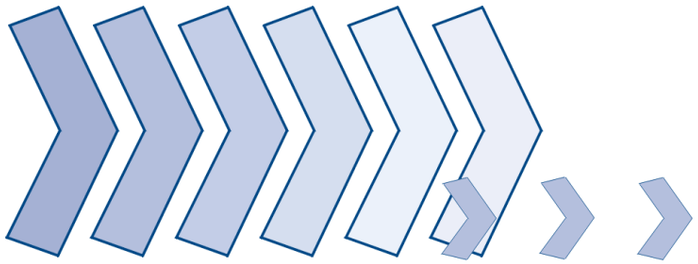
HUMBOLDT-UNIVERSITÄT ZU BERLIN



# Praktikum Kommunikationssysteme

Dr.-Ing. Siegmar Sommer

Wintersemester 2014/15

# Durchführung

- Gruppen mit (genau) vier Studenten
  - in Goya eintragen, Termin: 31.10.
- Zwischenabnahme:
  - wird noch bekannt gegeben
  - Zeitraum: Anfang bis Mitte Dezember 2014
- Endabnahme im Labor, RUD 25, R 4.309, in Form einer Netzwerkkonfiguration, Vorführung der eigenen Software und Gespräch
  - Zeitraum: Mitte Januar bis Mitte Februar 2014, Zeitplan wird noch bekannt gegeben
- Konsultationsmöglichkeiten: Mo von 15.00 bis 16:30 Uhr, RUD 25, R 4.303 stehe ich für Fragen zur Verfügung, falls keine anderen Termine (z.B. Übungen) angesetzt sind
- Weitere Anleitung nach Vereinbarung

# Hinweise

- Fangen Sie sofort mit der Bearbeitung an
- Informieren Sie sich auch zu Themen, die noch nicht im Unterricht behandelt wurden
- Beschäftigen Sie sich mit den Möglichkeiten des von Ihnen verwendeten Entwicklungstools (Debugging, Unit-Testing, Versionskontrolle (empfehlenswert: Git),...), das spart letztendlich viel Zeit
- Nehmen Sie bei Problemen rechtzeitig die Konsultationstermine war
- Wenn Sie wollen, können Sie einen Labor- bzw. Abnahmetermin bereits vor Mitte Januar erhalten

# Hinweise (2)

- Die Abnahme erfolgt mit eigenem Notebook
- Als Grundlage für die Softwareentwicklung wird Ihnen ein Beispiel/Framework (Download über die Kurs-Seite) zur Verfügung gestellt
- Eine Dokumentation finden Sie in dessen Verzeichnisbaum („doc/index.html“)
- Das Framework verwendet die Programmiersprache „Groovy“. Diese Sprache basiert auf der Sprache Java, ist jedoch entschieden „angenehmer“ in der Verwendung (groovy.codehaus.org).
- Es existieren verschiedene freie Entwicklungstools für Groovy, *groovy.codehaus.org*, *Eclipse-STS* oder noch besser *IDEA IntelliJ-Community-Edition*, sind zu empfehlen

# Inhalt des Praktikums

- Erster Teil: Implementierung/Komplettierung von Softwarekomponenten zur Netzwerkkommunikation (HTTP-Hyper Text Transfer Protocol, TCP-Transmission Control Protocol, IP-Internet Protocol, ARP-Address Resolution Protokoll)
- Zweiter Teil: Versuche in realen TCP/IP-Umgebungen: Konfiguration eines Netzwerks im Labor und Anwendung der selbst entwickelten Software
- Dritter Teil: Abnahme des Praktikums

