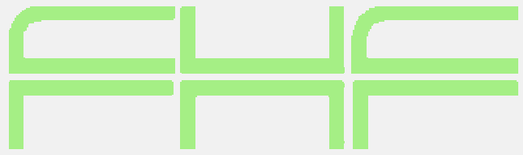
Performancemessung und –vergleich von SOAP-basierten Web Services



Prof. Mario Jeckle

Fachhochschule Furtwangen
mario@jeckle.de
http://www.jeckle.de

Web Services/XML One, München, 2003-05-14

Inhaltsübersicht

- Einführung und Motivation:
  Verteilungsarchitekturen und Einflußfaktoren auf ihre Leistungsfähigkeit
- Methoden der Leistungsbewertung
- Methodik zur technologieneutralen Leistungsbewertung
- Meßergebnisse und ihre Interpretation
  - Java Remote Method Invocation (RMI)
  - Common Object Request Broker Architecture (CORBA)
  - SOAP
- Ableitung übertragbarer Optimierungsansätze
  Ansätze eines analytischen Modells

Einführung und Motivation

Was ist *Performance*?

- Antwortzeit
- Netzlast
- Auslastung

Viel ist besser als mehr!

... aber ...

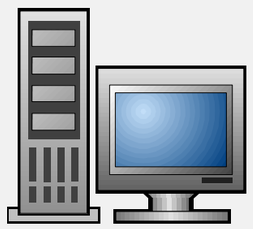
- wieviel ist genug?
- was liefert welche (Tuning-)Maßnahme?
- was bieten existierende Alternativen?

Einführung und Motivation

„klassische“ 4-Schichtenarchitektur

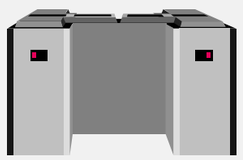
Präsentation

HTTP, SSL, SOAP, ...



Präsentations-aufbereitung

RMI, CORBA, ...

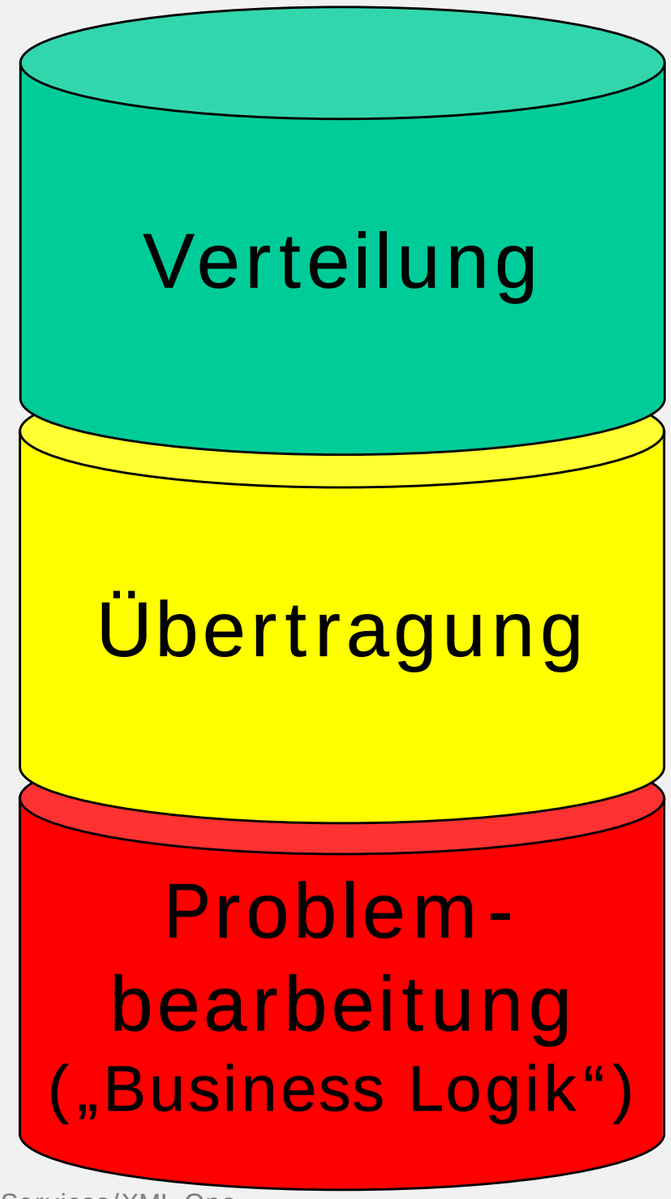


Anwendungslogik

ESQL, JSQL, O/JDBC,...

Datenhaltung

Einflußfaktoren auf die „gefühlte“ Performance

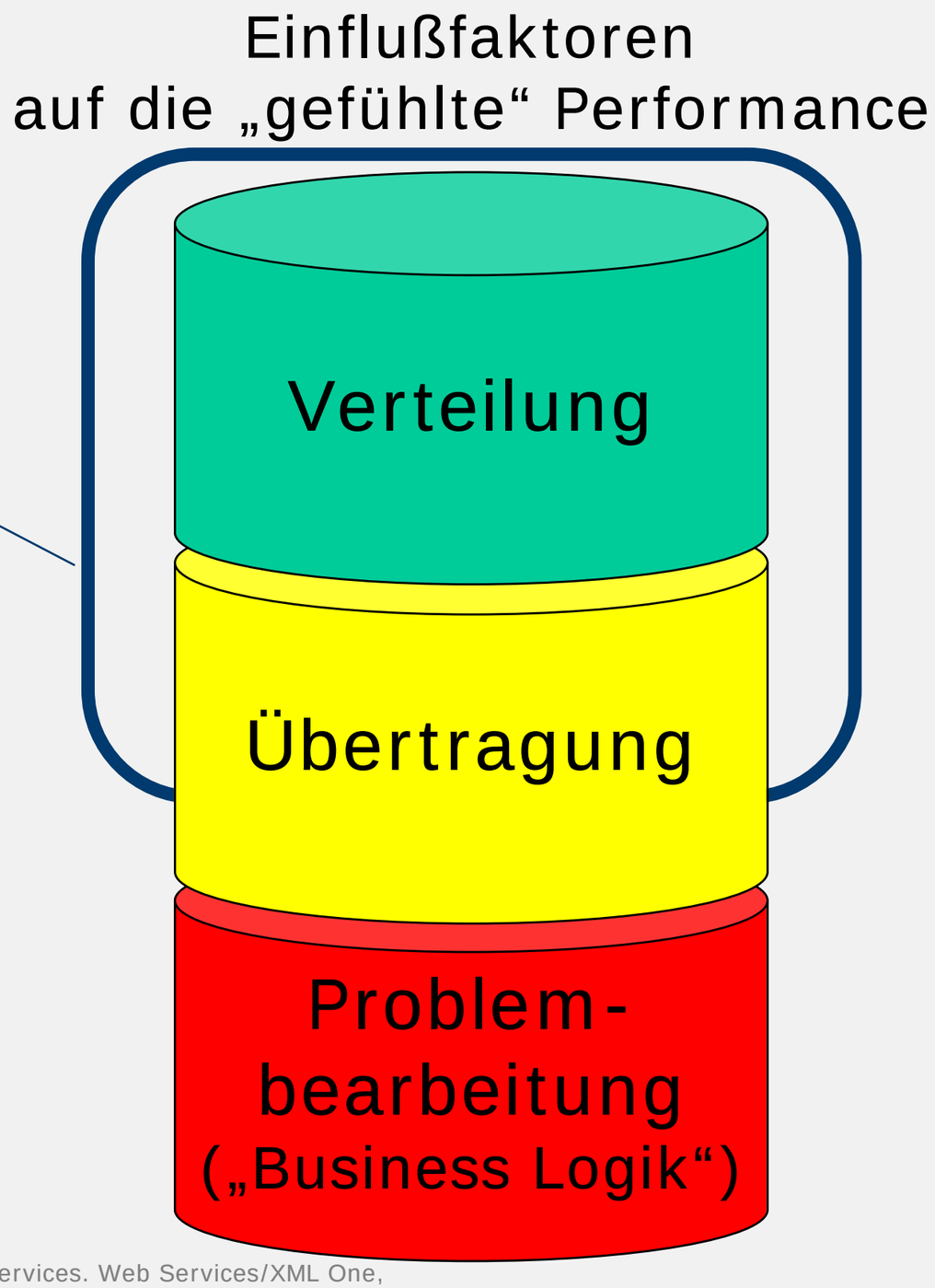


Einführung und Motivation

In diesem Vortrag betrachtet:

- RMI
- CORBA
- SOAP
- HTTP

- Antwortzeit
- Netzlast
- CPU-Last



Methoden zur Leistungsbewertung

	Analytische Methoden	Simulation	Messung
Einsatz	Entwurf, Konfigurations-bewertung, Engpaßanalye		Systemauswahl
Ansatz	mathematisch	programmier-technisch	Beobachtung
Realitäts-nähe und Detaillie-rungsgrad	gering bis mäßig	gering bis hoch	mäßig bis sehr hoch
Werkzeuge	gut	sehr gut	häufig Spezialent-wicklung notwendig

Methoden zur Leistungsbewertung

	Gewählter Ansatz	Messung
Einsatz	Existierende (kommerziell) verfügbare Systeme	Systemauswahl
Ansatz	Interpretation systematischer Messungen	Beobachtung
Realitäts-nähe und Detaillie-rungsgrad	Umfangreiche Messungen, hohe Realitätsnähe	mäßig bis sehr hoch
Werkzeuge	Eigenentwickeltes Verfahren, verfügbare Hilfstools	häufig Spezialent-wicklung notwendig

Getestete Konfigurationen

- Standards: SOAP v1.1, HTTP/1.1
- Ethernet 10Mbit/s, Rechner durch Hub verbunden
- Client:
  - AMD Athlon 1800Mhz
  - SUN J2SE v1.4.1_02-b06
  - Apache Axis v1.1 RC2
  - Windows XP Home
    (v5.1 Build2600.xpsp1.020828-1920, SP1)
- Server:
  - Dual AMD Athlon 1200Mhz
  - SUN J2SE v1.4.1_02-b06
  - Apache Axis v1.1 RC2
  - Jakarta Tomcat v4.1.26
  - Apache v1.3.27
  - Perl v5.8.0
  - Suse Linux v8.1 (Kernel 2.4.19)

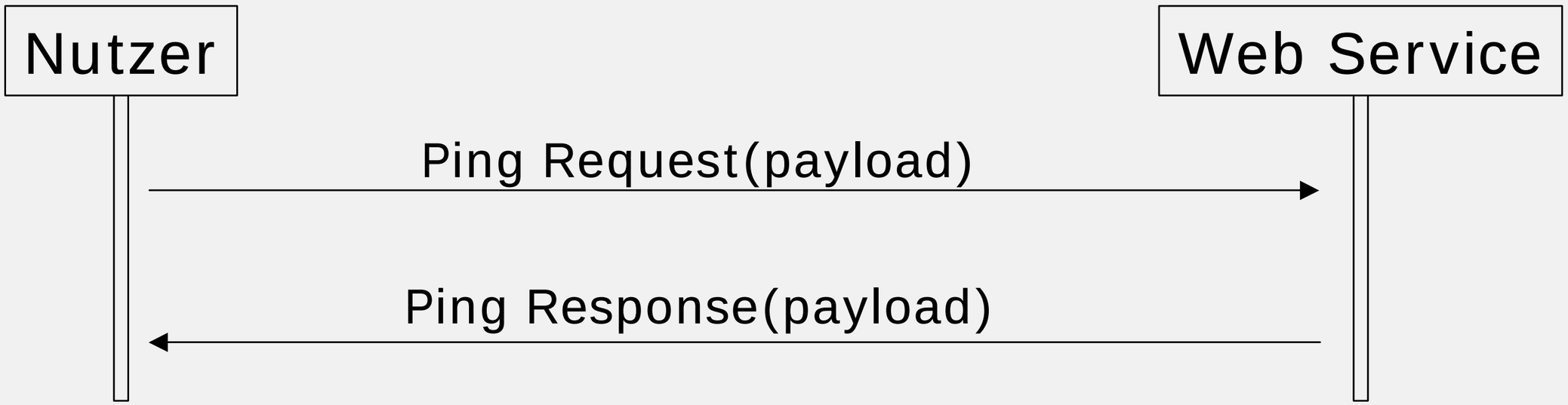
Durchgeführte Messungen

- >1.200.000 Einzelmessungen mit RMI, CORBA und SOAP
- Ausgewertet hinsichtlich
  - Antwortzeit
  - CPU-Last
- Interpretiert hinsichtlich
  - Skalierungsverhalten
  - Fehlersituationen
- Keinerlei Optimierungen
  - Prozeßprioritätsmodifikationen
  - Zeitgeber
  - Netzverkehr

