TRANSFERRING EARTH OBSERVATION DATA AROUND THE GLOBE

Christos Argyropoulos
cargious@noc.grnet.gr

Copernicus, Europe's eyes on Earth

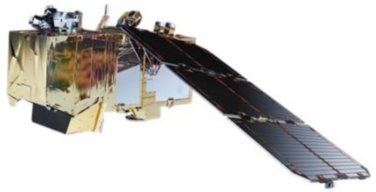
- The European Union's Earth Observation Programme
- Looking at our planet and its environment for the benefit of all European citizens
- It offers information services based on satellite Earth Observation and in situ (non-space) data.

grnet

Copernicus, Europe's eyes on Earth

- Sentinel-1
  - All weather, day-n-night
  - Radar imaging for land & ocean services
  - Able to see through clouds and rain
  - Main instrument: C-band synthetic aperture radar

Copernicus, Europe's eyes on Earth



- Sentinel-2
  - Medium resolution multispectral optical satellite for the observation of land, vegetation and water
  - 13 spectral bands with 10, 20 or 60m resolution and 290km swath width
  - Main instrument: Multispectral instrument (MSI)

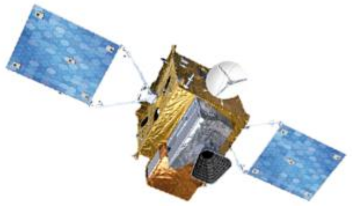
Copernicus, Europe's eyes on Earth

- Sentinel-3
  - Measures sea-surface topography (300m resolution)
  - Sea & Land surface temperature (0.3K accuracy)
  - Colour with a resolution of 1km
  - Measures water vapour, cloud water content & thermal radiation emitted by the Earth
  - Global sea surface temperatures (0.3K accuracy)
  - Main Instruments:
    - Ocean and Land Colour Instrument (OLCI)
    - Sea and Land Surface Temperature Radiometer (SLSTR)
    - SAR Radar ALtimeter (SRAL)

Copernicus, Europe's eyes on Earth

- Sentinel-5P
  - Observation of key atmospheric constituents including ozone, nitrogen dioxide, sulphur dioxide
  - Improves climate models and weather forecasts
  - Provides data continuity during the five-year gap between the retirement of Envisat and the launch of Sentinel-5
  - Main Instrument: TROPOspheric Monitoring Instrument (TROPOMI)

Copernicus, Europe's eyes on Earth



- Sentinel-4 (launch in 2021)
  - Provides air quality with data on trace gas concentrations and aerosols in the atmosphere
  - Main Instruments:
    - UVN Multispectral Spectrometer
    - Infra-Red Sounder (IRS)

Copernicus, Europe's eyes on Earth

- Sentinel-5 (launch in 2021)
  - Measures air quality and solar radiation, monitors stratospheric ozone and the climate
  - Global coverage of Earth’s atmosphere with an unprecedented spatial resolution
  - Carried aboard EUMETSAT’s MetOp Second Generation satellites
  - Main Instrument: Passive Grating Imaging Spectrometer

grnet

Copernicus, Europe's eyes on Earth

- Sentinel-6 (launch in 2020)
  - Observes changes in sea surface height with an accuracy of a few centimeters
  - Global mapping of the sea surface topography every 10 days
  - Enables precise observation of ocean currents and ocean heat storage; vital for predicting rises in sea levels
  - Main Instruments:
    - Synthetic Aperture Radar Altimeter (POSEIDON-4)
    - Microwave Radiometer (AMR-C)

Copernicus operations

- ESA developed a dedicated ground segment which includes
  - Sentinel Flight Operations Segment (FOS) → The brain of the control-plane
  - Sentinel Payload Data Ground Segments (PDGS) comprising → The brain of the data-plane

- Core Ground Stations (CGS) for Data Acquisition and product generation in Near Real Time
  - Matera (eGeos, IT)
  - Svalbard (K-Sat, NO)
  - Maspalomas (Inta, SP)
  - Alaska (K-Sat, USA)

- Sentinel Processing and Archiving Centers (PAC)
  - S1 (Astrium/UK, DLR/DE)
  - S2 (Astrium/UK, Indra/SP)
  - S3 (OLCI Land DLR/DE, SRAL CLS/FR, SLSTR-SYN ACRI/FR)
  - S3 (OLCI Marine EUMETSAT/DE)

