Win7 UDP Performance Internals

End to End Gigabit Ethernet IPv4 Networking

TINE multicast, unicast

Application: live video transport

Stefan Weisse
DESY IT/Controls
Zeuthen

TINE Core Meeting, December 2016

Win7 UDP Performance Internals
Overview

> What is possible? (MB/s)

> How one can achieve that?

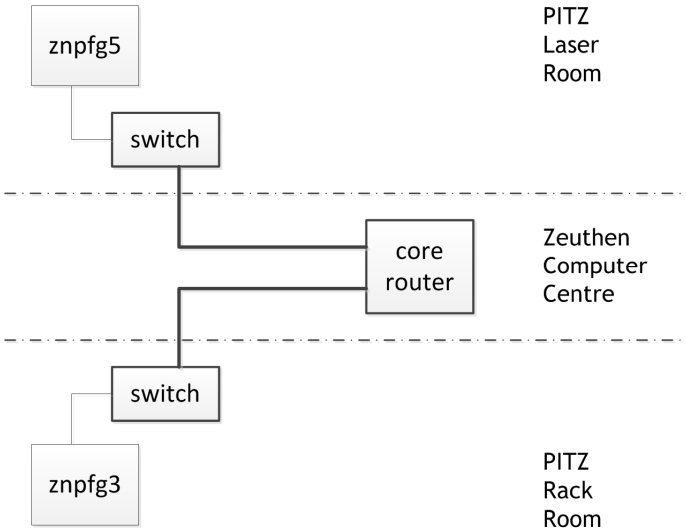
> What has shown to fail in practice? (how one can not achieve that)

> Reproduce cases on Testbed with two hosts:

- host znpfg5: sender with a dedicated GigE camera
- host znpfg3: receiver

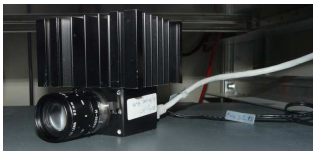
- both hosts monitored via Microsoft Remote Desktop Connection (RDP)!

Win7 UDP Performance Internals
Network Setup



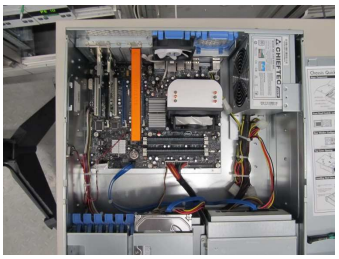
Win7 UDP Performance Internals
Host Details

- Sender's system specs:
  - Hostname: znpfg5
  - CPU: Intel Core 2 Duo E8400: 3.0 GHz, 2 cores
  - Motherboard: Intel D**X38**BT
  - 8 GB RAM (Non-ECC)
  - Windows 7 64-bit Enterprise (DESY: Firewall and Virus Scanner enabled)
  - Intel 82566DC-2 Gigabit Network Connection (on-board): DESY LAN
  - Intel Gigabit CT Desktop Adapter (PCIe x1) : Camera Input (via 1:1 Cable to Camera)



Win7 UDP Performance Internals
Host Details

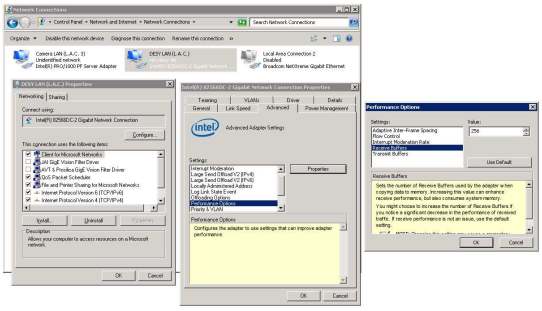
- Receiver's system specs:
  - Hostname: znpfg3
  - CPU: Intel Core 2 Duo E8400: 3.0 GHz, 2 cores
  - Motherboard: Intel D**X38**BT
  - 4 GB RAM (Non-ECC)
  - Windows 7 64-bit Enterprise (DESY: Firewall and Virus Scanner enabled)
  - Intel 82566DC-2 Gigabit Network Connection (on-board): DESY LAN



