Implementing Quality-of-Service using SMB2 Crediting

Christian Ambach

IBM / Samba Team

Why Quality-of-Service (QoS)?

- From Wikipedia: *Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.*
- Many users would like to establish QoS for their central fileservers to make sure that privileged clients will not be starved out by not-so-important clients
- The SMB2 protocol introduced the crediting mechanism that should be able to help with this requirement

The role of credits in SMB2

- Client requests credits in each request and consumes them by sending requests
  - One credit gives the client to permission to send a request for up to 64 KiB of data
  - Credits can be combined to issue a Multi-Credit request of up to 1 MiB instead of multiple small requests
  - Server does not need to grant the amount of credits the client has requested
    - But it must make sure that client never runs out of credits
- The more credits a client possesses, the more requests it can have outstanding at the same time
- The server can send interim responses to requests and grant credits in them, enabling the client to send more requests while others are still in-flight
- Thus by controlling the amount of credits a client has, it should be possible to control the load the client can drive against the server

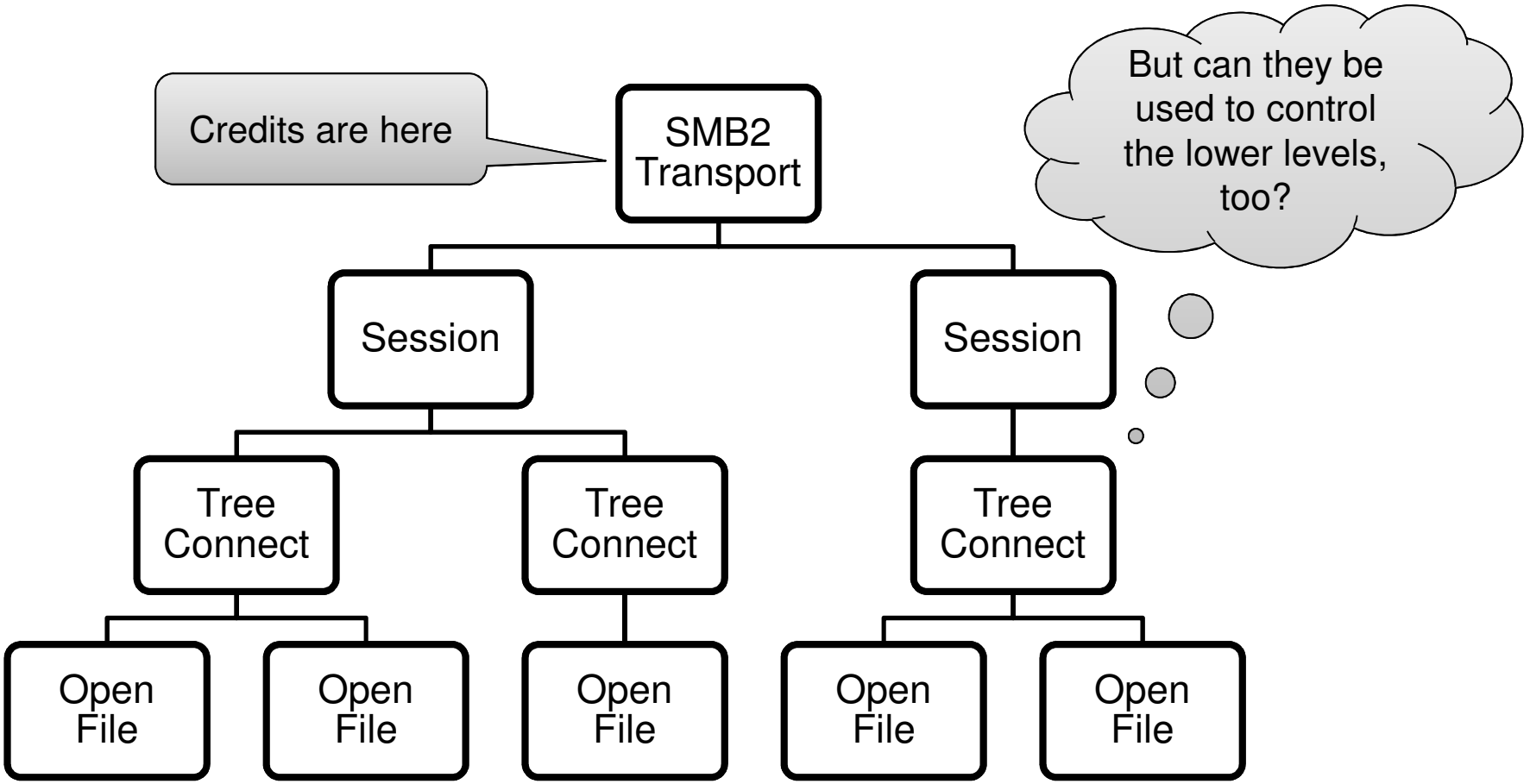
Potential requirements

- Limit bandwidth a client can drive
  - Prefer application servers over office clients
- Limit bandwidth a user can drive
  - Prefer employees over students
- Limit bandwidth for a share
  - Prefer vital data over not-so-important data
- Limit bandwidth for a file
  - Prefer business critical files over temporary or leisure files

Additional ideas

- Use credits to prevent overdriving the storage subsystem of a server
  - limit the total amount of credits granted to all clients
  - Establish a feedback loop between file system / storage to reduce amount of credits when saturation point is near

SMB2 datamodel



Test setup

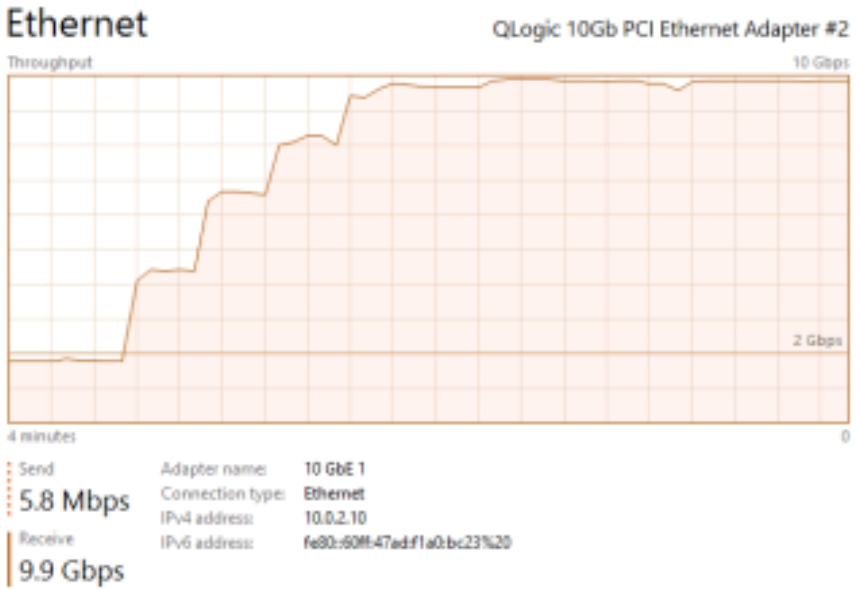
- Server
  - IBM x3650 M3, 24 cores, 96 MB RAM
  - RHEL 6.3
  - Samba master code
- Clients
  - Two IBM x3650 M2, 16 cores, 64 MB
  - Windows 2012 Server RC
- Network
  - 10 GbE QLogic CNA cards

Settings / Test application

- Protocol dialect: SMB2.1 with Large MTU
- Turned off oplocks to prevent clients from caching
- Turned on async I/O in Samba
  - enables interim responses
- Wrote a small application that sequentially reads file contents in 1 MiB chunks

Baseline testing

- Server serves data from RAM disk
- With two instances of application, line-speed can be achieved

