BETRIEBLICHE INFORMATIONSSYSTEME

Präsentation der Ergebnisse im Praktikum

Angriffe auf AMQP-Messagebroker PT03

Philipp Sieder, Marcel Mielke

6. Juli 2015

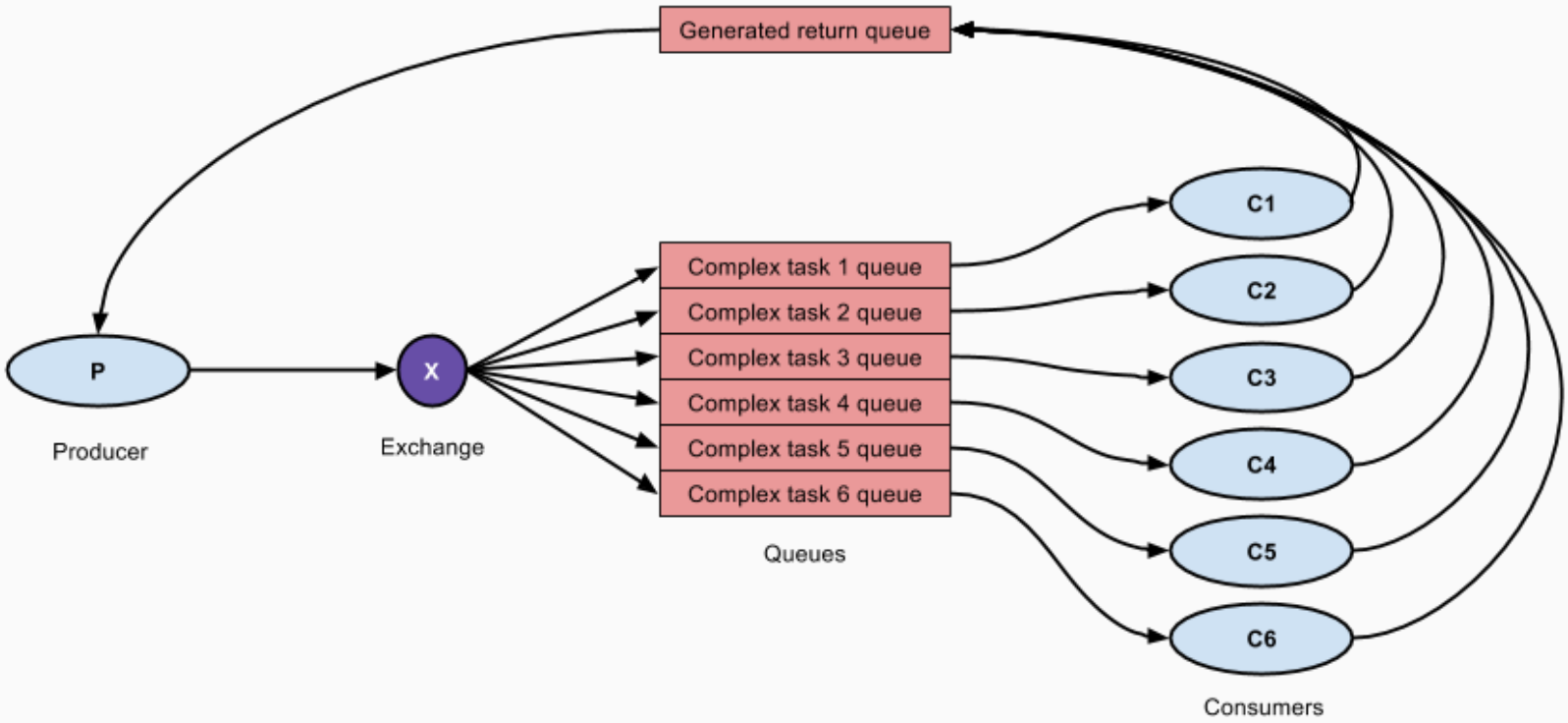
Hochschule für Technik, Wirtschaft und Kultur Leipzig / Universität Leipzig

AUFGABENSTELLUNG

ÜBERBLICK

- AMQP als Kommunikationsprotokoll für Message-orientierte Middleware (MOM)
- Demonstration von Angriffen auf Verfügbarkeit von AMQP-Brokern
  - Auf Basis der Open Source Message Broker Software „RabbitMQ“
  - Geschrieben in der Programmiersprache Erlang
- Fokus auf Denial-of-Service

BEISPIEL


Generated return queue
C1
C2
C3
C4
C5
C6
Complex task 1 queue
Complex task 2 queue
Complex task 3 queue
Complex task 4 queue
Complex task 5 queue
Complex task 6 queue
P
X
Producer
Exchange
Queues
Consumers

RAHMENBEDINGUNGEN

- Identifikation und Bewertung potenzieller Angriffsvektoren auf AMQP, RabbitMQ
- Messung von Latenz, Speicher, CPU-Auslastung, ...
- Implementierung von Beispielen auf Basis der RabbitMQ Java Client Library
- Vorschläge für Schadensbegrenzung

Identifikation von Angriffsvektoren

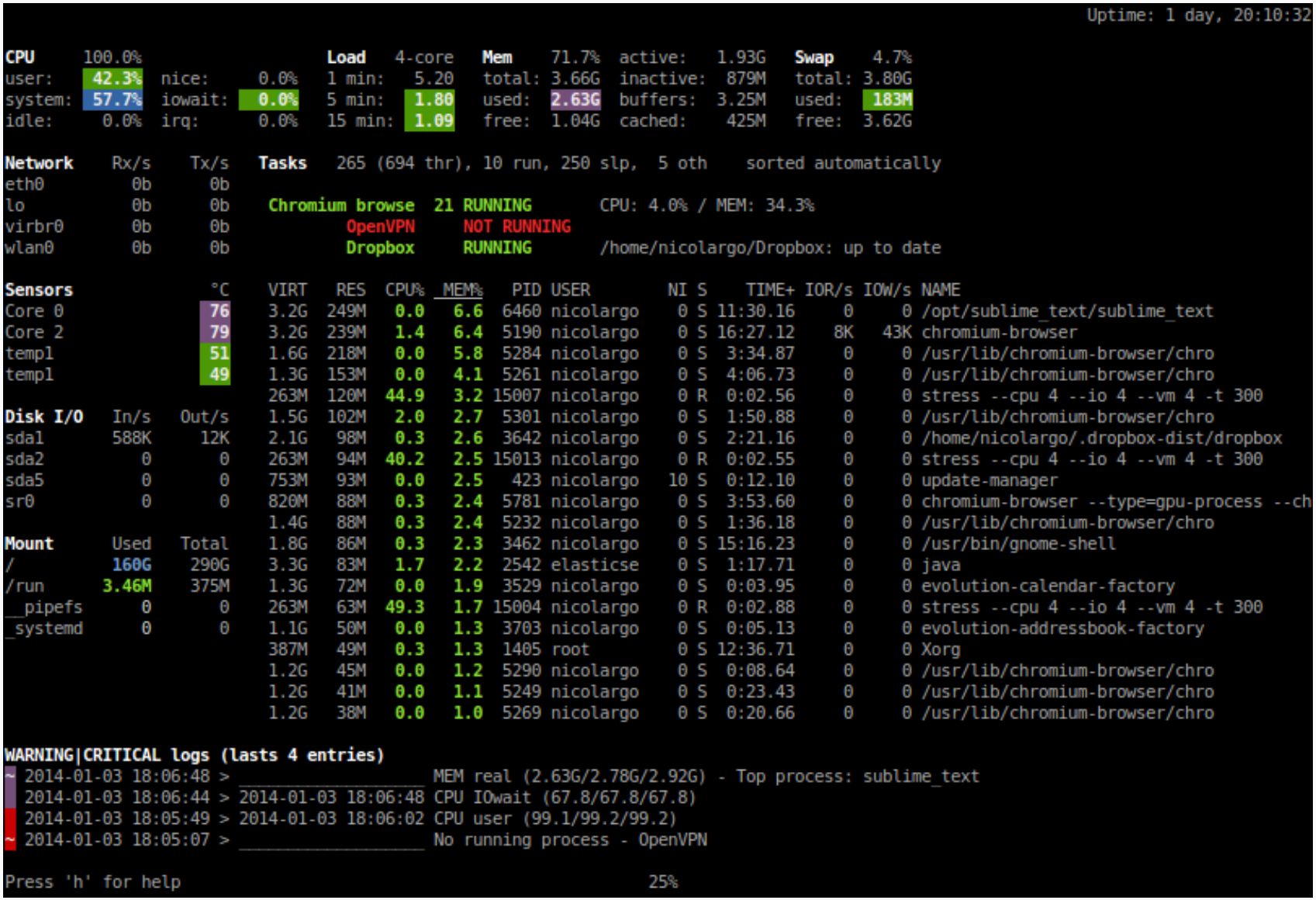
- Viele Verbindungen
- Bruteforce-Attacke auf Benutzercredentials
- Hohe Datenrate (viele kleine, wenig große Nachrichten)
- Große Header, kleiner Payload
- Langsamer Verbindungsaufbau
- Unvollständiger Verbindungsaufbau
- Pause im Protokollablauf

WERKZEUGE

VIRTUALBOX

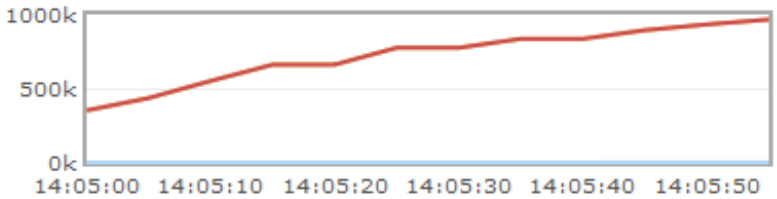
- Konfiguration der virtuellen Hardware
- Erstellung von Sicherungspunkten
- Portabilität der Testumgebung
- Als Grundlage dient die Servervariante von Ubuntu 14.04.2 LTS

GLANCES


Uptime: 1 day, 20:10:32
CPU 100.0%
user: 42.3% nice: 0.0%
system: 57.7% iowait: 0.0%
idle: 0.0% irq: 0.0%
Load 4-core
1 min: 5.20
5 min: 1.80
15 min: 1.09
Mem 71.7% active: 1.93G
total: 3.66G inactive: 879M
used: 2.63G buffers: 3.25M
free: 1.04G cached: 425M
Swap 4.7%
total: 3.80G
used: 183M
free: 3.62G
Network Rx/s Tx/s
eth0 0b 0b
lo 0b 0b
virbr0 0b 0b
wlan0 0b 0b
Tasks 265 (694 thr), 10 run, 250 slp, 5 oth sorted automatically
Chromium browse 21 RUNNING CPU: 4.0% / MEM: 34.3%
OpenVPN NOT RUNNING
Dropbox RUNNING /home/nicolargo/Dropbox: up to date
Sensors °C
Core 0 76
Core 2 79
temp1 51
temp1 49
Disk I/O In/s Out/s
sda1 588K 12K
sda2 0 0
sda5 0 0
sr0 0 0
Mount Used Total
/ 160G 290G
/run 3.46M 375M
__pipefs 0 0
_systemd 0 0
VIRT RES CPU% MEM% PID USER NI S TIME+ IOR/s IOW/s NAME
3.2G 249M 0.0 6.6 6460 nicolargo 0 S 11:30.16 0 0 /opt/sublime_text/sublime_text
3.2G 239M 1.4 6.4 5190 nicolargo 0 S 16:27.12 8K 43K chromium-browser
1.6G 218M 0.0 5.8 5284 nicolargo 0 S 3:34.87 0 0 /usr/lib/chromium-browser/chro
1.3G 153M 0.0 4.1 5261 nicolargo 0 S 4:06.73 0 0 /usr/lib/chromium-browser/chro
263M 120M 44.9 3.2 15007 nicolargo 0 R 0:02.56 0 0 stress --cpu 4 --io 4 --vm 4 -t 300
1.5G 102M 2.0 2.7 5301 nicolargo 0 S 1:50.88 0 0 /usr/lib/chromium-browser/chro
2.1G 98M 0.3 2.6 3642 nicolargo 0 S 2:21.16 0 0 /home/nicolargo/.dropbox-dist/dropbox
263M 94M 40.2 2.5 15013 nicolargo 0 R 0:02.55 0 0 stress --cpu 4 --io 4 --vm 4 -t 300
753M 93M 0.0 2.5 423 nicolargo 10 S 0:12.10 0 0 update-manager
820M 88M 0.3 2.4 5781 nicolargo 0 S 3:53.60 0 0 chromium-browser --type=gpu-process --ch
1.4G 88M 0.3 2.4 5232 nicolargo 0 S 1:36.18 0 0 /usr/lib/chromium-browser/chro
1.8G 86M 0.3 2.3 3462 nicolargo 0 S 15:16.23 0 0 /usr/bin/gnome-shell
3.3G 83M 1.7 2.2 2542 elasticse 0 S 1:17.71 0 0 java
1.3G 72M 0.0 1.9 3529 nicolargo 0 S 0:03.95 0 0 evolution-calendar-factory
263M 63M 49.3 1.7 15004 nicolargo 0 R 0:02.88 0 0 stress --cpu 4 --io 4 --vm 4 -t 300
1.1G 50M 0.0 1.3 3703 nicolargo 0 S 0:05.13 0 0 evolution-addressbook-factory
387M 49M 0.3 1.3 1405 root 0 S 12:36.71 0 0 Xorg
1.2G 45M 0.0 1.2 5290 nicolargo 0 S 0:08.64 0 0 /usr/lib/chromium-browser/chro
1.2G 41M 0.0 1.1 5249 nicolargo 0 S 0:23.43 0 0 /usr/lib/chromium-browser/chro
1.2G 38M 0.0 1.0 5269 nicolargo 0 S 0:20.66 0 0 /usr/lib/chromium-browser/chro
WARNING|CRITICAL logs (lasts 4 entries)
2014-01-03 18:06:48 > MEM real (2.63G/2.78G/2.92G) - Top process: sublime_text
2014-01-03 18:06:44 > 2014-01-03 18:06:48 CPU IOwait (67.8/67.8/67.8)
2014-01-03 18:05:49 > 2014-01-03 18:06:02 CPU user (99.1/99.2/99.2)
2014-01-03 18:05:07 > No running process - OpenVPN
Press 'h' for help
25%

RABBITMQ MANAGEMENT TOOL (I)

Queued messages (chart: last minute) (?)



Ready	1,007,369
Unacked	397
Total	1,007,766

Message rates (chart: last minute) (?)

Publish	12,158/s
Deliver	2,628/s
Acknowledge	2,620/s

Global counts (?)