- Sentinel Missions Performance Centers (MPC)
  - S1 (CLS/FR)
  - S2 (CS/FR)
  - S3 (ACRI/FR)

- Sentinel Precise Orbit Determination (POD) operated by GMV-led (Tres Cantos, Spain)
- Payload Data Management Centre (PDMC) (European Space Research Institute – ESRIN, Frascati/IT)

Space-to-Earth data delivery

- Orbiting from pole to pole about 700 km up, Sentinel-1A transmits data to Earth routinely, but only when it passes over its ground stations in Europe. However, geostationary satellites, hovering 36 000 km above Earth, have their ground stations in permanent view so they can stream data to Earth all the time

- Marking a first in space, Sentinel-1A and Alphasat (telco sat) have linked up by laser stretching almost 36 000 km

- Large volumes relay of data very quickly so that information from Earth-observing missions can be even more readily available (used maritime safety and helping to respond to natural disasters)

Copernicus Data Hubs

- **Copernicus Services Hub**
  - Over 200 users
  - No rolling policy
  - Data on demand service is available
  - Service Level Agreement
- **International Hub** (hosted by **GRNET**, operated by **N**ational **O**bservatory of **A**thens)
  - 4 users (**N**ational **A**eronautics and **S**pace **A**dministration/USA, **N**ational **O**ceanic & **A**tmospheric **A**dministration/USA, U.S. Geological Survey/USA, Geoscience/AUSTRALIA)
  - 3-week rolling policy
  - Service Level Agreement
- **Collaborative Hub** (hosted by **GRNET**, operated by **NOA**)
  - Over 25 users (including hub relays)
  - 2-4 weeks rolling policy
  - Service Level Agreement
- **Copernicus OpenAccess Hub**
  - 150.000 users
  - No rolling policy
- **Hub Relays, national mirror sites etc.**
  - Increase the overall capacity
  - Different flavors of data/metadata

Copernicus added-value services

- **The Copernicus Land Monitoring Service** (CLMS)
  - provides access to information on land-use and land-cover products (e.g. vegetation state or the water cycle)
- **The Copernicus Atmosphere Monitoring Service** (CAMS)
  - provides data and information on atmospheric composition (current situation, few days forecasts)
- **The Copernicus Marine Environment Monitoring Service** (CMEMS)
  - one-stop-shop for marine data
- **The Emergency Management Service** (EMS)
  - supports crisis management (disasters caused by natural hazards, man-made hazards and humanitarian crisis)
- **The Copernicus Climate Change Service** (C3S)
  - provides high quality data and graphics for climate change
- **Security service aims to support EU policies**
  - information in response to Europe's security challenges (crisis prevention, border surveillance, maritime surveillance)

Copernicus Data Metrics

ESA (all hubs)

8 Million published products

60 PB downloads from users

Sentinel-1 delivery: 1h (Near Real Time), 24h (Non Time Critical)

Sentinel-3 delivery: <3h (Near Real Time), <1month (Non Time Critical)

GRNET/NOA Consumes

- ~20k products /day
- ~13 TiB /day
- Data downloaded from the distribution DC
- *For Sentinel-5P GRNET will be the seeder*

GRNET/NOA Provides

- 36k products = 13 TiB / day downloads (IntHub)
- 51k products = 17 TiB / day downloads (ColHub)
- 61k products = 15 TiB / day downloads (DIASHub)
- Overall avg product size: 700 MB (min: 15 MiB, max: 1.7GiB)

The Mission

"Transfer satellite products from the main service distribution DC (central Europe), to our DC in Greece"

