden aber keine Daten direkt von schicht n einer maschine auf schicht n einer anderen maschine übertragen. Jede schicht leitet Daten und Steuerinformationen an die unmittelbar darunterliegenden schicht weiter, bis die unterste schicht erreicht ist. Unter schicht 1 befindet sich das physische Medium, über das die Kommunikation stattfindet.

Zwischen 2 angrenzenden schichten befindet sich eine Schnittstelle. Diese definiert, welche Operationen und Dienste die untere der oberen schichten anbietet. Jede schicht führt eine bestimmte Anzahl von wohlverstandenen Funktionen aus.

Netzarchitektur: Gruppe von schichten und Protokolle.

Protokolle: abstrakte Objekte, aus denen ein Netzwerksystem zusammengesetzt ist.

Protokollstack: folge von benutzten Protokollen.

Beispiel:

Von einem Anwendungsprozess auf schicht 5 wird Nachricht M produziert und an schicht 4 zur übertragen abgegeben. Schicht 4 setzt einen Header vor der Nachricht, um sie zu identifizieren, dann gibt sie das Ergebnis an schicht 3 ab. Der Header beinhaltet Steuerinformationen, z.B. laufende Nummern. Dadurch kann schicht 4 an der Zielmaschine die nachrichten in der richtigen Reihenfolge zustellen, falls die unteren schichte die Reihenfolge nicht einhält.

Schicht 3 teilt dann z.B. die Nachricht in Pakete auf und versehen jedes Paket mit einem Header, dann werden sie an schicht 2 weitergeleitet. Diese fügt nicht nur ein Header, sondern auch ein Trailer ein (Endanweisung) und gibt die so zusammengestellte Einheit an schicht 1 zur physischen übertragen weiter (wobei hier multiplexen wichtig wird).

In der Empfängermaschine wandert die Nachricht von schicht zu schicht nach oben, und die Header werden nacheinander wieder entfernt.

Die niedrigen schichten einer Protokoll Hierarchie werden häufig in Hardware oder Firmware implementiert.

Designprinzipien

In jeder schicht muss es ein Mechanismus für den Verbindungsaufbau geben.

Ein Prozess muss deutlich machen, mit wem er Verbindung aufnehmen will. → form der Adressierung ist erforderlich

Regeln des Datentransfers: Simplex, halb Duplex, Full Duplex

Fehlererkennung und –Korrektur Codes

Wie müssen zerstückelte nachrichten wieder in der richtigen Reihenfolge zusammengesetzt werden.

Schnelle Sender müssen gehindert werden, langsame Empfänger mit Daten zu überschwemmen.

Aufteilung der Nachricht in Pakete und deren Zusammensetzung.

Multiplexen, Demultiplexen.

Verbindungsorientierte und verbindungslose dienste

Schichten können den jeweils übergeordneten schichten zwei verschiedenen Dienstarten bieten: Verbindungsorientierte und verbindungslose.

Ein verbindungsorientierter (quasi pessimistisch) dienst ist nach dem Telefonsystem ausgelegt. Der dienstnutzer baut eine Verbindung auf, benützt diese Verbindung und löst sie wieder. Der wichtige Aspekt der Verbindung ist, dass sie wie ein Rohr funktioniert: der Sender schiebt Bits an einem ende hinein, und der Empfänger entnimmt sie am anderen ende in der gleichen Reihenfolge. Kurze Paketbezeichnung, teurer Verbindungsaufbau, Speicherung von Verbindungsaufbau, alle Verbindungen über Router bei Ausfall beendet, einfachere Überlastüberwachung. z.B. TCP (teuer, exactly once, Beibehaltung der Reihenfolge).

Ein verbindungsloser dienst basiert auf das Postsystem, jede Nachricht trägt eine volle Adresse und wird unabhängig von allen anderen durch das System geschleust. Normalerweise kommt auch diejenige Nachricht, die zuerst abgeschickt wurde, auch zuerst an. Es kann aber passieren, dass sich diese verzögert sodass die zweite als erste ankommt. Dass ist im Verbindungsorientiertem dienst unmöglich. Jedes Paket volle Adresse, kein Verbindungsaufbau, keine Statusinformationen, unabhängiges Routing, geringe folgen bei Routefehler, schwierige Überlastüberwachung. z.B. UDP (billig, Paketverlust, Paketverdoppelung, kein FIFO)

Jeder dienst kann nach einer Dienstqualität klassifiziert werden (Daten verlieren bzw. nie verlieren). Bei einem zuverlässigen dienst wird jeder erhalt einer Nachricht bestätigt, sodass der Sender die Gewissheit hat, dass die Nachricht auch wirklich angekommen ist. Z.B. für FTP kommt ein zuverlässiger Verbindungsorientierter dienst in frage, bei sprach- oder Filmübertragung stören Wartezeiten viel mehr als kleine Artefakte.

Das OSI-Referenzmodell:

Wurde von der ISO entwickelt und ist auf dem weg zur internationalen Standardisierung

Prinzipien für das Schichtenmodell:

1. eine neue schicht sollte dort entstehen, wo ein neuer abstraktionsgrad benötigt wird.
2. jede schicht sollte eine genau definierte Funktion ausführen
3. bei der Funktionswahl sollte die Definition international genormter Protokolle berücksichtigt werden
4. die grenzen zwischen den einzelnen schichten sollte so gewählt werden, dass der Informationsfluss über die Schnittstellen möglichst gering ist.
5. die Anzahl der schichten sollte so gross sein, dass keine Notwendigkeit besteht, verschiedenen Funktionen auf eine schicht zu packen, aber so klein, dass die gesamte Architektur nicht unhandlich wird.

Die Bitübertragungsschicht (Physical Layer)

Betrifft die Übertragung von rohen Bits über ein Kommunikationskanal. Die Wertigkeit eines Bits (0 oder 1) muss während der gesamten übertragen konstant bleiben. Frage: wie viel Volts sollten einer logischen 1 entsprechen und wie viel eine 0, wie viel Mikrosekunden soll ein Bit dauern, duplex oder Simplex, wie kommt die erste Verbindung zu stande und wie wird sie am ende wieder getrennt. Nimmt und sendet einfach nur einen Strom von Bits, ohne sich dessen Bedeutung oder Struktur zu kümmern.