Meßmethodik

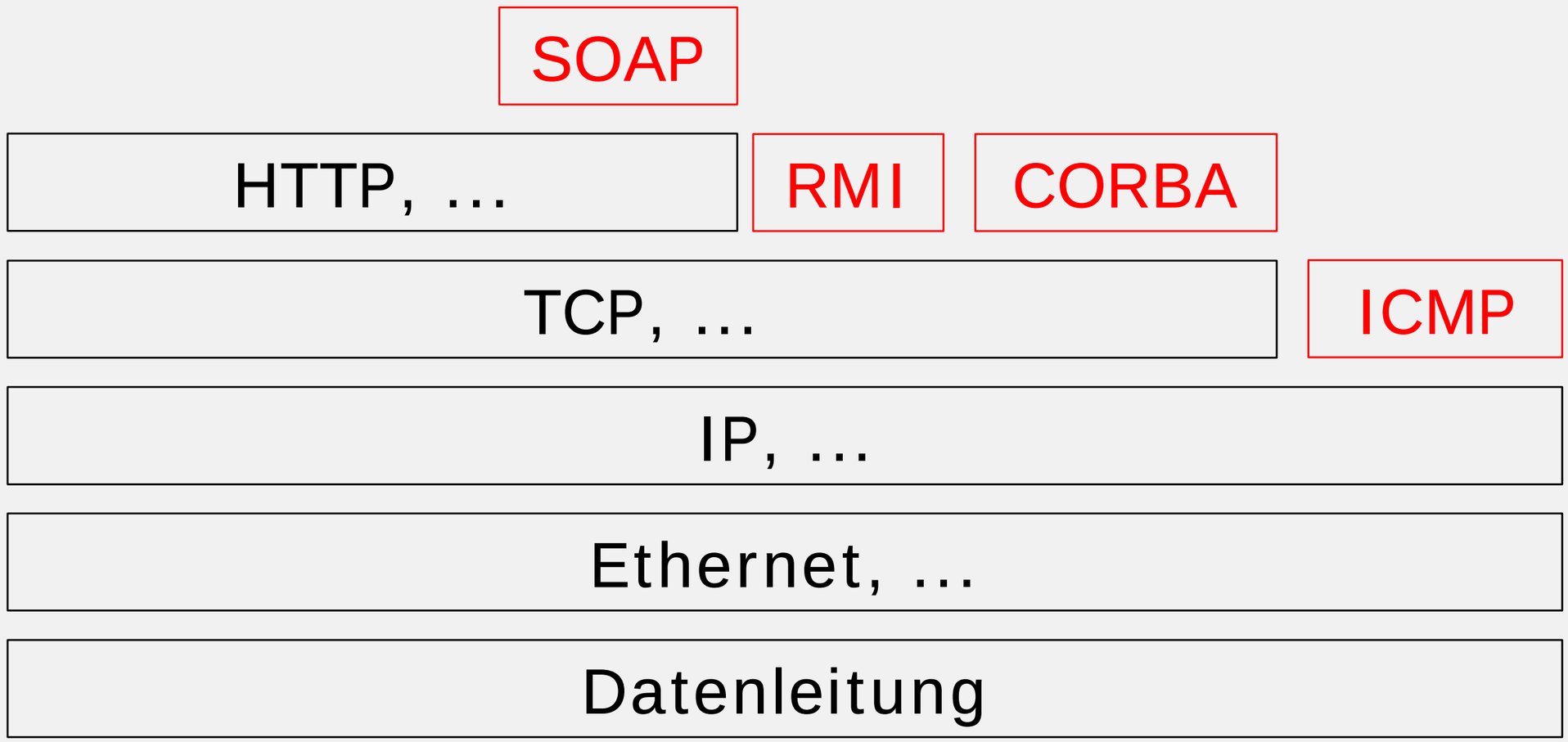
- Ziel:
  - **Ausschaltung verzerrender Backendeinflüsse**
    d.h. keine zusätzlichen Systemanforderungen oder Kommunikationsschritte
  - **Messungen am praxisnahen Beispiel**
    d.h. kein „Spiel“-Web Service
  - **Skalierbarkeit der Meßmethodik**
    d.h. unveränderter Dienst zur Messung beliebiger Kommunikationsgrößen
  - **Portabilität** (horizontal und vertikal)
    d.h. hinsichtlich eingesetzer Verteilungsarchitektur (RMI, CORBA, SOAP) und eingesetzer Ausprägung einer konkreten Verteilungsarchitektur (SOAP-Toolkit, konkreter ORB)
  - **Anwendbarkeit für verschiedene Kommunikationstypen**
    d.h. Unterstützung für RPC- und Message-artige Kommunikation
  - **Protokollneutralität**
    d.h. keine Festlegung auf HTTP

Meßmethodik



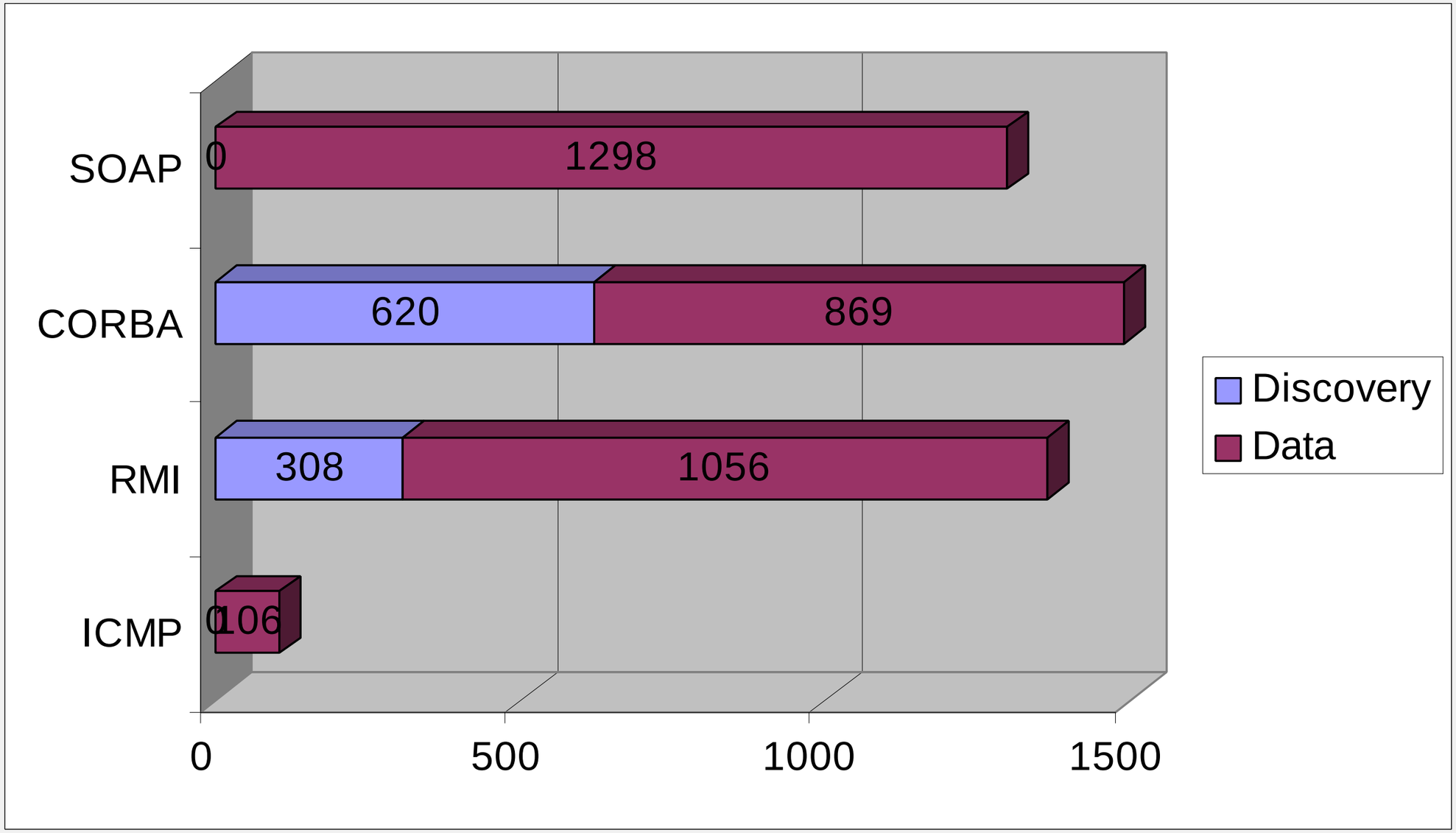
- Ansatz:
  - Orientierung am ICMP Echo-Dienst nach RFC 792 (*Ping*)
    - Symmetrische Netzlasten
    - Keine Serverlogik
    - Praktisch keine Ansprüche an Ausführungsumgebung
  - Ultra-schlanker Dienst (3-Zeilen Servercode!)
  - Portabler Java-Client für Messungen