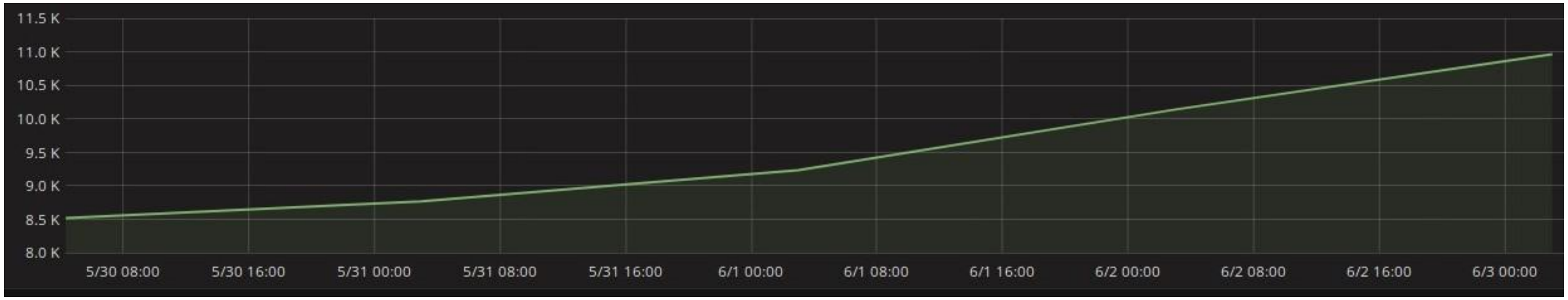
OR

"A constant flow of large files should be delivered daily from DC *alpha* to DC *beta* residing thousands miles away, over Long Fat Networks (LFNs)"

- Datasets should be transferred *asap* to the scientific community and authorities

- The software handling the file download is using TCP connections for data transferring and resides on multiple Virtual Machines on multiple servers across the DC.