Die Sicherungsschicht (Data Link Layer)

Die Hauptaufgabe ist es, eine rohe Übertragungseinrichtung in eine Leitung zu verwandeln, die sich der Vermittlungsschicht frei von unerkannten Übertragungsfehlern darstellt. Diese Aufgabe wird dadurch erfüllt, dass der Sender die Eingangsdaten in Datenrahmen (dataframes) aufteilt, diese sequentiell überträgt und die vom Empfänger erzeugten Bestätigungsrahmen (acknowledgement frames) verarbeitet. Fügt Rahmengrenzen ein und erkennt sie (am Anfang und am Ende werden spezielle Bitmuster gesetzt).

Falls Rahmen in der Übertragungsleitung komplett zerstört werden, muss die Software auf der Sicherungsschicht der Quellmaschine erneut übertragen. Gefahr der Rahmenduplizierung (falls z.B. ein Bestätigungsrahmen zerstört wird).

Pufferregelung wenn Sender schneller als Empfänger ist → Verkehrsregelung.

Die Vermittlungsschicht (Network Layer)

Hat mit der Steuerung des Teilnetzbetriebs zu tun. Eine wichtige Frage ist die Auswahl der Paketrouten, bzw. das Routing vom Ursprung zum Bestimmungsort. (statisch (in Tabellen festgelegt) oder dynamisch (werden immer neu bestimmt)).

Stausteuerung falls zu viele Pakete gleichzeitig sich in einem Teilnetz befinden.

Die Transportschicht (Transport Layer)

Hat die grundlegende Aufgabe, Daten von der Sitzungsschicht zu übernehmen, notfalls in kleinere Pakete zu zerlegen, sie danach die Vermittlungsschicht zu übergeben und danach zu sorgen, dass alle Teile richtig am anderen Ende ankommen.

Sie ist eine Ende-zu-Ende-Schicht → ein Programm auf der Quellmaschine führt ein Gespräch mit einem ähnlichen Programm auf der Zielmaschine und verwendet dabei die Nachrichten-Header und Steuerdaten.

Viele Hostrechner arbeiten im Mehrprogrammbetrieb, wodurch viele Verbindungen zum und vom Host entstehen. Deshalb muss bestimmt werden, welche Nachricht zu welcher Verbindung gehört.

Diese Schicht muss sich auch noch um den Aufbau und die Trennung von Verbindungen im Netz kümmern.

Die Sitzungsschicht (Session Layer)

Ermöglicht es den Benutzern an verschiedenen Maschinen, Sitzungen untereinander aufzubauen. Eine Sitzung ermöglicht den normalen Datentransport und bietet zusätzliche erweiterte Dienste, die für bestimmte Anwendungen nützlich sind.

Ein spezieller Dienst ist die Dialogsteuerung, Sitzungen können ermöglichen, dass Verkehr gleichzeitig in eine oder zwei Richtungen fließen. Kann der Verkehr nur in eine Richtung fließen, entscheidet die Sitzungsschicht, wer jeweils an die Reihe kommt.

Token Management: nur die Seite, die den Token hat kann eine Operation ausführen.

Synchronisation: Fixpunkte (Checkpoints) in den Datenstrom einfügen. Nach einer Störung werden nur die Daten ab dem nächsten Fixpunkt erneut übertragen.

Die Darstellungsschicht (Präsentation Layer)

Führt bestimmte Funktionen aus, die durch ihre häufige Verwendung eine allgemeine Lösung rechtfertigen. Kümmert sich nicht nur über die zuverlässige Übertragung von Bits, sondern auch um

dessen Syntax und Semantik. Diese Schicht übernimmt auch die Aufgabe, abstrakte Datenstrukturen zu handhaben und die interne Darstellungsform des Computers in die Standarddarstellung des Netzes zu konvertieren.

Die Verarbeitungsschicht (Applikation Layer)

Enthält eine Vielzahl häufig benötigter Protokolle, z.B. die Dateitransfer. Verschiedene Dateisysteme haben auch verschiedene Konventionen für Dateinamen, die Darstellung von Textzeilen... wenn man eine Datei zwischen 2 verschiedenen System hin und herschieben will, muss man sich mit diesen Unverträglichkeiten herumschlagen.

Internet Architektur – 4 Schichten

Die Internet Schicht

Es handelt sich um ein paketvermitteltes Netz auf der Grundlage einer verbindungsloser Vernetzungsschicht. Es ist die Sicherheitsnadel, die die gesamte Architektur zusammenhält. Ihre Aufgabe ist es, den Host zu ermöglichen, Pakete in jedes beliebige Netz einzuschleusen und unabhängig an das Ziel zu befördern. Sie können möglicherweise sogar in eine andere Reihenfolge ankommen, als sie abgeschickt wurden. In diesem Fall ist es Aufgabe der höheren Schichten, sie wieder richtig anzuordnen.

Die Internetschicht definiert ein offizielles Paketformat und Protokoll namens IP (Internet Protocol). Die Internetschicht hat die Aufgabe, IP-Pakete richtig zuzustellen.

→ Paketrouting

→ Vermeidung von Überlastungen.

Die Transportschicht

Sie soll die Partnereinheiten an die Quell- und Zielhosts die Kommunikation ermöglichen. Hier wurden zwei Ende-zu-Ende-Protokolle definiert:

TCP: zuverlässiges Verbindungsorientiertes Protokoll, durch das ein Bytestrom von einer Maschine zuverlässig und fehlerfrei im Internet einer anderen Maschine zugestellt wird. Es zerlegt den eingehenden Bytestrom in einzelne Nachrichten und leitet sie an die Internet Schicht weiter. Am Ziel werden sie vom empfangenden TCP-Prozess wieder zu einem Ausgabestrom zusammengesetzt. TCP handelt auch Flusssteuerung, um sicherzustellen, dass langsame Empfänger nicht von schnellen Sendern mit Nachrichten überschwemmt werden.