Ergebnisse



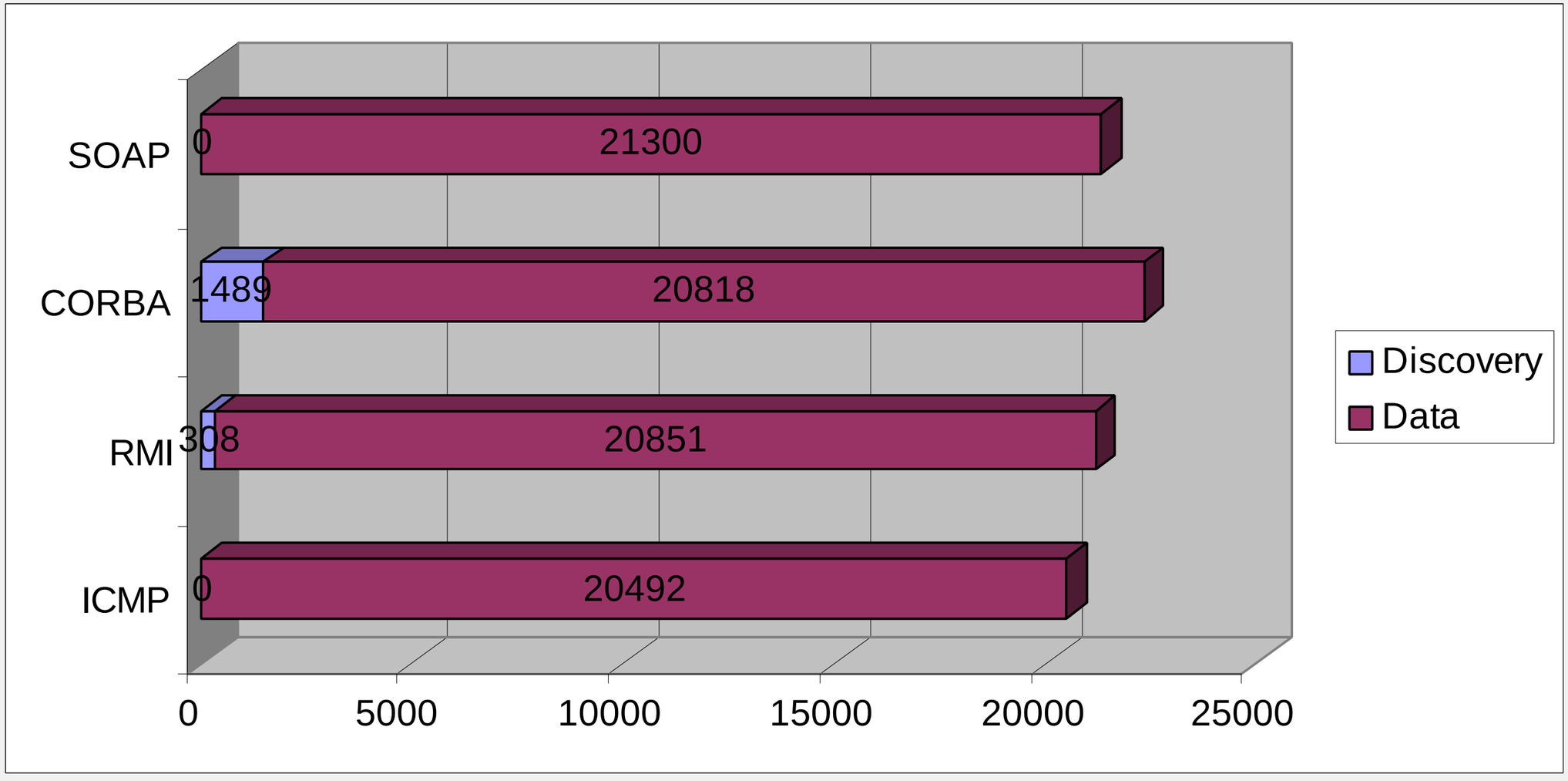
- Direkter Vergleich der Meßergebnisse konzeptionell nur schwer möglich, da Protokolle auf unterschiedlichen Ebenen operieren...
- In der praktischen Anwendung wird dies jedoch vernachlässigt!

Ergebnisse
I Kommunikationsgröße



Gemessen: 0-Byte großes Nutzdatenpaket

Ergebnisse
I Kommunikationsgröße



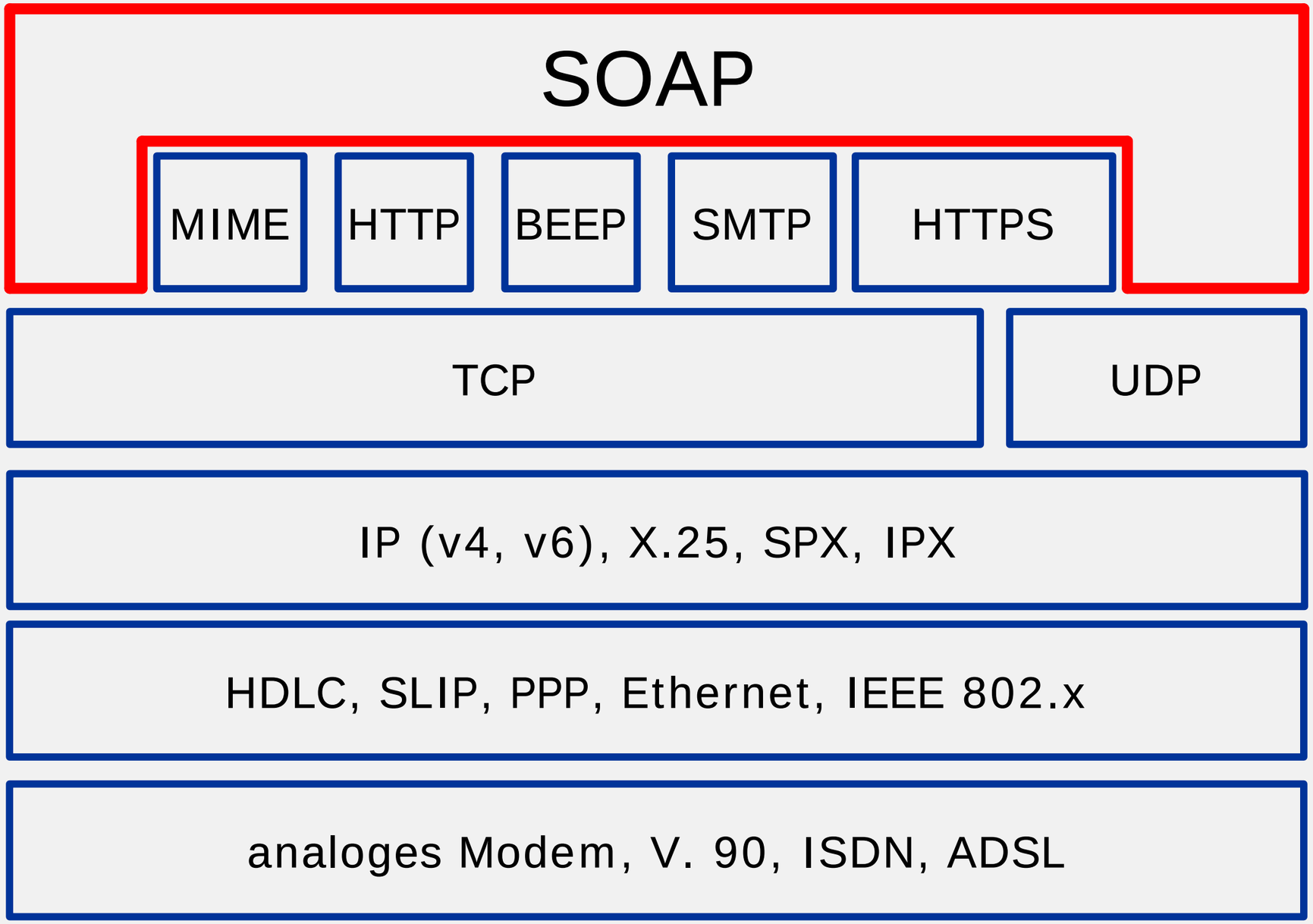
Gemessen: 10.000-Byte großes Nutzdatenpaket

Ergebnisse
I Kommunikationsgröße

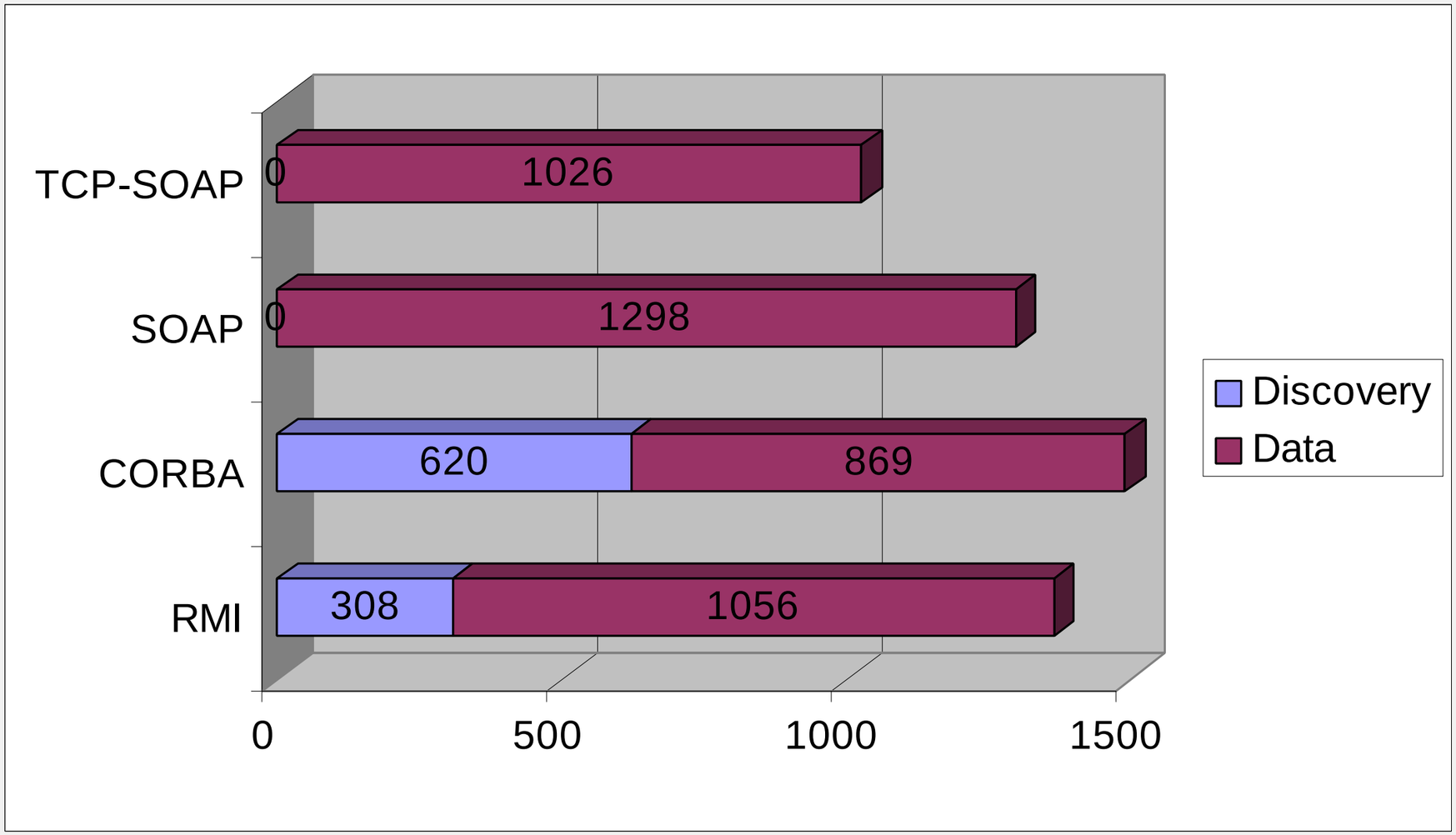
- SOAP:
  - 0-Byte Nutzlast ≙ 1298 Byte (gemessen)
  - 20.000-Byte Nutzlast ≙ 21298 (Erwartungswert)
    21300 (gemessen)
    (Abweichung 0,1%)
- CORBA:
  - 0-Byte Nutzlast ≙ 869 Byte (gemessen)
  - 20.000-Byte Nutzlast ≙ 20869 (Erwartungswert)
    20818 (gemessen)
    (Abweichung 0,2%)
- RMI
  - 0-Byte Nutzlast = 1056 Byte (gemessen)
  - 20.000-Byte Nutzlast = 21056 (Erwartungswert)
    20851 (gemessen)
    (Abweichung 1%)

=> Kommunikationsgröße skaliert linear!