Connections: 7 | Channels: 7 | Exchanges: 16 | Queues: 1 | Consumers: 2

Node

Node: rabbit@SW01 **(More about this node)**

File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Info	+/-
29 1024 available	8 829 available	255 1048576 available	307MB 801MB high watermark	3.2GB 48MB low watermark	Disc 1 Stats	

RABBITMQ MANAGEMENT TOOL (II)

HTML PERFORMANCE TOOLS

- Definition von Anwendungsszenarien
- Verwendung von **PerfTest** (Bestandteil der RabbitMQ-Client-Bibliothek)
- automatisierte Ausführung
- Alle Ergebnisse werden in einer JSON-Datei gesichert und über HTML, JS, CSS visualisiert
- `https://github.com/rabbitmq/rabbitmq-perf-html`

AMQPSTRESS

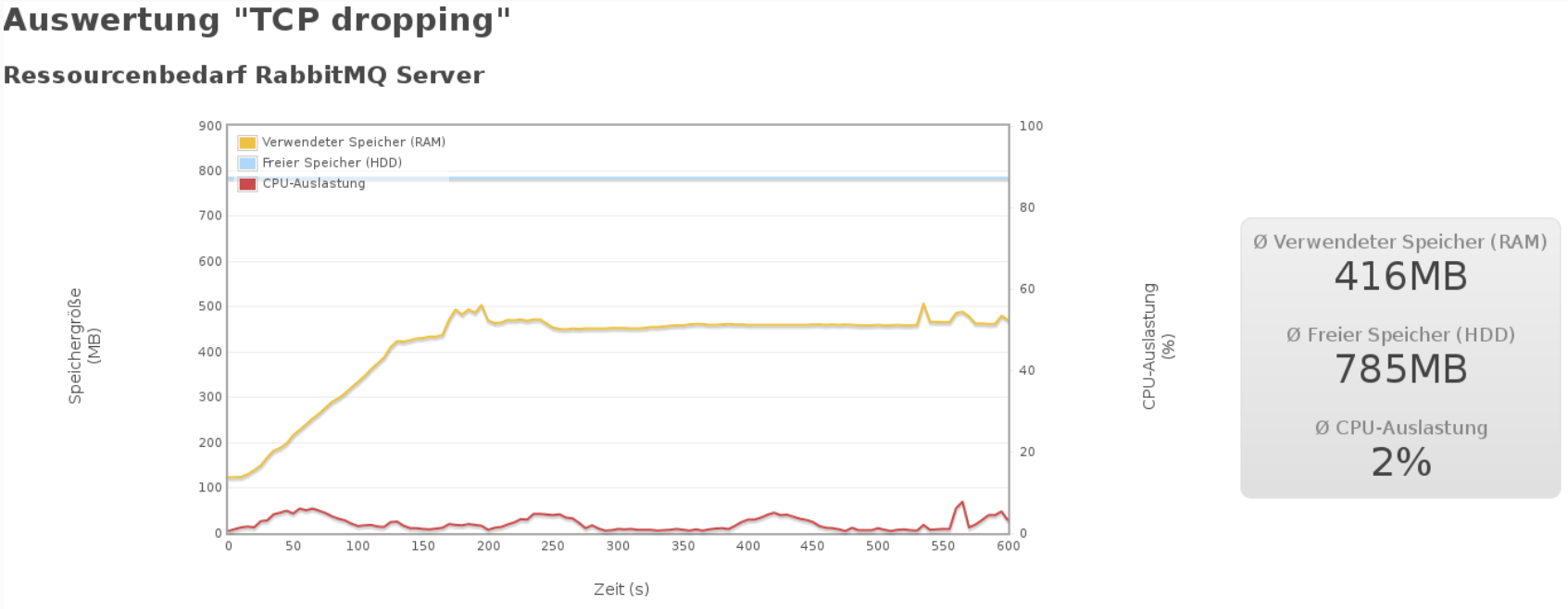
- Implementierung von Clients für mögliche Angriffsvektor
- Zahlreiche Einstellungsmöglichkeiten
- Projekt und Dokumentation zugänglich über GitHub

https://github.com/philippsied/amqp-stress-test

AMQPSTRESS - ARGUMENTE

```
usage: amqptest [-c <count>] [-cl <count>] [-co <commit>] -dc | -dm <responsetyp> | -dq <queueAction> |
                -hb | -lh | -mc | -sh | -tc | -ts | -tx   [-h]  [-hs <size>] [-i <milliseconds>]
                [-mct <count>] [-mp] [-ms <size in bytes>] [-p <count>] [-pc <count>] -u <uri>
Options
 -c,--consumer <count>           Set count of parallel running consumers
 -cl,--clients <count>           Set count of parallel running clients
 -co,--commit_messages <commit>  Set for committing messages - true/false
 -dc,--dropcon                   open connection and immediately close the Socket (Without keep-alive,
                                 max. heartbeat) and send TCP RST
 -dm,--dosmsg <responsetyp>      DoS with messages, type is one of "ACK","NO","NACK","REJECT"
 -dq,--dosqueue <queueAction>    DoS with queues, type is one of "NO","MSG"
 -hb,--heartbeat                 use small heartbeats to stress server, use -i to set heartbeat timeout,
                                 -cl to set amount of sending clients
 -hs,--headersize <size>         Set the size of the Headerfield - Number of entrys
 -i,--minterval <milliseconds>   Set interval for a simple action, interpreted as milliseconds
 -lh,--largeheader               Send messages with large header
 -mc,--manych                    Send messages over one Connection and many Channels; set over -p and -c
 -mct,--messagecount <count>     Set the number of messages
 -mp,--persistent                Set messages/queues persistent
 -ms,--msize <size in bytes>     Set the size of each message
 -p,--producer <count>           Set count of parallel running producers
 -pc,--pendingcount <count>      Set count of cached elements, i.e. the count of message to NACK all at once
 -sh,--slowhand                  Slow down the connection handshake
 -tx,--txmode                    Used the transaction-mode
```

- Aggregation der Informationen und Visualisierung aller Messergebnisse
- Verwendet Web-Technologien
- basiert auf der Visualisierung der HTML Performance Tools



ANGRIFFE

Angriff - Channel-Flooding

Aktion: 1. Client öffnet viele Channel auf *einer* Verbindung
2. Client sendet Nachrichten über Channel

Ziel: CPU, RAM, Netzwerkbandbreite, Festplatte

Ansatz: Server muss für jeden Channel Ressourcen allokieren und die Kommunikation aufrechterhalten. Die Channel teilen sich dabei eine TCP-Verbindung.

ANGRIFF - CHANNEL-FLOODING

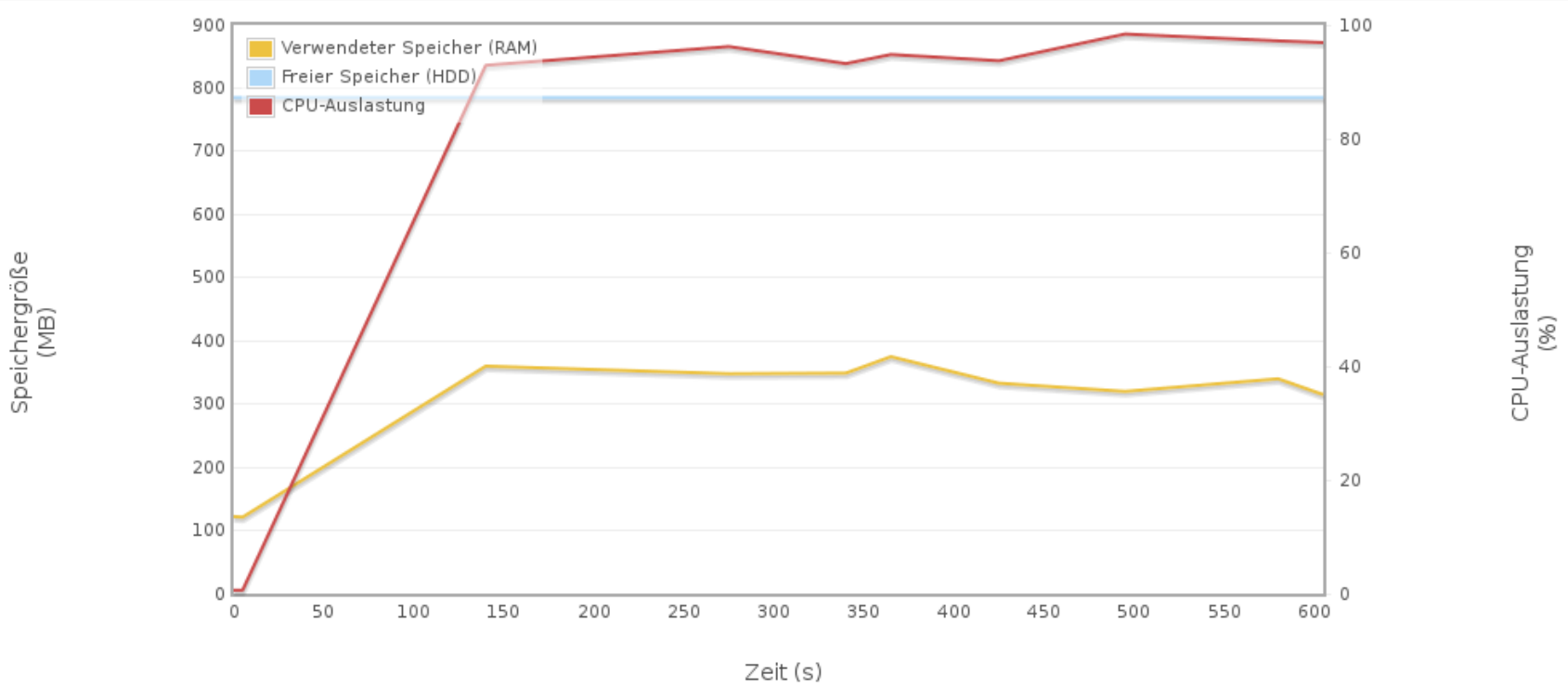


Abbildung 1: Channel-Flooding - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

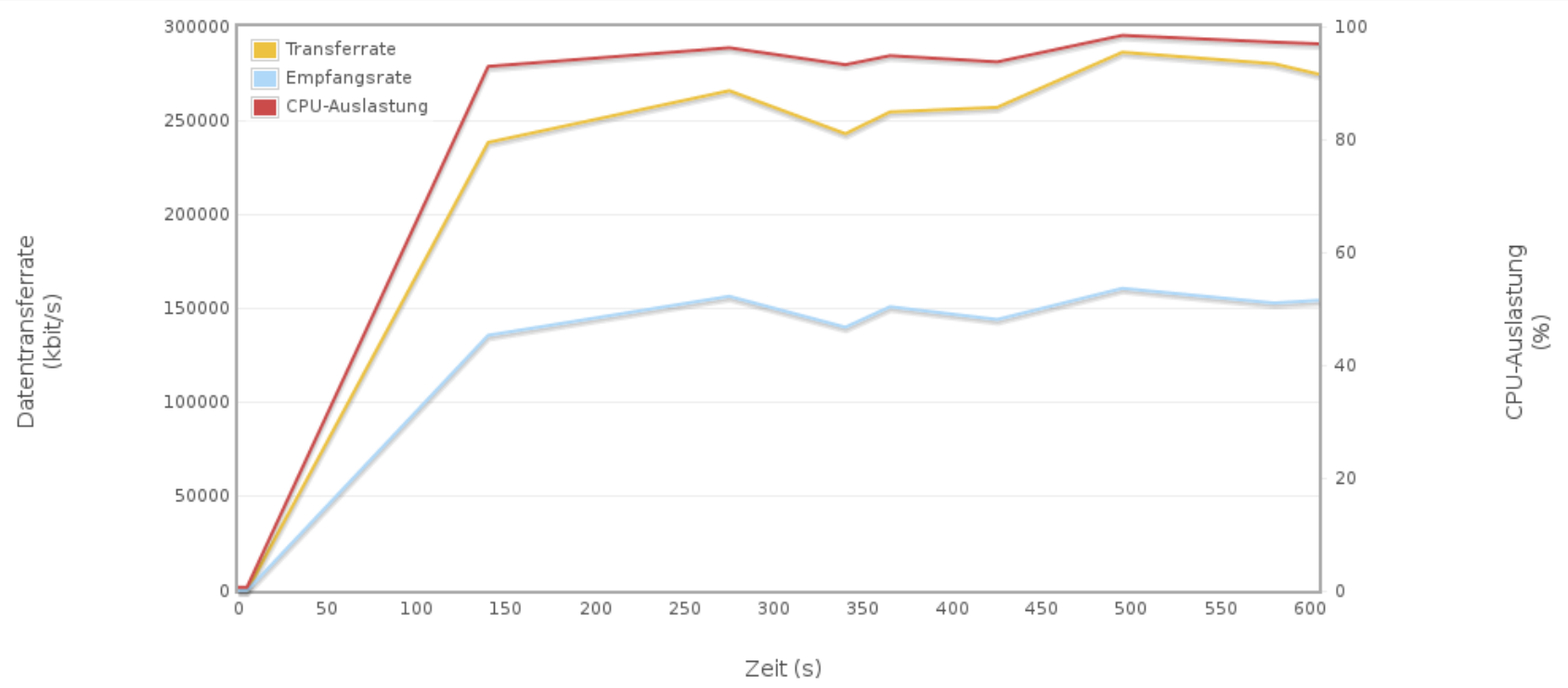


Abbildung 2: Channel-Flooding - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

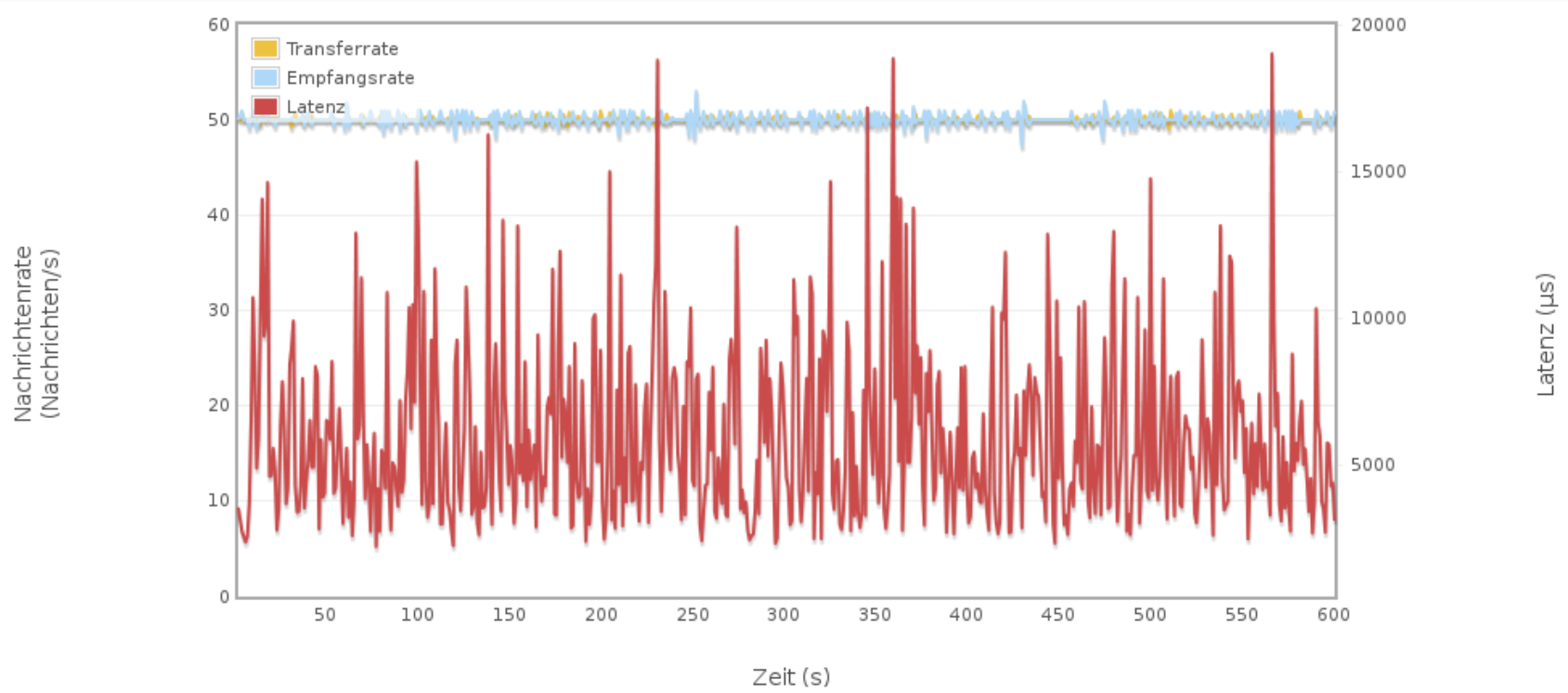


Abbildung 3: Channel-Flooding - Verlauf der Transfer-, Empfangsrate und Verlauf der Latenz im Anwendungsszenario

Übertragungsrate (schreiben)	Producer	Consumer	Nachrichtengröße (Byte)
7.000m/s	100	10	100
800m/s	100	10	1.000
100m/s	100	10	10.000

Tabelle 1: Mehrere Channel

Übertragungsrate (schreiben)	Producer	Consumer	Nachrichtengröße (Byte)
24.000m/s	100	10	100
2.000m/s	100	10	1.000
100m/s	100	10	10.000

Tabelle 2: Mehrere Connections

Angriff - Versenden von Nachrichten mit großem Header

Aktion:
1. Client erzeugt Nachrichten mit sehr großem Header
2. Header enthält Weiterleitungsoption zu ungültigem Ziel
3. Client sendet Nachricht an Server

Ziel: CPU, RAM, Festplatte

Ansatz: Server muss alle Headerinformationen prüfen, auch ungültige Ziele von Weiterleitungsoptionen.