Das zweite Protokoll auf dieser Ebene ist das UDP-Protokoll (User Datagram Protocol). Das ist ein unzuverlässiges, verbindungsloses Protokoll für Anwendungen, die anstelle der Abfolge oder Flusskontrolle von TCP diese Aufgabe lieber selbst bereitstellen. Wird vorwiegend für einmalige Abfragen und Anwendungen in Client/Server Umgebungen benutzt, in denen die Schnelligkeit der Zustellung wichtiger ist als ihre Genauigkeit.

Die Verarbeitungsschicht

Das TCP/IP-Modell hat keine Sitzungs- und keine Darstellungsschicht, für diese Schichten besteht in den meisten Anwendungen kein Bedarf.

Diese Schicht umfasst alle höherschichtigen Protokolle, z.B. Telnet, ftp, smtp. Das virtuelle Terminalprotokoll ermöglicht es dem Benutzer, sich von seiner Maschine auf eine entfernte Maschine anzumelden und dort zu arbeiten.

Host-an-Netz Schicht

Ein Host muss sich durch ein bestimmtes Protokoll ans Netz anschliessen, um IP Pakete darüber senden zu können.

OSI versus TCP/IP

A priori versus a posteriori

A priori (OSI, Modell wurde zuerst entwickelt, da gab es noch gar keine Protokolle. Protokolle wurde später dazu genommen, aber man musste merken, dass dies nicht ohne Probleme ging; Protokolle passten nicht zu Dienstspezifikationen).

A posteriori (TCP/IP, Protokolle wurden zuerst geschrieben, das Modell wurde später dazu entwickelt. Nur diese Modell passte zu keinem anderen Protokollstapel)

OSI unterscheidet klar zwischen Diensten (was macht eine Schicht), Schnittstellen (teilt den darüberliegenden Prozessen mit, wie sie auf diese bestimmte Schicht zugreifen können, definiert welche Parameter zu brauchen sind, und welche Ergebnisse erwarten werden) und Protokollen (sind die „Privatangelegenheit“ der jeweiligen Schichten. Es muss nur die Aufgabe erfüllt werden, also die richtigen Dienste zu Verfügung stellen).

OSI-Modell unterstützt auf Vermittlungsschicht sowohl die verbindungslose wie auch die Verbindungsorientierte Kommunikation. Auf der Transportschicht jedoch nur die Verbindungsorientierte. TCP/IP Modell hat auf Vermittlungsschicht nur verbindungslos, aber auf Transportschicht beide Modi.

Kritik an OSI:

Schlechtes Timing: OSI Standard Protokolle wurden in die Enge getrieben, da TCP/IP Protokolle bei Forschungsinstitutionen und Unis schon einsetzten, bevor OSI fertig war.

Schlechte Technologie: Modell und Protokolle sind schwächlich, 7 Schichten waren nicht nötig. Einige waren zu voll, andere fast leer (z.B. Sitzungsschicht und Darstellungsschicht).

Schlechte Implementierung: wegen der enormen Komplexität des Modells und seiner Protokolle waren seine Implementierungen riesig, unhandlich und langsam → OSI=schlechte Qualität

Schlechte Strategien:

Kritik an TCP/IP

Keine deutliche Trennung zwischen Dienst, Schnittstelle und Protokoll.

TCP/IP Modell kann nicht benützt werden, um andere Protokollstapel zu benützen.

Keine Trennung zwischen Bitübertragung- (Übertragungsmerkmale von Kupfer, Glasfaser,...) und Sicherungsschicht (Anfang und Ende eines Rames zu beschränken und mit der gewünschten Zuverlässigkeit von einem Ende zum anderen zu übertragen)

Optimales Hybridmodell als Referenzmodell:

5 Verarbeitungsschicht

4 Transportschicht

3 Vermittlungsschicht

2 Sicherungsschicht

1 Bitübertragungsschicht

Alternativen:

Optimistisch (reaktiv): optimiert auf den problemfreien Fall, teuer im Fall, dass Problemen behandelt werden müssen, Ethernet optimiert für Niederlastsituationen. z.B. bei codieren: Fehlererkennung

Pessimistisch (präventiv): optimiert auf die Problembehandlung, Kosten fast unabhängig von Problemen, teuer im problemfreien Fall; Token-Ring für Hochlastsituationen. Bei codieren: Fehler Korrektur

CORINNE

Einschub

Bezeichnungen:

LAN : Local Area Network (Ethernet (gemeinsame Mediennutzung, koaxial oder verdrehte Kabel; die angeschlossenen Stationen verwenden CSMA-CD-Methode für die gemeinsame Nutzung eines physischen Mediums), Token Ring, FDDI (Fiber Distributed Data Interface – gemeinsame Mediennutzung mit 100mbs auf Lichtwellenleiter oder verdrehten Kabelpaaren, verwendet Token-passing-methode)..)

MAN : Metropolitan Area Network (100-200km Durchmesser)

WAN : Wide Area Network (X.25, Frame Relay, IP, ATM (asynchronous Transfer Mode, für Fernnetze mit höher Bitrate, asynchronen zeitmultiplex. Kleine Dateien fester Länge (53 bytes))

SAN : System Area Network (oder storage area network), Anwendung bei Cluster computing, also Koppelung von Arbeitsplatzrechner zu functions-, daten- und Lastverbänden)

Eigenschaften:

Grösse des Netzwerkes schränkt verwendbare Technologie ein.

Internetworking

Internetwork: Sammlung von Zusammengeschachtelten autonomen Systemen, die Pakete zwischen 2 Hosts übertragen

TCP/UDP-IP Protokollhierarchie

Anwendungsschicht: TelNet, FTP, SMTP, HTTP, DNS, NFS

Transportschicht: TCP, UDP

Internetschicht (OSI 3): IP

Netzzugangsschicht: Paketorientierter, zuweilen gesicherter Datentransport zu direkt verbundenen Knoten.