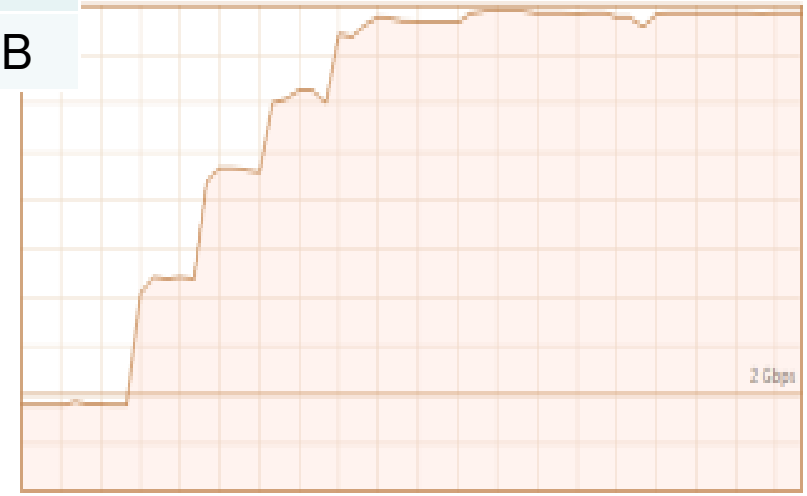


Limiting client bandwidth

- Testing with low credit values showed that clients need at least 64 credits
  - otherwise strange things will happen, e.g. Explorer hangs when listing directory
- But granting up to 64 credits means that the client can already drive 6.4 Gb/s of bandwidth with zero filesystem latency
- Further limitation possible by not sending interim responses so client has to wait for credits to be granted in the final response

Measurements

Max Credits	Throughput [Gb/s]	Readsizes
8192	9.9	1 MiB
512	9.9	1 MiB
256	9.9	256 KiB – 1 MiB
128	9.8	256-512 KiB
96	8.4	64-256 KiB
64	6.4	64-128-192 KiB