ANGRIFF - VERSENDEN VON NACHRICHTEN MIT GROßEM HEADER

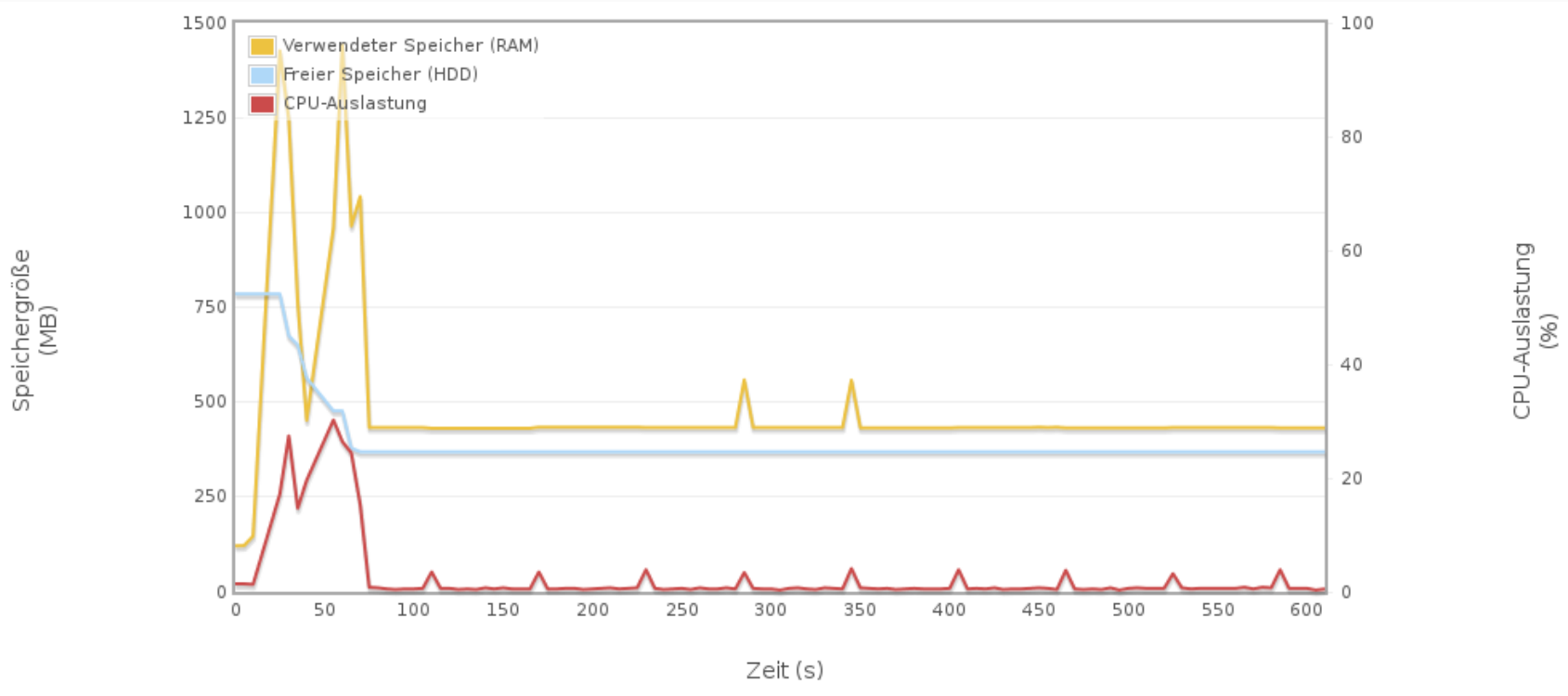


Abbildung 4: Nachrichten mit großem Header - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - VERSENDEN VON NACHRICHTEN MIT GROßEM HEADER

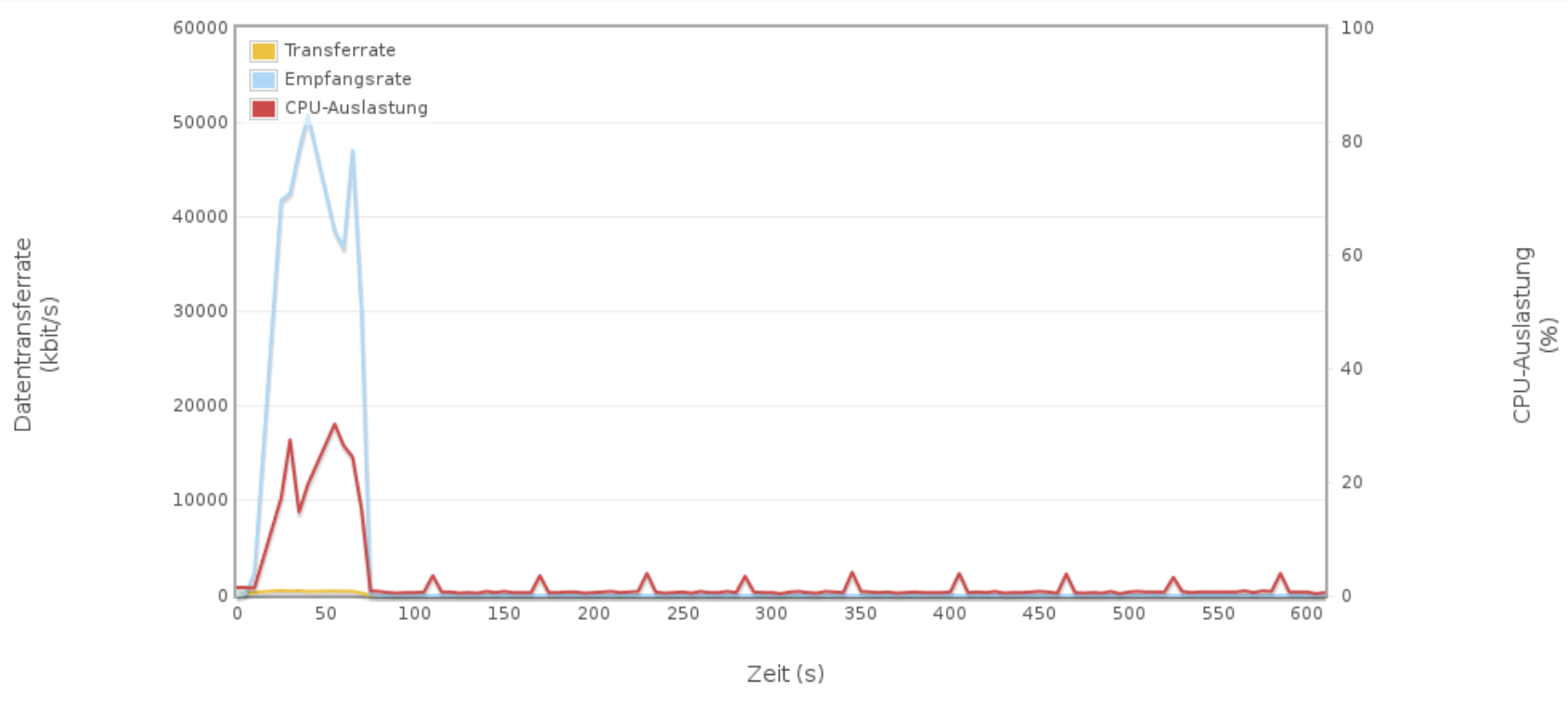


Abbildung 5: Nachrichten mit großem Header - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

Angriff - Versenden von Nachrichten mit großem Header

Keine Messung möglich

Anwendung zum Testen des Szenarios konnte nicht beendet werden - Keine Messergebnisse

Angriff - Versenden von Nachrichten mit großem Header

Übertragungsrate nach Nachrichtengröße			Headergröße
10.000 Byte	1.000 Byte	100 Byte	Einträge (Byte)
140m/s	260m/s	350m/s	200 (8.800)
80m/s	120m/s	180m/s	500 (22.000)
30m/s	50m/s	70m/s	1.000 (44.000)
20m/s	30m/s	40m/s	2.000 (88.000)
10m/s	10m/s	20m/s	2.500 (110.000)
270m/s	3000m/s	10000m/s	Kein Header

ANGRIFF - TRANSAKTIONSMODUS

Aktion:
1. Clients starten Transaktionen
2. Clients senden Nachrichten an Server
3. **Keiner** der Clients führt einen *commit* aus

Ziel: CPU, RAM

Ansatz: Server muss die Daten von Transaktionen zwischenspeichern bis ein *commit* oder ein *rollback* stattgefunden hat.

ANGRIFF - TRANSAKTIONSMODUS

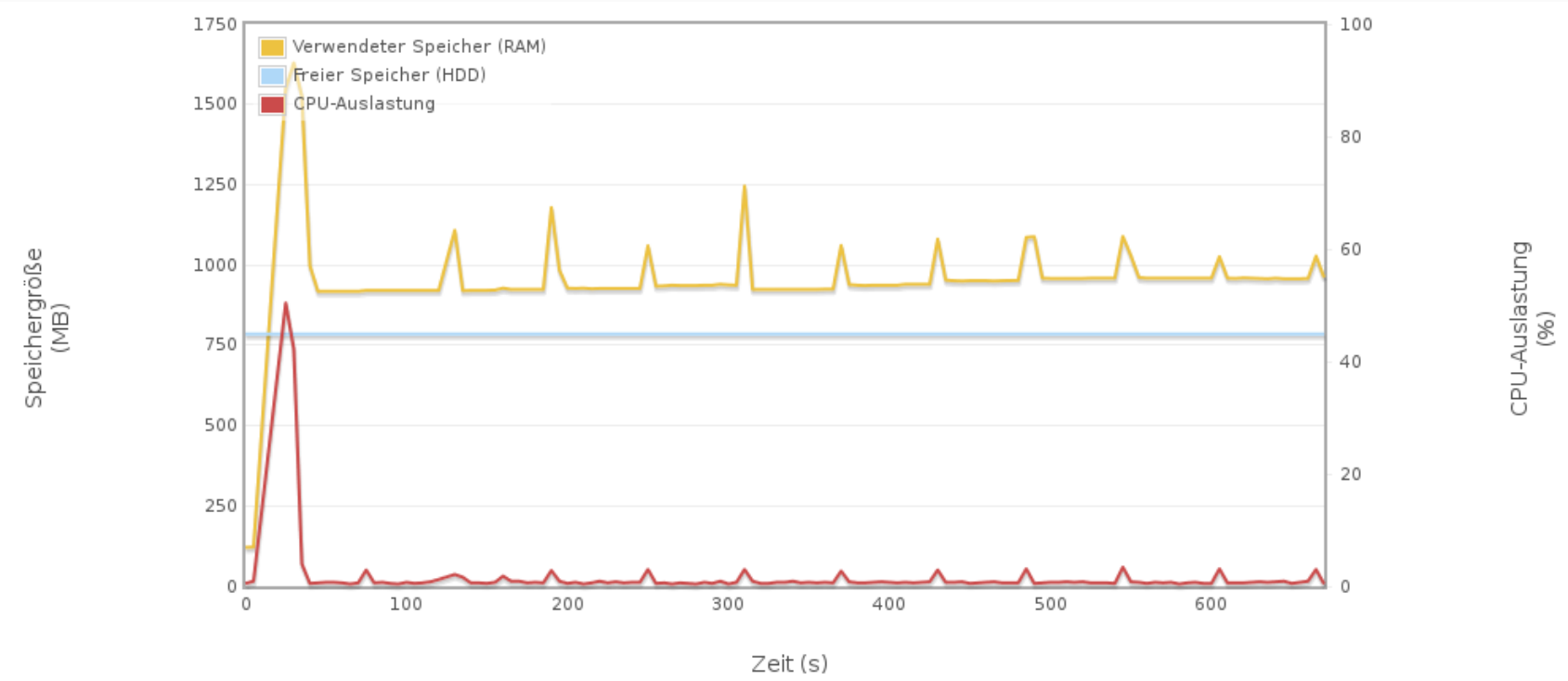


Abbildung 6: Ausbleiben von Commits im Transaktionsmodus - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - TRANSAKTIONSMODUS

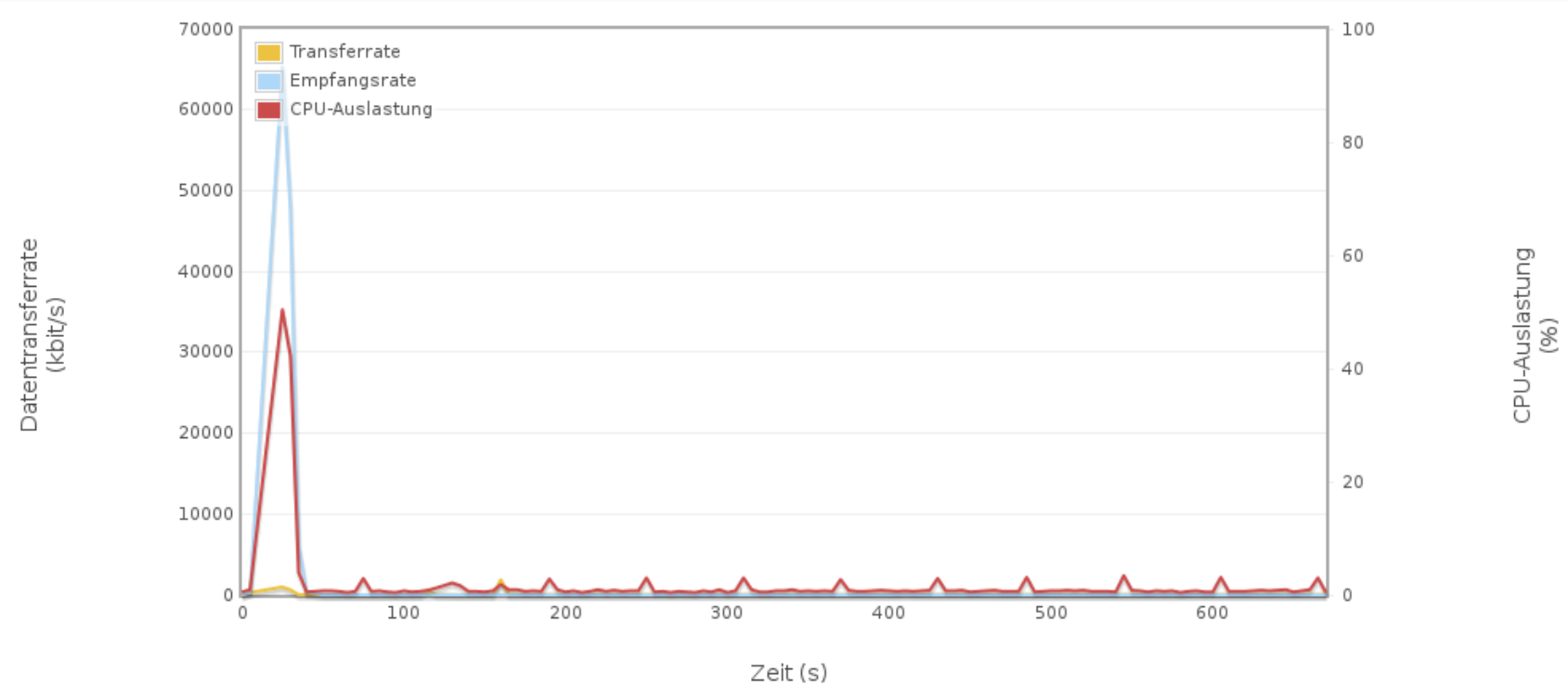


Abbildung 7: Ausbleiben von Commits im Transaktionsmodus - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

Keine Messung möglich

Anwendung zum Testen des Szenarios konnte nicht beendet werden
- Keine Messergebnisse

Angriff - Ignorieren von Nachrichten

Aktion:
1. Producer sendet Nachrichten an Server
2. Mehrere Consumer erhalten Nachrichten
3. Consumer ignorieren die Quittierung

Ziel: RAM, Festplatte

Ansatz: Server muss alle Nachrichten zwischenspeichern bis der Erhalt quittiert wurde.