Internet Protokoll (IP)

Zur Beförderung von Datagrammen von der quelle zum ziel

Einheitliche Adressierung (IP-Adressen)

Fragmentierung und Zusammensetzung überlanger Datagramme

Protokolle zum unterhalt konsistenter Routinginformationen

Methoden zur Linderung von überlast

Generierung von Fehlermeldungen

Keine quantitativen oder qualitativen Garantien

Keine Bestätigung, keine reihenfolgetreue, kein Schutz vor duplizieren

Dienstmodell IP

Adressierungsschema + „best effort“ datagramm-modell

Ein IP-Datagramm besteht aus einem Header- und einem Textteil.

IP-Header

Der Header hat einen festen 20byte grossen teil und einen optionalen teil mit variablen Länge (Max 40byte).

1. Version (4 Bit), d.h. 0100; verfolgt die Version des Protokolls, zu dem das Datagram gehört (ist für Version Erkennung bei Upgrades)
2. Header Length (4 Bit): nicht konstant, daher wird ein Feld IHL benötigt, es gibt die headerlänge in 32-bit Wörtern an (min 5 → keine Optionen vorhanden, Max 15 → Header ist dann 60byte und somit das Option Feld 40byte)
3. Type of Service (8 Bit): ursprünglich Priorität, der Host kann hiermit dem Teilnetz den gewünschten dienst bekannt geben. Hier sind verschieden Kombinationen aus Zuverlässigkeit und Geschwindigkeit möglich. z.B. für digitalisierte Sprache ist Schnelligkeit wichtiger als Genauigkeit. Für Dateitransfer ist fehlerfreie Übertragung wichtiger als Schnelligkeit. Enthält noch 3 Bit als Flags D (Delay), T (Durchsatz) und R (Zuverlässigkeit). QoS wird von Routern ignoriert
4. Total Length (16 Bit): enthält das ganze Datagram – Header und Daten. Die Höchstlänge ist 65535 bytes.
5. Identification (16 Bit): dieses Feld wird gebraucht um an einem Host mitzuteilen, zu welchem Datagram ein neu angekommenes Fragment gehört. Alle Fragmente eines Datagrammes enthalten den gleichen identifikations-wert.
6. danach folgt ein unbenütztes Bit und zwei 1 Bit Felder. DF bedeutet don't defragment. Dies wird gesetzt, so dass ein Router ein Paket nicht fragmentiert, weil der ziel Host es nicht mehr zusammen setzen kann. MF bedeutet more Fragments, wird bei allen Segmenten ausser dem letzten gesetzt. Fragment Offset bezeichnet, an welche stelle im Datagram ein Fragment gehört. Alle Fragmente ausser das letzte müssen ein vielfaches von 8 bytes sein. Da 13 Bit verfügbar sind, sind maximal 8192 Fragmente pro Datagram möglich, was eine maximale Datagrammlänge von 65536 ergibt.
7. Das Feld Time to Live (8 Bit) ist ein Zähler, mit dem die Lebensdauer von Paketen begrenzt werden kann. Maximaler Lebensdauer ist 255 Sekunden. Der Zähler wird bei jeder Teilstrecke gesunken, wenn es 0 erreicht wird es entfernt und eine entsprechende Fehlermeldung wird an den start Host geschickt.
8. durch das Feld Protocol (8 Bit) kann die Vermittlungsschicht erkennen, an welchen Transportprozess das Datagram weiterzugeben ist (z.B. UDP oder TCP)
9. Header Checksum (16 Bit): prüft nur den Header nach Fehler, die durch falsche Speicherwörter in einem Router erzeugt wurden.
10. Source Address (32 Bit) und Destination Address (32 Bit): bezeichnet die Netz- und Hostnummer.
11. Optionen (32 Bit): ermöglicht erweiterte Versionen des Protokolls, z.B. für Sicherheit (wie geheim ein Datagram ist), striktes source routing (bestimmt den kompletten Pfad), loses source routing (gibt eine liste von Router an, die nicht zu verfehlen sind), Routenaufzeichnung (veranlasst jeden Router, seine IP-Adresse anzuhängen), Zeitstempel (veranlasst jeden Router, seine Adresse und Zeitstempel anzuhängen)

IP-Adressen

Jeder Host und Router hat eine IP-Adresse, die die netz- und Hostnummer codiert, sie ist eindeutig, 32 Bit lang und wird bei source- und Destination Address benützt.

Netzadressen sind 32 Bit grosse zahlen. Es werden alle 4 bytes dezimal geschrieben, von 0 bis 255 X.Y.Z.K (mit X,Y,Z,K bytes von 0 bis 255).

126 class-A-netze mit je ca. 16mio Rechner (7 Bit Netzanteil $0 < X < 126$, 24bit Rechneranteil)

16000 class B netze mit je ca. 65000 Rechner (14bit netzanteil $126 < X < 192$, 16 Bit Rechneranteil)

2mio class C netze mit je 254 Rechner (21bit Netzanteil $192 < X < 224$, 8 Bit Rechneranteil)

Sonderadressen:

0.0.0.0 bei start des Hosts

0. bzw. 0.0 bzw. 0.0.0 als Netznummer bezeichnen das aktuelle netz

255.255.255.255 Broadcast im lokalen netz

127. schleifen test (lokal verarbeitet und als Eingangspakete behandelt)

Teilnetze

Alle Hosts im netz müssen die gleiche Netznummer haben.

Ein netz kann für interne Verwendungszwecke in mehrere teile aufgeteilt werden

//CORINNE

Ende zu Ende

Argumente:

So tief wie möglich (ist billiger im Protokollstack (braucht weniger Verarbeitung) nach Richtigkeit zu prüfen → höhere Schichten müssen sich nicht darum kümmern

Dienste/Qualitäten auf unteren Schichten garantieren nicht notwendigerweise selbige auf höheren Schichten, z.B. Verbindungen, exactly once, ...