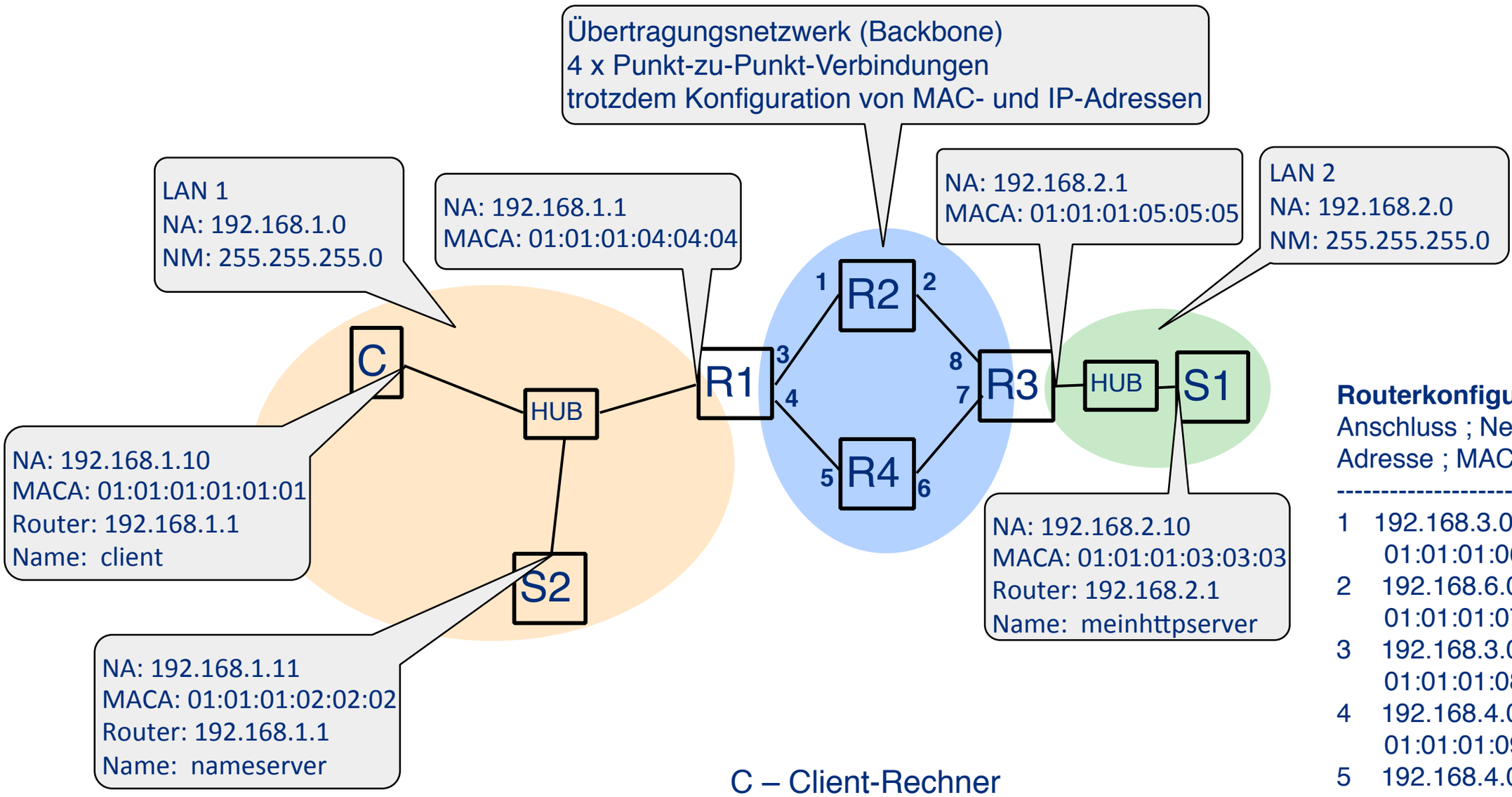
# Lokale Versuche

- Bauen Sie schrittweise das nachfolgend dargestellte Rechnernetzwerk unter Zuhilfenahme des bereitgestellten Frameworks auf
- Dabei müssen verschiedene Funktionalitäten der Netzwerkkommunikation (Netzwerkprotokolle) komplettiert werden
- Für diese Versuche benötigen Sie kein Labor
- Sie brauchen nur die für die Durchführung der Versuche absolut notwendigen Verhaltensweisen der Netzwerkprotokolle zu implementieren!

# Hinweise für TCP

- Implementiert werden muss:
  - Fehlerkontrolle
  - passiver Verbindungsaufbau und -abbruch
- Nicht implementiert werden braucht (kann jedoch, wenn Sie Lust dazu haben ;-) ) z.B.:
  - Verwaltung von mehr als einer Verbindung
  - Staukontrolle
  - Nagle-Algorithmus
  - Silly Window-Behandlung
  - Quittungsverzögerung
  - dynamische Bestimmung des Retransmission Timeouts

# Das Netzwerk im Endzustand




**Routerkonfiguration**
Anschluss ; Netzmaske ; IP-Adresse ; MAC-Adresse

---

1 192.168.3.0, 192.168.3.2
01:01:01:06:06:06
2 192.168.6.0, 192.168.6.2,
01:01:01:07:07:07
3 192.168.3.0, 192.168.3.1,
01:01:01:08:08:08
4 192.168.4.0, 192.168.4.1,
01:01:01:09:09:09
5 192.168.4.0, 192.168.4.2,
01:01:01:0A:0A:0A
6 192.168.5.0, 192.168.5.1,
01:01:01:0B:0B:0B
7 192.168.5.0, 192.168.5.2,
01:01:01:0C:0C:0C
8 192.168.3.6, 192.168.6.1,
01:01:01:0D:0D:0D

LAN – Local Area Network
NA – Netzwerk-Adresse (IPv4)
NM – Netzwerk-Maske (IPv4)
MACA – Medium Access Control – Adresse

C – Client-Rechner
S – Server-Rechner
R – Router
HUB – verbindendes, passives Gerät („Knotenpunkt“)

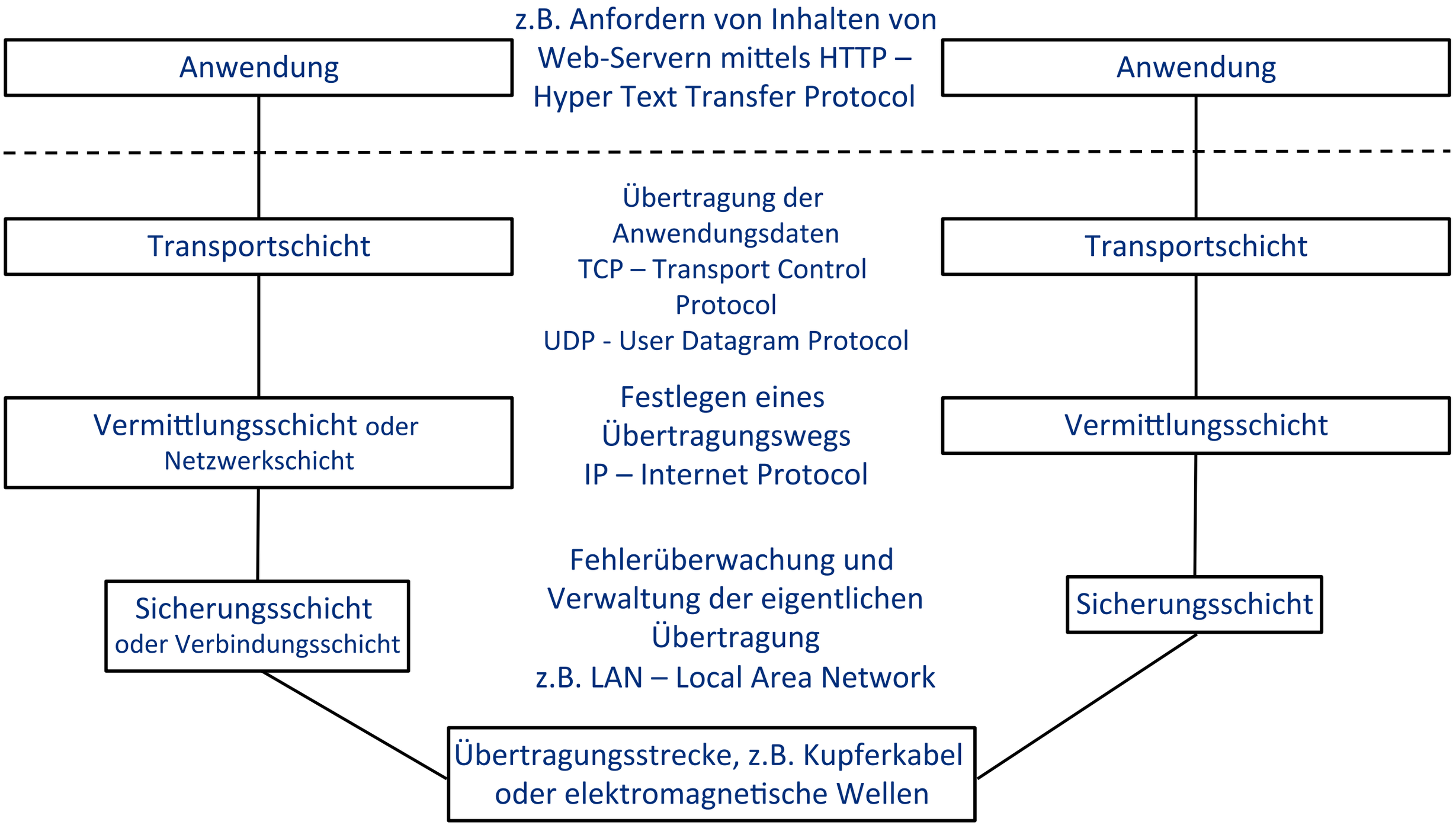
# Vollständige Funktionalität des Netzwerks zur Abnahme

