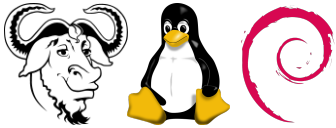
Das *Trivial File Transfer Protocol (TFTP)*

Protokolle, RFCs und ein Zauberlehrling

Andreas B. Mundt
andi@debian.org

Dornbirn Linux Day 2022

24. September 2022

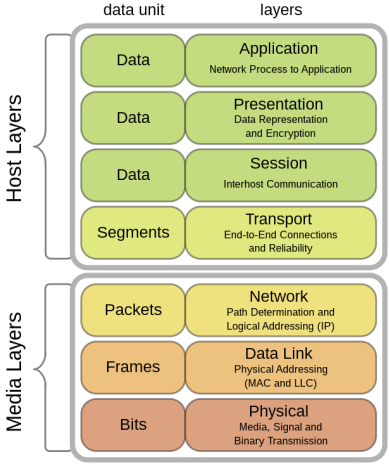


Vorgeschichte und Ausblick

... Release Critical (RC) Bug #988456 (missing dependency) → fix ...

1. Wie kann man Daten in einem Netzwerk übertragen?
2. Das TFT Protocol
3. TFTP Anwendung
4. Der Zauberlehrling und weitere Defizite
5. RFCs: Requests for Comments
6. TFTP Option Extension
7. Optimierungen und Tests

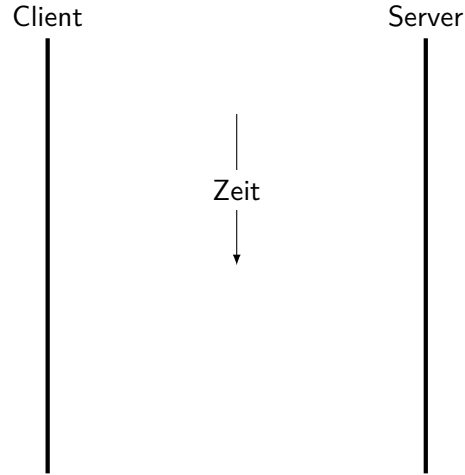
Das ISO/OSI-Referenzmodell



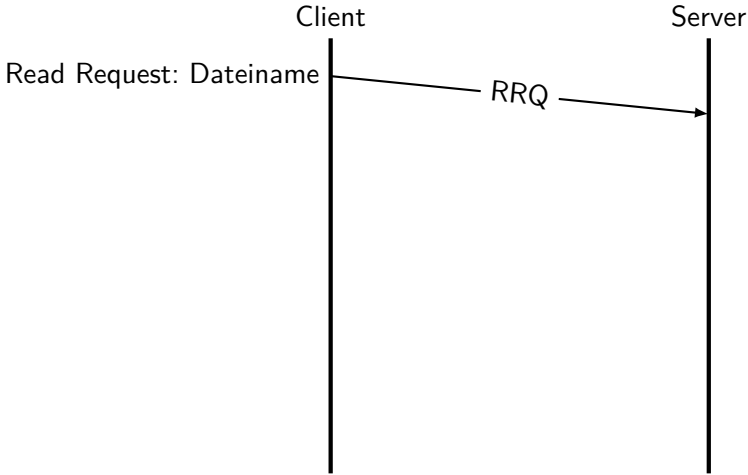
- Physical und Data Link Layer → Daten an das nächste Gerät im LAN senden
- Network Layer (IP) → Rechner in verschiedenen Netzen adressieren
- Transport Layer (UDP, TCP, . . .) → Prozesse auf verschiedenen Rechnern verbinden

Im einfachsten Fall (**Trivial** FTP) mit dem User Datagram Protocol (UDP)