ANGRIFF - IGNORIEREN VON NACHRICHTEN

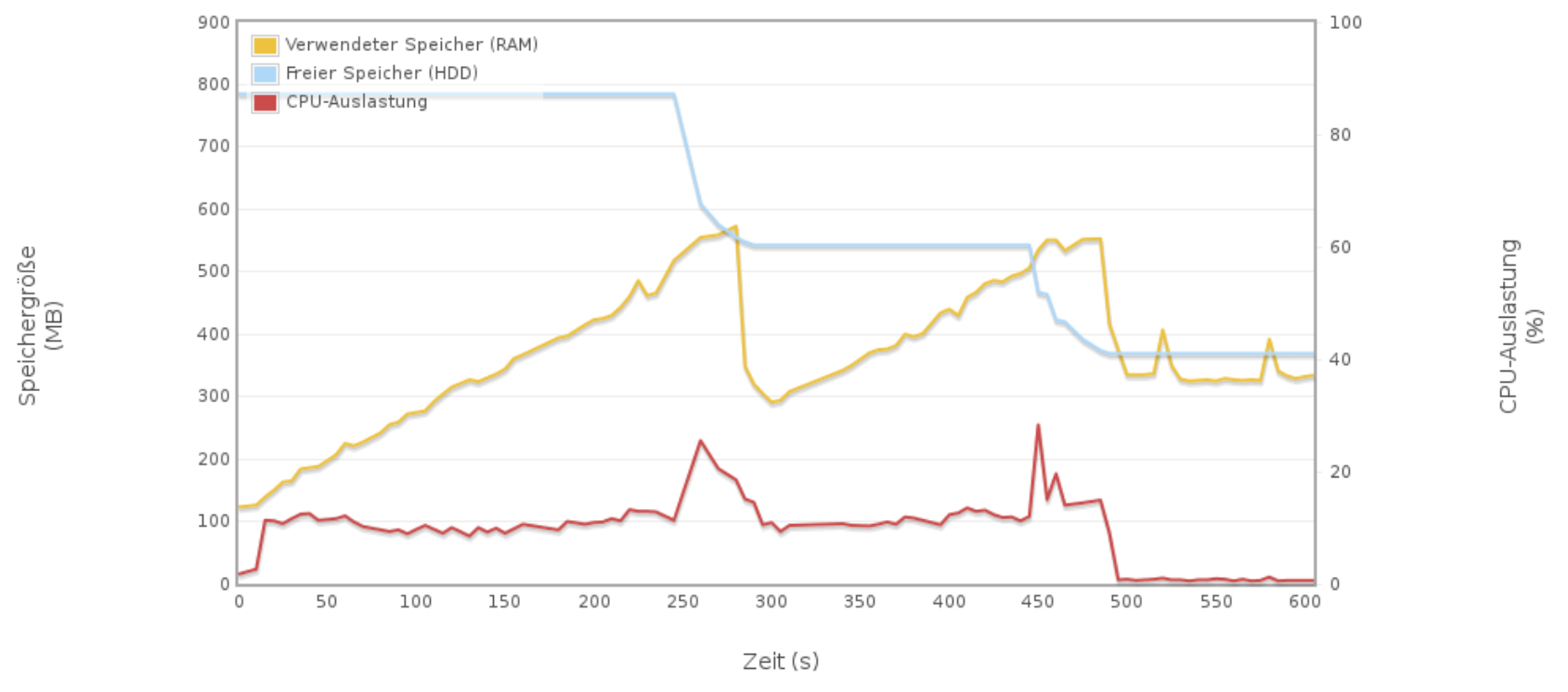


Abbildung 8: Ignorieren von Nachrichten - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - IGNORIEREN VON NACHRICHTEN

Abbildung 9: Ignorieren von Nachrichten - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - IGNORIEREN VON NACHRICHTEN

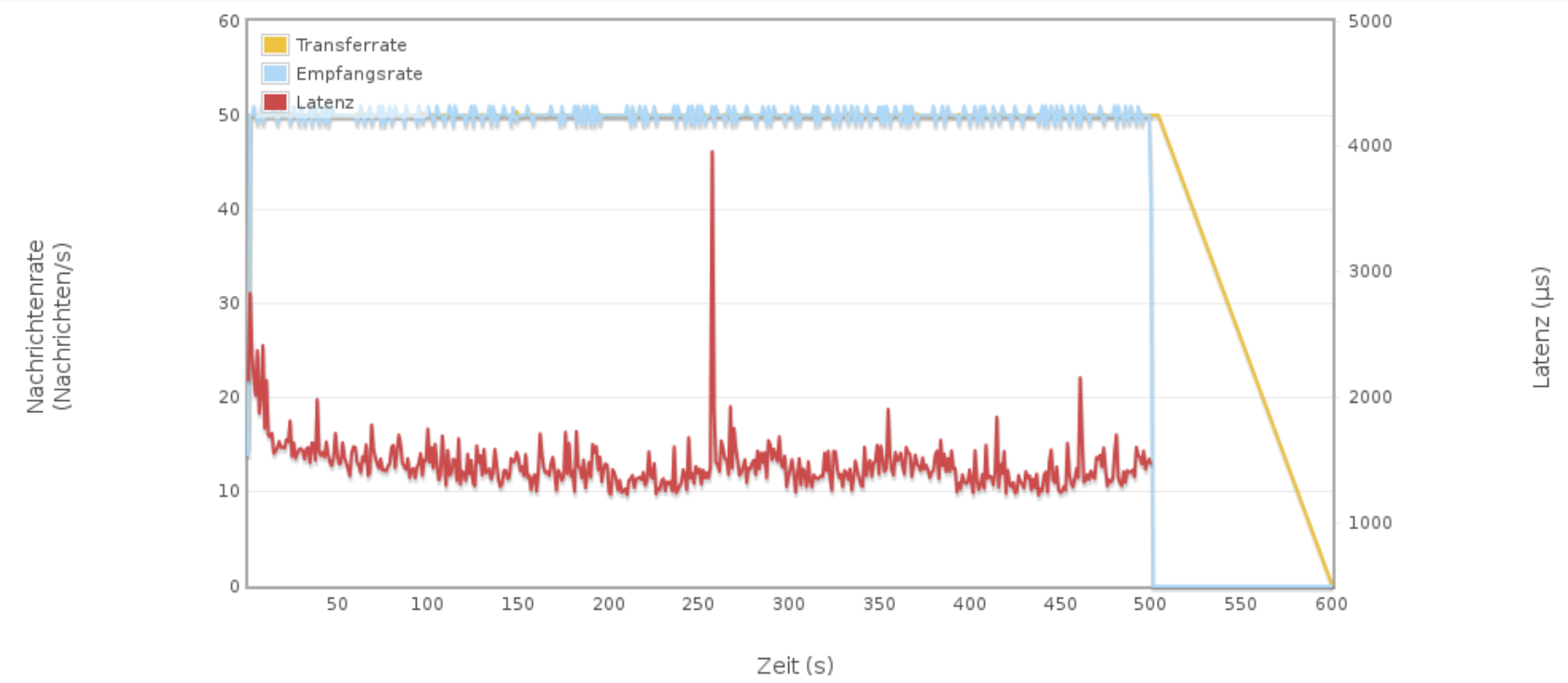


Abbildung 10: Ignorieren von Nachrichten - Verlauf der Transfer-, Empfangsrate und Verlauf der Latenz im Anwendungsszenario

Angriff - sofortiges abweisen von Nachrichten

Aktion:
1. Producer sendet Nachrichten an Server
2. Mehrere Consumer erhalten Nachrichten
3. Consumer senden jedoch sofort eine *Negative Bestätigung*

Ziel: RAM, Festplatte, Netzwerkbandbreite

Ansatz: Server muss alle Nachrichten zwischenspeichern bis der Erhalt *erfolgreich* quittiert wurde und muss abgewiesene Nachrichten erneut senden.

ANGRIFF - SOFORTIGES ABWEISEN VON NACHRICHTEN

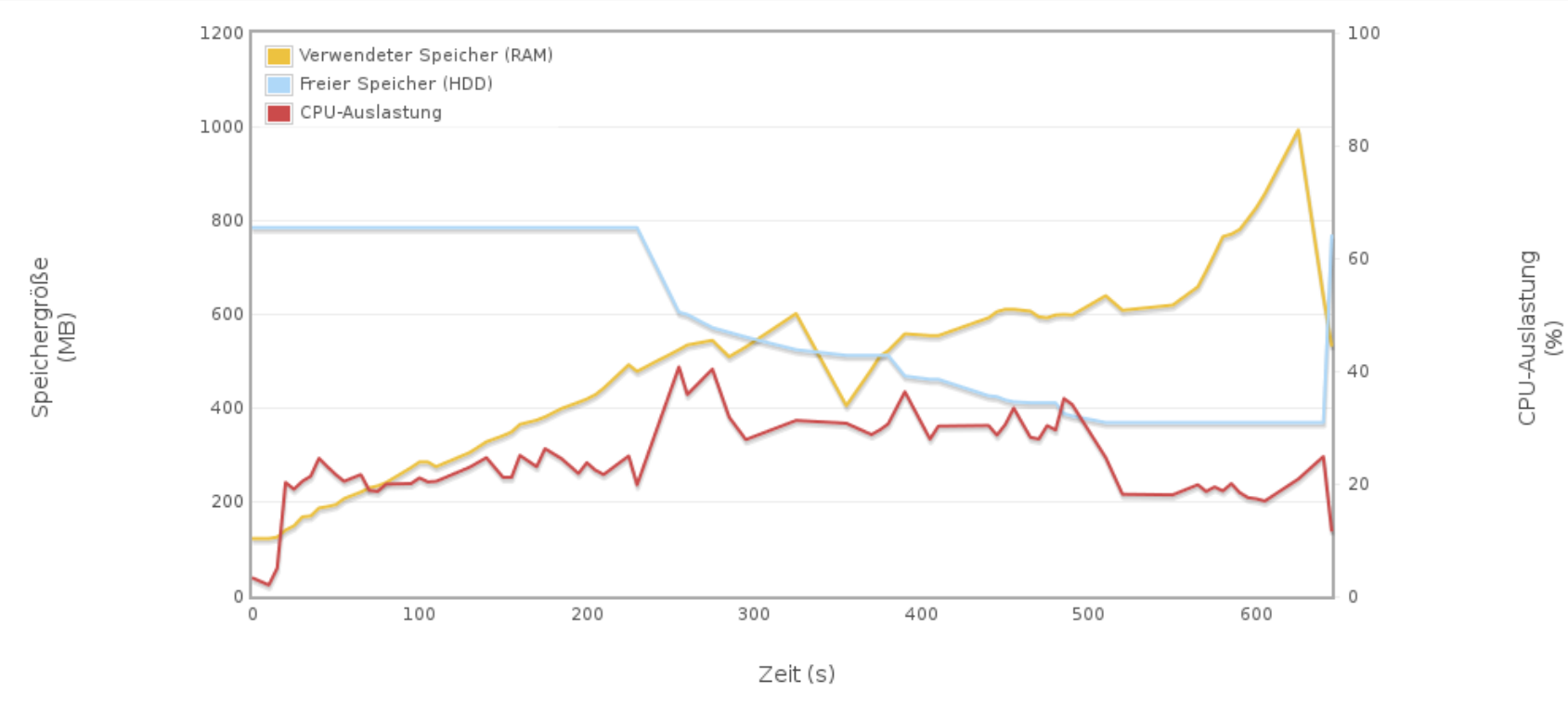


Abbildung 11: Sofortiges Abweisen von Nachrichten - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - SOFORTIGES ABWEISEN VON NACHRICHTEN

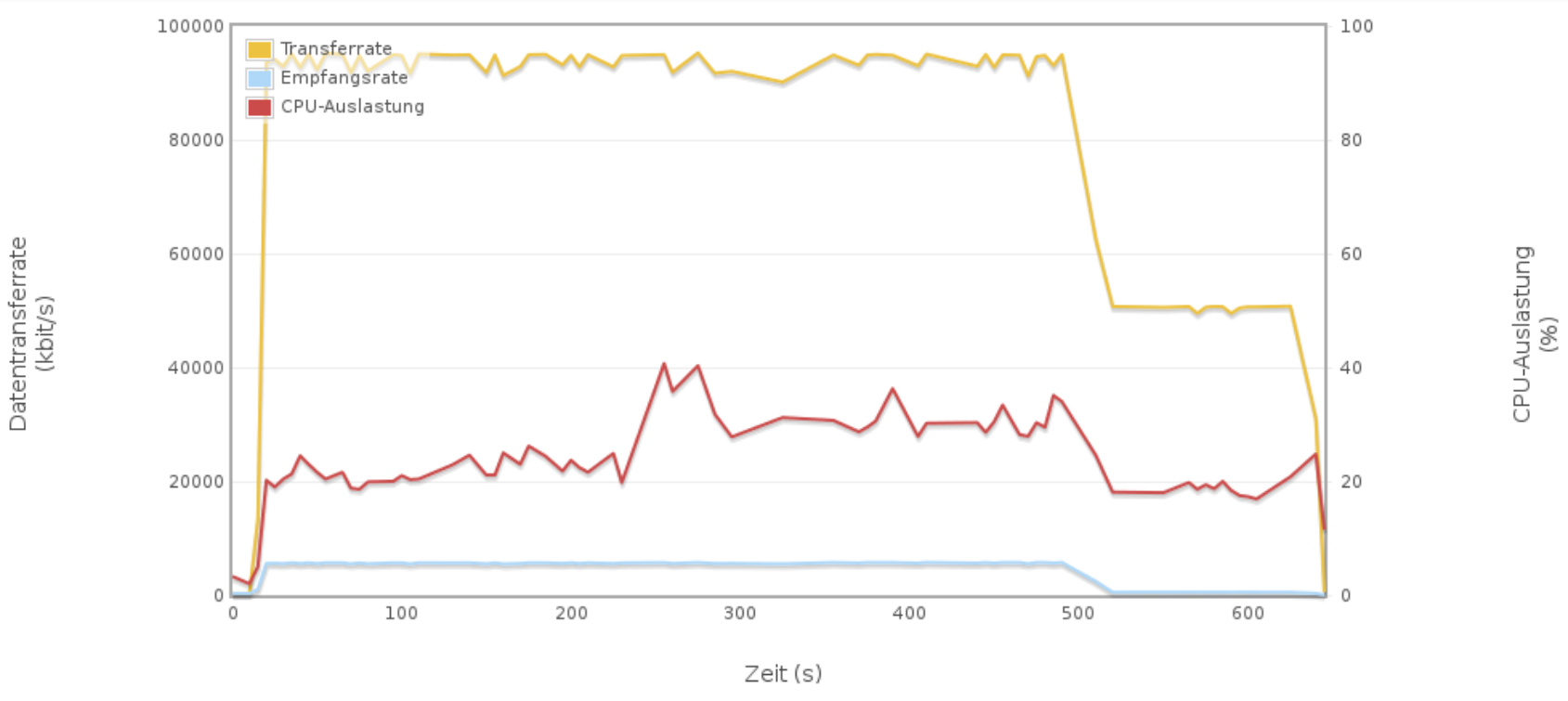


Abbildung 12: Sofortiges Abweisen von Nachrichten - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - SOFORTIGES ABWEISEN VON NACHRICHTEN