- Client *C* löst den Namen des HTTP-Servers *S1* unter Verwendung des Name-Servers *S2* in die entsprechende IP-Adresse auf
- Client *C* beauftragt den Server *S1* zur Übertragung von Daten
- Die Router *R1* bis *R4* führen ein Routingprotokoll zur Gewinnung von Routing-Informationen
- Es erfolgt eine fortlaufende Übertragung von Daten vom Server *S1* über die Router *R1*, *R2* und *R3* zum Client *C*. Währenddessen Unterbrechung der Verbindung zwischen *R1* und *R3* (beispielsweise durch manuelles Abschalten von *R2*)
- Router erkennen das Problem und ändern ihre Routingtabellen
- Nach kurzer Zeit erfolgt die automatische Wiederaufnahme die Datenübertragung, diesmal über *R4*
- Die durch den Ausfall des Routers *R2* aufgetretene Datenpaketverluste werden durch Sendewiederholungen korrigiert

# Hinweise zur Bearbeitung der lokalen Versuche

- Nehmen Sie das Beispiel (bestehend aus client1 und server1) aus dem Framework in Betrieb
  - spätestens bearbeitbar: Oktober
- Erweitern Sie den HTTP-Server
  - übertragen Sie nach Aufforderung durch den Client größere Mengen von Datenpaketen vom Server zum Client
  - verwenden Sie zur Vereinfachung immer Daten-Paketgrößen kleiner/gleich 1000 Byte
  - spätestens bearbeitbar: Oktober
- Konfigurieren Sie ein LAN bestehend aus Client, Server und Name-Server, verbunden über einen HUB
  - spätestens bearbeitbar: Oktober
- Implementieren Sie ein minimales Namensdienst-Protokoll
  - spätestens bearbeitbar: Oktober
- Fügen Sie einen Router und ein weiteres LAN hinzu und implementieren Sie im Router das Weiterleiten („forwarding“) von Paketen
  - spätestens bearbeitbar: Dezember/Januar
- Komplettieren Sie alle notwendigen Protokollfunktionalitäten, konfigurieren Sie das komplette Netzwerk und implementieren Sie ein Routing-Protokoll
  - spätestens bearbeitbar: Januar

# Kommunikation in Schichten (TCP/IP-Protokollfamilie)



# Protokoll-Schichten

- Die Schichten des TCP/IP-Protokoll-Stacks erfüllen folgende Aufgaben
  - oberste Schicht: Zuverlässiger Transport (Transportschicht)
    - Übertragung der Daten einer absendenden Anwendung zu einer empfangenden Anwendung (zuverlässig: TCP, unzuverlässig: UDP)
  - mittlere Schicht: Wegewahl (Vermittlungs- oder Netzwerkschicht)
    - bestimmen des nächsten Empfängers für Datenpakete auf dem Weg zum Zielsystem aufgrund von Adressinformationen (u.U. über mehrere Zwischensysteme (Router))
  - unterste Schicht: Sicherung der Übertragung (Verbindungs- oder Sicherungsschicht)
    - geordnete Übertragung von Datenpaketen über ein Übertragungsmedium, z.B. Kupfer- oder Lichtleitkabel
    - Erkennen bzw. vermeiden von physikalischen Übertragungsfehlern

# Protokoll-Schachtelung

z.B. ein HTTP-Kommando (ist ein “lesbarer” Text!):
“GET /index.html HTTP/1.1”