The Problem
Products Backlog



GRNET/GÉANT networks can achieve speeds of many Gbps

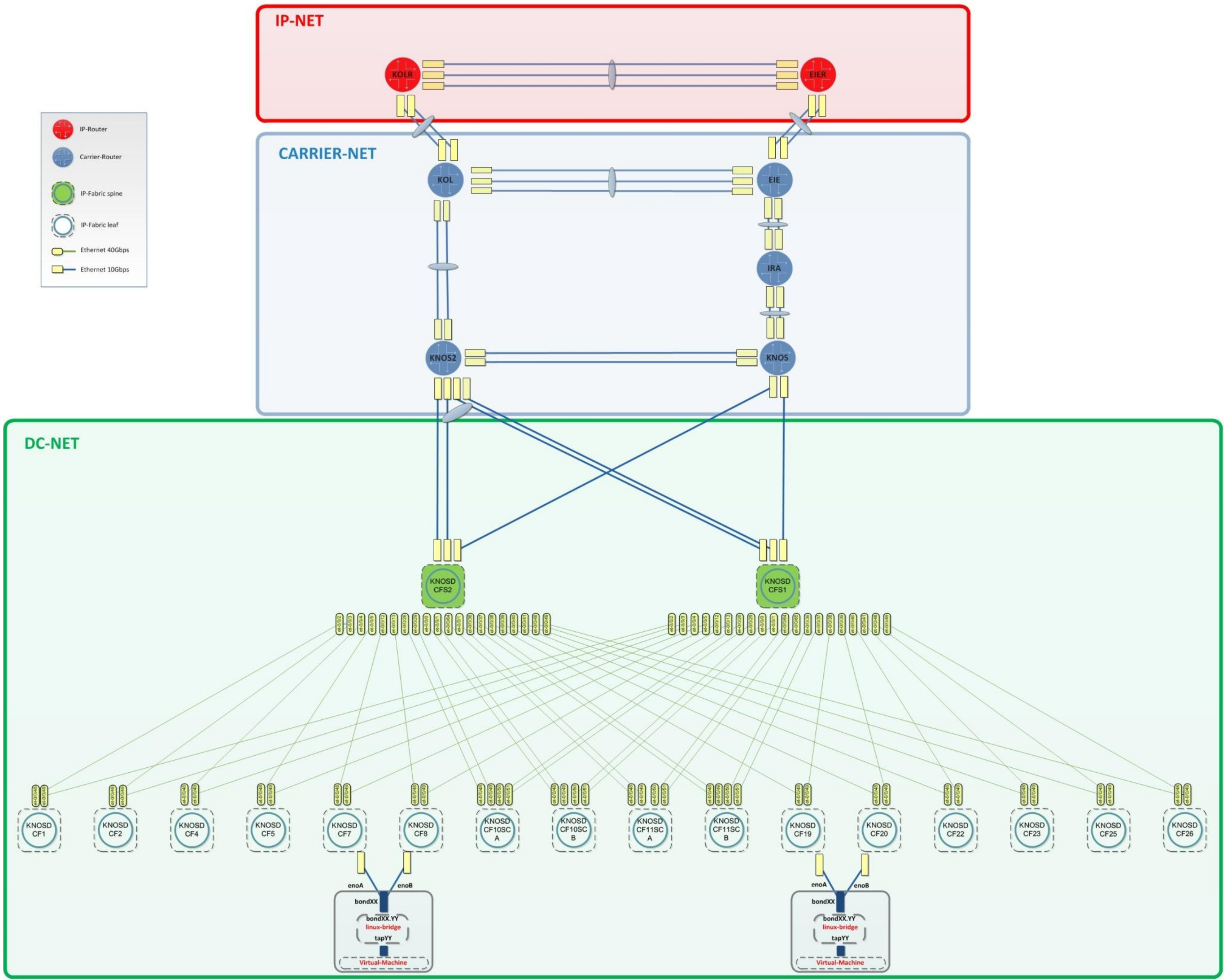
but

- The transfer speed was not adequate
- The number of the datasets (files) per day, with a total size of many Terabytes, that were in the queue waiting to be transferred to our DC was increasing
- One of our first thoughts →distance (a.k.a propagation delay in networking) leading to TCP throttling

Ecosystem to deal with

- Newly-built DC with spine-leaf collapsed CLOS topology
- EVPN-VXLAN for L2 connectivity (between servers & DC-routers)
- Spine switches act as DC-routers
- Carrier routers running MPLS/IS-IS provides L2VPNs to connect DC-to-IP
- Carrier routers links over optical services (DWDM optical network)
- Each layer of the aforementioned WAN/DC Fabric are redundant to node, link and routing engine level
- The overall number of the paths between end-hosts located to different GRNET's DCs is at least 128
- Servers running the service VMs also multi-homed to TOR leaf switches
- Ganetti/KVM for IaaS deployment
- Service specific software is running to download the datasets
- The main DC is located in central Europe and our capacity view is limited to the GRNET/GEANT network, two of the multiple pieces of the network topology puzzle. No view to the rest of the network links

GRNET KNOSSOS-DC IP-FABRIC: L2 topology



Troubleshooting Approach

Many software/hardware components taking part to the service provisioning there was no other path rather than..

Trying to make an educated guess as a starting point and adopt a top-down approach for our troubleshooting.

The educated guess

(1) taking into consideration the report that another DC somewhere in central Europe is downloading with higher speeds from the main DC

(2) not knowing exactly what the numbers behind this vague description are

(3) having assured there is no congestion inside our DC and WAN topology, we start thinking the TCP characteristics.

the bandwidth-delay (BD) product effect is throttling the TCP bandwidth of each connection