Semantik nur auf Anwendungsschicht bekannt (erleichtert Fehlererkennung, -korrektur)

Ausfälle nur auf darüberliegenden Schicht behandelbar (falls genügend Statusinfos vorhanden ist)

Ein Protokoll auf Applikationsebene zur Überprüfung, ob alles funktioniert, ist immer notwendig

Internet Transportprotokolle

Das Internet hat auf der Transportschicht 2 Protokollarten, eine Verbindungsorientierte (TCP) und eine verbindungslose (UDP, ähnlich wie IP mit kleinerem Header).

TCP wurde spezifisch zur Bereitstellung eines zuverlässigen Bytestroms von ende zu ende in einem unzuverlässigen Netzverbund (wo verschiedene teile eventuell total unterschiedliche Topologien, bandbreiten, Verzögerungen, Paketgrößen und andere Parameter haben) entwickelt

UDP (User Data Protocol)

Die Internet-Protokollfolge unterstützt auch verbindungsloses Transportprotokoll, das UDP. UDP bietet Anwendungen die Möglichkeit, gekapselte rohe IP-Datengramme zu übertragen, ohne eine Verbindung aufzubauen.

Ein UDP Segment besteht aus einem 8 bytes Header, gefolgt von den Daten. Quell- und Zielport identifizieren die quell- und Zielmaschine. Das Feld UDP Length beinhaltet eine 8 bytes Header und die Daten. Das Feld UDP Checksum beinhaltet den UDP-Header und die UDP-Daten, die bei Bedarf auf eine gerade Bytezahl aufgefüllt werden. Es ist optional und wird auf 0 gesetzt, wenn keine Prüfsumme berechnet wird.

Es ist ein einfacher Demultiplexen, d.h. unzuverlässiger Datentransport zwischen 2 Programmen. Beim Demultiplexen werden die Pakete in Warteschlange transferiert.

Kein Schutz vor Paketverlust, Paketduplizierung und Veränderung der Reihenfolge.

TCP (Transmission Control Protocol)

Zuverlässiges Verbindungsorientiertes Protokoll, durch das ein Bytecode von einer Maschine fehlerfrei im Internet einer anderen Maschine zugestellt wird. Es zerlegt den eingehenden Bytestrom in einzelne Nachrichten und leitet sie an die Internet-Schicht weiter. Am Ziel werden sie vom empfangenden TCP-Prozess wieder zu einem Ausgabestrom zusammengesetzt. TCP handhabt auch Flusssteuerung, um sicherzustellen, dass ein langsamer Empfänger nicht von einem schnellen Sender mit Nachrichten überschwemmt wird.

Die IP-Schicht gibt keinerlei Gewähr, dass Datagramme richtig zugestellt werden. Deshalb obliegt es TCP, ein Timeout zu veranlassen um Daten bei Bedarf erneut zu übertragen. Es ist auch möglich, dass Datagramme in der falschen Reihenfolge ankommen. In diesem Fall werden sie von TCP wieder in die richtige Reihenfolge gebracht.

Um einen TCP-Dienst zu nutzen, müssen Sender und Empfänger Endpunkte (sockets) erstellen. Jedes Socket hat eine Socketnummer (Adresse), die aus der IP-Adresse des Hosts und einer 16bit Nummer für den lokalen Port des Hosts besteht. Ein Socket wird für mehrere Verbindungen gleichzeitig genutzt. Verbindungen werden nach Socketbezeichner an beiden Enden identifiziert.

TCP Verbindungen sind immer Vollduplex und Punkt-zu-Punkt (jede Verbindung hat genau 2 Endpunkte). TCP unterstützt weder Multicasting noch Broadcasting.

Eine TCP Verbindung ist ein Bytestrom, kein Nachrichtenstrom. (Die Grenzen von Nachrichten werden von Ende zu Ende beibehalten, TCP hat kein Erhalt von Nachrichtengrenzen). Der Empfänger kann nicht bestimmen, in welche Datenblockgrösse die Pakete geschickt wurden (Daten Grösse Max 64kb, meist aber 1,5kb). Der Datenaustausch passiert in Segmenten.

TCP kann vor dem Versenden Daten zwischenspeichern oder sofort weitergeben. Falls Anwendungen auf Befehl Daten verschicken müssen, wird ein so genannter PUSH-Flag gesetzt. TCP wird somit die Übertragung nicht verzögern.

Durch den Urgent Flag wird eine eingeleitete entfernte Operation abgebrochen, die sendende Anwendung setzt bestimmte Steuerinformationen mit Urgent Flag in den Datenstrom. Durch dieses Ereignis stoppt TCP mit dem Ansammeln von Daten und überträgt sofort alles, was für die betreffende Verbindung anliegt. Die empfangende Anwendung wird somit unterbrochen und diese liest die dringende Daten.

Übernommene Aufgaben:

Auf- und Abbau von Verbindungen

Reihenfolgegetreue Auslieferung von unverfälschten Daten: exactly once

Flusskontrolle

Multiplexen

Überlastüberwachung

Das TCP-Protokoll

Die sendende und empfangende TCP-Einheit tauschen Daten in Form von Segmenten aus. Ein Segment besteht aus einem festen 20-Byte-Header, gefolgt von Datenbytes. Die TCP-Software entscheidet über die Grösse der Segmente. Sie kann mehrere Schreiboperationen zu einem Segment zusammenfassen oder Daten aus einem Schreibvorgang auf mehrere Segmente aufteilen.

Jedes Segment wird begrenzt von:

Segment + TCP-Header müssen in das Ip-Nutzenfeld passen (65535 bytes)

Jedes netz hat eine maximale MTU (Maximum Transfer Unit), in die jedes Segment passen muss (einige tausend bytes).

TCP-Einheiten benützen das Schiebefensterprotokoll. Überträgt ein Sender ein Segment, startet er gleichzeitig ein Timer. Kommt das Segment am ziel an, sendet die empfangende TCP-Einheit ein Segment mit einer Bestätigungsnummer zurück. Diese Nummer entspricht die als nächste erwartete Folgenummer. Läuft der Timer des Senders ab, bevor die Bestätigung eingeht, überträgt der Sender das Segment erneut.