**Daten** — **Anwendung z.B HTTP** (HyperText Transfer Protocol)

---

**TCP-Header** **Daten** — **Transport (TCP/UDP)**

---

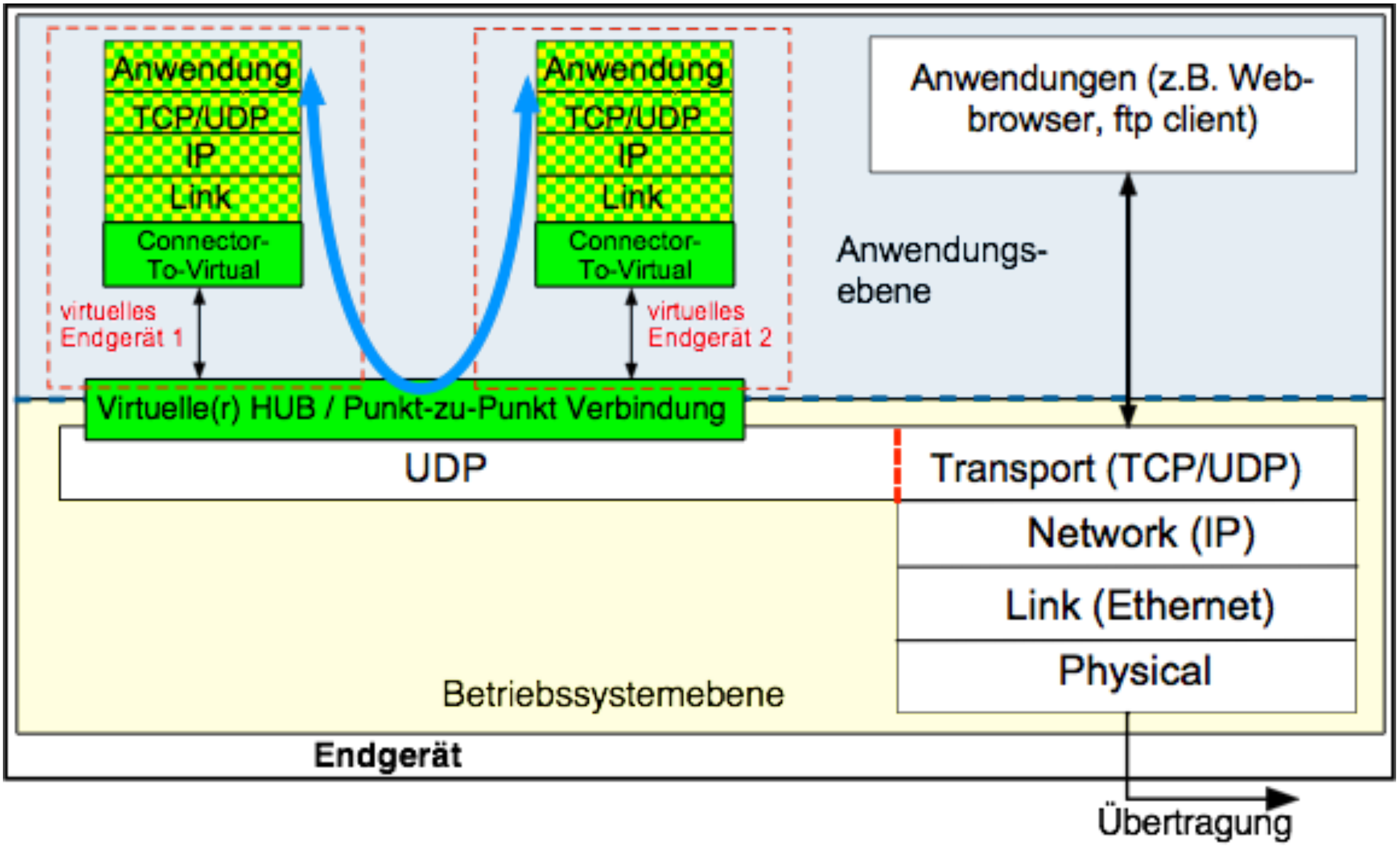
**IP-Header** **TCP-Header** **Daten** — **Vermittlung (IP)**

---

**MAC-Header** **IP-Header** **TCP-Header** **Daten** **MAC-Trailer** — **Verbindung/ Sicherungs z.B. IEEE 802.3 („Ethernet“)**

| MAC-Header | IP-Header | TCP-Header | Daten | MAC-Trailer |
|---|---|---|---|---|
| MAC-Ziel- und Quelladresse, Typ-Feld: 0800H für IP | IP-Ziel- und Quelladresse, Protokoll-Typ: 6 für TCP | Portnummer der Ziel- und der Quellanwendung, Sequenz- und Acknowledge-Nummer, verschiedene Steuerflags | Anwen-dungspro-tokoll und -daten | Fehler-prüfung |

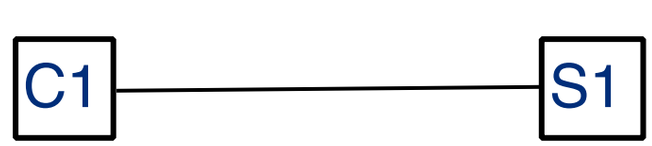
# Softwarestruktur für lokale Versuche



# Schritt 1

- Nehmen Sie den Versuch unter Verwendung der Geräte „client1“ und „server1“ in Betrieb, die einen HTTP-Client bzw. –Server darstellen

Netzwerk:




- Untypischerweise verwenden die Geräte C1 und S1 UDP zum Transport von HTTP
  - das vereinfacht die ersten Versuche
- Verwenden Sie UDP-Portadressen im Bereich 5100 bis 5199

# Schritt 2

Anwendung:

- Ergänzen Sie den Server *S1* so, das er bei einer HTTP-Abfrage des Dokuments *„daten“* durch den Client *C1* beliebigen Text in mehreren (20-30 oder mehr) Datenpaketen  an den Client sendet
  - der zeitliche Abstand zwischen den Paketen betrage ungefähr 300 ms (zur besseren Verfolgbarkeit der Vorgänge)
- Verwenden Sie Schritt 1 als Grundlage
- Generieren Sie die Daten auf beliebige Art und Weise

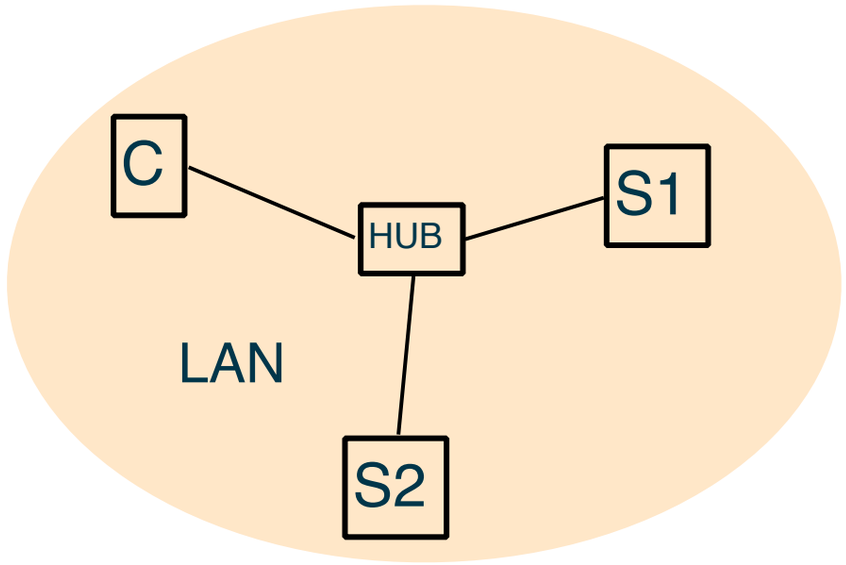
# Schritt 3

Vermittlungsschicht und Anwendung:

- Konfigurieren Sie ein LAN
- Implementieren Sie auf Gerät *S2* (*nameserver*) eine Anwendung (einen Dienst, genannt: Namensdienst, name service), welche Rechnernamen in Netzwerkadressen auflöst (z.B. „meinhttpserver“ in „192.168.1.10”)
  - welche „Fehlfunktion“ tritt auf, wenn der Client den Name-Server anspricht? Beobachten Sie den Server *S1*!
  - Sie müssen die Vermittlungsschicht ergänzen
  - nutzen Sie den Name-Server ab sofort für alle weiteren Versuche um die IP-Adresse des Servers *S1* zu erhalten

# Schritt 3 (2)

Netzwerk:



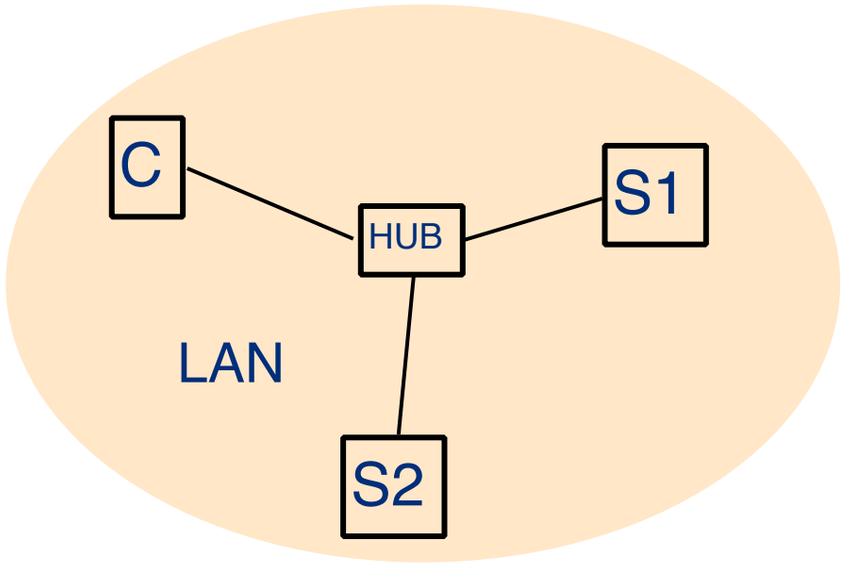
# Schritt 4

Übertragungsschicht:

- Im aktuellen Netzwerkausbau werden gesendete Daten von allen Geräten empfangen und erst die Vermittlungsschicht erkennt, ob das Gerät der Empfänger ist.
- Das ist ineffektiv, der Empfänger muss schneller erkannt werden! Führen Sie in der Übertragungsschicht eine weitere Adressierung bzw. Adresserkennung ein (*MAC*-Adressierung)
- Die Übertragungsschicht benötigt eine weitere Funktionalität: Bestimmung der MAC-Adresse aus der Netzwerkadresse des im selben LAN adressierten Gerätes.
- Verwenden Sie zuerst eine manuell verwaltete Tabelle (ARP–(Address Resolution Protocol) Tabelle) zu Abbildung von IP-Adressen auf MAC-Adressen
- Nehmen Sie dann das Protokoll ARP in Betrieb

# Schritt 4 (2)

Netzwerk:



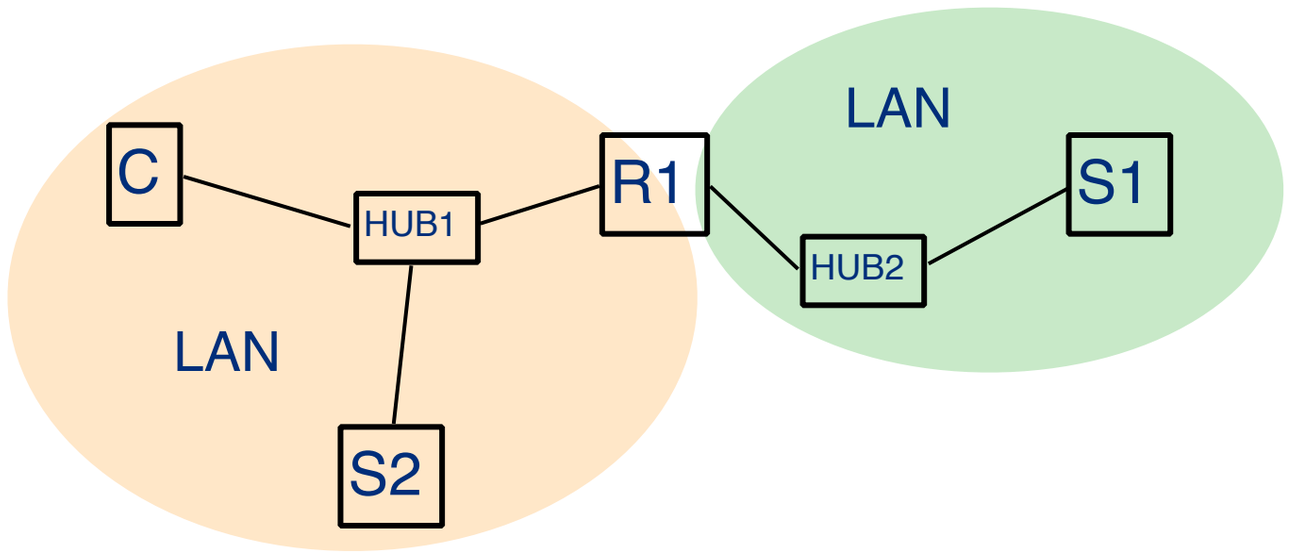
# Schritt 5

Vermittlungsschicht: Routing bzw. Forwarding

- Erzeugen Sie ein Netzwerkgerät mit zwei Netzwerk-Adaptern, einen „Router“ *R1*
  - verbinden Sie Client, Name-Server *S2* und Router über den HUB1, den Datei-Server *S1* verbinden Sie mittels eines zweiten Hubs *HUB2* mit dem zweiten Anschluss des Routers
  - werden Daten zwischen Client und Server *S1* übertragen?
  - in der Netzwerkschicht muss das Weiterleiten von Daten-Paketen auf der Basis einer Routing-Tabelle eingeführt werden
  - achten Sie darauf, das Pakete an ein Gerät im eigenen IP-Netzwerk (und damit im eigenen LAN) nicht über einen Router transportiert werden!