Einordnung von SOAP in der Protokollhierarchie

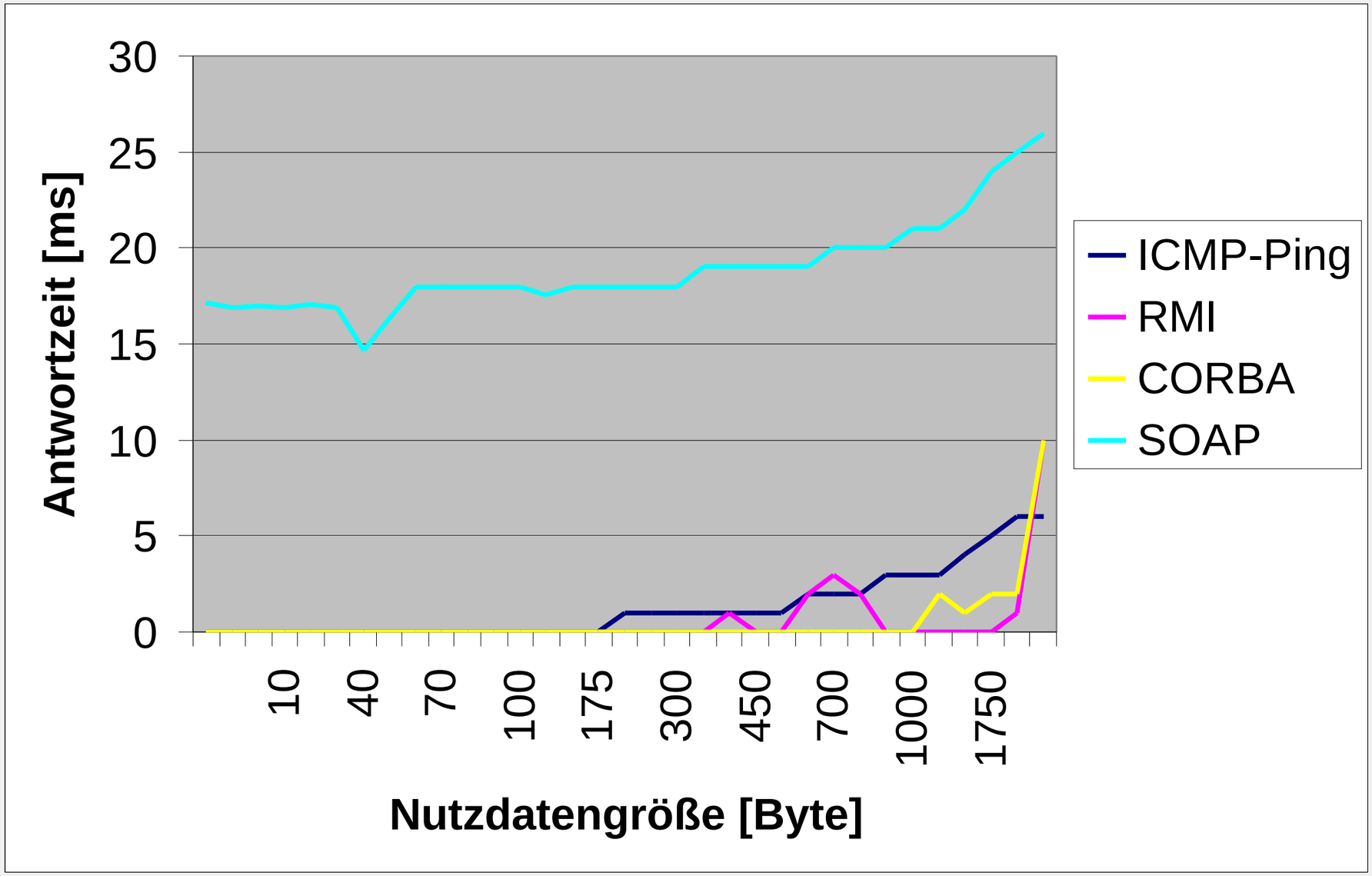


Ergebnisse
I Kommunikationsgröße



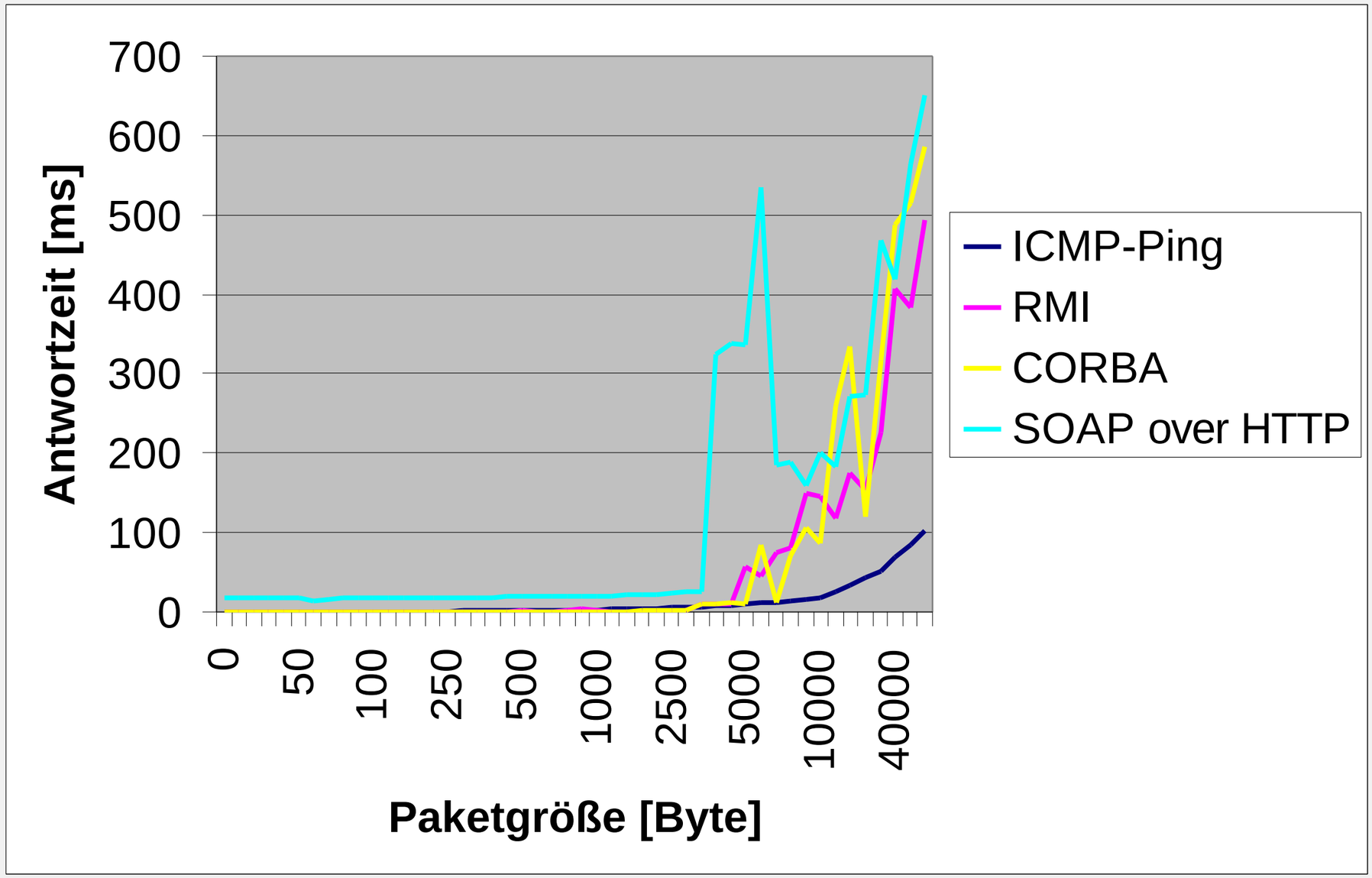
0-Byte großes Nutzdatenpaket

Ergebnisse
II Kommunikationszeit



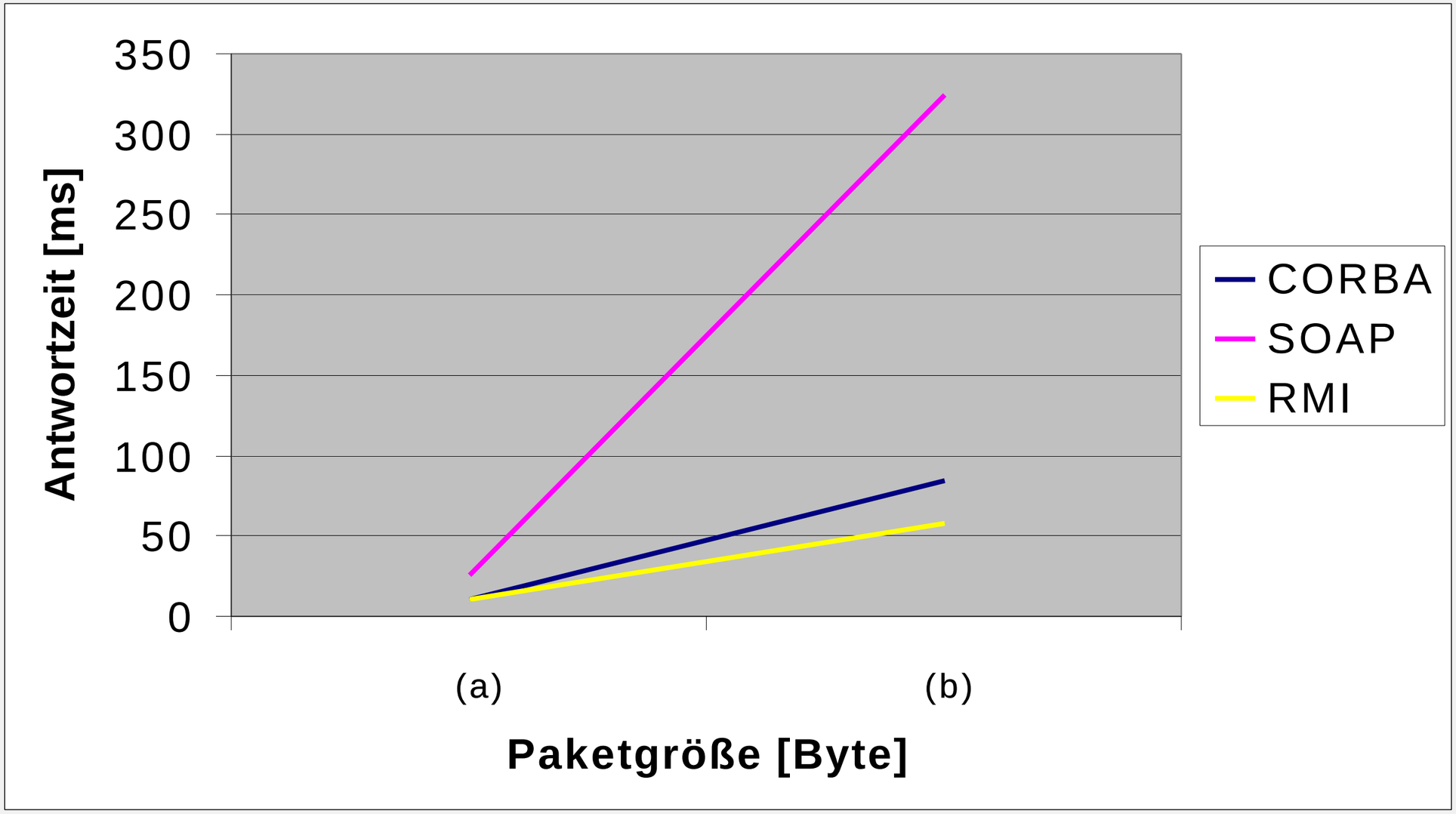
Deutlich sichtbar: HTTP-Overhead bei SOAP