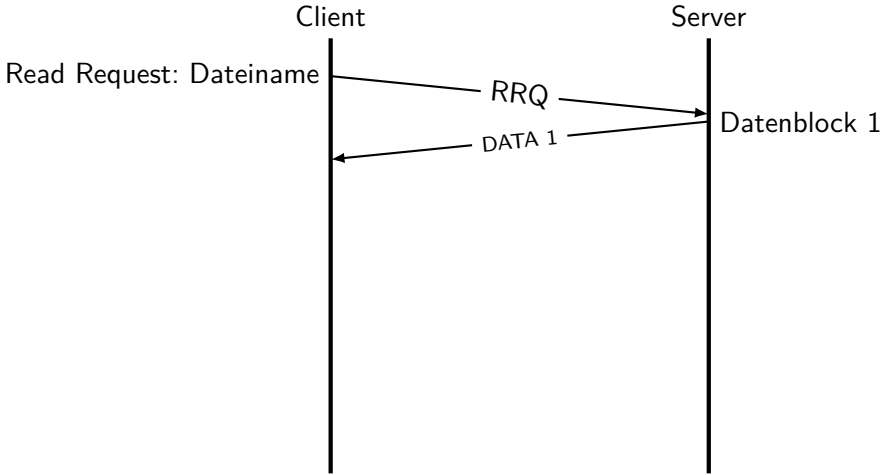
Trivial File Transfer Protocol



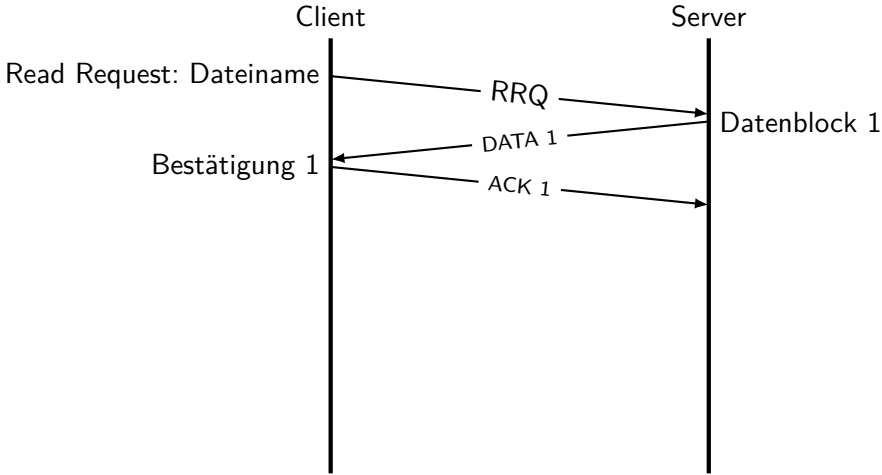
Trivial File Transfer Protocol



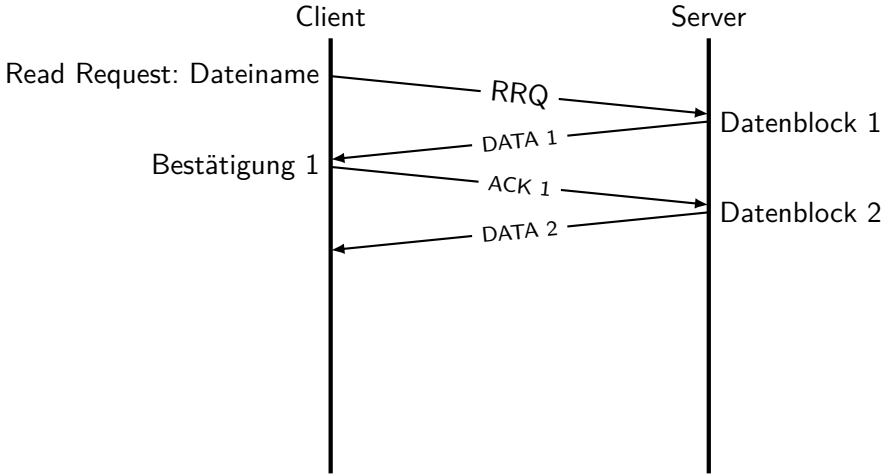
Trivial File Transfer Protocol



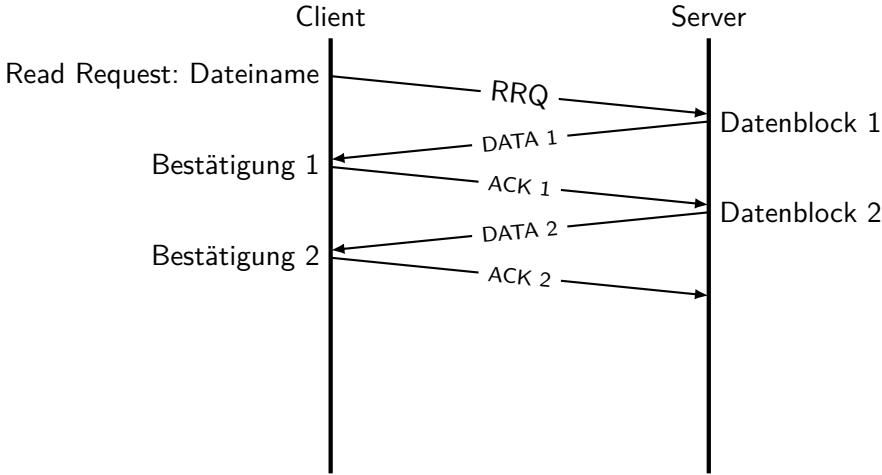
Trivial File Transfer Protocol



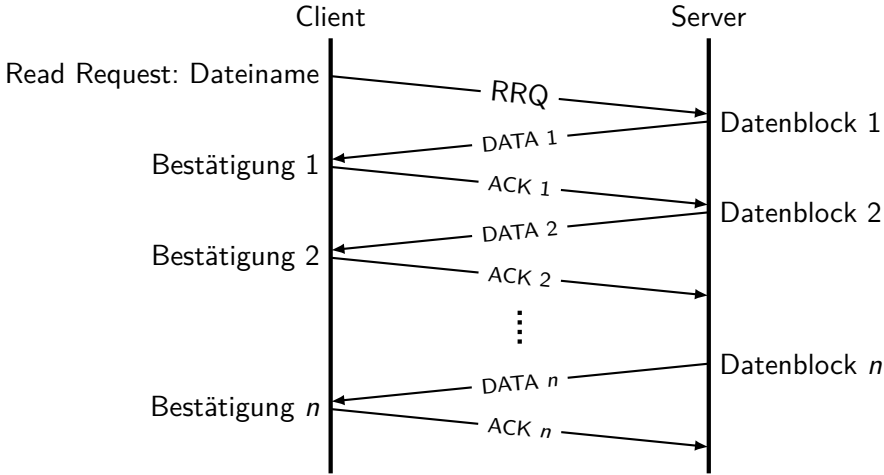
Trivial File Transfer Protocol



Trivial File Transfer Protocol



Trivial File Transfer Protocol



TFTP Algorithmus (ursprünglich)

Server:

- Nach dem Read Request RRQ: Sende Datenblock DATA 1
- Sende für Bestätigung ACK j den nächsten[1] Datenblock DATA $j + 1$

Client:

- Bestätige jeden Datenblock k mit dem zugehörigen Acknowledgement k
- Der letzte Datenblock ist kleiner als die vorhergehenden oder ganz leer.

Was, wenn eine „Antwort“ (DATA oder ACK) ausbleibt[2]?

$\rightarrow$ Sende nach Timeout einfach das letzte Datagram erneut.

[1] solange $j \neq n$

[2] z.B. durch Paketverlust

TFTP Algorithmus (ursprünglich)

Server:

- Nach dem Read Request RRQ: Sende Datenblock DATA 1
- Sende für Bestätigung ACK j den nächsten[1] Datenblock DATA $j + 1$

Client:

- Bestätige jeden Datenblock k mit dem zugehörigen Acknowledgement k
- Der letzte Datenblock ist kleiner als die vorhergehenden oder ganz leer.

Was, wenn eine „Antwort“ (DATA oder ACK) ausbleibt[2]?

$\rightarrow$ Sende nach Timeout einfach das letzte Datagram erneut.

[1] solange $j \neq n$

[2] z.B. durch Paketverlust

TFTP Algorithmus (ursprünglich)

Server:

- Nach dem Read Request RRQ: Sende Datenblock DATA 1
- Sende für Bestätigung ACK j den nächsten[1] Datenblock DATA $j + 1$

Client:

- Bestätige jeden Datenblock k mit dem zugehörigen Acknowledgement k
- Der letzte Datenblock ist kleiner als die vorhergehenden oder ganz leer.

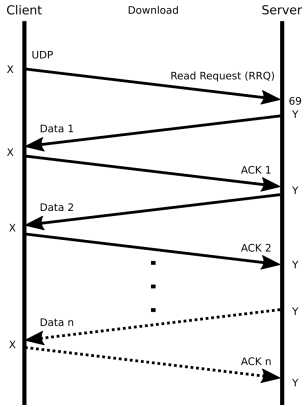
Was, wenn eine „Antwort“ (DATA oder ACK) ausbleibt[2]?

$\rightarrow$ Sende nach Timeout einfach das letzte Datagram erneut.

[1] solange $j \neq n$

[2] z.B. durch Paketverlust

TFTP Anwendung



No.	Time	Source	Source Port	Destination	Destination Port	Protocol	Length	Info
1	0.000000000	::1	41944	::1	69	TFTP	93	Read Request, File: d-i/n-
2	0.021296925	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 1
3	0.021360985	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 1
4	0.021398777	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 2
5	0.021429127	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 2
6	0.021469999	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 3
7	0.021498167	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 3
8	0.021527434	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 4
9	0.021556974	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 4
10	0.021573057	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 5
11	0.021604702	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 5
12	0.021619356	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 6
13	0.021631370	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 6
14	0.021644470	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 7
15	0.021656429	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 7
16	0.021669326	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 8
17	0.021681239	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 8
18	0.021697141	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 9
19	0.021744167	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 9
20	0.021758924	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 10
21	0.021772046	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 10
3645	0.061388936	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 1822
3646	0.061398571	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 1823
3647	0.061407236	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 1823
3648	0.061416838	::1	54995	::1	41944	TFTP	578	Data Packet, Block: 1824
3649	0.061425553	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 1824
3650	0.061437327	::1	54995	::1	41944	TFTP	418	Data Packet, Block: 1825 (last)
3651	0.061454647	::1	41944	::1	54995	TFTP	66	Acknowledgement, Block: 1825

[2] Z.B.: `atftpd --daemon --trace --no-fork --verbose=7 --logfile=/dev/stdout --port=6969 /tmp/`
`atftp --get --remote-file TFTP.toc --local-file /dev/null 127.0.0.1 6969 --trace --option "blksize 256"`

TFTP Anwendung

Anwendungsbereiche TFTP:

- Aufgrund der Einfachheit nur im lokalen Netz (LAN)
- Booten über das Netzwerk, Preboot Execution Environment (PXE)
- Embedded Systems: Firmware Images flashen auf Geräte wie Router, Firewalls, IP Phones, ...

Freie Open-Source TFTP-Server:

- `tftpd-hpa`: HPA's tftp server
- `atftpd`: advanced TFTP server
- `dnsmasq`: small caching DNS proxy and DHCP/TFTP server
- ...