Common off-the-shelf (COTS)
Rackmount PC

Win7 UDP Performance Internals
Host Details

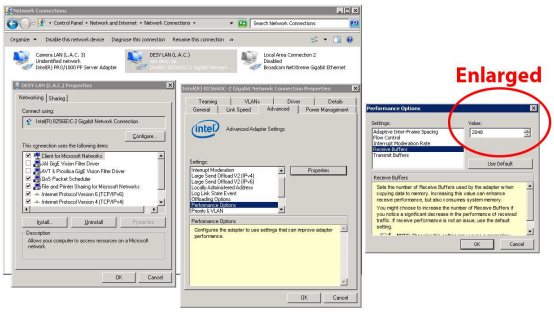
- Intel Network Interface Card (NIC) Tuning
  - Receive Buffers: 256 -> 2048 (max)



  - When dealing with the high throughput UDP traffic a GigE camera can provide, enlarging receive buffers has shown to improve transport reliability (few dropped packets or dropped frames -> no dropped packets/dropped frames)
  - According to a test which I've done it's not clear whether increasing number of receive buffers has a positive effect on transfer stability as shown in this presentation
  - No negative side effects have been observed

Win7 UDP Performance Internals
Host Details

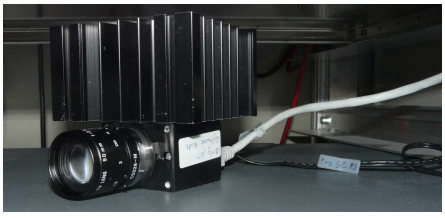
- Intel Network Interface Card (NIC) Tuning
  - Receive Buffers: 256 -> 2048 (max)



  - When dealing with the high throughput UDP traffic a GigE camera can provide, enlarging receive buffers has shown to improve transport reliability (few dropped packets or dropped frames -> no dropped packets/dropped frames)
  - According to a test which I've done it's not clear whether increasing number of receive buffers has a positive effect on transfer stability as shown in this presentation
  - No negative side effects have been observed

Win7 UDP Performance Internals
Image Source Details

- Prosilica GC1350M
  - 1360x1024 pixel
  - Monochrome
  - 12 bits per pixel (as 2 bytes per pixel)
  - Up to 20 Hz full frame read out (20.065 Hz)



  - Total possible (net) bandwidth: 53.13 MB/s (55.705.600 bytes per second) (1360 x 1024 x 2 x 20)

As comparison: Highest in practice observed Gigabit Ethernet bandwidth (with TCP): 109 MB/s (NetIO benchmark v1.31)

Win7 UDP Performance Internals
Goal

- A transfer is very stable if:
  - Less than 1 per 1000 frames is dropped, for more than 10 minutes

Win7 UDP Performance Internals
Settings which are not modified across tests runs

- Camera Prosilica GC1350M (connected to znpfg5)
  - delivering 1360x1024x12 bit -> 2.785.280 bytes / image
- TINE server (on znpfg5)
  - Lazy scheduling
  - CycleDelay=0
  - SGP_ProsilicaGigE.exe: C/C++ 32-bit Windows executable, using tine32.dll*
- TINE client (on znpfg3)
  - **Multicast,** 2000ms polling interval
  - socket receive buffer: 4.194.304 bytes
  - TineFrameTransferTestV3.exe: C/C++ 32-bit Windows executable, using tine32.dll*

*: tine32.dll v4.5.10 build 5248 (Jun 16, 2016)

Win7 UDP Performance Internals
Bandwidth Demands

- 1 video frame 1360x1024 pixel, 2 bytes per pixel
  -> 2.785.280 bytes (2.66 MB)

- UDP packet size: 1024 bytes
  - packets/s: 2720 @ 1 Hz, 27200 @ 10 Hz, 54400 @ 20 Hz
- UDP packet size: 1472 bytes
  - packets/s: 1893 @ 1 Hz, 18930 @ 10 Hz, 37860 @ 20 Hz
- UDP packet size: 9000 bytes
  - packets/s: 310 @ 1 Hz, 3100 @ 10 Hz, 6200 @ 20 Hz
- UDP packet size: 16384 bytes
  - packets/s: 170 @ 1 Hz, 1700 @ 10 Hz, 3400 @ 20 Hz

Rough Estimate!