Ergebnisse
II Kommunikationszeit



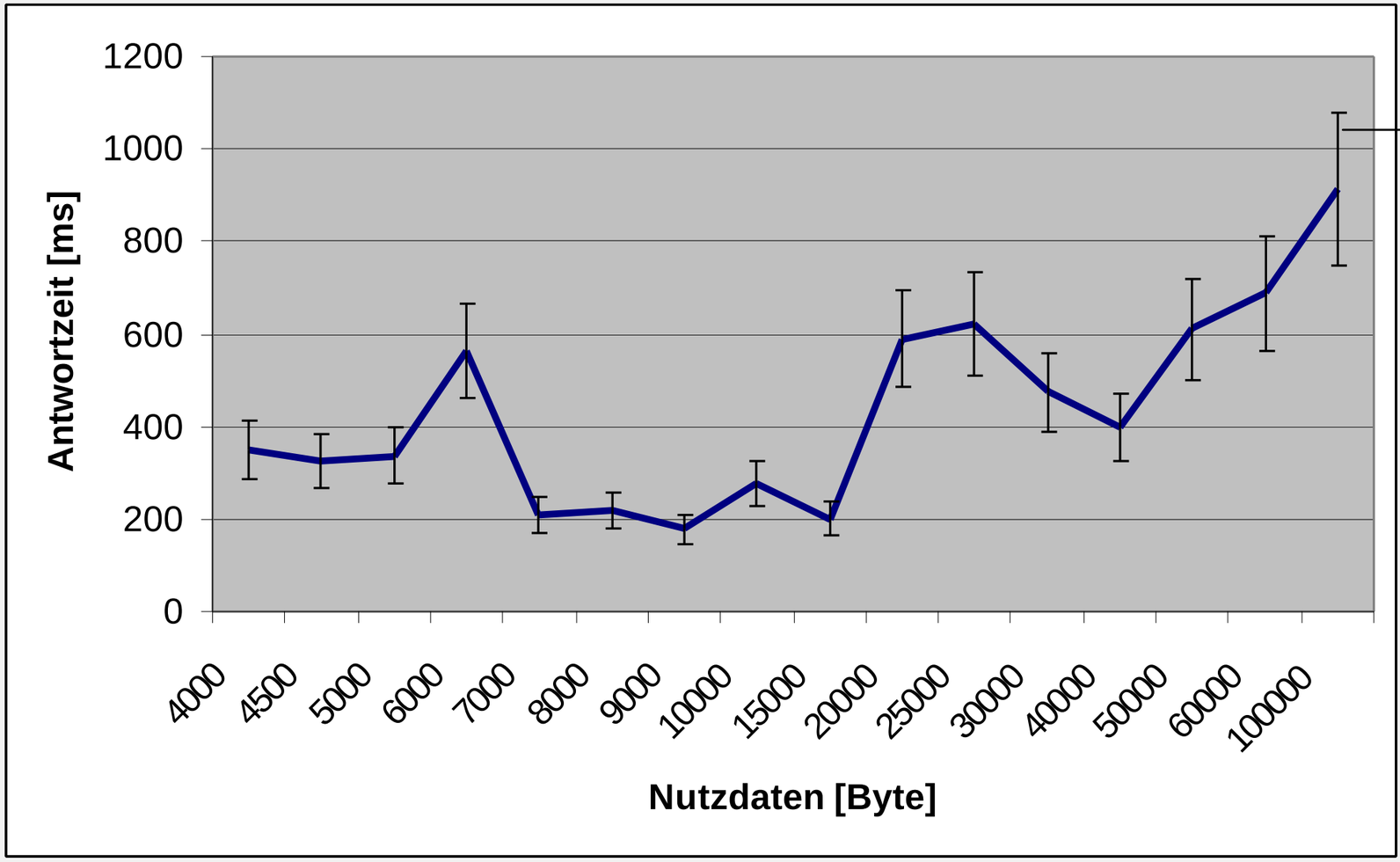
Kommunikationszeit skaliert exponentiell,
trotz linear skalierender Kommunikationslast?

Ergebnisse
II Kommunikationszeit



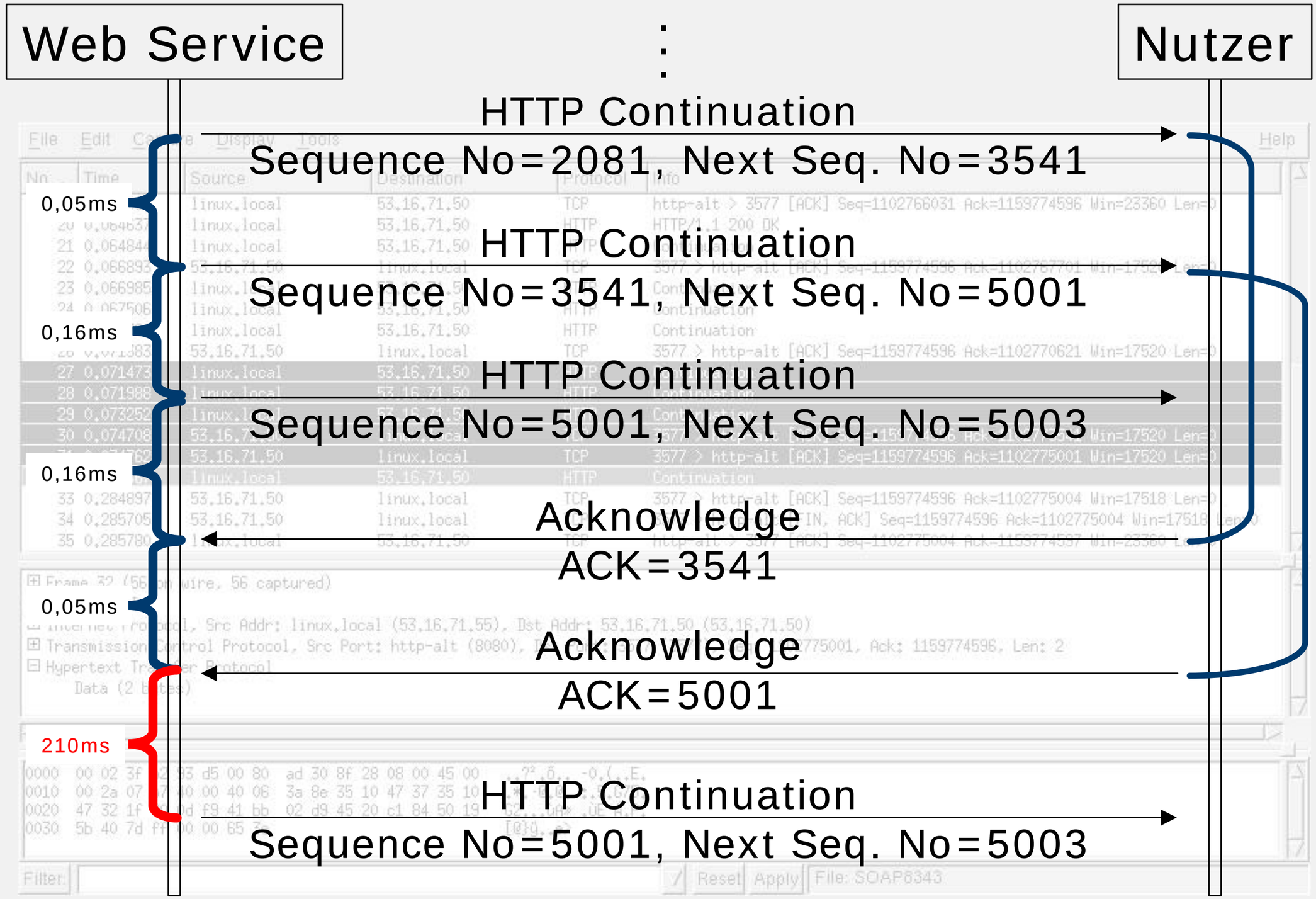
RMI	CORBA	SOAP
(a) 4500B: 10ms	5000B: 10ms	3500B: 26ms
(b) 5000B: 58ms	6000B: 60ms	4000B: 324ms

Ergebnisse
II Kommunikationszeit



- **Beobachtung**: Reproduzierbare emergente extreme Schwankungen („vorherseh- aber unvorhersagbare Ausreißer“)
- **Erklärung**: Ethernet-Kollisionsbedingte Überschreitung der TCP Retransmission-Time (RTO)

Ergebnisse
II Kommunikationszeit



Ergebnisse
II Kommunikationszeit

- **Hintergrund**:
  - TCP (RFC 675) errichtet ein verbindungsorientiertes Protokoll ausgehend vom verbindungslosen IP.
  - Einführung des *Retransmission Time Outs* durch RFC 2988 (Algorithmus nach V. Jacobson).

- **Retransmission Time Out**:
  - Ziel: Dynamische Erkennung der Zeitspanne nach der „sicher“ ein Paketverlust angenommen werden kann.

Ergebnisse
II Kommunikationszeit

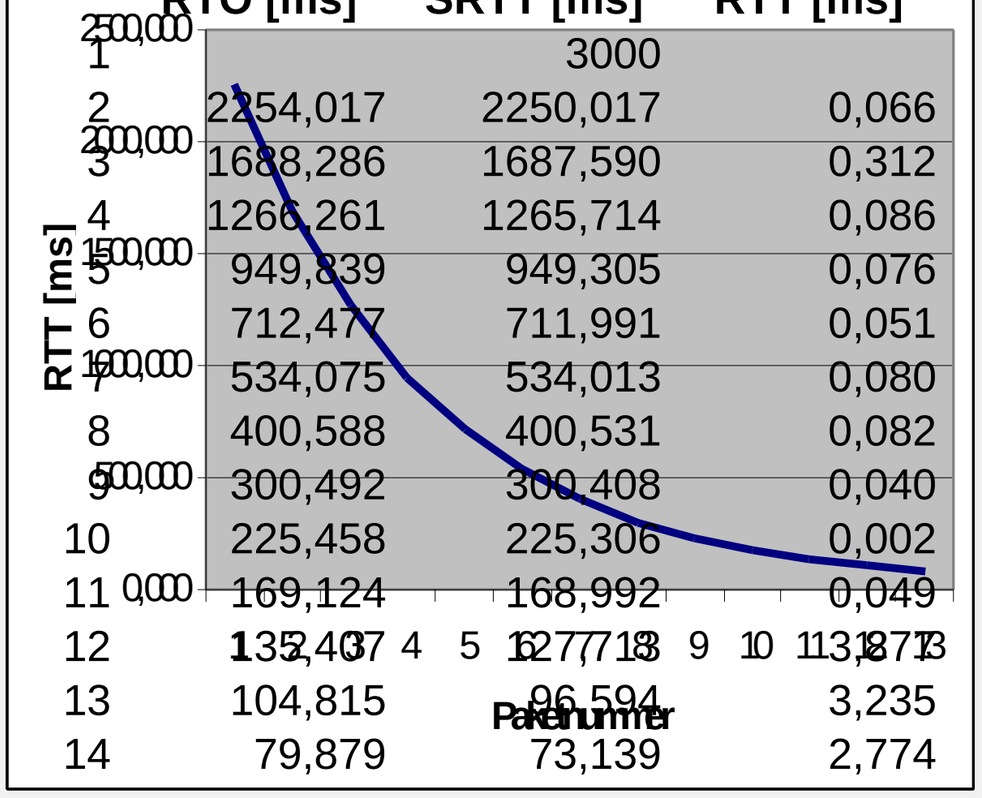
Retransmission Time Out

$SRTT_i = 0{,}75 * SRTT_{i-1} + 0{,}25 * RTT_i$

$RTO_i = SRTT_i + 4 * StaAbw(RTT_{i-1}, RTT_{i-2}, ...)$

- RTT: Gemessene Übertragungsdauern
- SRTT: Glättung durch Einbezug „historischer“ Werte
- RTO: Erwartungswert und Toleranzschwelle für zukünftige Übertragungsdauern