TCP-Header:

20byte Header (+optionaler Teil) plus Datenbytes

Sourceport (2bytes), Destination Port (2bytes)

Sequence number (4B)

Acknowledgement (4B)

Header Length (4bit), unbenütztes Feld (6bit), 6 1bit Flags (SYN (bei verbindungs-
Aufbau),FIN (bei Verbindungsabbau),RESET,PUSH,URG,ACK), advertised Window (2B)

Checksum (2B), Urgent Pointer (2B)

Options (variable, 0 oder 4nB)

Daten (optional=)

Die Flusssteuerung erfolgt in TCP anhand eines Schiebefensters mit variabler Grösse. Das windows-feld bezeichnet, wie viele Bytes ab dem bestätigten Byte gesendet werden können. Ein windows-feld von 0 bedeutet, dass die bytes bin einschliesslich zur acknowledgednummer -1 empfangen wurden, dass der Empfänger im Moment unbedingt eine Verschnaufpause braucht und vorläufig keine weitere Daten mehr will. Die Erlaubnis zum senden weiterer Daten kann später gewährt werden, indem ein Segment mit der gleichen Bestätigungsnummer und einem Window Feld von nicht null gesendet wird.

TCP-Verbindungsmanagement

3-schritt Verbindungsaufbau, getrennt für beide Richtungen

C zu S: SYN (SEQ=x) (synchronisation Nummer x wird dem Server geschickt)

S zu C: SYN + ACK (SEQ=y, ACK=x+1) (Server synchronisiert mit y und bestätigt mit x+1)

C zu S: ACK (SEQ=x+1, ACK=y+1) (Client bestätigt mit y+1 und synchronisiert mit x+1)

2-Schritt Verbindungsaufbau

FIN-bit und Bestätigung, bzw. Timeout

In beiden Richtungen, also 4 (ev. nur 3) nachrichten

Timed wait (doppelte Paketlebensdauer) bis zum endgültigen schliessen

Verbindungsnutzen: Funktionen der Transportinstanz in der Absenz von Fehlern&Fehlermanagement:

Verwerfen von Segmenten mit fehlerhafter Prüfsumme

Erneuter Transport unbestätigter Segmente

TCP-Überlastüberwachung

Heute sind Paketverluste aufgrund von Übertragungsfehler relativ selten, da die meisten Fernleitungen aus Glasfaser bestehen. Folglich sind die meisten Timeouts von Übertragungen im internet auf Überlastung zurückzuführen.

Beim Aufbau einer Verbindung muss darauf geachtet werden, dass das die gewählte Fenstergrösse auf der Grundlage der Puffergrösse des Empfängers basiert ist.

Die Internet Lösung besteht darin, 2 potentielle Probleme zu realisieren: die netz- und die Empfängerkapazität. Hierfür pflegt jeder Sender 2 Fenster, eines, das der Empfänger gewährt hat (advertised Window), und ein zweites, das so genannte Überlastungsfenster (Congestion Window). Jedes spiegelt die Anzahl von bytes wieder, die der Sender übertragen kann (das Minimum der beiden Fenster wird dann ausgewählt).

Bei jeder erfolgreichen übertragene und bestätigte Spitzenmenge wird das Überlastungsfenster verdoppelt.

Das Überlastungsfenster nimmt exponentiell zu, bis entweder ein Timer abläuft oder das Fenster des Empfängers erreicht wird. Es wird die letzte erfolgreich übertragene menge beibehalten. SLOW START Algorithmus.

- $\text{MaxWindow} = \min(\text{CengestionWindow}, \text{AdvertisedWindow})$
- $\text{EffectiveWindow} = \text{MaxWindow} - (\text{LastByteSent} - \text{LastByteAck})$

Im Internet wird allerdings der Fast Start benützt. Er wendet ein dritter Parameter an: der schwellwert (Threshold), am Anfang beträgt er 64kb, zusätzlich zum empfänger- und Überlastungsfenster. Läuft ein Timer ab, wird er auf die hälfte des aktuellen Überlastungsfenster gesetzt, und das Überlastungsfenster wird auf ein maximales Segment zurückgesetzt. Danach wird Slow start benützt. Wenn der schwellwert wieder erreicht ist, gibt es ein linearer Wachstum des Überlastungsfenster Dieser schwellwert braucht man aber nur um zwischen slow start und linearer Wachstum des Überlastungsfenster zu unterscheiden.

TCP-Timermanagement

Der Retransmission-Timer:

Wird ein Segment gesendet, startet ein retransmission Timer. Wird ein Segment bestätigt, bevor der Timer abläuft, wird der Timer gestoppt. Läuft aber der Timer ab bevor die Bestätigung kommt, wird das Segment erneut übertragen (und der Timer startet wieder von vorne). Das Problem ist: auf welchen wert setzen wir diesen Timer? Die zeit bis TCP eine Bestätigung bekommen kann ist sehr schwer abschätzbar, dafür können aber dynamische Algorithmen benützt werden die diese zeit kontinuierlich messen. Es wird hier der Jacobson Algo verwendet:

Für jede Verbindung verwaltet TCP die Variable RTT, das ist die momentan beste Schätzung der Rundreisezeit zum fraglichen ziel. Wird ein Segment gesendet, startet der Timer. Er überwacht, wie lange die Bestätigung braucht, und löst bei bedarf eine Neuübertragung aus. Kommt die Bestätigung zurück, bevor der Timer abläuft, misst TCP, wie lange die Bestätigung dauerte, z.B. M. dann wird RTT nach der Formel $\text{RTT} = \alpha \text{RTT} + (1 - \alpha)M$ aktualisiert. Dabei ist α ein Glättungsfaktor, der bestimmt, wie viel gewicht dem alten werte beigemessen wird (normalerweise ca. 7/8).

Eine weitere Möglichkeit für das Timer Management ist folgende:

Auch bei einem guten RTT wert ist es schwierig, eine gute Auswahl für Timeouts für erst Übertragungen zu bestimmen. Normalerweise wurde βRTT benützt, wobei β standardmässig auf 2 gesetzt war. Dies war aber nicht gut genug, denn dieser wert war nicht genügend flexibel, und verschlechterte sich mit steigender Abweichung.