- Not taking into account TINE UDP packet header size!

Win7 UDP Performance Internals
Bandwidth Demands

- High number of UDP packets require high effort
  - for each TINE packet, a TINE header is appended, lowering the possible bandwidth (but not much)
  - a lot of code in TINE is run for each packet, same is true on a lower level for operating system, driver of NIC
  - CPU is heavily utilized for a huge number of packets per second

Win7 UDP Performance Internals
IP fragmentation and reassembly?

- Common operating systems (at least non-embedded Linux, Windows) support IP fragmentation and reassembly
- Idea: set TINE PacketMTU to a huge number, e.g. 16384, effectively lowering the number of TINE packets that need to be produced, send, received and decoded
- TCP/IP firmware across the path will split the big UDP packet into fragments small enough to be transported on the underlying layer, client will reassemble the fragments (if all fragments arrive the client)
- Drawback: Not all operating systems support IP fragmentation and reassembly
- In principle a promising idea, but in practice…

Win7 UDP Performance Internals
IP fragmentation and reassembly??

- Transfer specs
  - 2.66 MB/frame, 20 Hz
  - PacketMTU=16384 bytes, BurstLimit=3, SocketSendBufferKB=64
  - CycleDelay=0, MicroDelay=100
- First observation after starting the transfer:
  - CPU load is pretty low, transfer is stable at a frame rate which was not expected to be reached (20 Hz -> 53.1 MB/s)
- But after some time (usually minutes)….
  - Transfer starts to drop more and more frames per second
- And some time later…
  - on client, incoming traffic is still present, but no data reaches the client application
  - eventually, some time later, network load drops to almost zero (multicast traffic stops)

Win7 UDP Performance Internals
IP reassembly buffer pool exhaust: Unfreeze Multicast

- Stop transfer
- Observe Windows Task Manager, that network load is getting low
- Wait two minutes
- Transfer should be able to be restarted, and should run with little or no data losses
  - If not then wait five minutes

- The above recipe does not always bring back a very stable transfer, even though an improvement is always there
  - network weather conditions?

Win7 UDP Performance Internals
Screencast: transfer break, unfreeze hanging transfer

- Video: ip-reassembly-buffer-pool-exhaust-mcast-DenOfSrv-Jul12-2016.mp4 (3m15s, 7.1 MB)
  - no video frame reaches the client application
  - in Windows task manager (networking tab), one can see a large network utilization on server as well as client side
  - netstat*, on client, shows expected high number of fragments incoming (~41000/s)
  - netstat*, on client, does not show expected number of successful reassemblies (less than 50/s are shown, but more than 3400/s expected)
  - at 0m51s, the transfer is stopped by hand
  - one can see in Windows task manager (networking tab) that network load drops almost immediately to zero
  - at 1m50s, the transfer is restarted
  - at 2m28s, first frame is dropped
  - up to the end: frame drops increase, but do not reach fatal level (yet…)

*: netstat -s -p IP -t 1

Win7 UDP Performance Internals
IP reassembly failure

- IP reassembly is not stable enough at high data rates
- IP reassembly buffer exhaust?
  - Due to UDP fragments not reaching the client, other fragments of the same macropacket block buffer space in the IP reassembly pool, if IP reassembly is successful, buffer space is released quickly, available for new packets as they come in, but if some fragment is missing, other fragments of the same macropacket block space in the IP reassembly pool for seconds to minutes ([technet], [rfc2460], [rfc791])
- Reassembly fragment number only 16 bit?
  - reassembled macroblock checksum mismatch? [rfc4963]