	RTO [ms]	SRTT [ms]	RTT [ms]
1		3000	
2	2254,017	2250,017	0,066
3	1688,286	1687,590	0,312
4	1266,261	1265,714	0,086
5	949,839	949,305	0,076
6	712,477	711,991	0,051
7	534,075	534,013	0,080
8	400,588	400,531	0,082
9	300,492	300,408	0,040
10	225,458	225,306	0,002
11	169,124	168,992	0,049
12	135,407	127,713	13,877
13	104,815	96,594	3,235
14	79,879	73,139	2,774



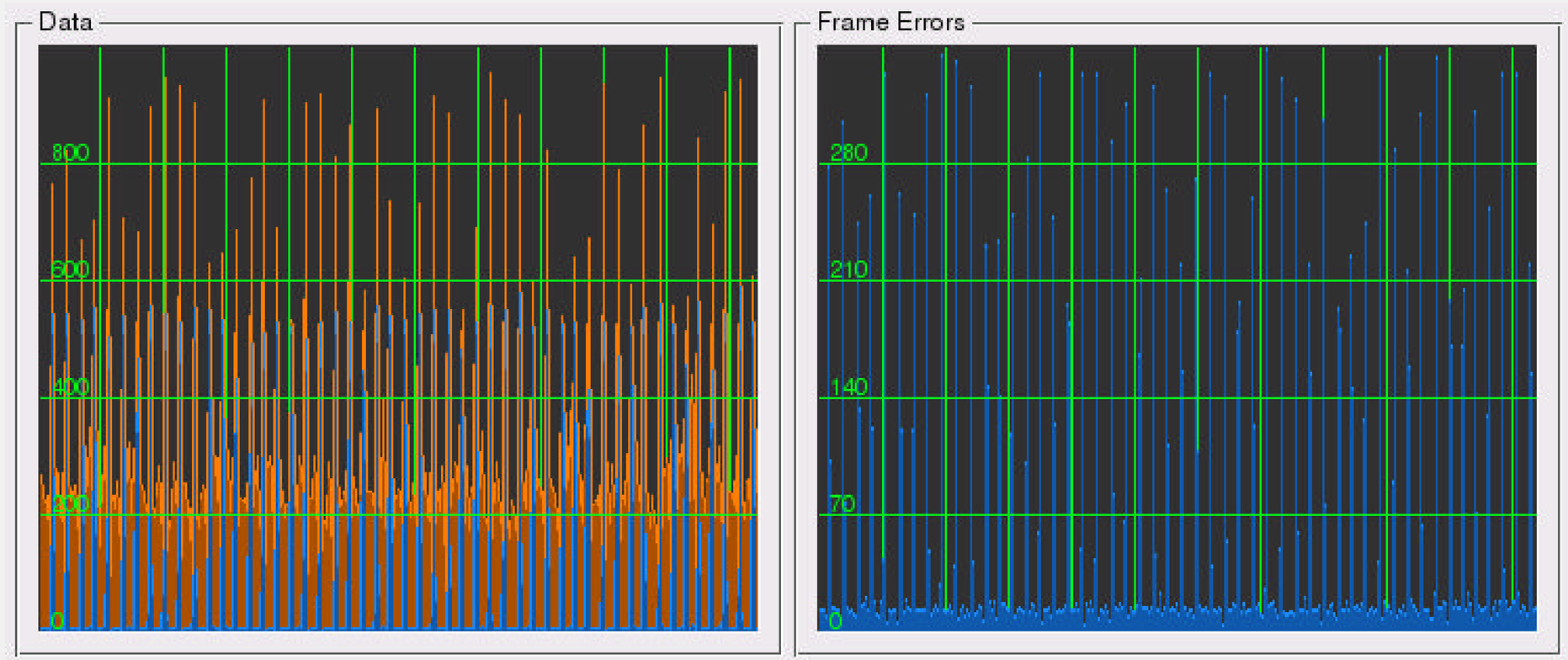
RTT: Round Trip Time
RTO: Retransmission Time Out
SRTT: Smoothed Round Trip Time

Ergebnisse
II Kommunikationszeit

Retransmission Time Out und CSMA/CD-Netze

- Spontanes unkoordiniertes Versenden von Paketen über Netzwerk kann zu *Kollisionen* führen.
- Im Kollisionsfalle unternehmen die betroffenen Netzwerkkomponenten *applikationstransparent* einen erneuten Sendevorgang.
- Nimmt dieser erneute Versuch zuviel Zeit in Anspruch oder muß (aufgrund erneuter Kollisionen) mehrmals vorgenommen werden, so können Pakete später eintreffen als die RTO-Schelle als Akzeptanzgrenze festlegt.
- Betroffen: Alle TCP-basierten Protokolle (damit auch RMI, CORBA und SOAP) mit besonderer Anfälligkeit für starkbelastete Netze.

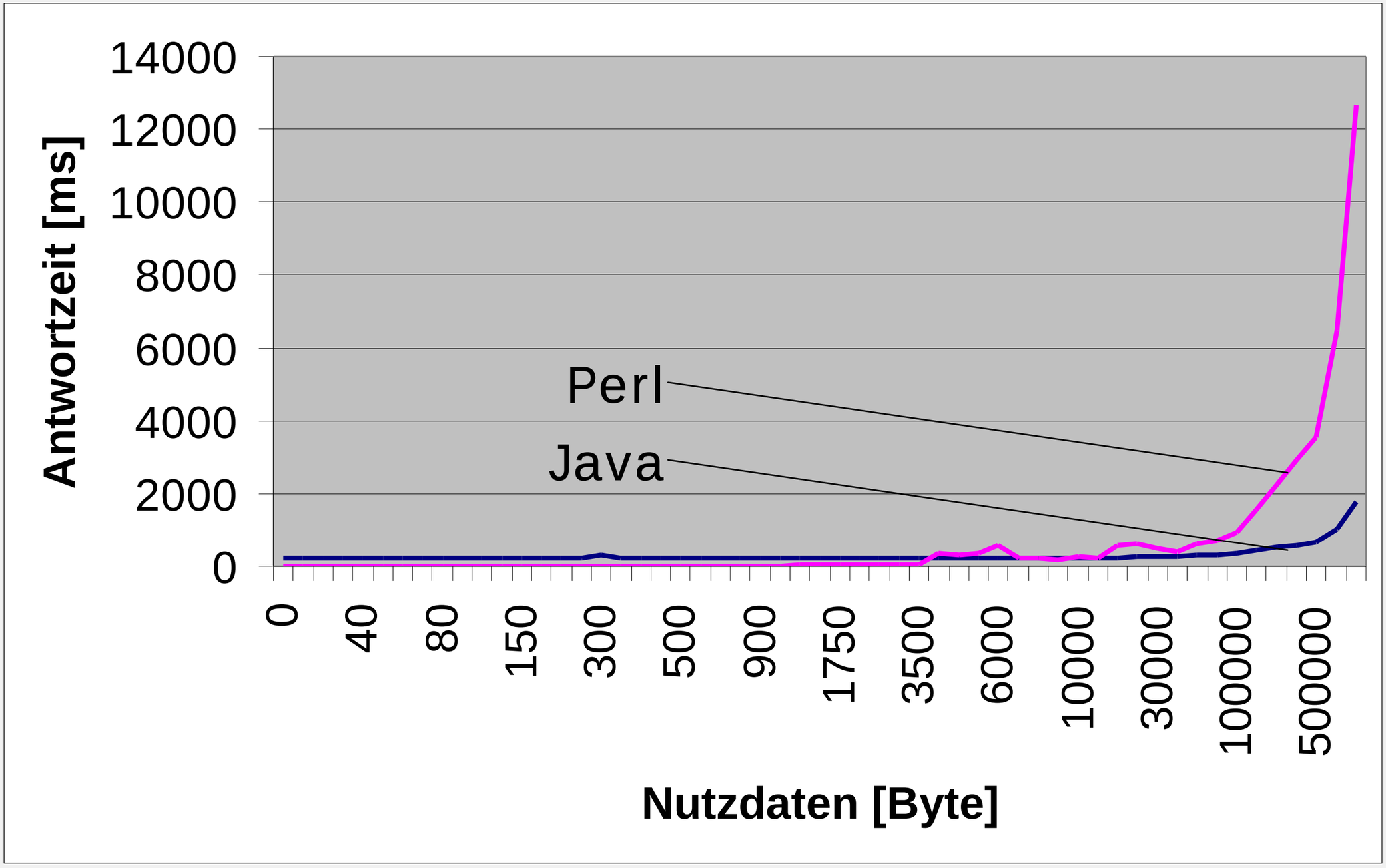
Ergebnisse
II Kommunikationszeit



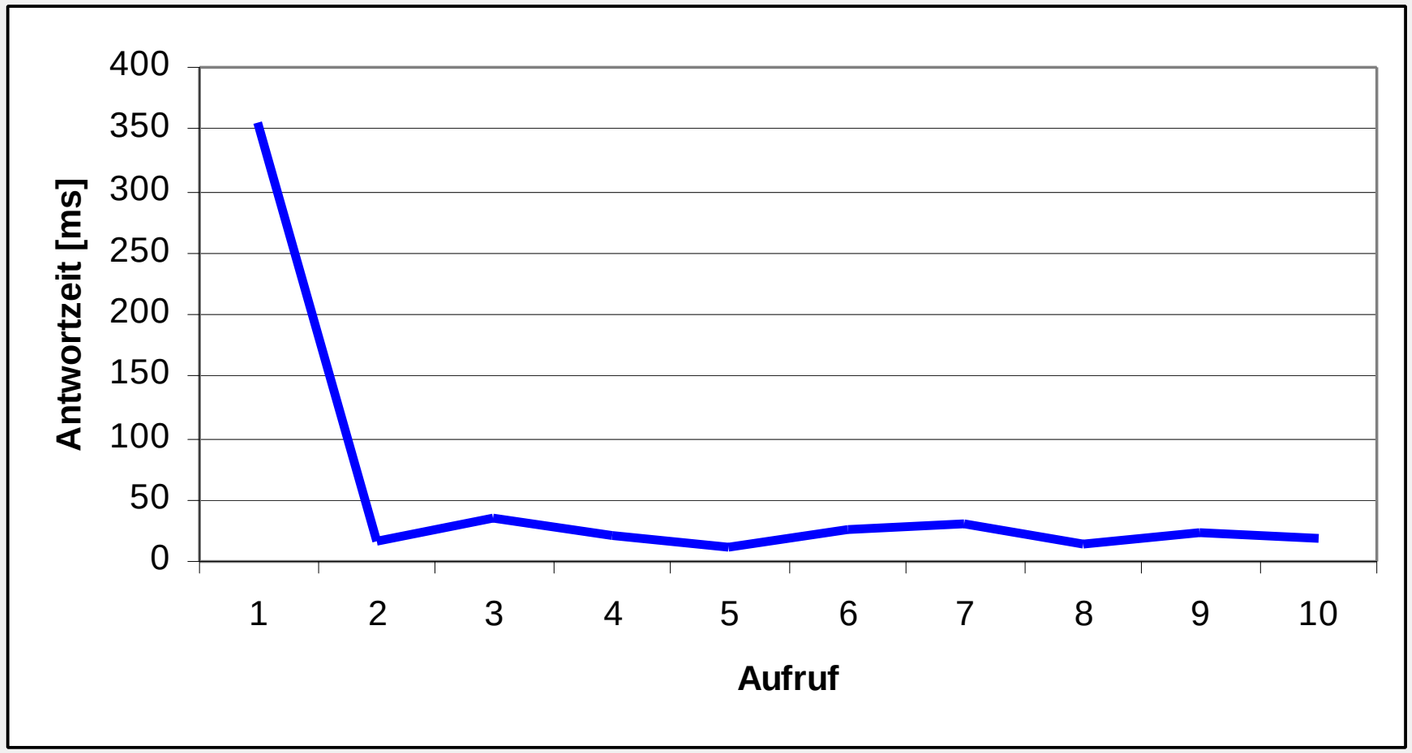
Auftretende Paketfehler („verlorene Pakete“) durch Überschreitung des Retransmission Time Outs.