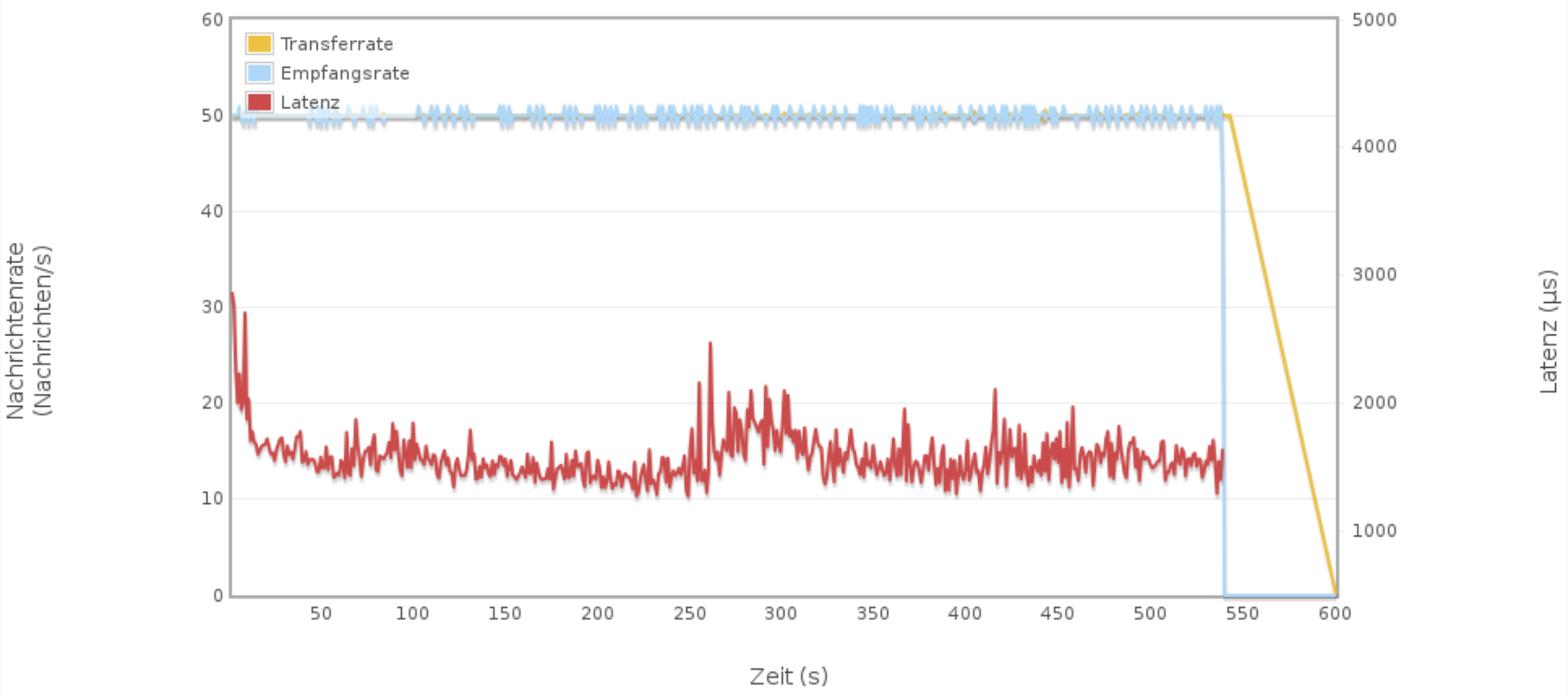


Abbildung 13: Sofortiges Abweisen von Nachrichten - Verlauf der Transfer-, Empfangsrate und Verlauf der Latenz im Anwendungsszenario

Angriff - Gebündeltes Abweisen von Nachrichten

Aktion:
1. Producer sendet Nachrichten an Server
2. Mehrere Consumer erhalten Nachrichten
3. Consumer ignorieren diese bis um Erreichen eines Schwellwertes
4. Alle empfangen Nachrichten werden *gebündelt* über eine *Negative Bestätigung* abgewiesen

Ziel: RAM, Festplatte, Netzwerkbandbreite, CPU

Ansatz: Server muss alle Nachrichten zwischenspeichern bis der Erhalt *erfolgreich* quittiert wurde und muss alle abgewiesene Nachrichten *stoßweise* erneut senden.

ANGRIFF - GEBÜNDELTES ABWEISEN VON NACHRICHTEN

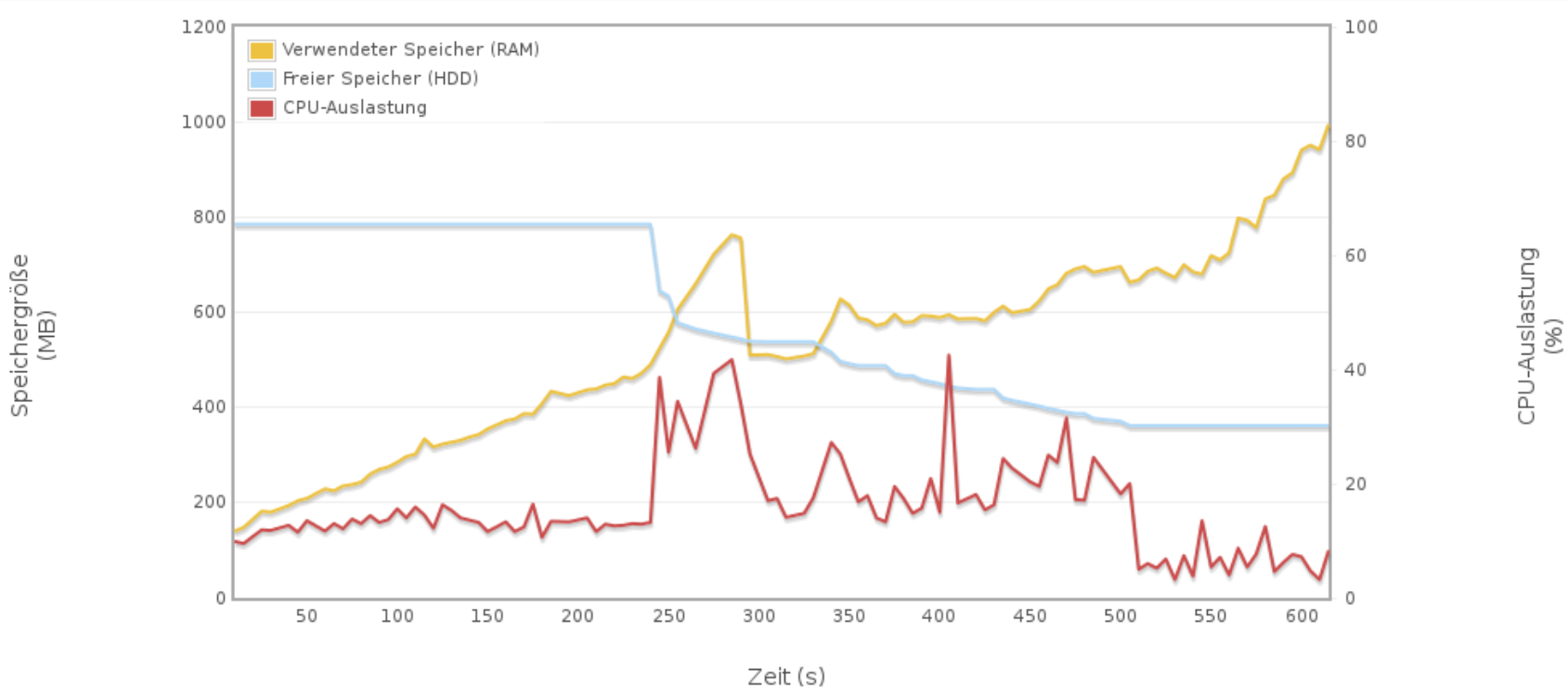


Abbildung 14: Gebündeltes Abweisen von Nachrichten - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - GEBÜNDELTES ABWEISEN VON NACHRICHTEN

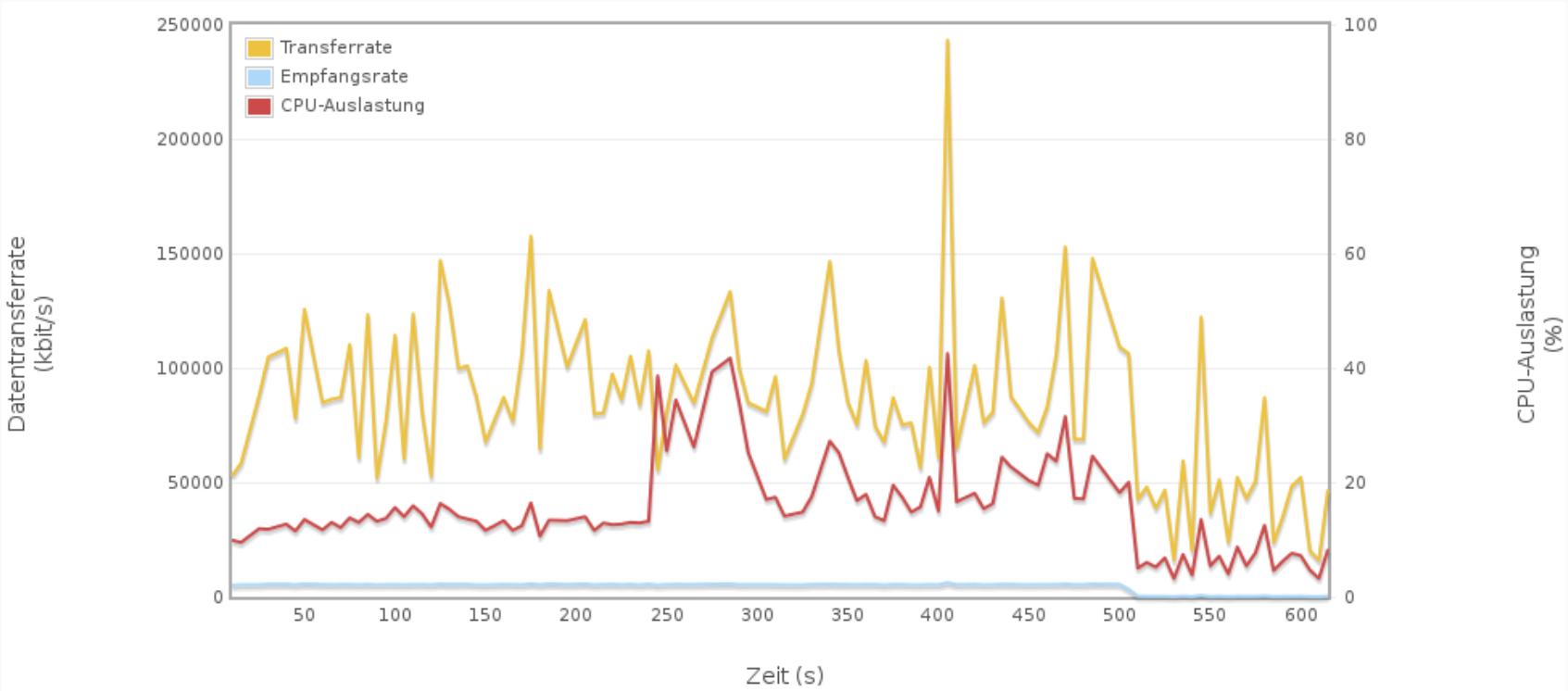


Abbildung 15: Gebündeltes Abweisen von Nachrichten - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - GEBÜNDELTES ABWEISEN VON NACHRICHTEN

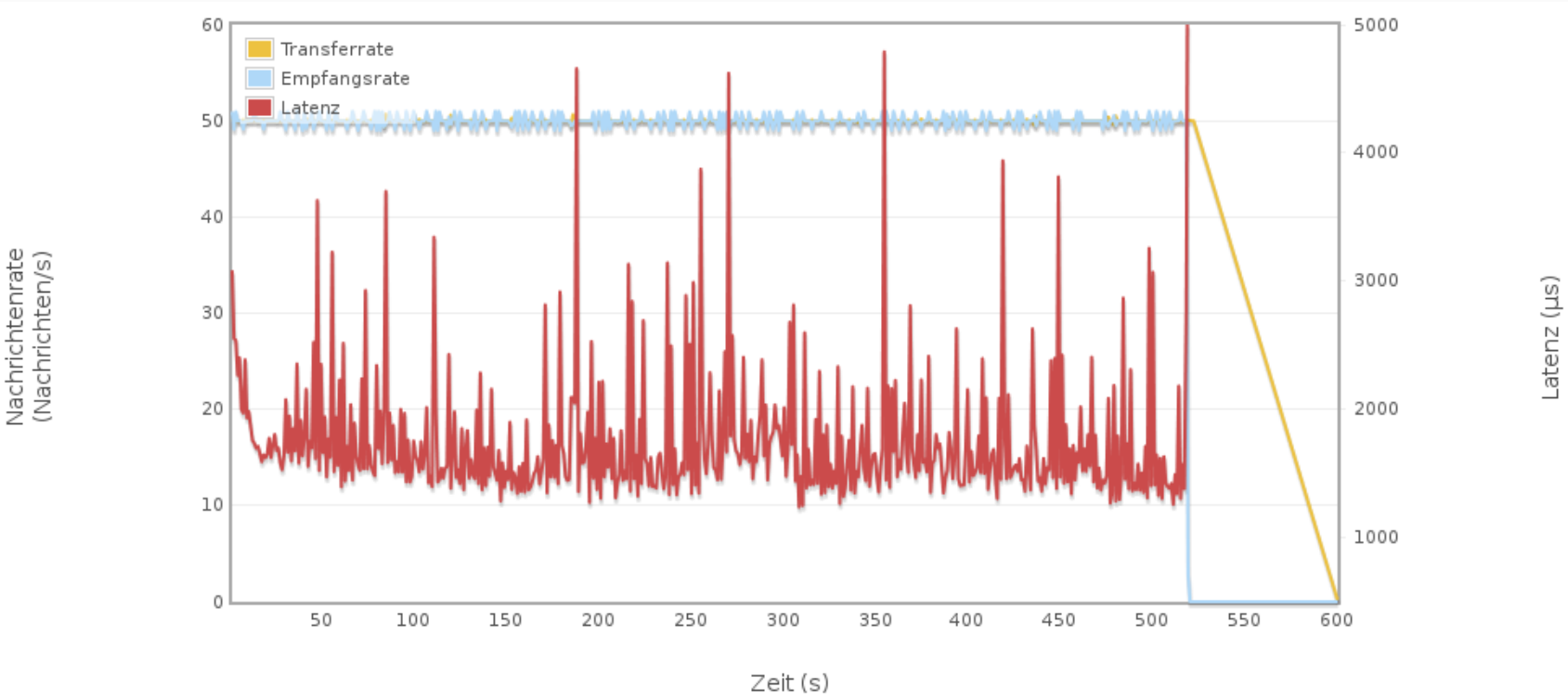


Abbildung 16: Gebündeltes Abweisen von Nachrichten - Verlauf der Transfer-, Empfangsrate und Verlauf der Latenz im Anwendungsszenario

Angriff - Queue-Churning

Aktion:
1. Client erzeugt fortlaufend neue Queues bis zum Erreichen eines Schwellwertes
2. **OPTIONAL** Client sendet Nachrichten an Queue
3. Client löscht schlagartig alle Queues

Ziel: CPU, RAM

Ansatz: Server muss Überreste der Queue beseitigen, während bereits neue Queues erzeugt werden.