[technet] http://blogs.technet.com/b/nettracer/archive/2010/08/10/3335600.aspx

[rfc2460] http://www.ietf.org/rfc/rfc2460.txt

[rfc791] http://www.ietf.org/rfc/rfc791.txt

[rfc4963] http://www.ietf.org/rfc/rfc4963.txt

Win7 UDP Performance Internals
IP reassembly buffer pool exhaust

- As far as I know
  - Limited possibilities to trace IP reassembly buffers (statistics)
  - No possibilities to tweak the pool (e.g. adjust timeout, enlarge buffer space)
- If the bandwidth demand is low, no issue has been observed
  - Multicast 3.66 MB/s (1600x1200x2 byte per pixel at 1 Hz) in Hamburg @ PETRA
- But if bandwidth demand is high, sooner or later degrading of transfer stability has been observed, even though it's not clearly reproducable
  - dependency on overall network load?
  - also dependency on client network load suspected
  - at 5 Hz (13.28 MB/s): no issue observed (running for more than two days)
  - at 7 Hz (18.59 MB/s): issue reproduced (took more than 20 hours until fatal)

Win7 UDP Performance Internals
Maximum fragment-free UDP Packet Size?

- If going high bandwidth, IP fragmentation and reassembly looks like no way to go right now…
- as a consequence, TINE needs to operate on small packet size, which puts high demand on CPU
  - modern desktop CPUs (Haswell, Skylake, especially 4+-core) are supposed to handle this, but 5 year old Core 2 Duo may not
- What's the maximum UDP packet size which is not fragmented? (no fragmentation and reassembly while retaining lowest possible number of packets per second)
  - in general, on LAN, 1472 bytes is a good guess

Win7 UDP Performance Internals
Maximum fragment-free UDP Packet Size

```
C:\Windows\System32>ping /?

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-4] [-6] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been deprecated
                   and has no effect on the type of service field in the IP Header).
    -r count       Record route for count hops (IPv4-only).
    -s count       Timestamp for count hops (IPv4-only).
    -j host-list   Loose source route along host-list (IPv4-only).
    -k host-list   Strict source route along host-list (IPv4-only).
    -w timeout     Timeout in milliseconds to wait for each reply.
    -R             Use routing header to test reverse route also (IPv6-only).
    -S srcaddr     Source address to use.
    -4             Force using IPv4.
    -6             Force using IPv6.
```

Win7 UDP Performance Internals
Maximum fragment-free UDP Packet Size

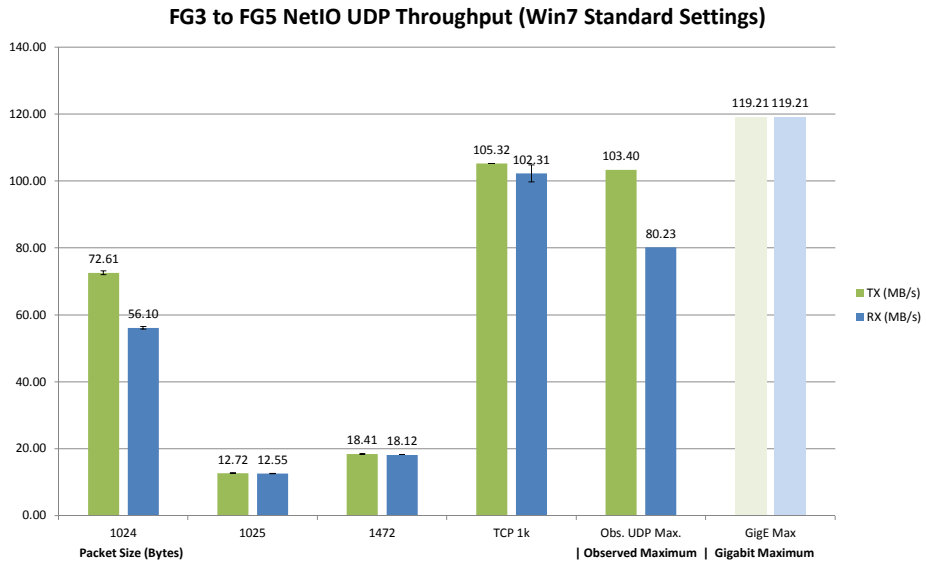
```
[znpfg3] C:\temp\sweisse\netio>ping -l 1472 -f znpfg5

Pinging znpfg5.ifh.de [141.34.30.215] with 1472 bytes of data:
Reply from 141.34.30.215: bytes=1472 time<1ms TTL=128
Reply from 141.34.30.215: bytes=1472 time<1ms TTL=128
Reply from 141.34.30.215: bytes=1472 time<1ms TTL=128

Ping statistics for 141.34.30.215:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
```

```
[znpfg3] C:\temp\sweisse\netio>ping -l 1473 -f znpfg5

Pinging znpfg5.ifh.de [141.34.30.215] with 1473 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.

Ping statistics for 141.34.30.215:
    Packets: Sent = 3, Received = 0, Lost = 3 (100% loss),
Control-C
^C
```