Jacobson schlug vor, β ca. proportional zur Standardabweichung der Ankunftszeit von Bestätigungen als Wahrscheinlichkeitsdichte zu verwenden (grosse Abweichung = grosses β).

Die variable D als Abweichung muss somit ständig verfolgt werden, kommt eine Bestätigung an, wird der unterschied zwischen dem erwarteten und dem beobachteten wert ($RTT-M$) berechnet (M =wie lange braucht die Bestätigung).

$$D = \alpha D + (1-\alpha)[RTT-M] \rightarrow \text{Timeout} = RTT + 4 * D$$

Der persistence Timer

Er soll folgendes verhindern: der Empfänger sendet eine Bestätigung mit einer Fenstergrösse von $=$ und weist damit den Sender zum warten an. Später aktualisiert der Empfänger das Fenster, aber das Paket mit der Aktualisierung ist verloren. Nun warten beide, Sender und Empfänger, bis einer etwas unternimmt. Läuft der persistence Timer ab, überträgt der Sender eine probe zum Empfänger. Die antwort auf diese probe ergibt die Fenstergrösse. Ist sie immer noch null, wird der persistence Timer nochmals gesetzt, ist sie nicht null, können Daten übertragen werden.

Der Keepalive Timer

Wenn eine Verbindung über längere zeit inaktiv ist, schaltet der Keepalive Timer ab, wodurch eine Seite veranlasst wird, zu prüfen, ob die andere Seite noch da ist. Kommt keine antwort, wird die Verbindung abgebrochen. Umstritten, weil eine gesunde inaktive Verbindung einfach beendet werden kann.

Timed Wait

Er läuft über das doppelte der maximalen Paketlebensdauer, um sicherzustellen, dass alle Pakete während der Schliessung einer Verbindung übertragen wurden, bevor die Verbindung geschlossen wird.

Drahtloses TCP und UDP

Theoretisch sollten Transport Protokolle von der Technologie der zugrunde liegende Netzschicht unabhängig sein. Insbesondere sollte es für TCP gleich sein, ob IP über Glasfaser oder funk läuft. In der Praxis spielt dies aber eine grosse rolle, weil die meisten TCP Implementierungen sorgfältig auf der Grundlage von annahmen optimiert werden, die auf verkabelte netze zutreffen, bei drahtlosen netzen aber versagen. Dies kann zu einer sehr schwachen Leistung der Implementierung führen.

Das Hauptproblem ist der Algorithmus für die Überlastüberwachung. Fast alle TCP-Implementierungen basieren heute auf der Annahme, dass Timeouts durch Überlastung, nicht aber durch Paketverlust verursacht werden. Läuft ein Timer ab, verlangsamt sich TCP (durch slow start Algo).

Das Problem bei drahtlosen Übertragungen ist dass sie ständig Pakete verlieren, und diese so schnell wie möglich nachgesendet werden müssten. Durch Verlangsamung der Übertragung verschlimmert sich die Situation nochmer.

Gehen Pakete in einem verkabelten netz verloren, sollte der Sender verlangsamen. Gehen in einem drahtlosen netz verloren, sollte sich der Sender mehr anstrengen.

Nur selten ist aber der Pfad von Sender zu Empfänger homogen, z.B. geht er zuerst über ein verkabeltes und später über ein drahtloses netz. Hier ist es noch schwieriger eine Entscheidung auf der Grundlage von Timeout zu treffen, da es eine rolle spiel, wo das Problem entstanden ist..

→ Indirekte TCP (von Bakne und Badrinath). Trennt die TCP Verbindung in 2 getrennte Verbindungen, die erste läuft vom Sender zur Basisstation, die zweite von der Basisstation zum Empfänger. Die Basisstation kopiert lediglich Pakete zwischen den Verbindungen in beide Richtungen. (somit sind beide Verbindungen homogen). Die TCP Semantik wird aber verletzt, denn der empfang einer Bestätigung beim Sender heisst lediglich, das die Basisstation das Paket erhalten hat.

→ Lösung von Balakrishnan: mehrere kleine Änderungen an der Basisstation auf der Vermittlungsschicht. Es wird ein Schnüffleragent hinzugefügt, der TCP Segmente von einem mobilen Host beobachtet und zwischenspeichert und deren Rückweg bestätigt. Bei verlorene Segmente die vom mobilen Host gesendet wurden stellt die Basis Station eine Lücke in den Folgenummern der Eingangsverkehr fest, erzeugt aber mit Hilfe einer TCP-Option eine anfrage für eine selektive Wiederholung der fehlenden bytes. Somit wird die drahtlose Verbindung zuverlässiger in beiden Richtungen.

Problem existiert vor allem auch für UDP, da da verloren gegangene Pakete nicht nachgefragt werden können. Man büsst einen grossen einschnitt in die Leistung.

Interprozess Kommunikation

Themen

- Ortstransparenz, sichere Kommunikation zwischen verschiedenen Prozessen über ein Netzwerk
- Schutz durch Firewalls und Kommunikation über Firewalls
- Saubere Schnittstellen zur modularen Strukturierung grosser verteilter Applikationen
- Statische und dynamische Rekonfiguration

Gewünschte Eigenschaften

- Einfache Implementierbarkeit
- Hohe Leistungsfähigkeit (Performance) und Zuverlässigkeit

Voraussetzung

- Netzwerke zur Übermittlung von Daten zwischen Knoten (heterogen und meist ohne Garantien)
- Transportschichtprotokolle (offerieren unterschiedliche Dienste: Übertragung von Paketen, Byteströmen oder {Anforderung, Antwort}-Paaren, Kerndienst Demultiplexing (zu UDP))