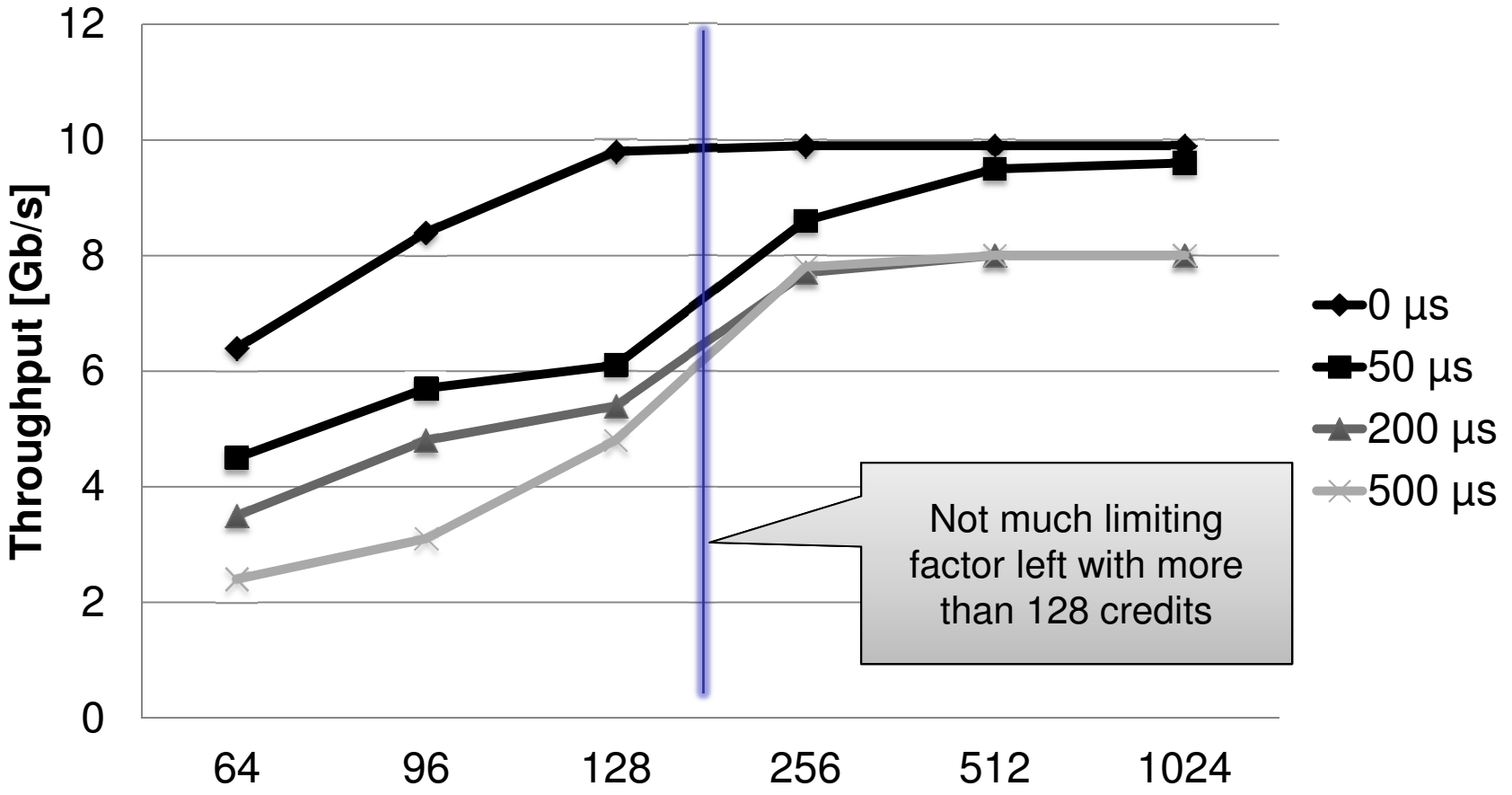


Measurements (continued)