TFTP Anwendung

Anwendungsbereiche TFTP:

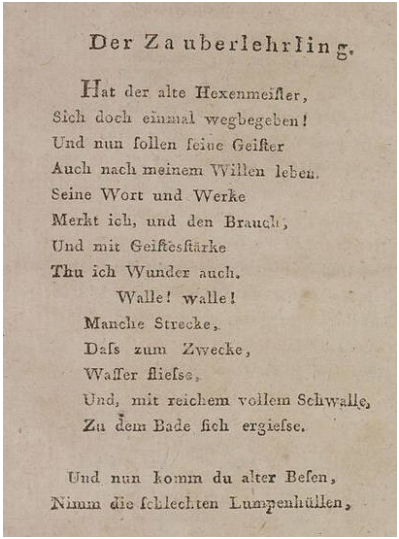
- Aufgrund der Einfachheit nur im lokalen Netz (LAN)
- Booten über das Netzwerk, Preboot Execution Environment (PXE)
- Embedded Systems: Firmware Images flashen auf Geräte wie Router, Firewalls, IP Phones, ...

Freie Open-Source TFTP-Server:

- `tftpd-hpa`: HPA's tftp server
- `atftpd`: advanced TFTP server
- `dnsmasq`: small caching DNS proxy and DHCP/TFTP server
- ...

Ausblick

Sorcerer's Apprentice Syndrome



Der Zauberlehrling.

Hat der alte Hexenmeister,
Sich doch einmal wegbegeben!
Und nun sollen seine Geister
Auch nach meinem Willen leben.
Seine Wort und Werke
Merkt ich, und den Brauch;
Und mit Geisteskräfte
Thu ich Wunder auch.
Walle! walle!
Manche Strecke,
Dass zum Zwecke,
Wasser fliesse,
Und, mit reichem vollem Schwalle,
Zu dem Bade sich ergiesse.

Und nun komm du alter Besen,
Nimm die schlechten Lumpenhüllen,

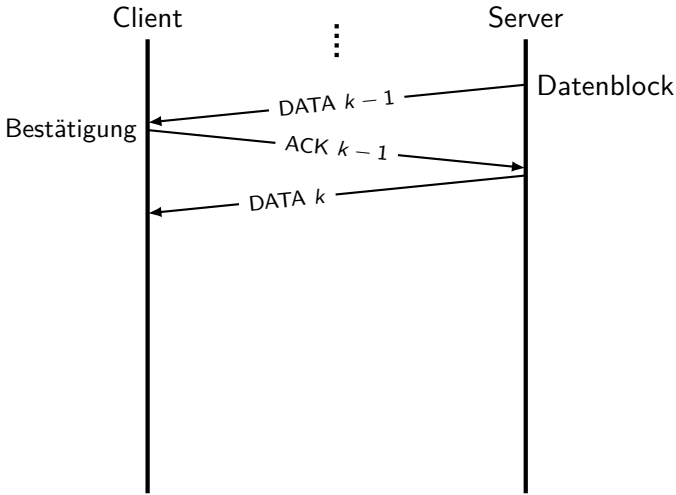
```
[...]
Seht da kommt er schleppend wieder!
Wie ich mich nur auf dich werfe,
gleich, o Kobold, liegst du nieder;
krachend trifft die glatte Schärfe.
Wahrlich, brav getroffen!
Seht, er ist entzwei!
Und nun kann ich hoffen,
und ich atme frei!

Wehe! wehe!
Beide Teile
stehn in Eile
schon als Knechte
völlig fertig in die Höhe!
Helft mir, ach! ihr hohen Mächte!
```

Sorcerer's Apprentice Syndrome

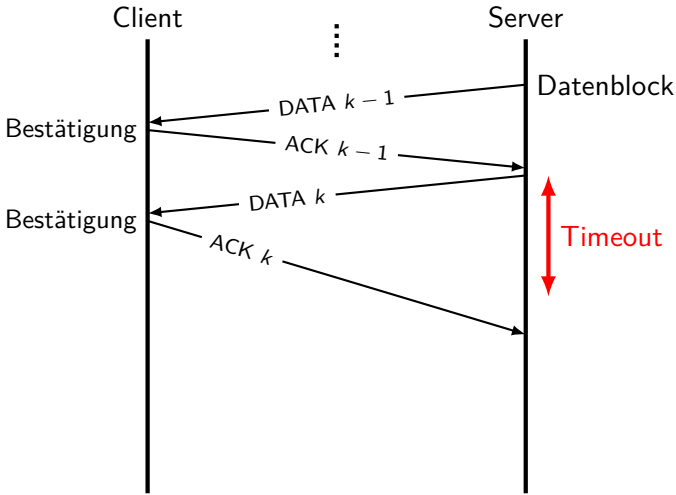
```
[...]
Seht da kommt er schleppend wieder!
Wie ich mich nur auf dich werfe,
gleich, o Kobold, liegst du nieder;
krachend trifft die glatte Schärfe.
Wahrlich, brav getroffen!
Seht, er ist entzwei!
Und nun kann ich hoffen,
und ich atme frei!

Wehe! wehe!
Beide Teile
stehn in Eile
schon als Knechte
völlig fertig in die Höhe!
Helft mir, ach! ihr hohen Mächte!
```

Und sie laufen! Naß und nässer
wirds im Saal und auf den Stufen.
Welch entsetzliches Gewässer!
Herr und Meister! hör mich rufen! -
Ach, da kommt der Meister!
Herr, die Not ist groß!
Die ich rief, die Geister
werd ich nun nicht los.