Protokolleigenschaften / Einschränkungen / Varianten

- Erkennung / Behebung von Fehlern gemäss Fehlermodell (z.B. UDP: einzige Fehlermassnahme: Überprüfung des Pakets anhand Prüfsumme)
- Keine Unterscheidung von Netzwerkzusammenbruch und Partnerausfall
- At-least-once plus Idempotenz (= wenn eine Operation n-mal wiederholt wird und sich an der Ausführung nie etwas ändert, wie z.B. eine Datei öffnen)
- At-most-once basierend auf eindeutig identifizierten Sitzungen

Sehr unterschiedliche Anforderungen / Lösungen

- At-most-once, niedrige Latenz, Unterstützung für sehr grosse nachrichten
- Positiver Feedback, Fehlermeldungen, Timer, Nummerierung (Sitzungen, Operationen, nachrichten, Pakete), Alive-Anfragen (z.B. go-back Bits)

Anwendungstypen

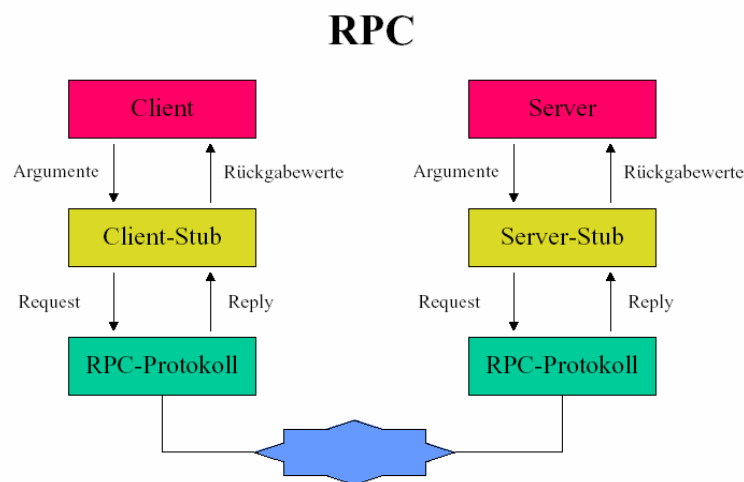
- Remote Operation (Anfrage /Antwort)
 - Client sendet Arbeitsauftrag und Server antwortet mit Resultat (RPC= Remote Procedure Call, RMI in JAVA)
- Bulk Data Transfer

- Möglichst effizienter Transport sehr grosser Datenmengen, quasi ein read bzw. write des Clients in einem C/S-Schema (File Transfer Access Method)
- One-to-many Kommunikation
 - Multicast und Broadcast, Ordnung erleichtert Anwendungsprogrammierung
- Continuous Media
 - Konstante Datenrate mit geringer Latenz, ATM Netzwerke sind dafür sehr geeignet

Remote Procedure Call

Birrel & Nelson (1984)

- Idee:
 - Nutzung der Semantik des lokalen Prozeduraufrufs
- Probleme:
 - Heterogenität der Architekturen und Darstellungsformate & Netzwerkkompatibilität
- Lösungskomponenten:
 - Protokoll zum Nachrichtenaustausch und Unterstützung von Programmiersprachen und Compilern, um Argumente / Rückgabewerte in Nachrichten zu Verpacken und wieder auszupacken
- Unterschiede zum Lokalen Prozeduraufruf:
 - Unabhängige Ausfälle von Client und Server
 - Unterschiedliche Adressräume von Client und Server: kein Pointer und globale Variablen, beschränktes Daten- und Function-passing



- Schichtenfrage
 - Nutzt UDP-Transportprotokoll
 - Transportprotokoll, bestehend aus Mikroprotokollen, die selber gültige Transportprotokolle sind
- Mikroprotokolle
 - Fragmentierung und Reassemblierung grosser Nachrichten (A)
 - Synchronisation von Anfrage- und Antwortnachrichten (B)
 - Verschickung der Anfragenachrichten an den richtigen Prozess (C)
 - ... sind nicht standardisiert
- Einfacher RPC-Stack
 - C - B - A - IP - Ethernet

- Unterstützung grosser Nachrichten
 - Idee: Garantierte Übertragung wird durch andere Protokolle gewährleistet, Fragmentneuübertragung wird hingegen zur Performance-Verbesserung unterstützt
- Realisierung z.B.:
 - Sender: Verschicken von Fragmenten (letztes besonders gekennzeichnet) + Timer (um bei Scheitern zu entscheiden Speicherplatz dann freizugeben) + Neuübertragung nach Selective Retransmission Requests (SSR)
 - Empfänger: Reassemblierung + 2 Timer (LAST-FRAG und RETRY, um SSR zu senden) + Versenden SSR (nach Ablauf von LAST_FRAG und RETRY-Timern)
- Synchronisation von Anfrage- und Antwortnachrichten
 - Idee: Nachrichtenübertragung über logischen Anfrage/Antwort-Kanal (für jede Nachricht separat), at-most-once (letzteres nicht von allen RPC-Protokollen unterstützt) und Synchronisation
 - Realisierung z.B. mit Huckepack/implizite Bestätigung, Alive-Checks (Alternative Heartbeats) und Numerierung - resultiert in 2 RETRANSMIT-Timern (Client und Server) und einem PROBE-Timer (beim Client)
 - Synchrone, asynchrone und intermediäre Protokolle
- Verschickung der Anfragenachrichten an den richtigen Prozess
 - Idee: separates synchrones Demultiplexen für RPC (zur einfachen Änderung des Adressraums für Prozeduren und zur Behandlung von Nebenläufigkeit)
- Marshalling
 - Darstellungsformatierung der Daten der Anwendungsprogramme für den Netzwerktransfer: Kodierung (Argument-Marshalling) und Dekodierung (Unmarshalling)
 - Notwendig beim RPC um Datenstrukturen von einem Adressraum in einen anderen zu übertragen
 - Wird von Client- und Server-Stubs übernommen, welche für die Serialisierung zuständig sind und die Konvertierung der Struktur aus der Repräsentation des aufrufenden Prozesses in die Repräsentation des aufgerufenen Prozesses
 - Alternative für Stubs: Kompilierung (effizient) versus Interpretation
 - („generische“ Stubs mit Parametersetzung, flexibel)