Gemessen: SOAP 200.000-Byte Nutzdatengröße

Ergebnisse
II Kommunikationszeit

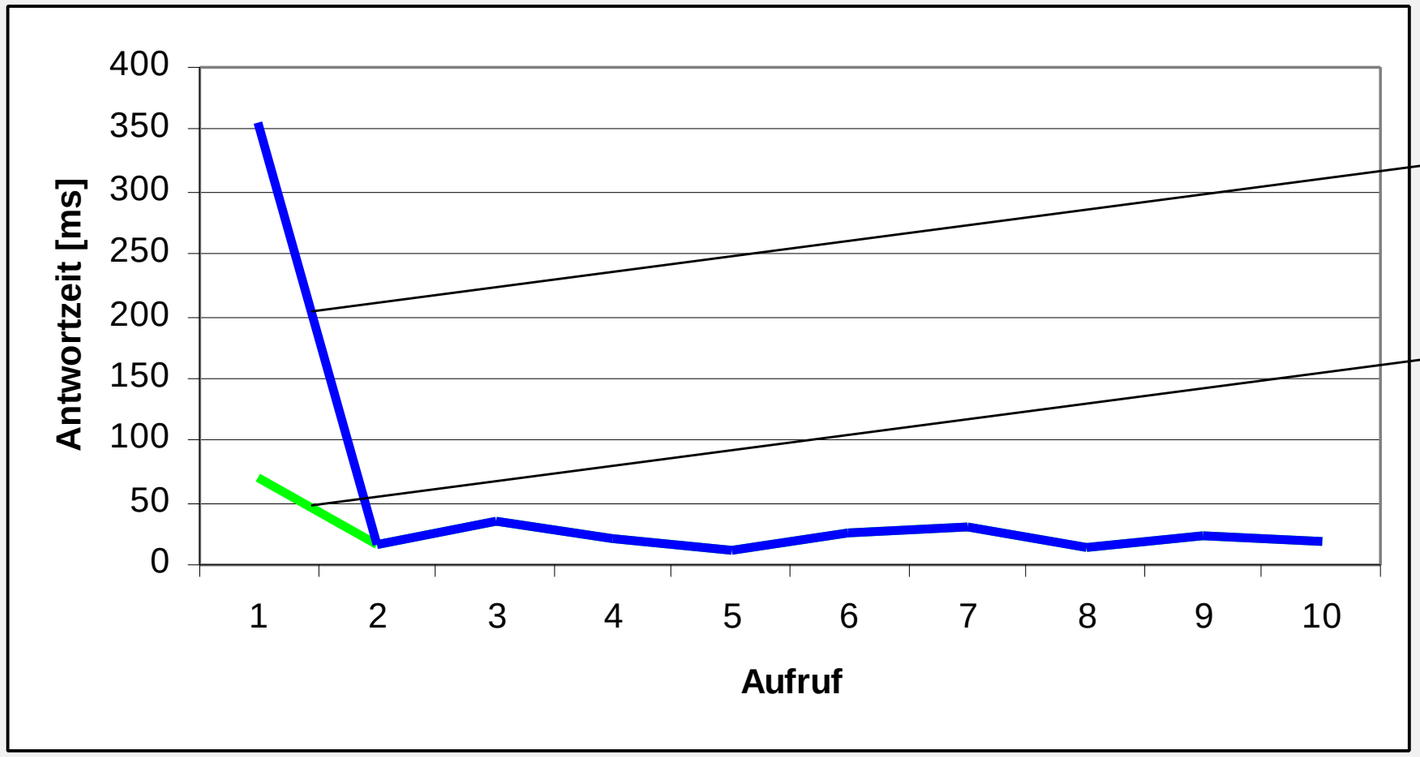


Ergebnisse
II Kommunikationszeit



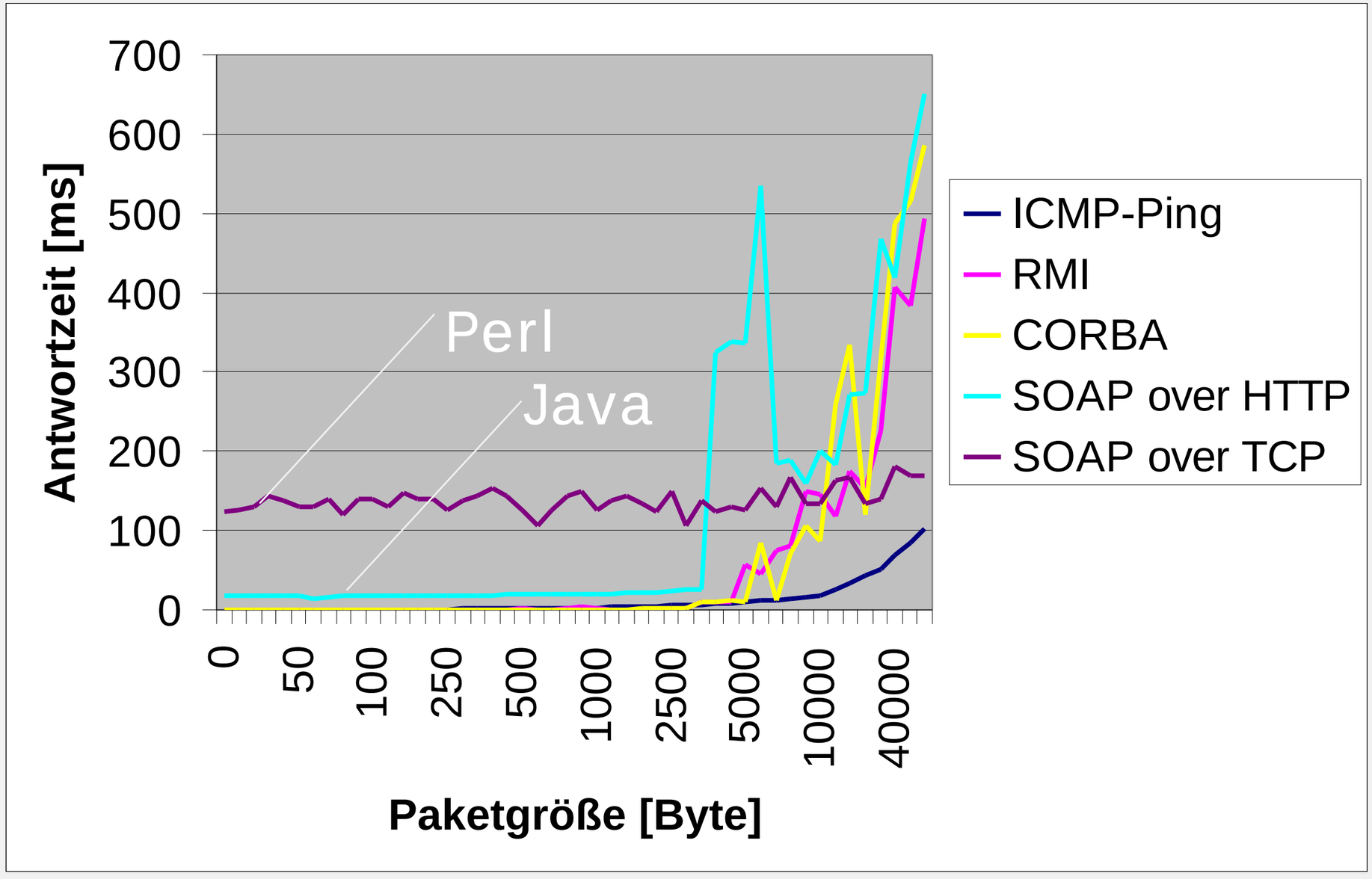
- AXIS SOAP mit nicht vorübersetztem Web Service (JWS deployment).
- Zusätzlicher Aufwand bei erstem Dienstaufruf durch serverseitige Compilierung (16-fache Antwortzeit!).
- Keine Pufferung übersetzter Daten.
  => Nach Ablauf eines Timeouts ist der nächste Aufruf wiederum der „Erste“.
  => Bei „selten“ genutzten Diensten ist jeder Aufruf der Erste.

Ergebnisse
II Kommunikationszeit



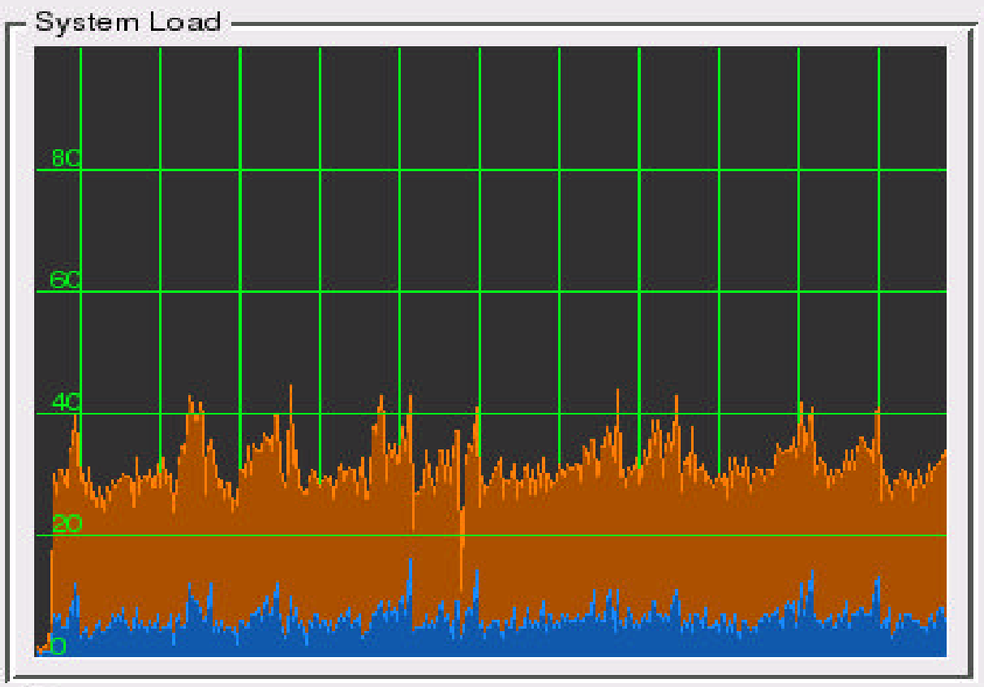
- Reduktion des initialen Aufwandes durch Ablage des vorübersetzten Dienstes auf dem Server.
- Verbleibender Restaufwand bei erstem Aufruf durch Anlage serverinterner Verwaltungsstrukturen.