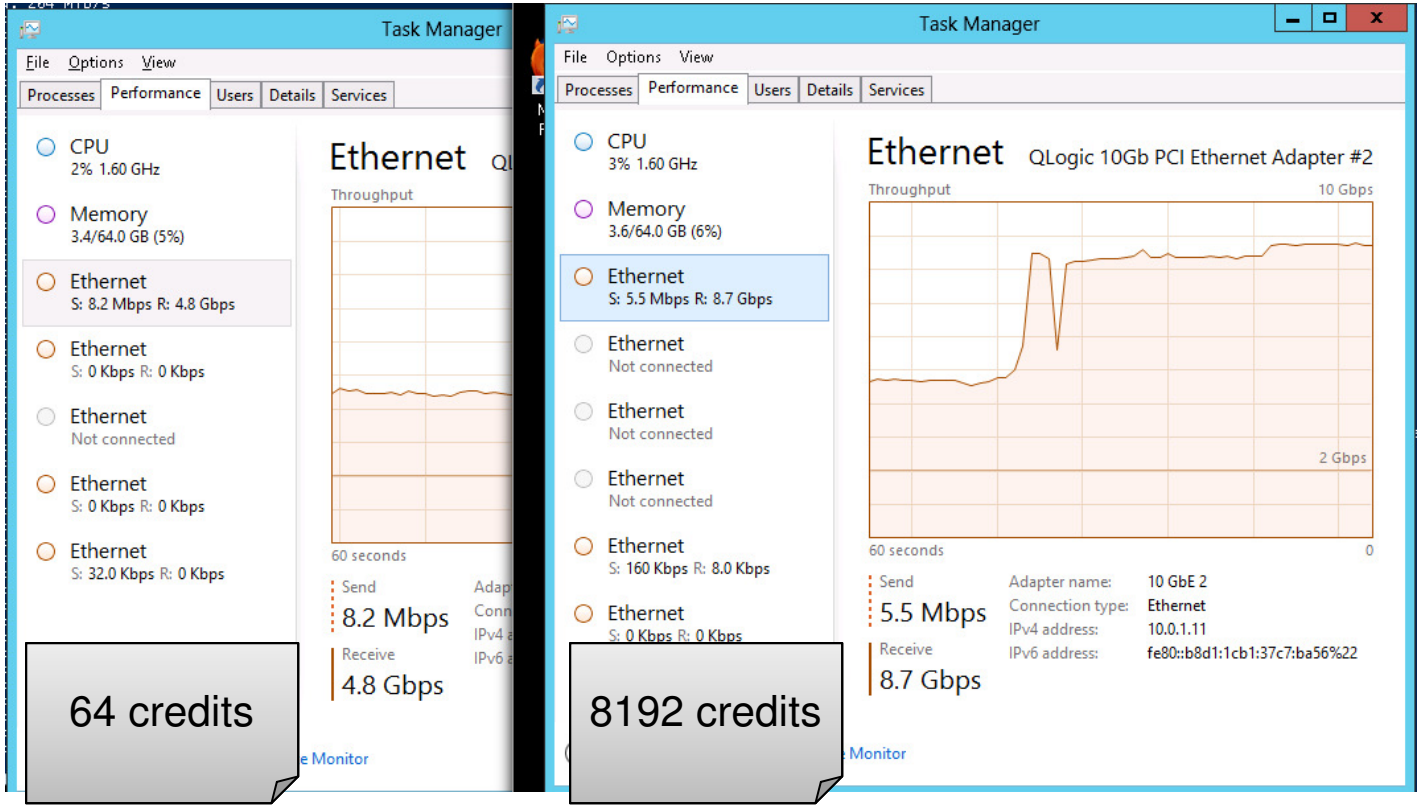
Filesystem delay 50 µs	
Credits	Throughput [Gb/s]
64	4.5
96	5.7
128	6.1
256	8.6
512	9.5
1024-8192	9.6

Filesystem delay 200 µs	
Credits	Throughput [Gb/s]
64	3.5
96	4.8
128	5.4
256	7.7
512	8.0
1024-8192	8.0

Throughput by credits and latency



Example results



Limiting bandwidth by session

- Although multiple sessions on the same transport are possible, they are rarely used
- Not possible via regular user commands

- Modern Terminal Servers do not seem to use single transport for all users any more (as Windows 2000 did), they open a transport per user
- Linux multi-user mount as only exploiter?