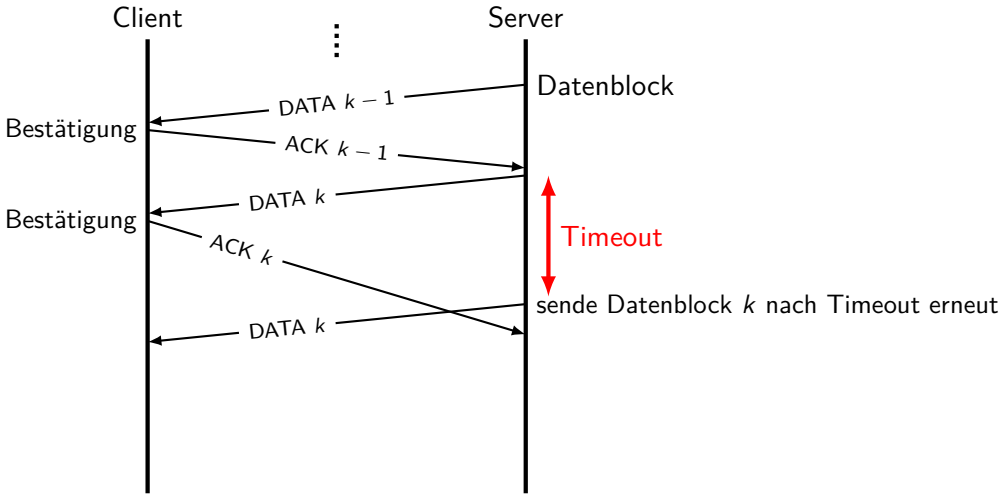
Sorcerer's Apprentice Syndrome



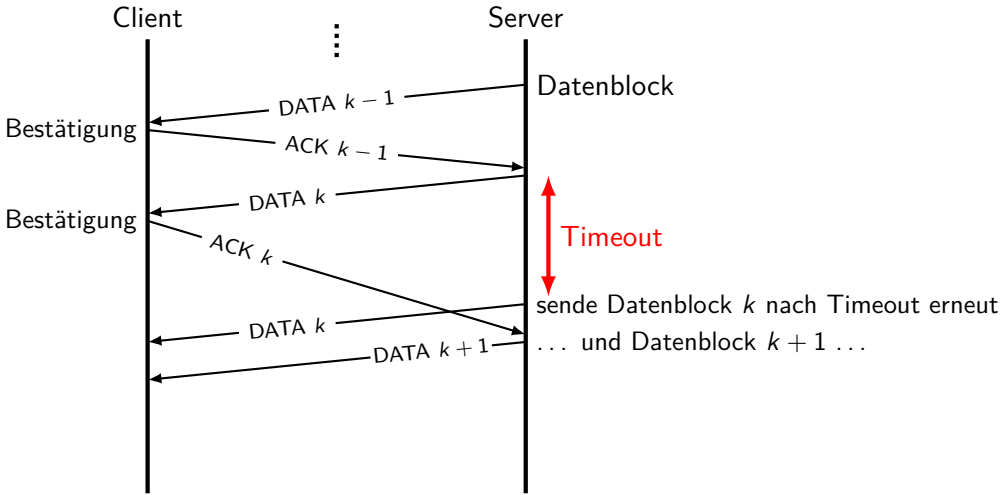
Sorcerer's Apprentice Syndrome



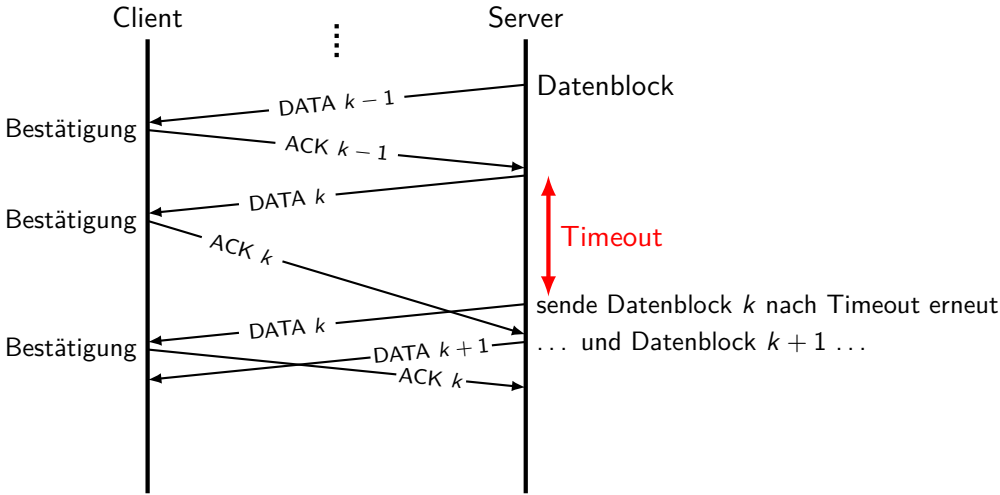
Sorcerer's Apprentice Syndrome



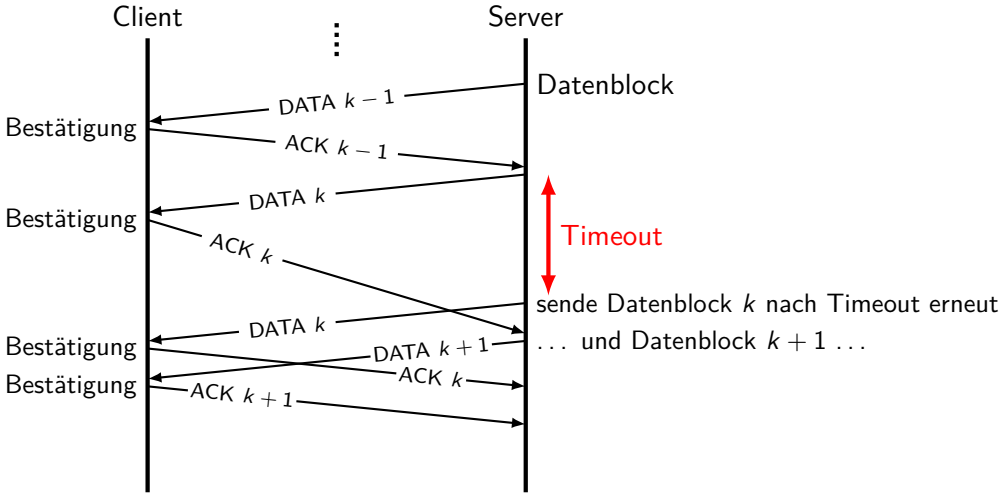
Sorcerer's Apprentice Syndrome



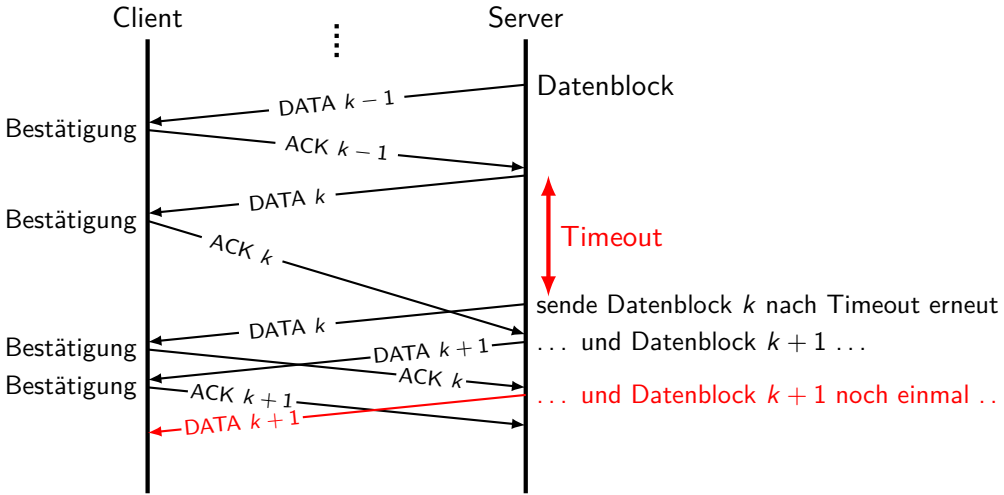
Sorcerer's Apprentice Syndrome



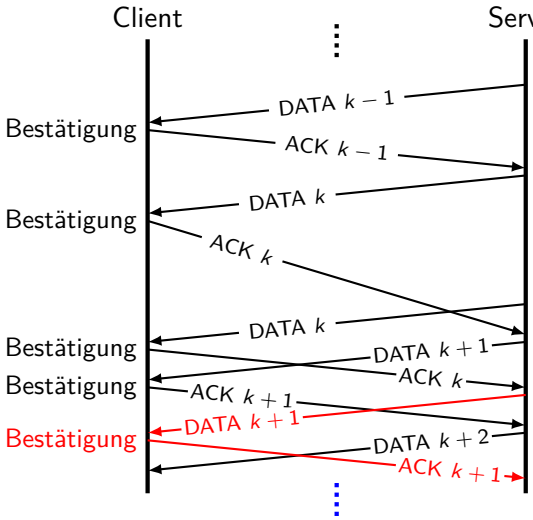
Sorcerer's Apprentice Syndrome



Sorcerer's Apprentice Syndrome



Sorcerer's Apprentice Syndrome



sende Datenblock k nach Timeout erneut

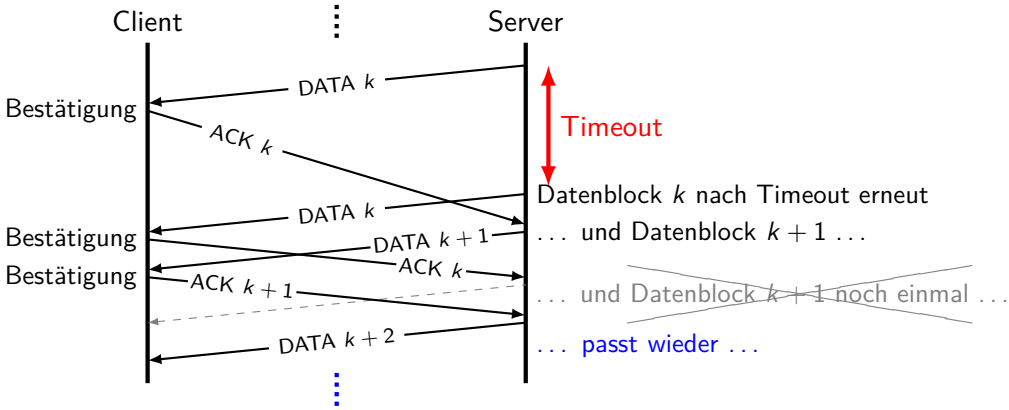
... und Datenblock $k+1$ ...