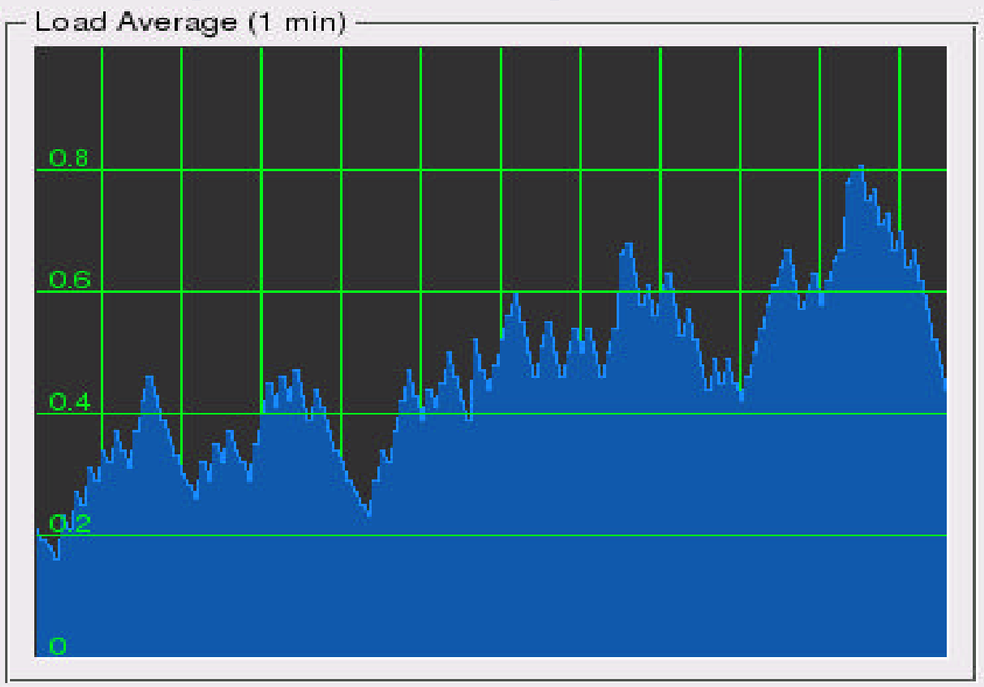
Ergebnisse
II Kommunikationszeit

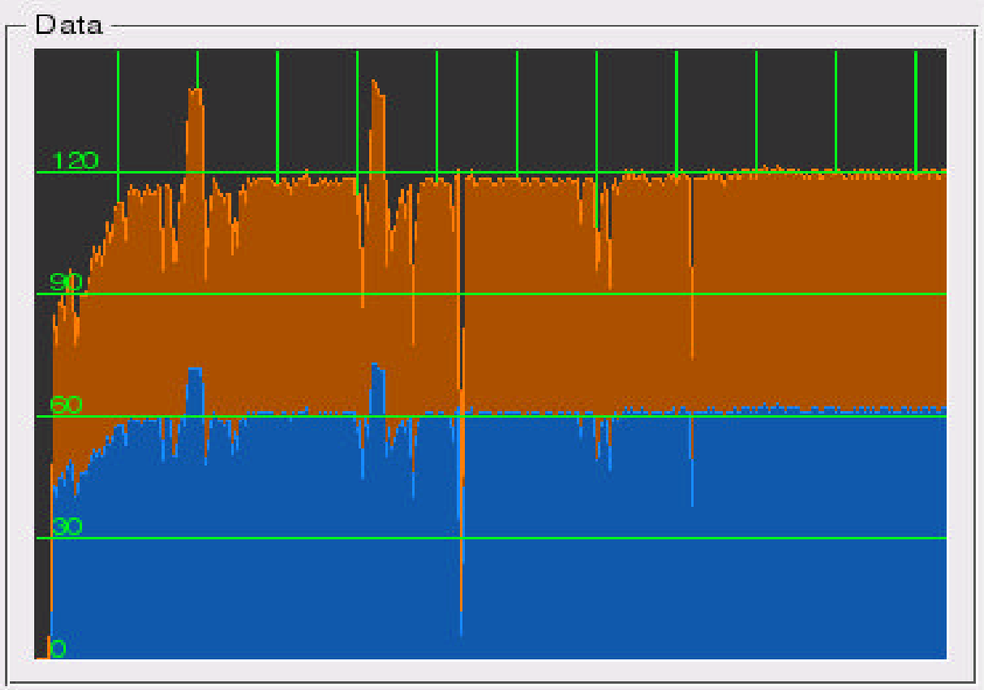


- Vergleich zwischen verschiedenen Transportprotokollen

Ergebnisse
III Lastverhalten (Server)

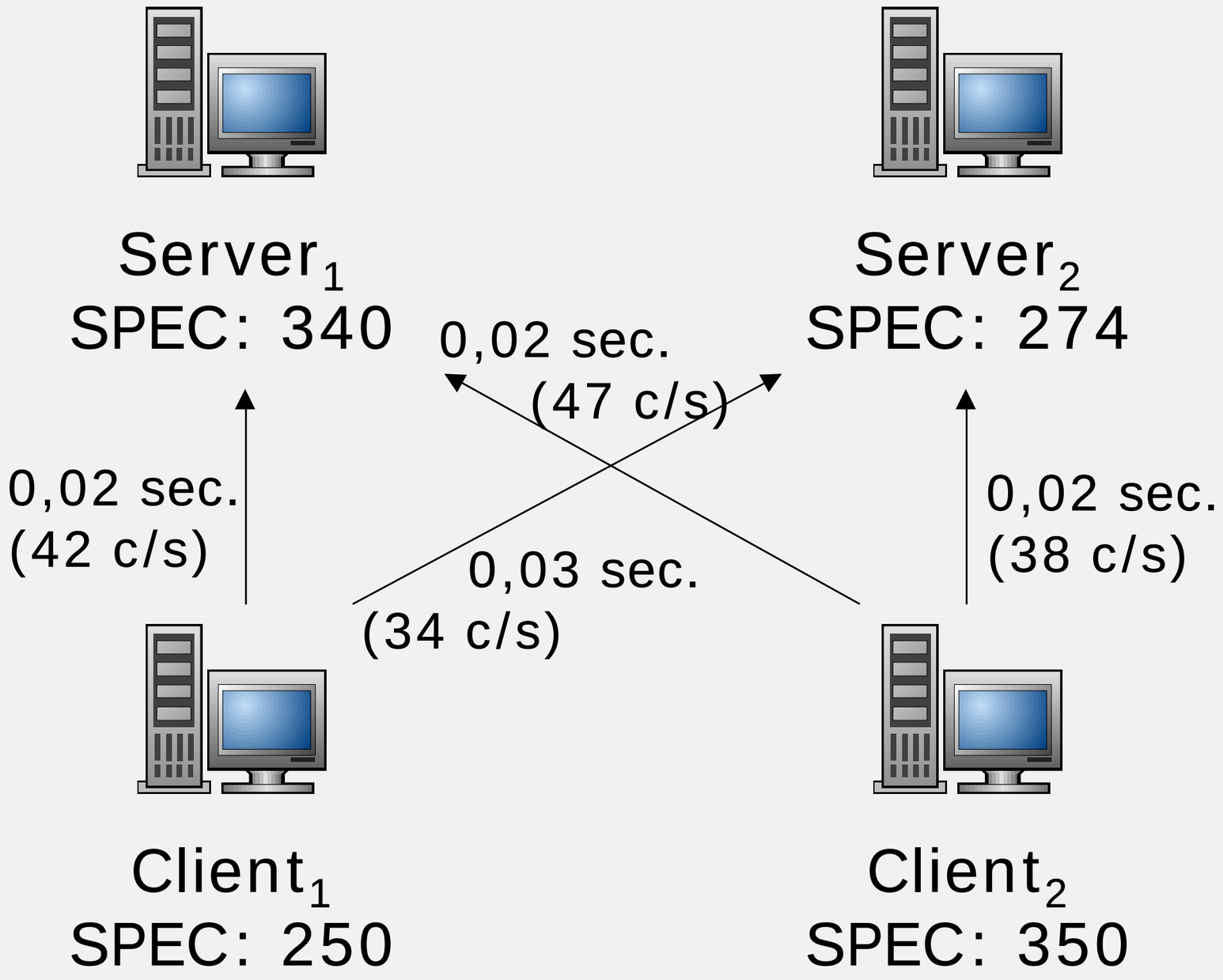




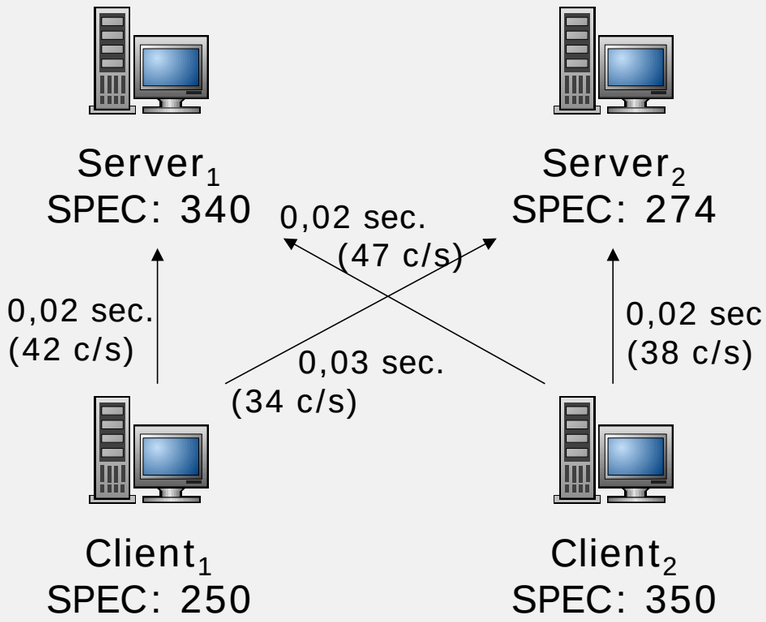


SOAP-Serverlast
bei kontinuierlichem
Zugriff
(58 Aufrufe/sec.).

Ergebnisse
III Lastverhalten



Ergebnisse
III Lastverhalten



- Gemessen: 10000 Aufrufe mit jeweils 3000Byte Nutzlast
- Geschwindigkeitsabschätzung gemäß veröffentlichter $SPEC_{2000}$-Werte

- 40% schnellerer Client erhöht mögliche Aufrufanzahl lediglich um 10%
- 20% schnellerer Server liefert 20% mehr Aufrufe

=> SPEC-Werte liefern eine zuverlässige Abschätzung der erwarteten Aufrufperformance.

Ergebnisse
III Lastverhalten

Server-Maschine	**Zu erwartende Performance** **[calls/sec. bei 3000Byte Nutztransfervolumen]**
Pentium III 750Mhz (SPEC 340)	47
Pentium IV 1.3Ghz (SPEC 561)	77
Pentium IV 2.66Ghz (SPEC 957, Aktueller „Mediamarkt-PC“)	132
Dual Intel Xeon 2.8Ghz (SPEC 926)	128
Dual AMD Opteron 1.8Ghz (SPEC 1023)	141
Intel Xeon 3.06Ghz (SPEC 1113)	155

Performanceoptimierung und -vorhersage

- Netzwerklast (beeinflußt durch Paketgröße) hat m.U. großen Einfluß auf Gesamtperformance.
- SOAP ist (teilweise deutlich) besser als erwartet.
- HTTP bildet den Flaschenhals.
- SPEC-Benchmarkwerte liefern gute Abschätzung zukünftiger Server-Performance.
- Optimierungspotential liegt überwiegend auf Seiten des Servers.
- SOAPing ist gut geeignet schnelle Vergleiche zwischen SOAP-Implementierungsalternativen anzustellen.