# Schritt 5 (2)

Netzwerk:



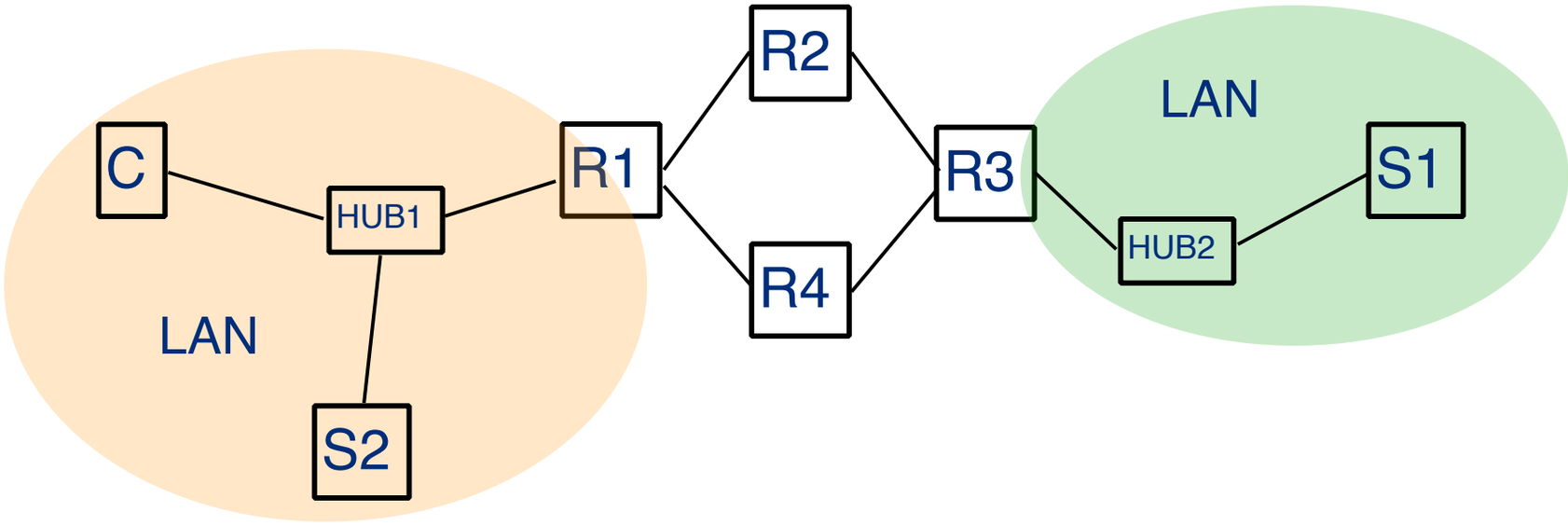
# Schritt 6

Vermittlungsschicht:

- Konfigurieren Sie das endgültige Netzwerk nach Aufgabenstellung
  - implementieren Sie einen einfachen Algorithmus, der zwischen den Routern periodisch Informationen über die ihnen bekannten Routing-Ziele (Adressen von Netzwerken) austauscht
    - bevorzugt ein Distance-Vector-Routing-Protokoll mit minimaler Funktionalität , als Anwendung implementiert (Verwendung von UDP als Transportprotokoll)
  - die Routingtabellen werden automatisch modifiziert, wenn sich die Netzwerkstruktur durch Ausfälle/Umstrukturierung ändert
  - die Router besitzen anfangs nur Wissen über die IP-Netzwerke, an die sie selbst angeschlossen sind, sie sollen in unserem Beispiel auch die IP-Adressen ihrer benachbarten Router kennen
  - führen Sie einen länger anhaltenden Datentransfer zwischen Client und Server *S1* über Router *R2* durch
  - während der Datenübertragung schalten Sie *R2* aus
  - nach kurzer Zeit sollte die Übertragung, diesmal über *R4*, wieder aufgenommen werden

# Schritt 6 (2)

Netzwerk:



# Schritt 7

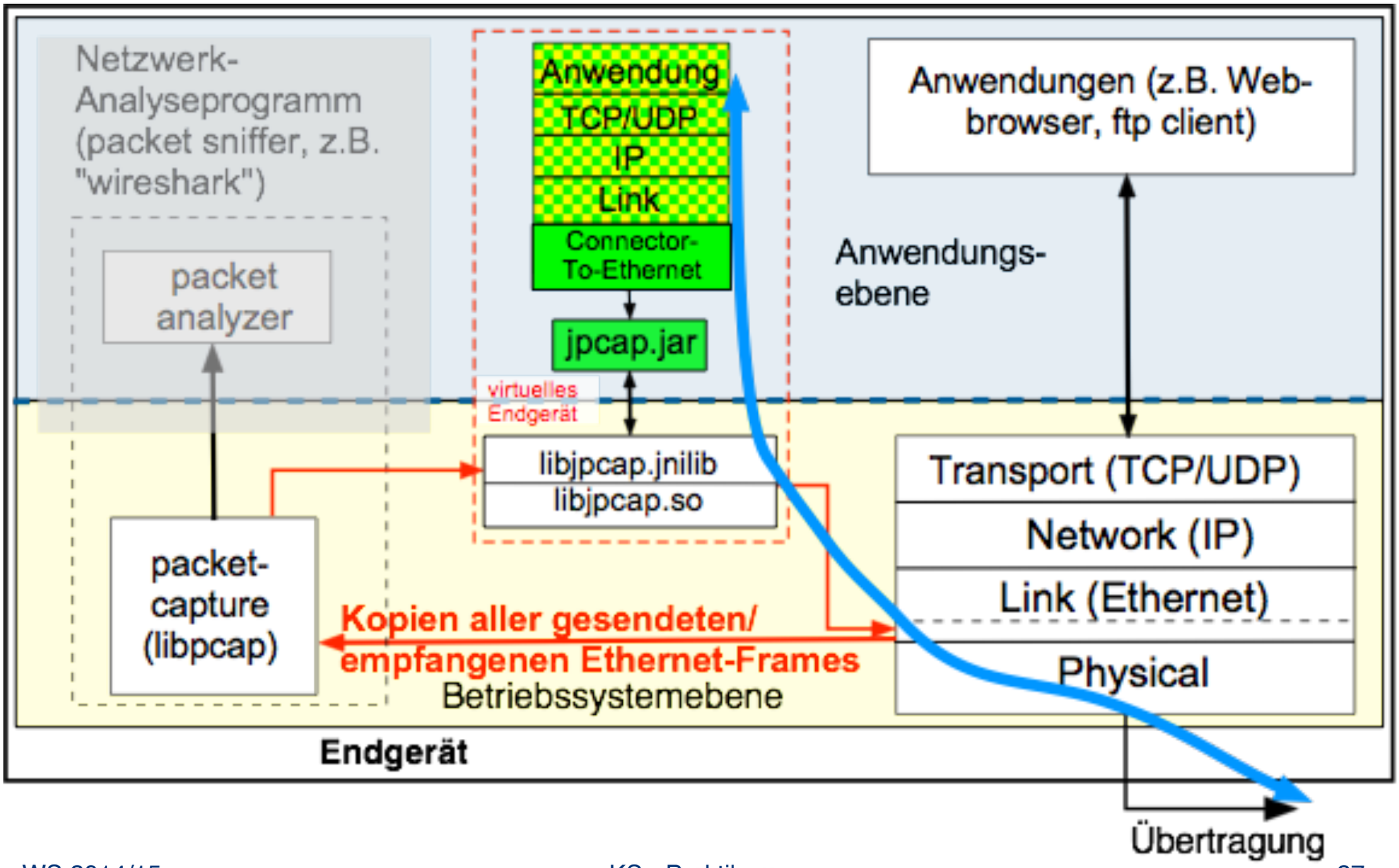
Transportschicht:

- Übertragen Sie von Ihnen bisher implementierte Funktionalitäten in die Geräte „client2“ (Bezeichnung sei weiterhin *C1*) und „server2“ (Bezeichnung sei weiterhin *S1*)
- Diese Geräte verwenden HTTP über TCP, so wie in realen Umgebungen
- Sie müssen nun TCP komplettieren, in dem Sie fehlende Funktionalitäten implementieren
  - dazu gehören die passive Verbindungsaufnahme und –beendigung sowie die Fehlerbehandlung und Flusssteuerung
- Nach Fertigstellung muss der Versuch aus Schritt 6 mit Fehlerkorrektur durch Sendewiederholung nach Paketverlust durchführbar sein

# Externe Versuche

- Nehmen Sie Ihren Geräte „*client2*“ und „*server2*“ in einer realen Umgebung in Betrieb
- Als MAC- und IPv4-Adressen verwenden Sie die Ihres Rechners, ebenso die IPv4-Adresse des Default-Routers
- Sie müssen (wahrscheinlich) die Firewall des Betriebssystems konfigurieren - hier eine Firewall-Regel für viele Linux-Varianten:
  - „iptables -A INPUT -p tcp --dport 5100:5199 -j DROP“
  - damit reagiert der TCP/IP-Stack des Betriebssystems nicht auf eingehende Pakete an den im Framework verwendeten TCP-Portbereich

# Softwarestruktur für externe Versuche



# Externe Versuche (2)

- Aufgabe 1: Nehmen Sie „client2“ in Betrieb und laden Sie ein HTML-Dokument von einem öffentlichen Web-Server Ihrer Wahl
- Aufgabe 2: Testen Sie ihren HTTP-Server „server2“ in einer realen Umgebung, d.h. von einem 2. Rechner aus

# Externe Versuche (3)

- Zusatzaufgabe: Bestimmen Sie die Werte von drei aufeinanderfolgenden TCP-Retransmission Timeouts des Web-Servers aus Aufgabe 1
  - öffnen Sie eine TCP-Verbindung und senden Sie ein GET-Request
  - warten Sie auf das erste Datenpaket vom Server und speichern Sie den Zeitpunkt des Eintreffens, bestätigen Sie das Datenpaket jedoch nicht
  - warten Sie auf das wiederholte Eintreffen des Datenpakets
  - berechnen Sie die Zeitdifferenz, usw.
- Externe Versuche spätestens bearbeitbar: Dezember

# Externe Versuche (4)

- Beachten Sie: bei jedem Versuch sollte am Schluss die TCP-Verbindung zum Server ordnungsgemäß beendet werden, da beim Server sonst Fehlerzustände erkannt werden könnten

# Labortermin

- Konfiguration der Netzwerkkomponenten (Verkabelung, IP-Konfiguration)
- Überprüfung der Funktionstüchtigkeit des Routers
- Inbetriebnahme der eigenen Versuchsprogramme
- Abnahme: Vorführung der eigenen Programme und Diskussion der Versuchsergebnisse mit allen Gruppenmitgliedern
- Für die Laborarbeit werden rechtzeitig weitere Unterlagen ausgegeben
- Ort: RUD25, 4.309

# Zeitliche Aufteilung

- Siehe Angaben in der Aufgabenstellung
- Zwischenabnahme ab 08.12.2014
  - Näheres wird noch bekannt gegeben
- Da die Aufgaben von Vierer-Gruppen bearbeitet werden (Arbeitsteilung!), sind für den Softwareentwicklungs-Anteil mindestens 6 SWS vorgesehen (plus 2 SWS für die „Koordination“ der Gruppe)
- Endabnahme im KS-Labor ab 26.01.2015
- Für jede Gruppe ein Termin mit 2.5 Stunden Dauer, weitere Einzelheiten werden noch bekanntgegeben und Listen ausgelegt