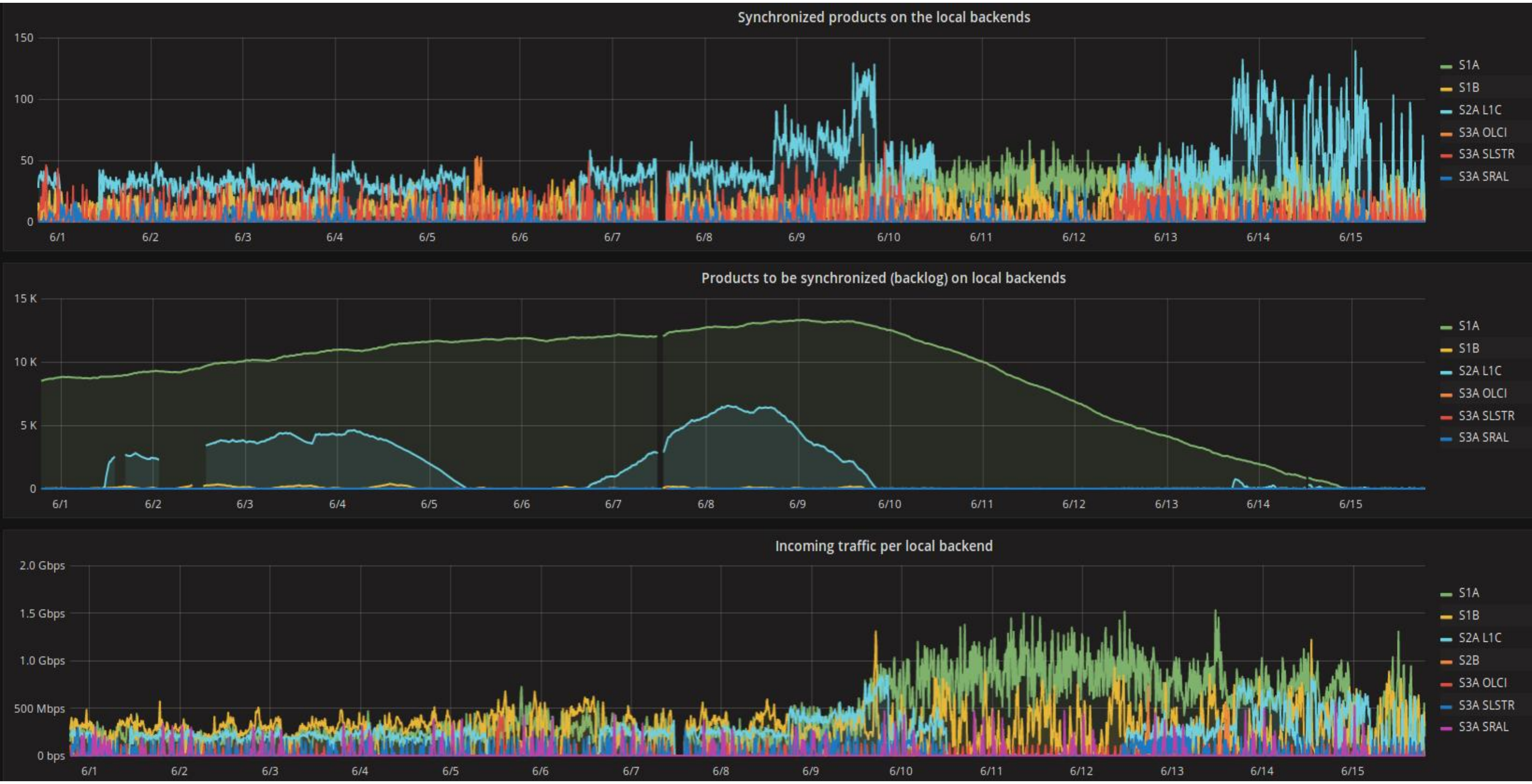
BUT

With so many domains/operation teams between the host and not controlling both ends we had to prove it and suggest proper tuning

Fast work-around

- Each satellite has its own datasets, handled separately: one FIFO queue of datasets per satellite waiting to the main DC to be downloaded to our DC
- Multiple TCP connections are used in order to transfer datasets in parallel
- The immediate work-around we decided to test if it really works was to increase the number of parallel TCP connections
- IF the BD product the only cause of the “low speeds”
- IF no packet loss (buffer overflow, link congestion etc.)
- THEN the congestion window would not be activated, letting each TCP connection to carry as many bits as the BD product would let to fly on wire
- *This was the obvious way to increase the aggregated bitrate*

Work-around impact



grnet

Assumptions

- **Argument:** “The BD product was the root cause”
- **Baseline:**

  RTT= propagation + transmission + processing + queuing

  buffer(size) = bandwidth * RTT

1st assumption:

The propagation delay →

the dominant type of delay for long-distance paths →

estimate the buffer size using the equation →

buffer size = bandwidth * propagation delay

2nd assumption:

No throttling/bottleneck (e.g. congestion back-off algorithm)

Hypothesis: *With constant RTT the bandwidth is proportional to the buffer size.*

Fully(?)-controlled Environment

- Small testbed deployment inside our premises between hosts located to GRNET's distant DCs (200 miles away)
- Iperf consecutive tests with 1 TCP connection
- No congestion/packet loss to our WAN/DC networks
- Results between 2 hosts with the default kernel parameters:

  HOST A, DC A –> HOST B, DC B
  1st trial: 0.0-10.0 sec 1.58 Gbits/sec
  2nd trial: 0.0-10.0 sec 2.00 Gbits/sec
  3rd trial: 0.0-10.0 sec 3.11 Gbits/sec

- >> 1Gbit/s vs. GRNET-Central_EU DC

First conclusions:

- In LFNs the propagation delay is the dominant type of delay
- buffer size = bandwidth * propagation delay
- longer connections suffer more (200Mbit/s vs. 2Gbit/s)

Fully(?)-controlled Environment

- Kernel rx (max) buffer size → 6Mbyte

- Extra space for internal kernel structures

- The rx buffer size in linux kernel is the upper bound of the window size but is not equivalent to the window size

- The % of the receive buffer used for "administrative purposes", not allocable for the tcp window was stated wrong to the man-page → dig to the kernel code

- With window size scaling & auto-tuning on the correct upper bound for the default window size is (only) 3072 Kbyte

Fully(?)-controlled Environment

- Calculating the throughput for 8ms RTT the default window size value of 3072 Kbyte → <3.1 Gbit/sec, close to the max value we achieved , but not enough, because the 1st trial has achieved only 1.58 Gbit/s, almost the 1/2 of the max achieved value of the 3rd trial
- The iperf set of results, each time, had fluctuations, but there were no packet retransmissions, so the congestion algorithm was not kicked in
- RTT was changing ?
  After several ping checks → RTT ~ [8, 12]ms
  The alternative paths were above 128
- The maths for the 12ms RTT results to an upper bound of 2.1 Gbit/sec
- Understand why the ping results were steadily 7.8ms or in rare cases steadily 11.9ms was a nightmare (multiple links between DC/Carrier/IP network)
- The reason for such a RTT difference was eventually found to a DWDM optical service that was following a much longer path, inserting propagation delay

Fully(?)-controlled Environment

- The results between those 2 hosts when we increased the rmem buffer (receiver buffer that defines the announced window size of the end-host) were better

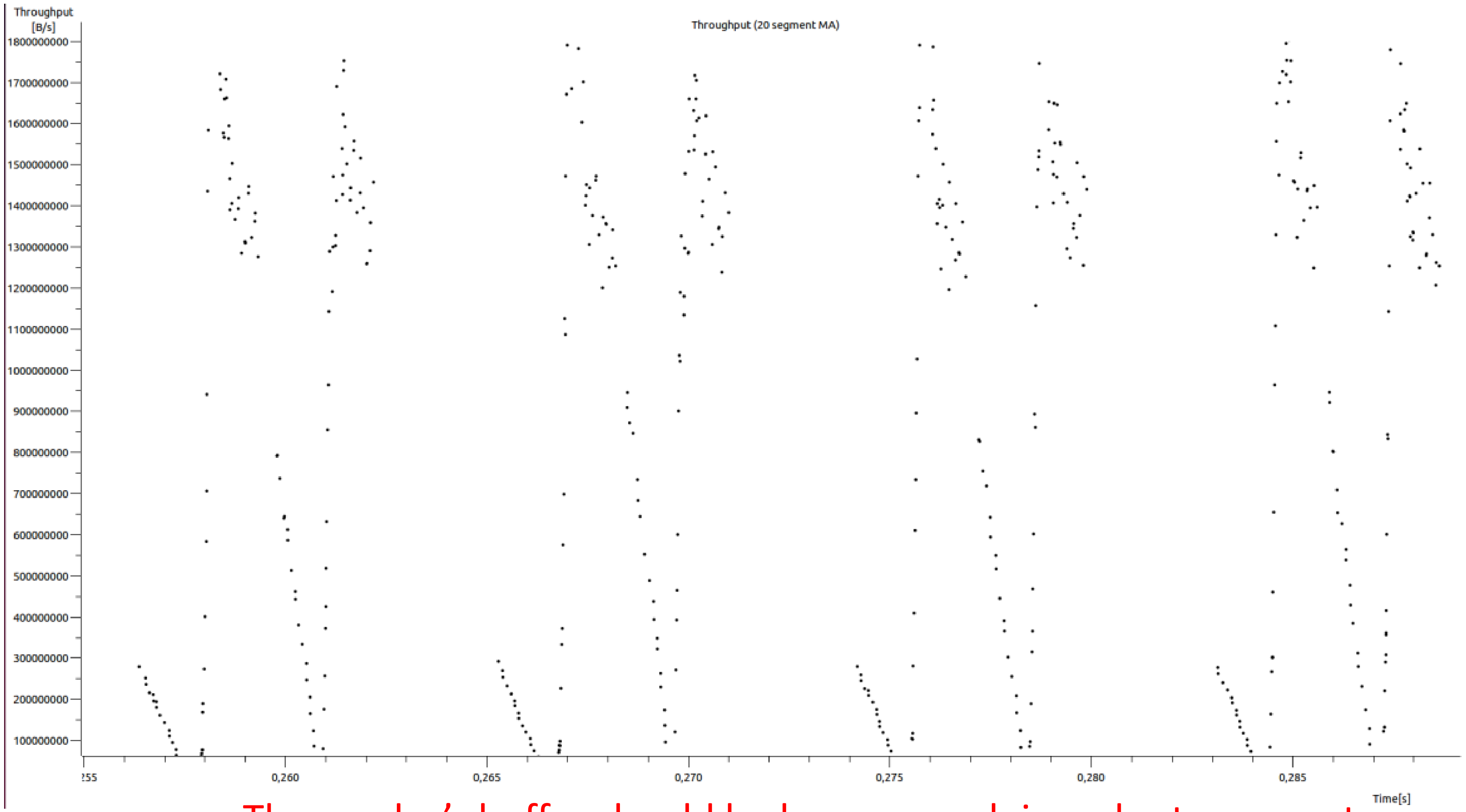
- Good news, but something unexpected derived from this tests, except the large deviation between the trials we made in both directions. On this round of measurements the results were not even approaching the theoretical value of the maximum throughput →
  16Mbytes gives <= 8.4 Gbit/sec for 8ms RTT