... und Datenblock $k+1$ noch einmal ...

... ab jetzt wird alles doppel verschickt. Und wenn dann schon mal ein Datagram stecken blieb ...

Sorcerer's Apprentice Syndrome

Lösung: Sende einen Datenblock nur einmal, es sei denn ein Timeout tritt auf.



Weitere Defizite des ursprünglichen TFTP

- Die Größe eines Datenblocks ist mit 512 Bytes fix vorgegeben.
- Der Empfänger einer Datei hat bis zum letzten Block keine Kenntnis über die insgesamt zu erwartende Datenmenge.
- Timeouts sind fix konfiguriert.
- Schlechte Performance: Pro Datenblock eine Round Trip Time (RTT).

RFCs Trivial File Transfer Protocol

Request **F**or **C**omments:

- RFC 783 (First Revision, 1981)
- RFC 1123 (1989): *Sorcerer's Apprentice Syndrome*[3]
- RFC 1350 (Second Revision, 1992)
- RFC 2347 (TFTP Option Extension, 1998)
- RFC 2348 (TFTP Blocksize Option, 1998)
- RFC 2349 (TFTP Timeout Interval and Transfer Size Options, 1998)
- RFC 7440 (TFTP Windowsize Option, 2015)

[3] Cf. J.W. von Goethe, „Der Zauberlehrling“: „Die ich rief, die Geister werd ich nun nicht los.“ (1797)
(„The spirits that I summoned / I now cannot rid myself of again“)

RFCs Trivial File Transfer Protocol

Request **F**or **C**omments:

- RFC 783 (First Revision, 1981)
- RFC 1123 (1989): *Sorcerer's Apprentice Syndrome*[3]
- RFC 1350 (Second Revision, 1992)
- RFC 2347 (TFTP Option Extension, 1998)
- RFC 2348 (TFTP Blocksize Option, 1998)
- RFC 2349 (TFTP Timeout Interval and Transfer Size Options, 1998)
- RFC 7440 (TFTP Windowsize Option, 2015)

[3] Cf. J.W. von Goethe, „Der Zauberlehrling“: „Die ich rief, die Geister werd ich nun nicht los.“ (1797)
(„The spirits that I summoned / I now cannot rid myself of again“)

RFCs Trivial File Transfer Protocol

Request **F**or **C**omments:

- RFC 783 (First Revision, 1981)
- RFC 1123 (1989): *Sorcerer's Apprentice Syndrome*[3]
- RFC 1350 (Second Revision, 1992)
- RFC 2347 (TFTP Option Extension, 1998)
- RFC 2348 (TFTP Blocksize Option, 1998)
- RFC 2349 (TFTP Timeout Interval and Transfer Size Options, 1998)
- RFC 7440 (TFTP Windowsize Option, 2015)

[3] Cf. J.W. von Goethe, „Der Zauberlehrling“: „Die ich rief, die Geister werd ich nun nicht los.“ (1797)
(„The spirits that I summoned / I now cannot rid myself of again“)

RFCs Trivial File Transfer Protocol

Request **F**or **C**omments:

- RFC 783 (First Revision, 1981)
- RFC 1123 (1989): *Sorcerer's Apprentice Syndrome*[3]
- RFC 1350 (Second Revision, 1992)
- RFC 2347 (TFTP Option Extension, 1998)
- RFC 2348 (TFTP Blocksize Option, 1998)
- RFC 2349 (TFTP Timeout Interval and Transfer Size Options, 1998)
- RFC 7440 (TFTP Windowsize Option, 2015)

[3] Cf. J.W. von Goethe, „Der Zauberlehrling“: „Die ich rief, die Geister werd ich nun nicht los.“ (1797)
(„The spirits that I summoned / I now cannot rid myself of again“)

RFCs Trivial File Transfer Protocol

Request **F**or **C**omments:

- RFC 783 (First Revision, 1981)
- RFC 1123 (1989): *Sorcerer's Apprentice Syndrome*[3]
- RFC 1350 (Second Revision, 1992)
- RFC 2347 (TFTP Option Extension, 1998)
- RFC 2348 (TFTP Blocksize Option, 1998)
- RFC 2349 (TFTP Timeout Interval and Transfer Size Options, 1998)
- RFC 7440 (TFTP Windowsize Option, 2015)

[3] Cf. J.W. von Goethe, „Der Zauberlehrling“: „Die ich rief, die Geister werd ich nun nicht los.“ (1797)
(„The spirits that I summoned / I now cannot rid myself of again“)

RFCs Trivial File Transfer Protocol

Request **F**or **C**omments:

- RFC 783 (First Revision, 1981)
- RFC 1123 (1989): *Sorcerer's Apprentice Syndrome*[3]
- RFC 1350 (Second Revision, 1992)
- RFC 2347 (TFTP Option Extension, 1998)
- RFC 2348 (TFTP Blocksize Option, 1998)
- RFC 2349 (TFTP Timeout Interval and Transfer Size Options, 1998)
- RFC 7440 (TFTP Windowsize Option, 2015)

[3] Cf. J.W. von Goethe, „Der Zauberlehrling“: „Die ich rief, die Geister werd ich nun nicht los.“ (1797)
(„The spirits that I summoned / I now cannot rid myself of again“)

RFCs Trivial File Transfer Protocol

Request **F**or **C**omments:

- RFC 783 (First Revision, 1981)
- RFC 1123 (1989): *Sorcerer's Apprentice Syndrome*[3]
- RFC 1350 (Second Revision, 1992)
- RFC 2347 (TFTP Option Extension, 1998)
- RFC 2348 (TFTP Blocksize Option, 1998)
- RFC 2349 (TFTP Timeout Interval and Transfer Size Options, 1998)
- RFC 7440 (TFTP Windowsize Option, 2015)

[3] Cf. J.W. von Goethe, „Der Zauberlehrling“: „Die ich rief, die Geister werd ich nun nicht los.“ (1797)
(„The spirits that I summoned / I now cannot rid myself of again“)

RFC 2347: TFTP Option Extension (1998)

Der Client kann dem Server gewisse Parameter des Transfers vorschlagen/übermitteln:

- `tsize` → transfer size
- `blksize` → block size
- `timeout` → timeout :-)
- `windowsize` → Anzahl der „am Stück“ gesendeten Datenblöcke

Der Server kann diese dann bestätigen (OACK) oder bei den „defaults“ bleiben.