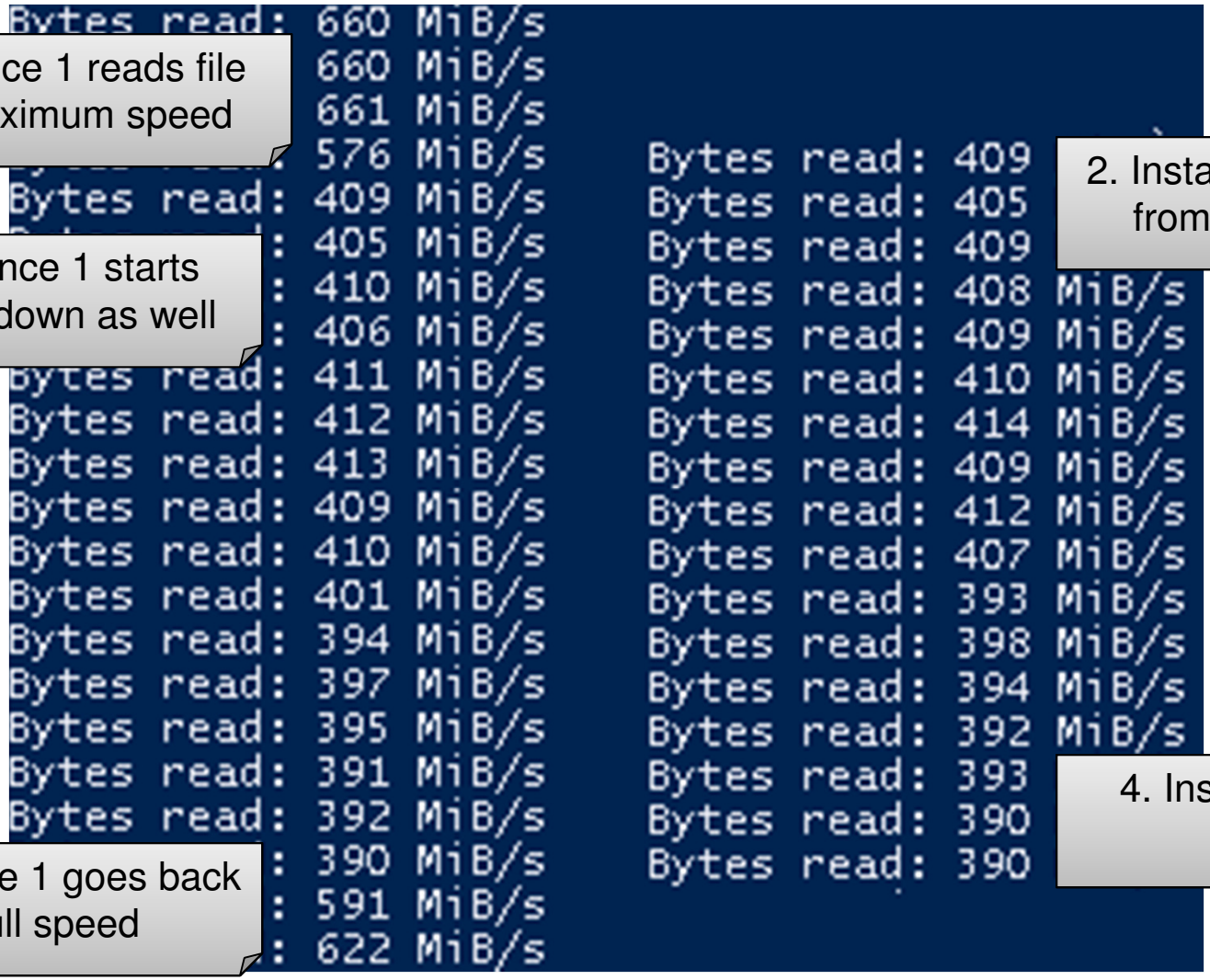
Limiting bandwidth by session

- If every user has his own transport, bandwidth for a user can be controlled by limiting the amount of credits on the transport level
- This is essentially the same scenario as limiting on transport level and results are similar

Limiting bandwidth by tree

- Approach
  - only grant a few credits for operations on a share configured to be „slow“
  - Do not send out interim responses
- Result
  - access to other shares also slows down

Limiting bandwidth by tree



1. Instance 1 reads file with maximum speed

2. Instance 2 reads file from „slow“ share

3. Instance 1 starts slowing down as well

4. Instance 2 stops reading

5. Instance 1 goes back to full speed

Limit bandwidth by file

- Approach as for share
  - Give less credits than client had asked for for operations on files configured to be „slower“
- Result
  - With the experiences made with limiting by tree, result is not surprising: it does not work

Summary

- Limiting client bandwidth is possible by controlling the amount of credits granted to the client
  - Starting point is limited due to 64 credit minimum
- Controlling bandwidth for a user is possible (and can be done with the same mechanism used above)
- Working on tree and file level is not really possible and leads to unexpected/undesired results

Where to find the code

- My testing code can be found in the „sdc“ branch my private git repository on git.samba.org
  - git://git.samba.org/ambi/samba.git
  - https://gitweb.samba.org/?p=ambi/samba.git

Q&A