ANGRIFF - QUEUE-CHURNING

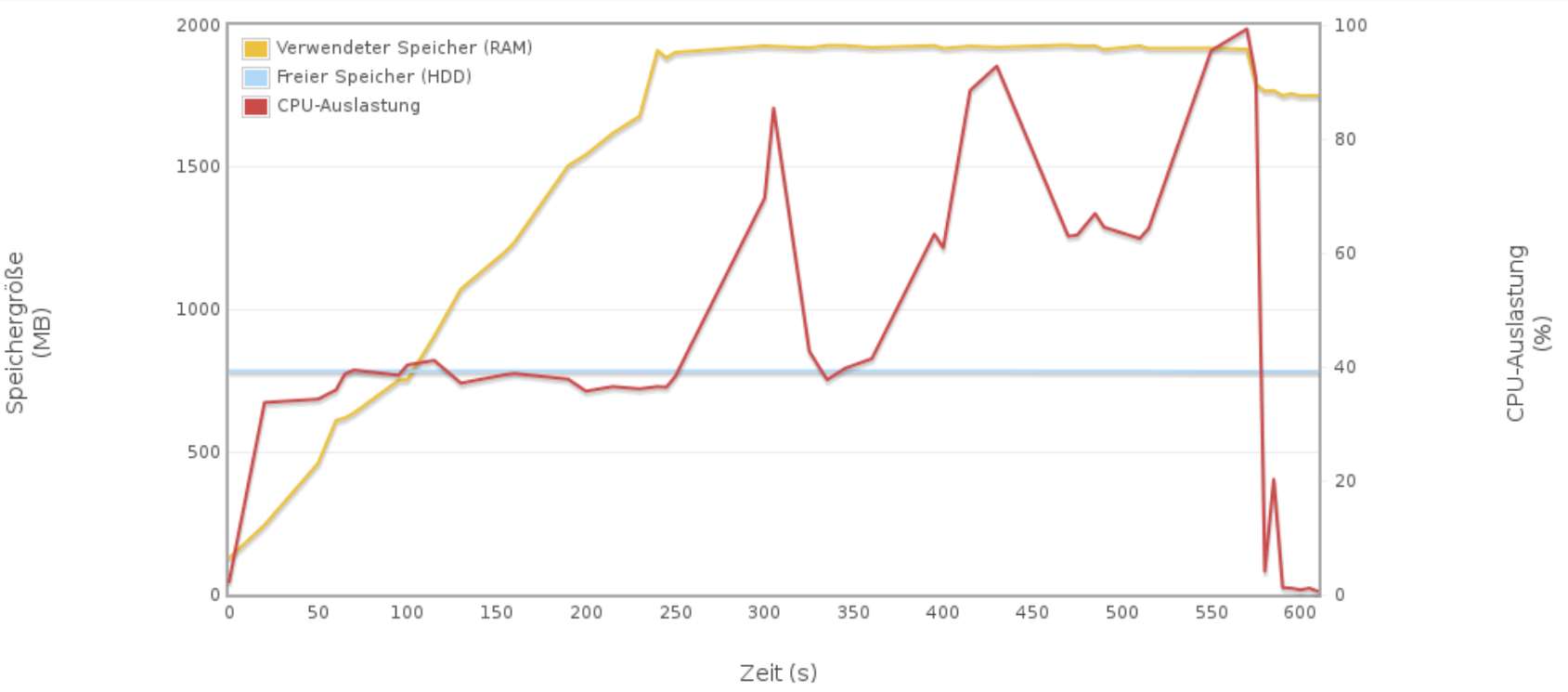


Abbildung 17: Queue-Churning - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

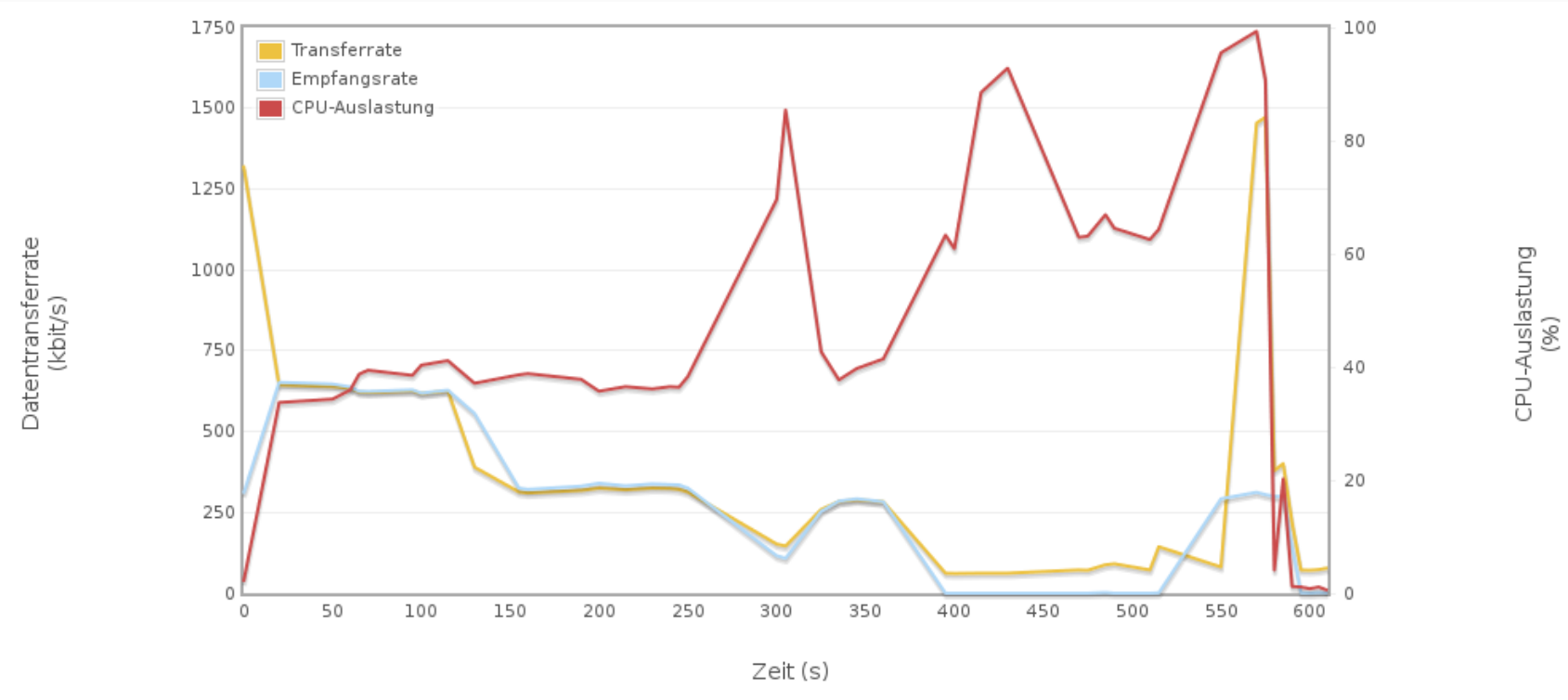


Abbildung 18: Queue-Churning - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

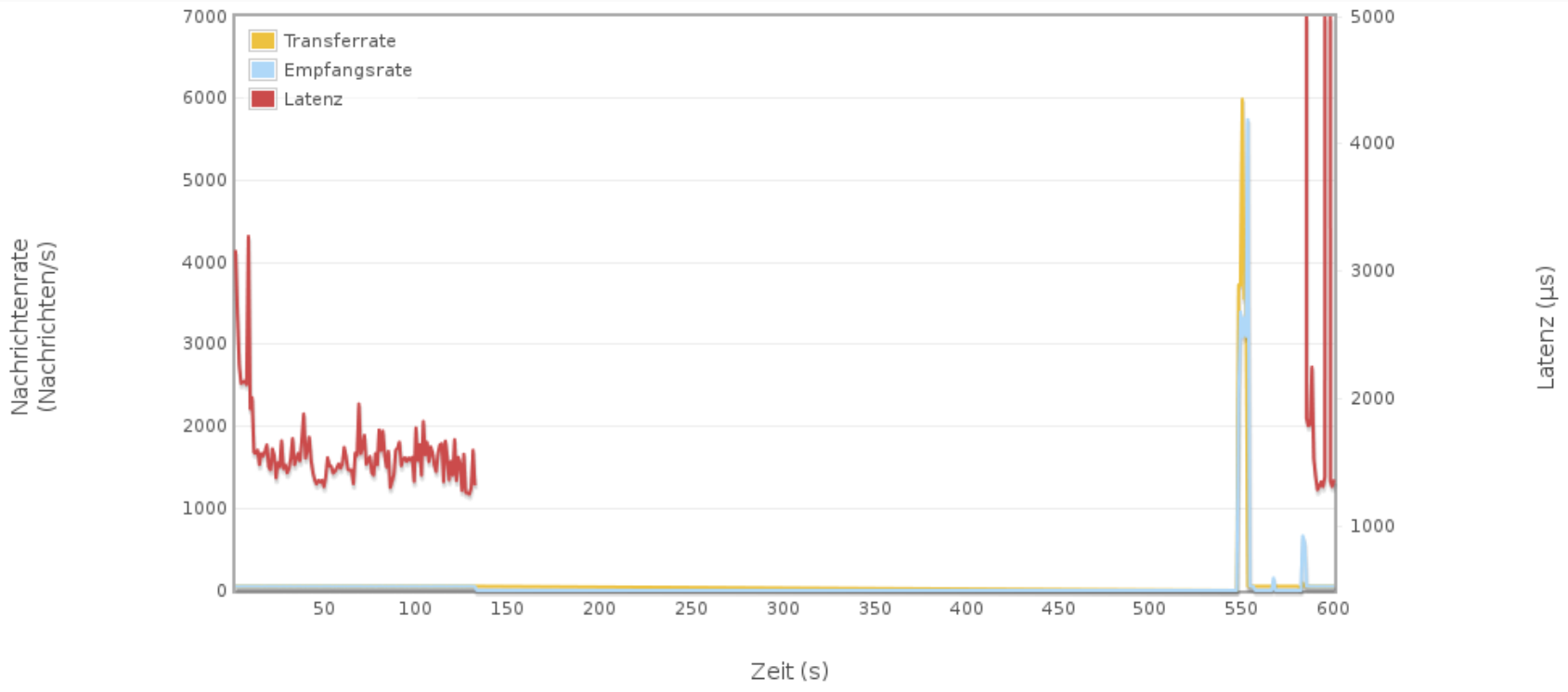


Abbildung 19: Queue-Churning - Verlauf der Transfer-, Empfangsrate und Verlauf der Latenz im Anwendungsszenario

Angriff - Handshake-Trickle

Aktion:
1. Mehrere Clients starten einen Verbindungsaufbau zum Server
2. Handshake wird in jeder Phase künstlich pausiert
3. Clients schließen Verbindung
4. Close-Handshake wird ebenfalls künstlich verlangsamt

Ziel: Unerwartetes Verhalten seitens des Server

Ansatz: Server muss für jeden begonnen Handshake Ressourcen zum Zustand des Verbindungsaufbaues bzw. Verbindungsabbaues allokieren.

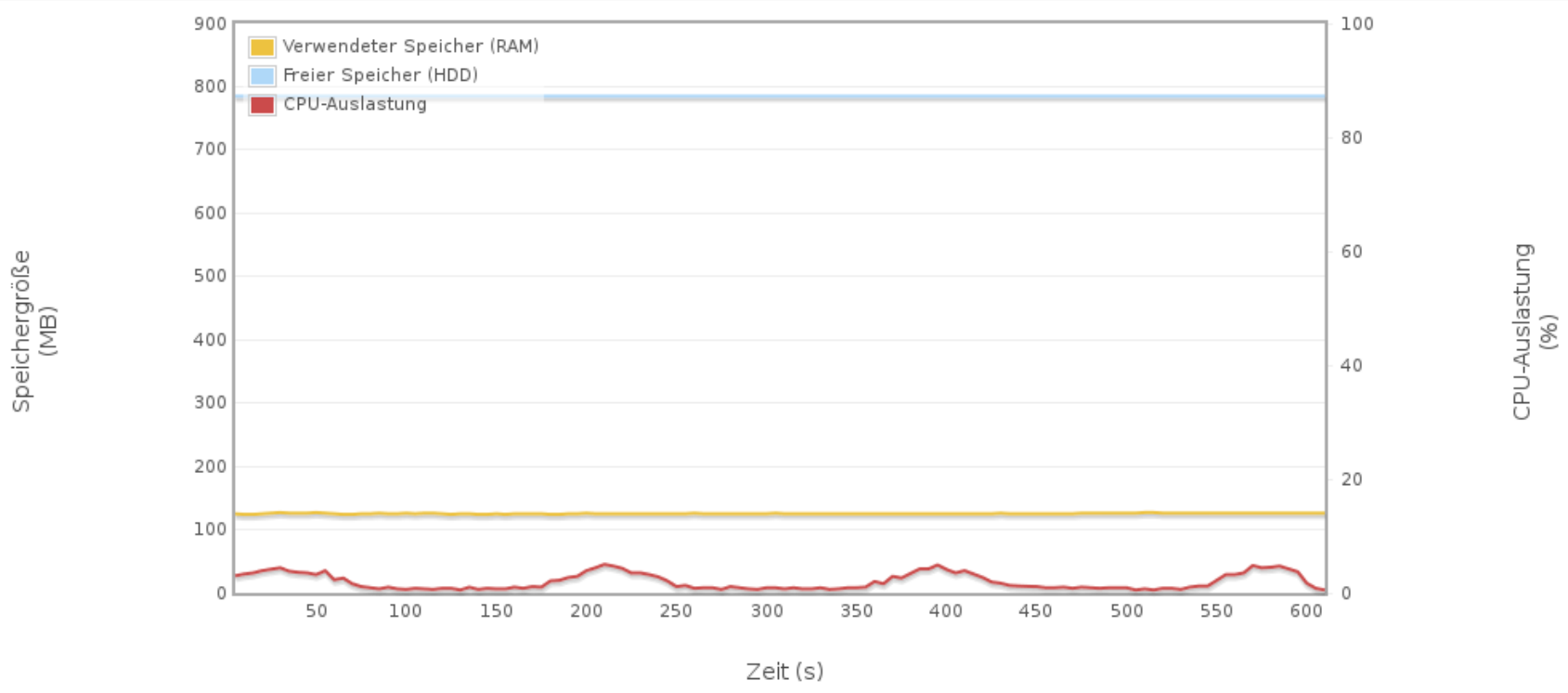


Abbildung 20: Handshake-Trickle - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