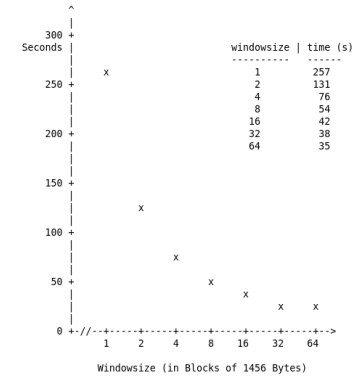
```
Read Request, File: d-i/n-a/menu.ipxe, Transfer type: octet, tsize=0, timeout=1, blksize=1434, windowsize=8
Option Acknowledgement, tsize=3593, timeout=1, blksize=1434, windowsize=8
Acknowledgement, Block: 0
Data Packet, Block: 1
Data Packet, Block: 2
Data Packet, Block: 3 (last)
Acknowledgement, Block: 3
```

TFTP Windowsize Option

RFC 7440 (2015)

```
[ DRCV ]      <---traffic--->      [ DSND ]
  ACK#      ->              <-   Data Block#   window block#
                    ...
                    <-           |DB n+01|          1
                    <-           |DB n+02|          2
                    <-           |DB n+03|          3
                    <-           |DB n+04|          4
|ACK n+04|          ->
                    <-           |DB n+05|          1
               Error |<-         |DB n+06|          2
                    <-           |DB n+07|          3
|ACK n+05|          ->
                    <-           |DB n+06|          1
                    <-           |DB n+07|          2
                    <-           |DB n+08|          3
                    <-           |DB n+09|          4
|ACK n+09|          ->
                    <-           |DB n+10|          1
               Error |<-         |DB n+11|          2
                    <-           |DB n+12|          3
|ACK n+10|          ->| Error
                    <-           |DB n+13|          4
                                - timeout -
                    <-           |DB n+10|          1
                    <-           |DB n+11|          2
                    <-           |DB n+12|          3
                    <-           |DB n+13|          4
|ACK n+13|          ->
                    ...
```

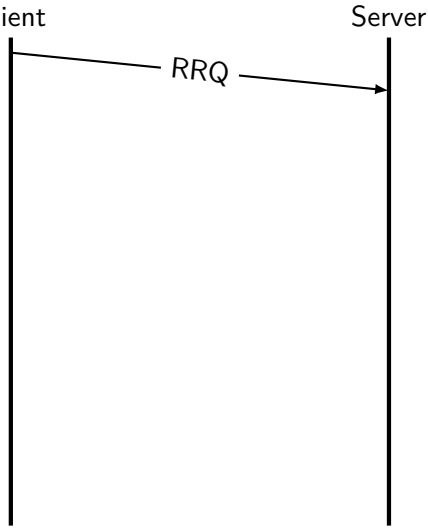
Performance tests were run on the prototype implementation using a variety of windowsizes and a fixed blocksize of 1456 bytes. The tests were run on a lightly loaded Gigabit Ethernet, between two Toshiba Tecra Core 2 Duo 2.2 Ghz laptops, in "octet" mode, transferring a 180 MByte file.



windowsize	time (s)
1	257
2	131
4	76
8	54
16	42
32	38
64	35

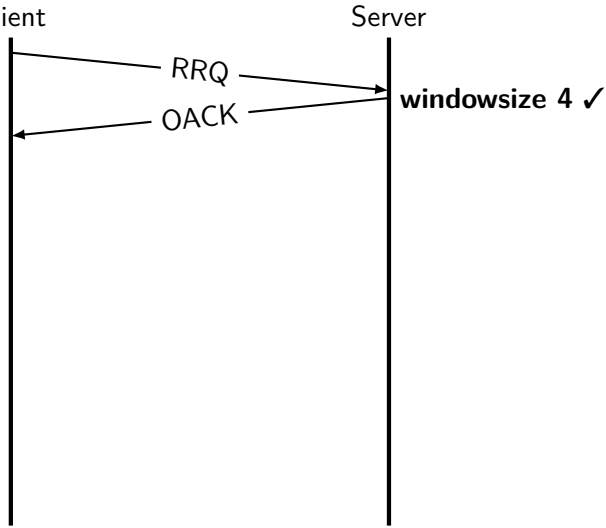
TFTP Windowsize Option

Read Request: Dateiname,
Option: **windowsize 4**

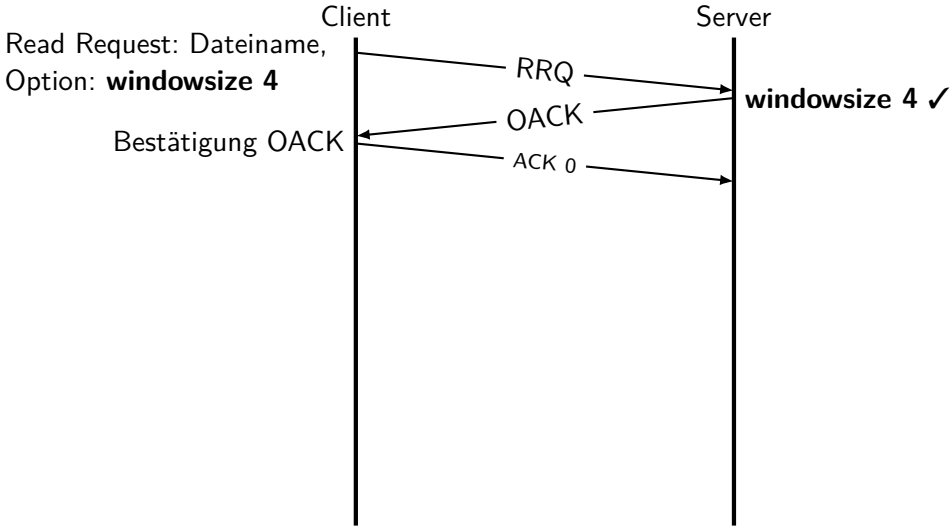


TFTP Windowsize Option

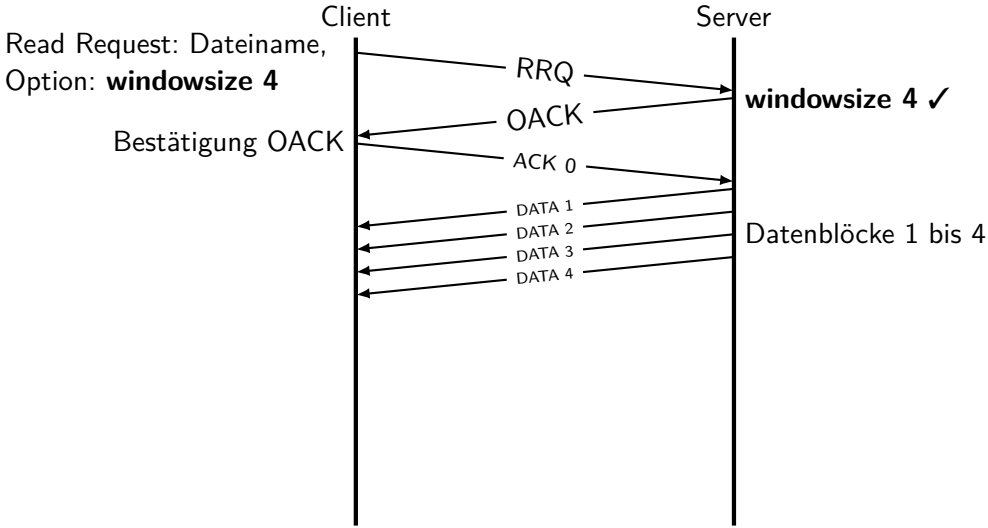
Read Request: Dateiname,
Option: **windowsize 4**