# Hinweise zur Labortermin

- Ort: RUD25, 4.309
- Alle Gruppenmitglieder sind anwesend
- Installation
- Vorführung
- Befragung aller Gruppenmitglieder zu allen Aufgaben
  - **in der Diskussion während der Abnahme sollte *jeder* den Quellcode des Beispiels und natürlich auch den eigenen Quellcode erläutern können**
  - **jeder sollte Überlegungen (z.B. Algorithmus, Anpassungen im Quelltext, Aufwandsabschätzung) zur Implementierung noch fehlender Funktionalitäten der verwendeten Protokoll-Implementierungen anstellen können**

# Framework: Hinweise (1)

- Das Framework beinhaltet grundlegende Module zur Programmierung von Netzwerksoftware
- Implementiert sind rudimentär die Protokolle TCP, UDP, IPv4, ARP, IEEE 802.3 Rahmenbildung
- “Kabel” werden entweder durch Kommunikationskanäle in einem existierenden IP-Netzwerk simuliert (“virtuelle Kabel”) oder es wird ein real existierendes Netzwerk verwendet
- Auf jedem Gerät kann nur eine einzelne Anwendung ausgeführt werden, die genau eine TCP-Verbindung verwendet; gleichzeitig kann über UDP kommuniziert werden
- Zu jedem Zeitpunkt wird nur eine einzelne Transport-Verbindung unterhalten (dadurch Vereinfachung der Transportschicht)
- Die MAC-Frames enthalten keine Fehlererkennungssequenz
- Um in realen Netzwerken kommunizieren zu können wird die Java-Library „Jpcap“ verwendet, die eine Java-Schnittstelle zur Library „libpcap“ bereitstellt

# Hinweise (2)

- Die Paketlänge und zeitliche Abstände der Übertragungen sind so zu wählen, dass die Wirkungsweise des Protokollstacks gut gezeigt werden kann
- Vorgänge während der Übertragung visuell verfolgbar protokollieren (Sende-Timeouts und –Wiederholungen, Änderungen in den Routing-Tabellen, ...)
- Der Protokollstack soll für alle Geräte (Endgeräte und Router) identisch sein

# Projekt mit IntelliJ IDEA

- Zip-Datei in einen Ordner entpacken („KS_Praktikum“)
- Anlegen eines Verzeichnisbaums aus den Quellen des Frameworks
  - ~/IdeaProjects/KS_Praktikum/doc/...
  - ~/IdeaProjects/KS_Praktikum/src/...
  - ...
- IntelliJ IDEA starten
- Neues Projekt beginnen
  - „Create New Project“
  - links „groovy“ auswählen, rechts „Project SDK“ (Java, ggf. Pfad angeben) und „Groovy library“ („Create“, Pfad zur Groovy library) entsprechend angeben, dann „Next“
  - „Project Location“ angeben (~/IdeaProjects/KS_Praktikum), dann „Finish“

# Projekt mit IntelliJ IDEA (2)

- Notwendige Einstellungen
  - Menü „file“, „project structure“, „modules“, „sources“, im Baum „doc“ auswählen und „excluded“ wählen, ok
  - Menü „file“, „project structure“, „libraries“, Hinzufügen (+) wählen, „KS_Praktikum/libs/jpcap.jar“ wählen, ok, ok
- Test der Konfiguration
  - Menü „build“, „make project“

# Projekt mit IntelliJ IDEA (3)

- Programm starten
  - ist nur bei installierter Programmierbibliothek „libjpcap“ und Arbeit mit Administrator- (root-) Rechten möglich
  - innerhalb der IDE
    - im Kontextmenü der Client-, Server- und Router-Klassen: „Run“
  - im Terminal
    - In den Scripts (start.sh) Pfad der kompilierten Klassen eintragen (~/IdeaProjects/KS_Praktikum/out/production/KS_Praktikum)
    - Scripts im Terminal ausführen
- Optionale Einstellungen
  - im Projekt-Panel Einstellungen wählen (Zahnrad-Symbol), beide „autoscroll...“ auswählen, „Compact Empty Middle Packages“ abwählen
  - rechter Fensterrand „ant build“ auswählen, Hinzufügen (+), „build.xml“ auswählen

# Projekt mit IntelliJ IDEA (4)

- Versions-Kontroll-System verwenden (VCS, hier:Git, optional)
  - Menü „VCS“, „import into version control“, „create git repository“, Projektverzeichnis auswählen, ok, „Git Init“-Dialog, ok
- In Git repository einchecken (optional)
  - Quelltexte, markieren
  - Menü „VCS“, „git“, „add“
  - „VCS“, „git“, „commit“
- Es kann ebenfalls mit Git-Servern (die Fachschaft unterhält einen, informieren Sie sich) oder z.B. SVN gearbeitet werden

# Informationsgewinnung

- Eigene IP- und MAC-Adresse bestimmbar durch das Kommando „ifconfig -a“ und suchen des Eintrags für das LAN-Interface z.B. „eth0“ bei Linux
- IP- (Netzwerk)-Adresse eines Rechner kann bestimmt werden durch das Kommando „host“ z.B. „host speedtest.qsc.de“
  - ebenfalls möglich „ping speedtest.qsc.de“
- IP-Adresse des eigenen (Default-) Routers bestimmbar durch das Kommando „netstat -nr“ und suchen des „destination“-Eintrags „default“
- Eine Liste der bekannten MAC-IP-Adressen-Zuordnungen wird durch das Kommando „arp -an” angezeigt
  - MAC-Adresse des Default-Routers durch ”arp -an | grep ”192.168.178.1” wenn ”192.168.178.1” die IP-Adresse des Default-Routers ist
- Beschäftigen Sie sich auch mit den Kommandos „traceroute“, „tcpdump“ und „wireshark“
- Achtung: „wireshark“ und das Framework können nicht gleichzeitig auf dem selben Rechner verwendet werden