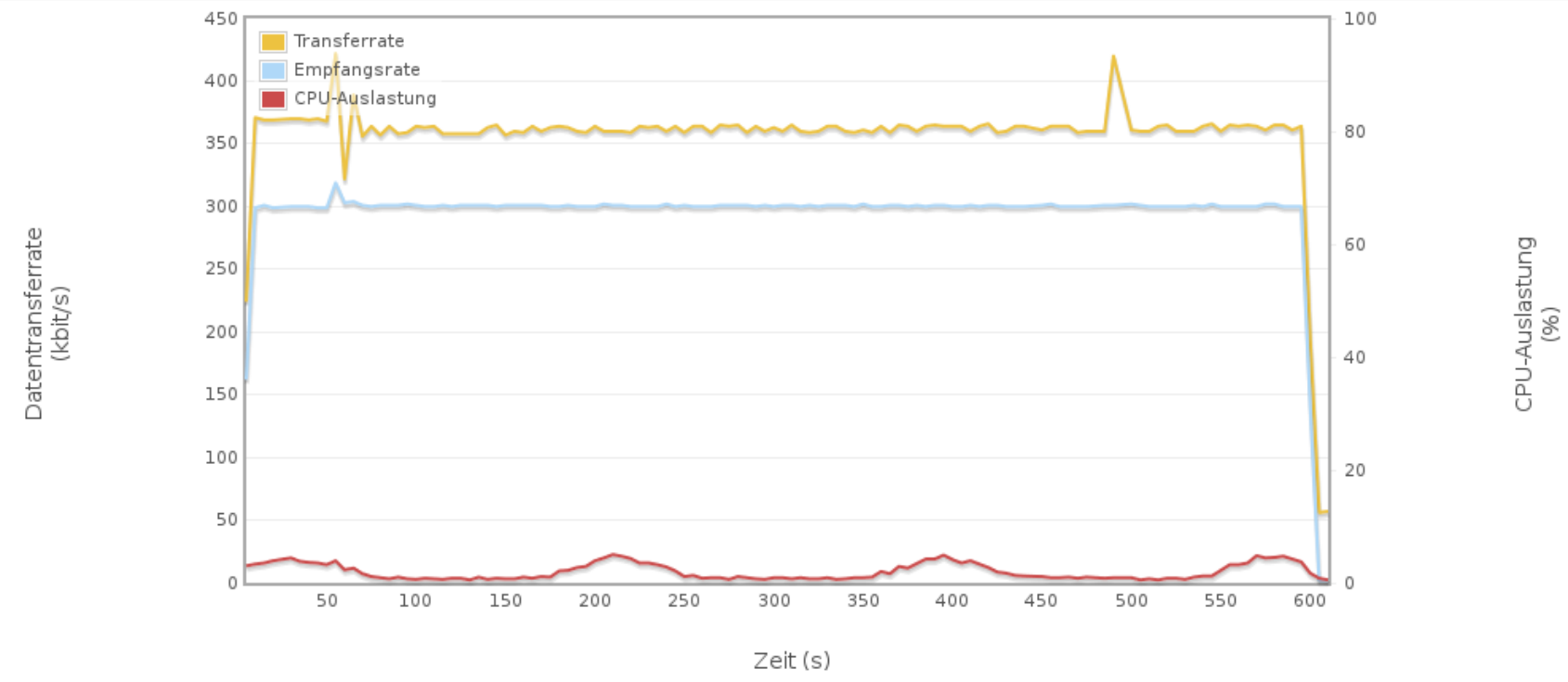


Abbildung 21: Handshake-Trickle - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

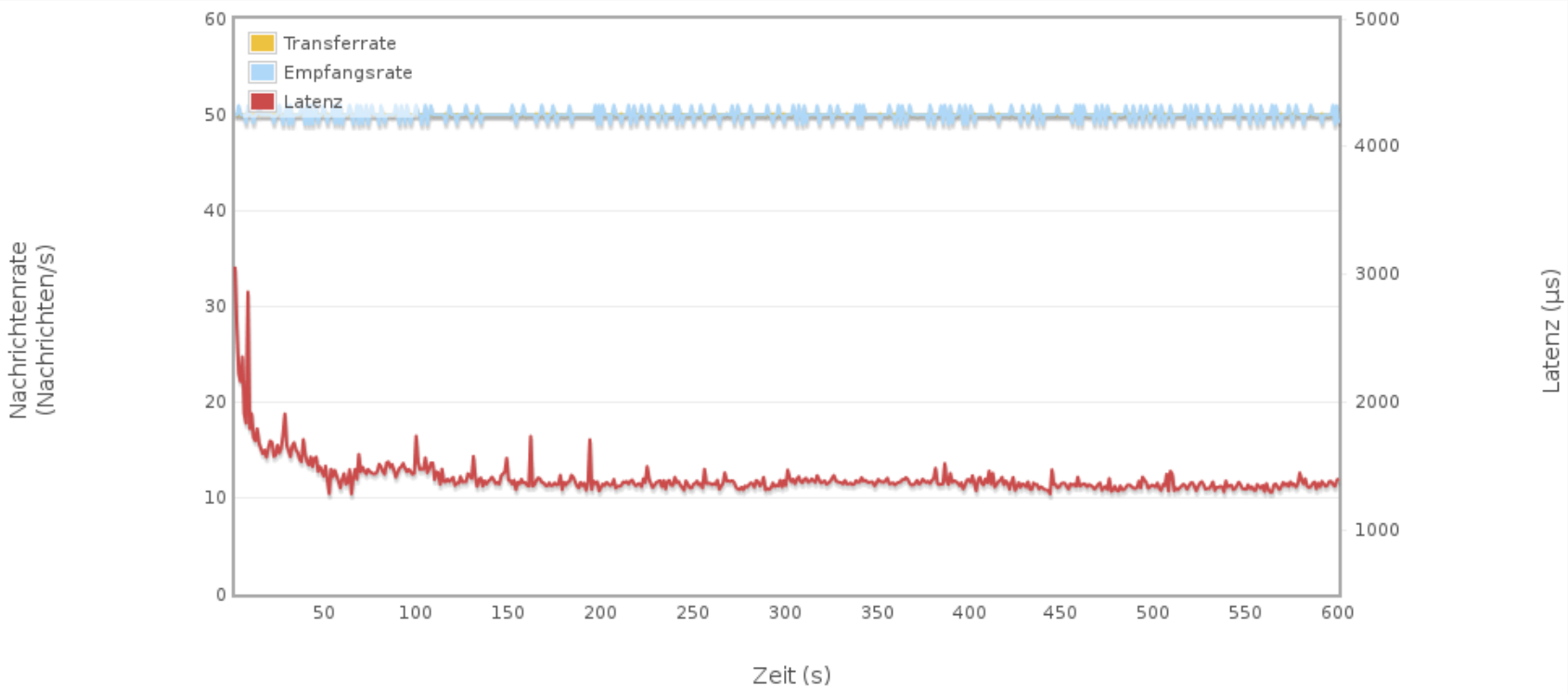


Abbildung 22: Handshake-Trickle - Verlauf der Transfer-, Empfangsrate und Verlauf der Latenz im Anwendungsszenario

Angriff - Heartbeat-Flooding

Aktion:
1. Mehrere Clients öffnen Connections zum Server
2. Clients stellen dabei Anfrage den Heartbeat herunterzusetzen

Ziel: CPU, Netzwerkbandbreite

Ansatz: Server muss bei der Hälfte des angeforderten Timeouts einen Heartbeat an die Clients senden.

ANGRIFF - HEARTBEAT-FLOODING

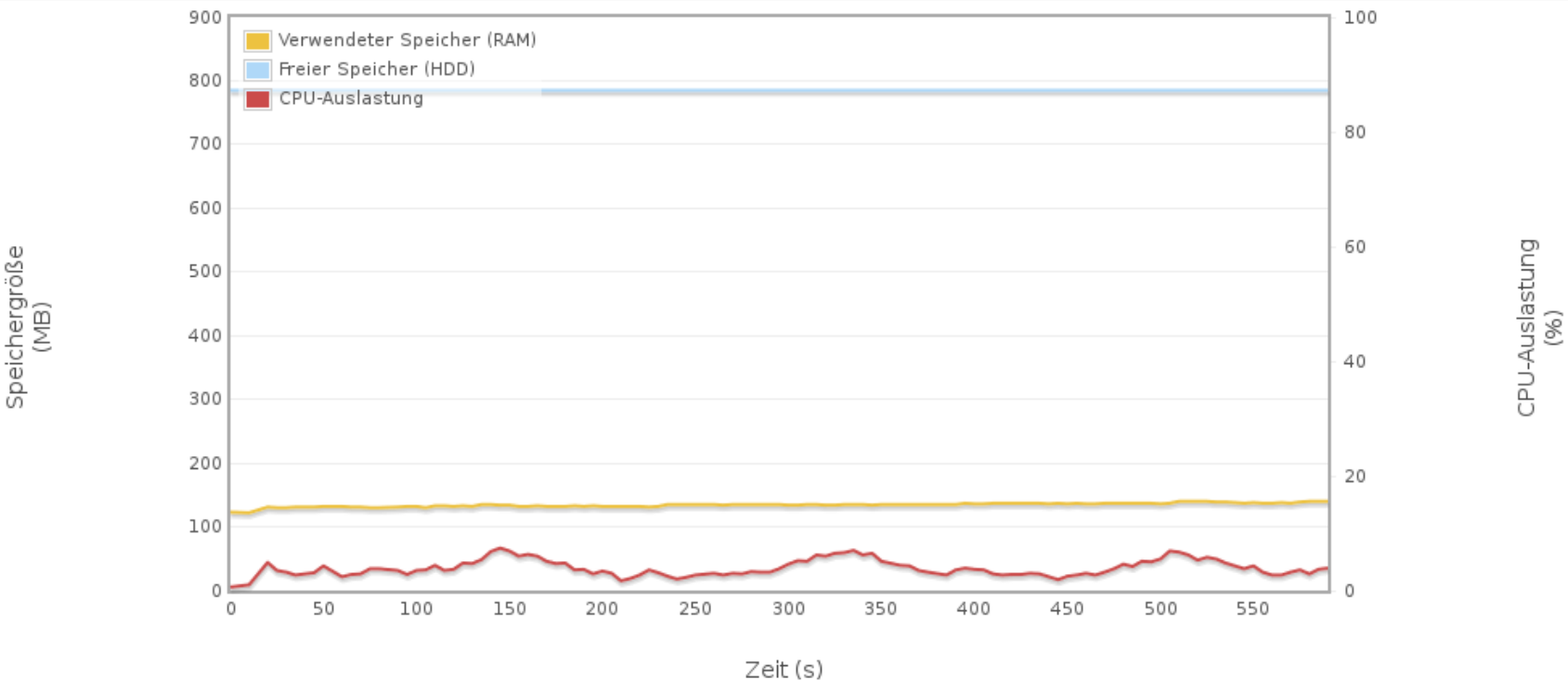


Abbildung 23: Heartbeat-Flooding - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

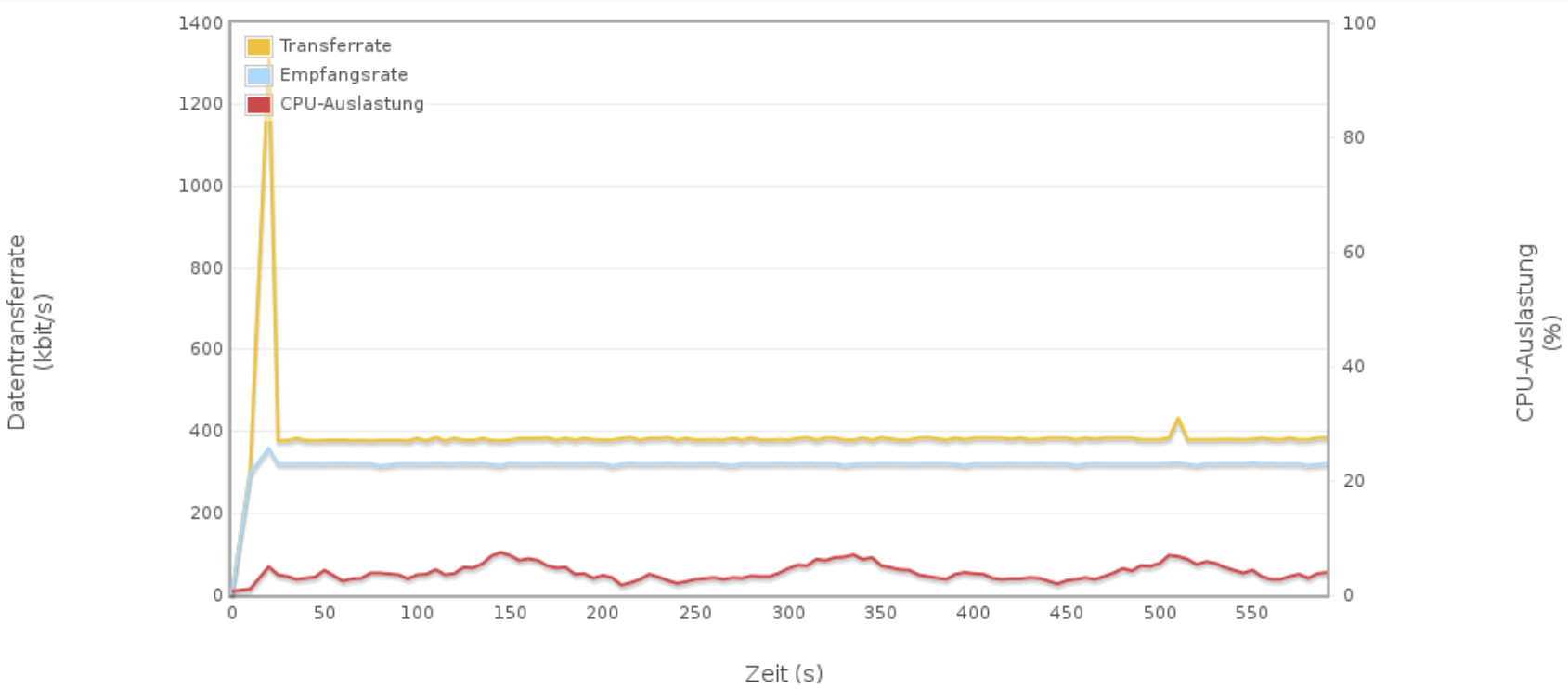


Abbildung 24: Heartbeat-Flooding - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

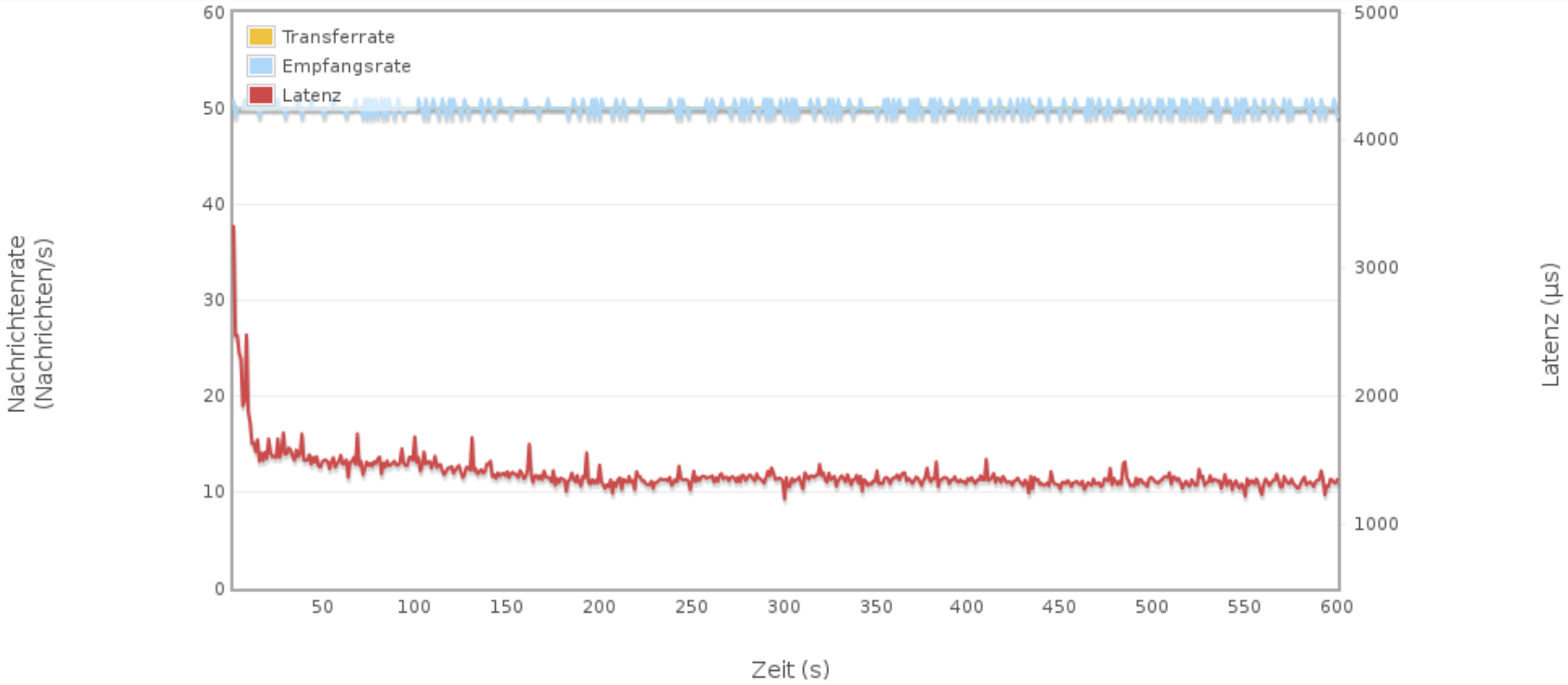


Abbildung 25: Heartbeat-Flooding - Verlauf der Transfer-, Empfangsrate und Verlauf der Latenz im Anwendungsszenario

Angriff - TCP-Connection-Dropping

Aktion:
1. Client öffnet Connection zum Server mit max. Heartbeat
2. Extrahiert hierbei TCP-Socket
3. Deaktiviert Keep-Alive, Aktiviert SO_Linger
4. Schließt TCP-Socket; Firewall blockiert RST-Paket

Ziel: RAM, Socketdeskriptoren

Ansatz: Server muss für jede „offene“ Verbindung Ressourcen bereitstellen. Alle Mechanismen zur Erkennung der toten Verbindung werden verzögert oder deaktiviert.