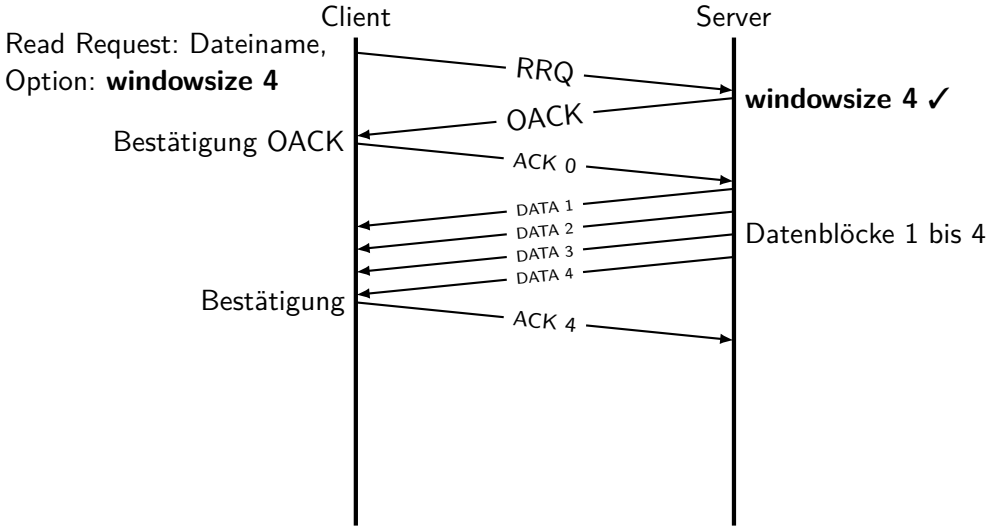
TFTP Windowsize Option



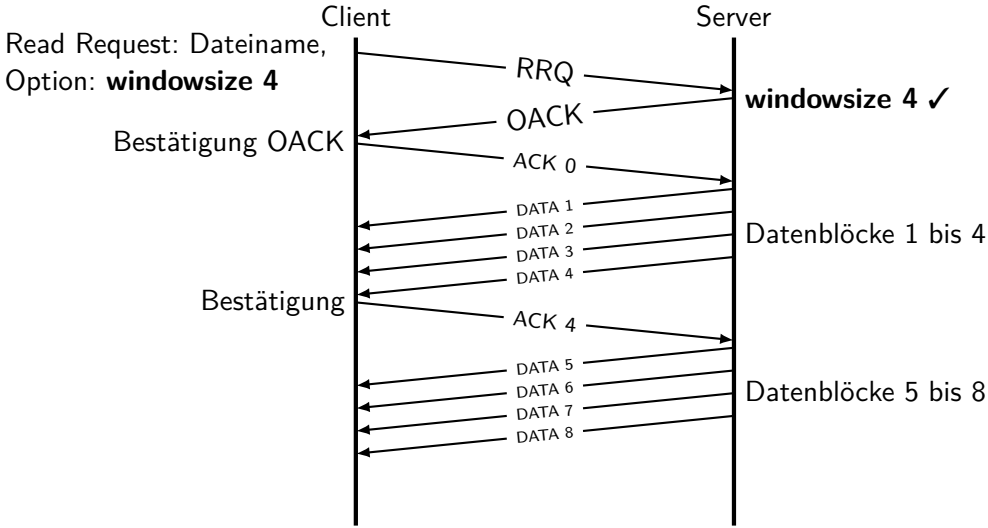
TFTP Windowsize Option



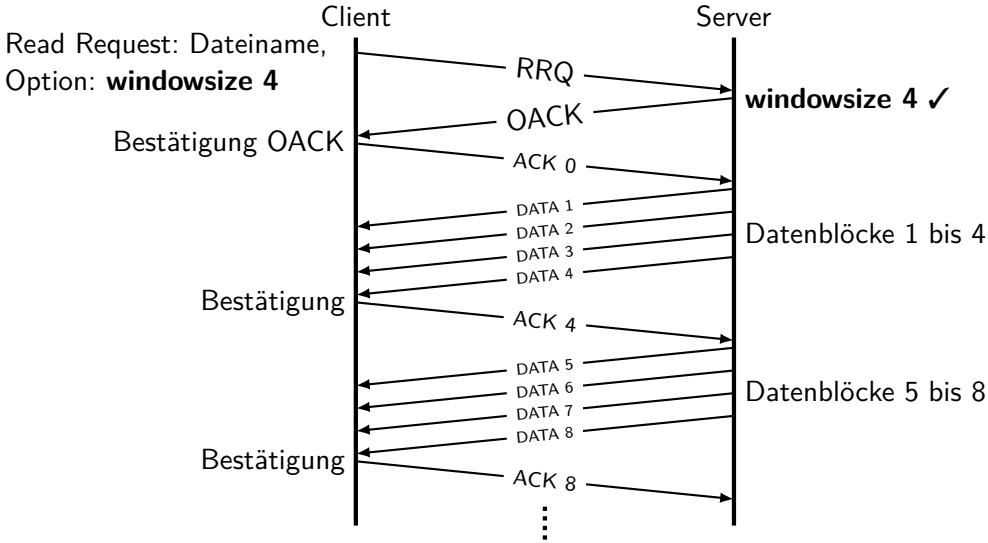
TFTP Windowsize Option



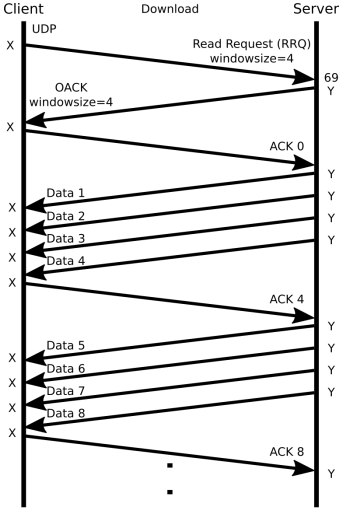
TFTP Windowsize Option



TFTP Windowsize Option



TFTP Windowsize Option



No.	Time	Source	Source Port	Destination	Destination Port	Protocol	Length	Info
1	0.000000000	::1	57394	::1	69	TFTP	101	Read Request, File: d-i/n-a/menu.ipx
2	0.019275714	::1	60556	::1	57394	TFTP	77	Option Acknowledgement, windowsize=4
3	0.019335292	::1	57394	::1	60556	TFTP	66	Acknowledgement, Block: 0
4	0.019597917	::1	60556	::1	57394	TFTP	578	Data Packet, Block: 1
5	0.019614407	::1	60556	::1	57394	TFTP	578	Data Packet, Block: 2
6	0.019622339	::1	60556	::1	57394	TFTP	578	Data Packet, Block: 3
7	0.019629606	::1	60556	::1	57394	TFTP	578	Data Packet, Block: 4
8	0.019651267	::1	57394	::1	60556	TFTP	66	Acknowledgement, Block: 4
9	0.019692844	::1	60556	::1	57394	TFTP	578	Data Packet, Block: 5
10	0.019707647	::1	60556	::1	57394	TFTP	578	Data Packet, Block: 6
11	0.019715360	::1	60556	::1	57394	TFTP	578	Data Packet, Block: 7
12	0.019724429	::1	60556	::1	57394	TFTP	75	Data Packet, Block: 8 (last)
13	0.019743221	::1	57394	::1	60556	TFTP	66	Acknowledgement, Block: 8
64	97.686022677	::1	54834	::1	69	TFTP	105	Read Request, File: d-i/n-a/grub/grub
65	97.703060307	::1	57246	::1	54834	TFTP	77	Option Acknowledgement, windowsize=4
66	97.703115588	::1	54834	::1	57246	TFTP	66	Acknowledgement, Block: 0
67	97.703368319	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 1
68	97.703383131	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 2
69	97.703390267	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 3
70	97.703396619	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 4
71	97.703415088	::1	54834	::1	57246	TFTP	66	Acknowledgement, Block: 4
72	97.703451215	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 5
73	97.703471905	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 6
74	97.703487414	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 7
75	97.703493439	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 8
76	97.703511357	::1	54834	::1	57246	TFTP	66	Acknowledgement, Block: 8
77	97.703540540	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 9
78	97.703558124	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 10
79	97.703563936	::1	57246	::1	54834	TFTP	578	Data Packet, Block: 11
80	97.703579999	::1	57246	::1	54834	TFTP	173	Data Packet, Block: 12 (last)
81	97.703646128	::1	54834	::1	57246	TFTP	66	Acknowledgement, Block: 12

```
▸ Frame 1: 101 bytes on wire (808 bits), 101 bytes captured (808 bits) on interface lo, id 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▸ Internet Protocol Version 6, Src: ::1, Dst: ::1
▸ User Datagram Protocol, Src Port: 57394, Dst Port: 69
▾ Trivial File Transfer Protocol
    Opcode: Read Request (1)
    Source File: d-i/n-a/menu.ipxe
    Type: octet
  ▾ Option: windowsize = 4
      Option name: windowsize
      Option value: 4
```