Win7 UDP Performance Internals
First non-fragmented transport: low max. bandwidth

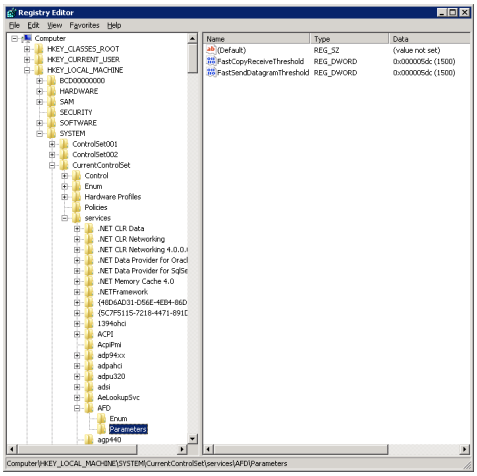
- Using maximum packet size without fragmentation (1472 bytes)
- Maximum achievable bandwidth is unexpected slow (17 MB/s)… why?

Win7 UDP Performance Internals
Out of the box: Poor UDP performance



Win7 UDP Performance Internals
Registry Settings

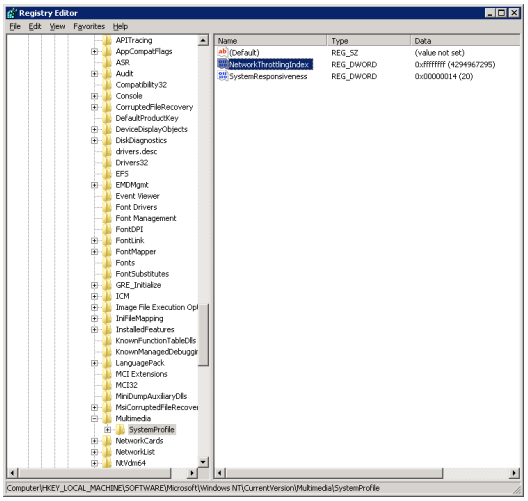
HKLM\SYSTEM\CurrentControlSet\services\AFD\Parameters



- FastCopyReceiveThreshold
  - REG_DWORD
  - Default: 1024 (0x400)
  - Recommended: 1500 (0x5dc)
  - little impact on receive assumed
- FastSendDatagramThreshold
  - REG_DWORD
  - Default: 1024 (0x400)
  - Recommended: 1500 (0x5dc)
  - **huge impact on send**