ANGRIFF - TCP-CONNECTION-DROPPING

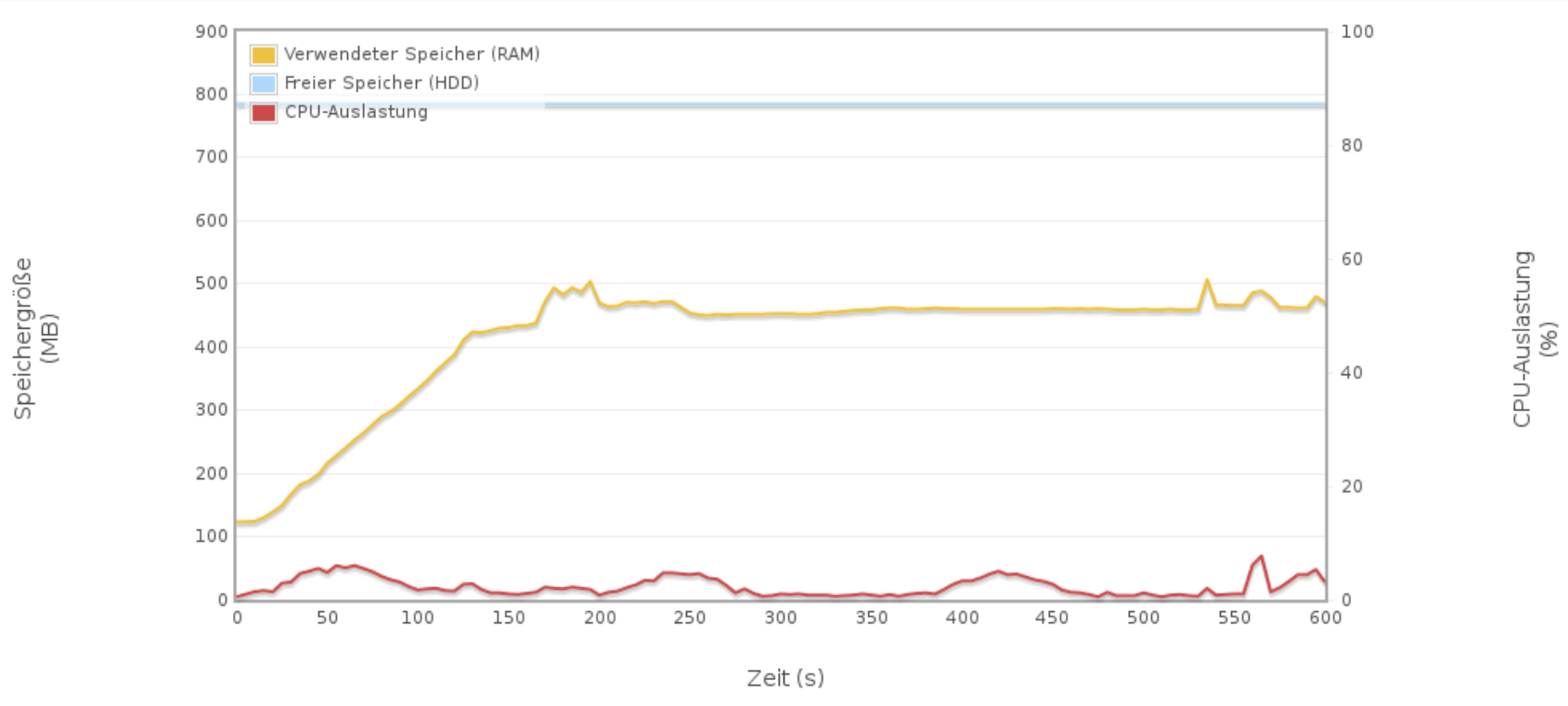


Abbildung 26: TCP-Connection-Dropping - Verlauf des Speicherbedarfs für RAM/HDD und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - TCP-CONNECTION-DROPPING

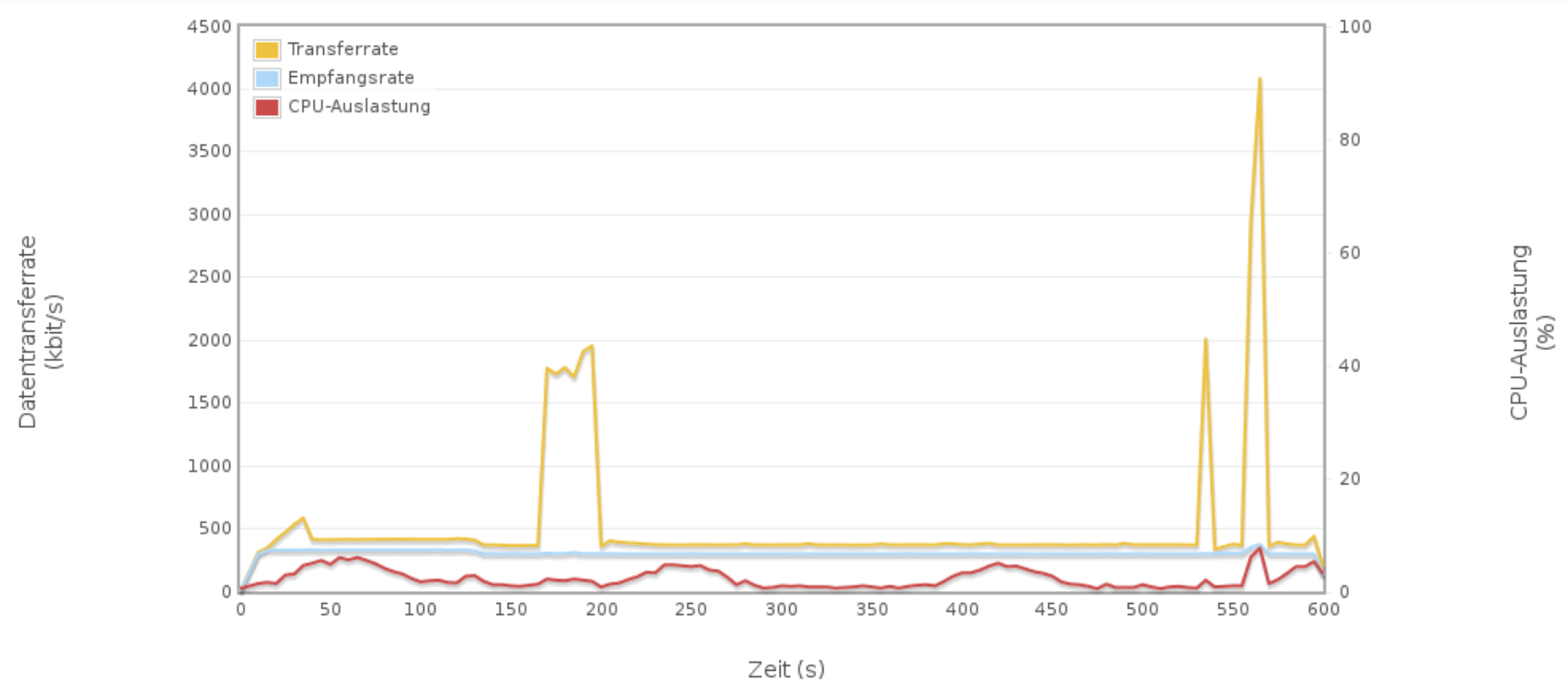


Abbildung 27: TCP-Connection-Dropping - Verlauf der Transfer-, Empfangsrate und Verlauf der CPU-Last auf dem RabbitMQ-Server

ANGRIFF - TCP-CONNECTION-DROPPING

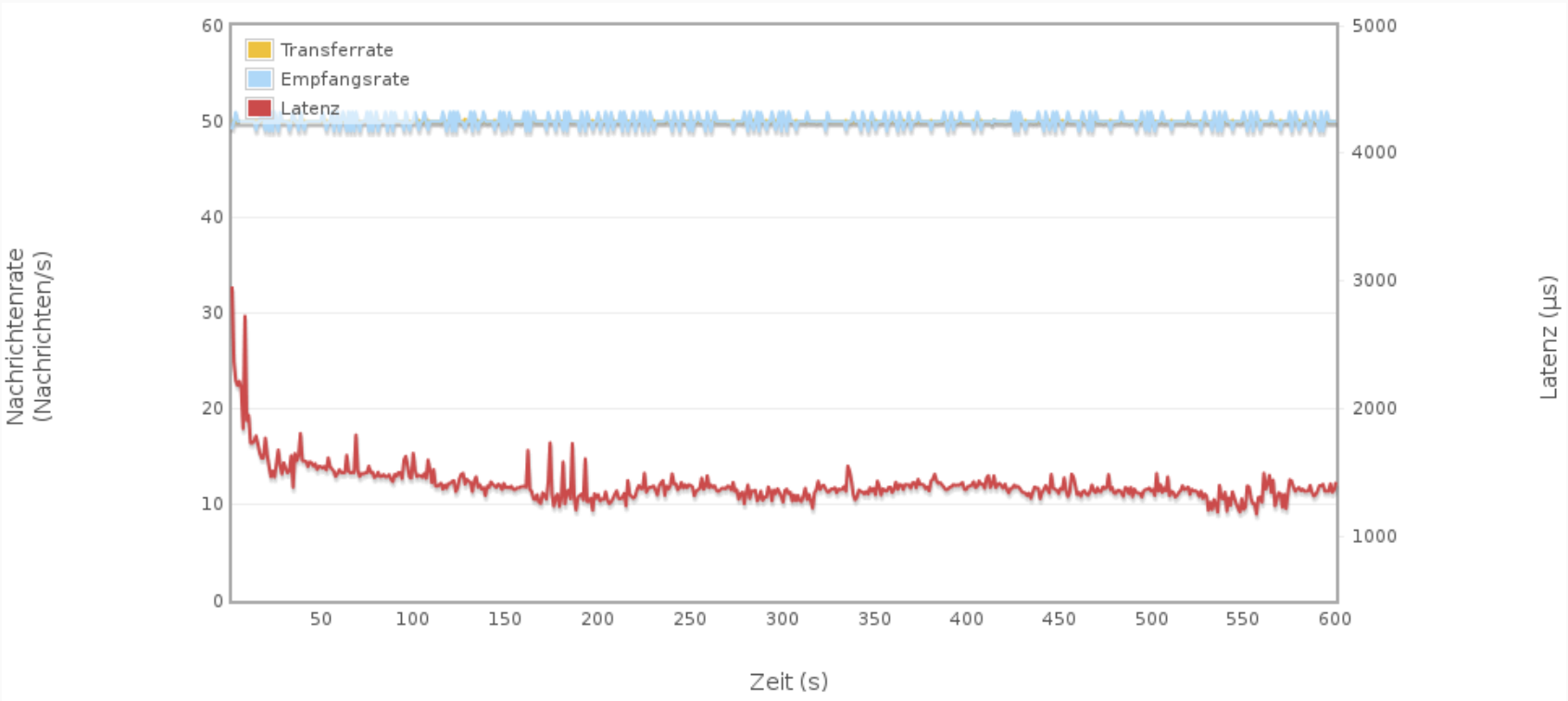


Abbildung 28: TCP-Connection-Dropping - Verlauf der Transfer-, Empfangsrate und Verlauf der Latenz im Anwendungsszenario

ABSCHLUSS

UMSETZUNG - MÖGLICHE ANGRIFFSVEKTOREN

- Viele Verbindungen
- Bruteforce-Attacke auf Benutzercredentials
- Hohe Datenrate (viele kleine, wenig große Nachrichten)
- Große Header, kleiner Payload
- Langsamer Verbindungsaufbau
- Unvollständiger Verbindungsaufbau
- Pause im Protokollablauf

UMSETZUNG - MÖGLICHE ANGRIFFSVEKTOREN

- Viele Verbindungen + Viele Channel ✔
- Bruteforce-Attacke auf Benutzercredentials ✖
- Hohe Datenrate (viele kleine, wenig große Nachrichten) ✔
- Große Header, kleiner Payload ✔
- Langsamer Verbindungsaufbau + Verbindungsabbau ✔
- Unvollständiger Verbindungsaufbau ~
- Pause im Protokollablauf ~
- Queue-Churning
- Heartbeat-Flooding
- Transaktionen
- Ausnutzen Message-Response

ZUSAMMENFASSUNG

- Umfangreiche Angriffsszenarien realisiert
  ➥ Genügend Potenzial für weitere Angriffsszenarien
- Angefragte Konfiguration durch Client oft ausschlaggebend
  ➥ Prefetching, Framegröße, Heartbeat, ...
- Konfigurierte Ressourcen-Limits werden teilweise ignoriert
  ➥ RAM bei Queues, Transaktionen

Fazit & Empfehlung

- Konfigurierte Ressourcen-Limits sollten verbindlich sein
- Konfigurationsspielraum für Clients sollte auf Einsatzgebiet eingeschränkt werden

  ➥ Anzahl Queues, Channel, Prefetching, Framegröße, Headergröße, Heartbeat, Transaktionsgröße

Fazit & Empfehlung

- Konfigurierte Ressourcen-Limits sollten verbindlich sein
- Konfigurationsspielraum für Clients sollte auf Einsatzgebiet eingeschränkt werden

  ➥ Anzahl Queues, Channel, Prefetching, Framegröße, Headergröße, Heartbeat, Transaktionsgröße

RabbitMQ ist stabiler Messagebroker, der in Standardkonfiguration jedoch nicht gefahrlos verwendet werden kann.

KONTAKT

✉ *marcel.mielke@stud.htwk-leipzig.de*

✉ *philipp.sieder@stud.htwk-leipzig.de*

FRAGEN?