windowsize option (RFC 7440, 2015)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	::1	::1	TFTP	101	Read Request, File: d-i/n-a/menu.ipxe, Transfer type: octet, windowsize=3
2	0.016803443	::1	::1	TFTP	77	Option Acknowledgement, windowsize=3
3	0.016852472	::1	::1	TFTP	66	Acknowledgement, Block: 0
4	0.016898404	::1	::1	TFTP	578	Data Packet, Block: 1
5	0.016914065	::1	::1	TFTP	578	Data Packet, Block: 2
6	0.016922623	::1	::1	TFTP	578	Data Packet, Block: 3
7	0.016935795	::1	::1	TFTP	66	Acknowledgement, Block: 3
8	0.016949373	::1	::1	TFTP	578	Data Packet, Block: 4
9	0.016960107	::1	::1	TFTP	578	Data Packet, Block: 5
10	0.016969592	::1	::1	TFTP	578	Data Packet, Block: 6
11	0.016980340	::1	::1	TFTP	66	Acknowledgement, Block: 6
12	0.017003561	::1	::1	TFTP	578	Data Packet, Block: 7
13	0.017017159	::1	::1	TFTP	75	Data Packet, Block: 8 (last)
14	0.017028243	::1	::1	TFTP	66	Acknowledgement, Block: 8
15	14.047689367	::1	::1	TFTP	101	Read Request, File: d-i/n-a/menu.ipxe, Transfer type: octet, windowsize=5
16	14.048077016	::1	::1	TFTP	77	Option Acknowledgement, windowsize=5
17	14.048132049	::1	::1	TFTP	66	Acknowledgement, Block: 0
18	14.048231740	::1	::1	TFTP	578	Data Packet, Block: 1
19	14.048271914	::1	::1	TFTP	578	Data Packet, Block: 2
20	14.048300157	::1	::1	TFTP	578	Data Packet, Block: 3
21	14.048316349	::1	::1	TFTP	578	Data Packet, Block: 4
22	14.048331901	::1	::1	TFTP	578	Data Packet, Block: 5
23	14.048343989	::1	::1	TFTP	66	Acknowledgement, Block: 5
24	14.048358115	::1	::1	TFTP	578	Data Packet, Block: 6
25	14.048374024	::1	::1	TFTP	578	Data Packet, Block: 7
26	14.048391615	::1	::1	TFTP	75	Data Packet, Block: 8 (last)
27	14.048402887	::1	::1	TFTP	66	Acknowledgement, Block: 8

iPXE hat windowsize implementiert:

```
Read Request, File: /d-i/n-a/menu.ipxe, Transfer type: octet, tsize=0, blksize=1468, windowsize=4
```

Optimierungen und Tests

- Wie kann man Packetverlusten oder -Verzögerung begegnen?
- Mit welchem zeitlichen Abstand sollte man die Datagramme innerhalb eines Fensters versenden?
- Congestion control?
- RFC 2090 (TFTP Multicast Option, 1997, experimental)

Network Emulation[4]:

`https://wiki.linuxfoundation.org/networking/netem`

netem provides Network Emulation functionality for testing protocols by emulating the properties of wide area networks. The current version emulates variable delay, loss, duplication and re-ordering.

Was ist ein realistisches Scenario ...

[4] https://en.wikipedia.org/wiki/Network_emulation

Optimierungen und Tests

- Wie kann man Packetverlusten oder -Verzögerung begegnen?
- Mit welchem zeitlichen Abstand sollte man die Datagramme innerhalb eines Fensters versenden?
- Congestion control?
- RFC 2090 (TFTP Multicast Option, 1997, experimental)

Network Emulation[4]:

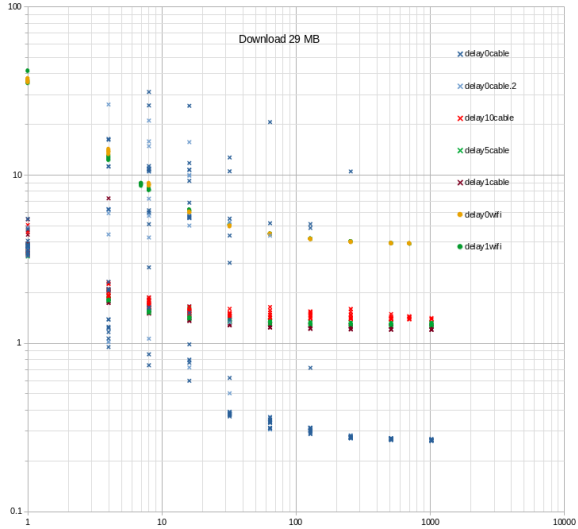
`https://wiki.linuxfoundation.org/networking/netem`

netem provides Network Emulation functionality for testing protocols by emulating the properties of wide area networks. The current version emulates variable delay, loss, duplication and re-ordering.

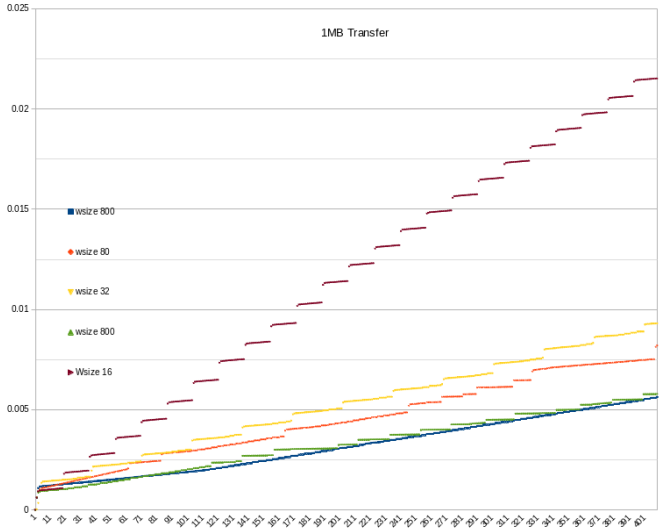
Was ist ein realistisches Scenario ...

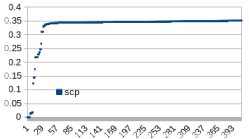
[4] https://en.wikipedia.org/wiki/Network_emulation

TFTP Windowsize Option



TFTP Windowsize Option





Zusammenfassung und Rückblick

1. Wie kann man Daten in einem Netzwerk übertragen?
2. Das TFT Protocol
3. TFTP Anwendung
4. Der Zauberlehrling und weitere Defizite
5. RFCs: Requests for Comments
6. TFTP Option Extension
7. Optimierungen und Tests

Keep the sourcerer alive!

Zusammenfassung und Rückblick

1. Wie kann man Daten in einem Netzwerk übertragen?
2. Das TFT Protocol
3. TFTP Anwendung
4. Der Zauberlehrling und weitere Defizite
5. RFCs: Requests for Comments
6. TFTP Option Extension
7. Optimierungen und Tests

Keep the sourcerer alive!

Vielen Dank für Interesse und Aufmerksamkeit!