Win7 UDP Performance Internals
Registry Settings

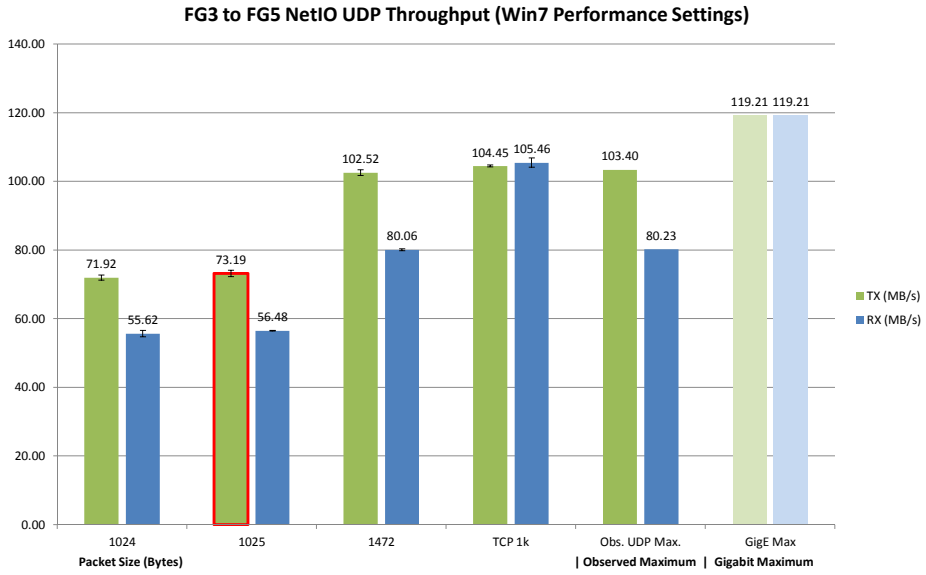
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Multimedia\SystemProfile



- NetworkThrottlingIndex
  - REG_DWORD
  - Default value: 0xa
  - Recommended: 0xffffffff (4294967295)
  - Impact unknown

In general: No negative side-effects observed.

Win7 UDP Performance Internals
Optimized UDP performance



Win7 UDP Performance Internals
Win 7 Standard Settings: What's possible

- Camera Framerate 11.0 Hz (29.22 MB/s)
- TINE Server: PacketMTU=1024, BurstLimit=30, MicroDelay=10, default SocketSendBufferSize=32 KB
- Result: less than 1 per 1000 frames dropped (runtime 15m19s: 10102 frames rcvd, 6 dropped)
- In comparison: 12.2 Hz (32.41 MB/s) results in 3 per 1000 frames dropped (13m47s: 10059 frames rcvd, 31 dropped)
- Main bottleneck assumed: CPU load (on server and on client around 50% on dual-core machine)

Win7 UDP Performance Internals
Win 7 Performance Settings: What's possible

- Camera Framerate 15.5 Hz (41.2 MB/s)
- TINE Server: PacketMTU=1472, BurstLimit=21, MicroDelay=10, default SocketSendBufferSize=32 KB
- Result: less than 1 per 1000 frames dropped (runtime 10m54s: 10119 frames rcvd, 9 dropped)
- In comparison:
  - 16.0 Hz (42.5 MB/s) results in 1.8 per 1000 frames dropped (10m49s: 10350 frames rcvd, 19 dropped)
  - 16.5 Hz (43.8 MB/s) results in 4 per 1000 frames dropped (10m21s: 10213 rcvd, 40 dropped)
- Main bottleneck assumed: CPU load (on server and on client around 50% on dual-core machine)

Win7 UDP Performance Internals
Win7 Performance vs. Standard (video transmission)

- Camera Framerate
  - Standard: 11.0 Hz (29.22 MB/s)
  - Performance: 15.5 Hz (41.2 MB/s)
- +11.95 MB/s
- Both Standard and Performance suffer from busy CPU (higher network bandwidth occupation on less busy CPU expected)

Win7 UDP Performance Internals
Summary

- Out of the box, UDP bandwidth of packet size >1024 bytes surprisingly low
- IP reassembly showed denial of service in real world environment
- In wide field: stick to Win7 standard settings, re-adjust TINE to PacketMTU=1024, keep transfer rate low (20 MB/s, maybe 25 MB/s)
- In lab: consider windows 7 performance settings, use TINE default PacketMTU=1472, a transfer rate of 40 MB/s is possible
- MicroDelay can be used as a handbrake to saturate possible TINE bandwidth before network level issues occur
- TINE Send buffer should be kept at default size (32 KB)
- on receive side, enlarging of socket buffer is highly recommended (rule of thumb: 1 video frame should fit in receive socket buffer)

Win7 UDP Performance Internals

Thank you for your attention!

Question? Comments?

Remarks?