- We thought that there may be another bottleneck, starting looking again to other system parameters

Fully(?)-controlled Environment

- We noticed that the throughput had been throttled many times per second, it was like the sender was slowing down again and again
- The sender was never sending packets with a rate that could reach the maximum window size value that the receiver was announcing through its acks.
- We also noticed that the “bytes in flight” (a.k.a bytes of TCP packets that has not been yet acked from the receiver) were not exceeding the sender buffer size (net.ipv4.tcp_wmem).
- Hence, the sender’s buffer should be large enough in order to support a large number of bytes in flight

Fully(?)-controlled Environment



The sender's buffer should be large enough in order to support a large number of bytes in flight

Fully(?)-controlled Environment

- First priority requirement to optimize the bitrate: tune the rx/tx buffers
- We run the tests again setting the write buffers to the same values with the receive buffers, in both end-host (rx/tx buffer size 16Mbyte)
- The results were much better:

  HOST A, DC A –> HOST B, DC B
  0.0-10.0 sec, 5.95 Gbytes, 5.11 Gbits/sec
- We were feeling that we could proceed with a longer step

  ⇩
- Throughput tests with hosts outside GRNET's network, residing worldwide, tuning the relevant receive/transmit buffers.

Production environment

Real-life conditions which define the environment we made the measurements/observations:

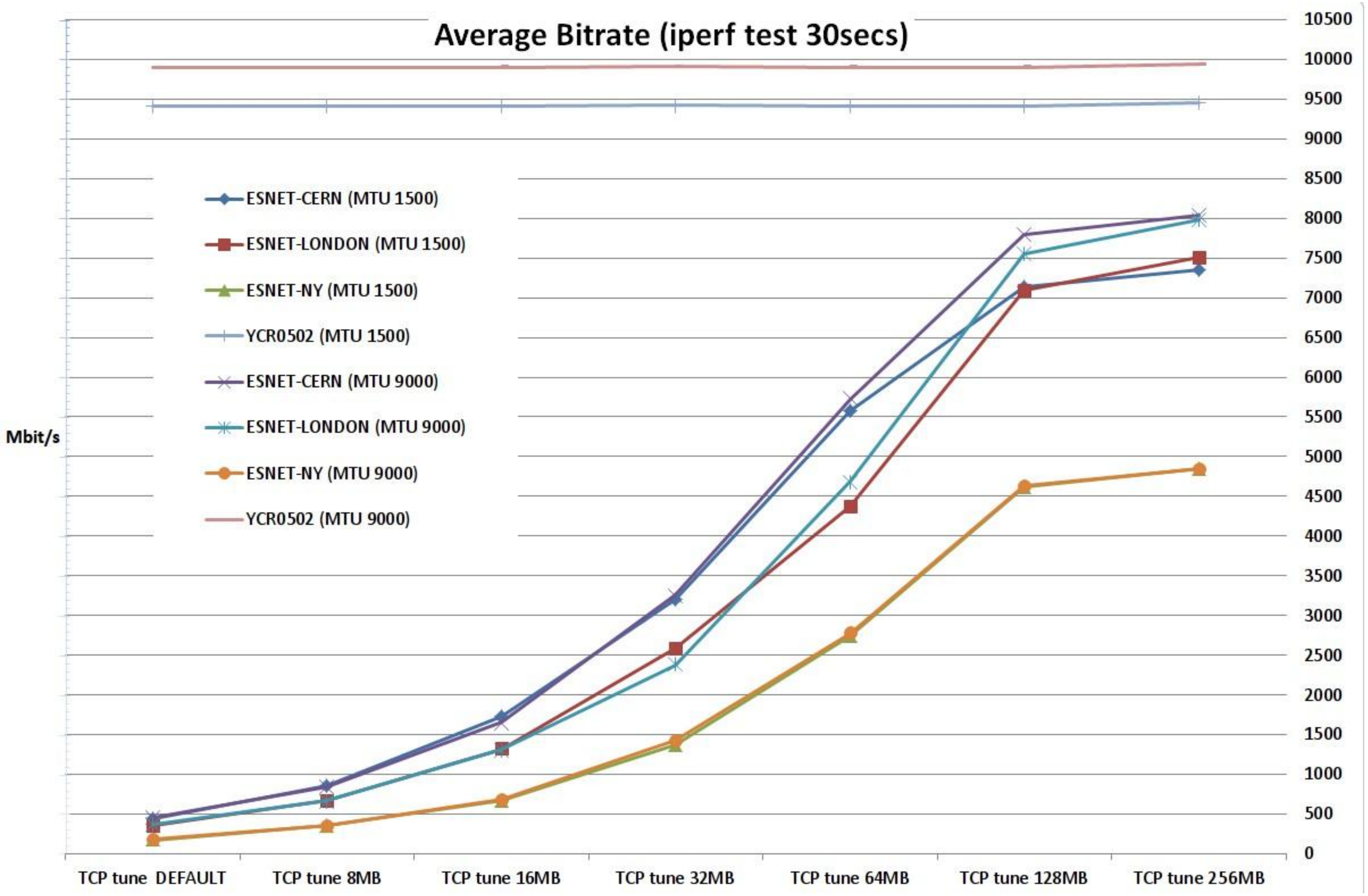
1) Use of perfSONAR monitoring suite for measurements

2) Use of ESnet servers (256 Mbyte buffers & 10Gbit NICs), capable of supporting high throughputs even in high-RTT links
   We could throttle the TCP connection from our end-host which was acting as the sender, tuning the write buffer

3) The intermediate links were not congested as they were belonging to overprovisioned research networks such as GEANT & ESnet

4) We estimated that the background traffic the moment of the tests was between 1-1,5Gbps out of 10Gbps that a single TCP connection could theoretically consume due to link speed constraints and hashing (Layer3-Layer4 hashing to nx10Gbit aggregated Ethernet bundles)

Production environment



grnet

Conclusions

1) ***The rx buffer size of the fat flow receiver has the highest impact to tcp performance*** → defines the TCP window size → should be tuned taking into consideration the RTT

2) ***The wx buffer size of the fat flow sender must also be tuned****, as it defines the maximum number of UN-acknowledged bytes that the sender side will allow to fly on wire*
Better to define in both tcp endpoints the rx & the wx buffers, as a TCP connection is bidirectional, unless you know which side is going to create the heavy traffic (sender) & which side is going to receive it

3) ***The buffers which (at least) should be tuned (for Linux kernel):***
net.core.rmem_max {max_value}
net.ipv4.tcp_rmem {min_value default_value max_value}
net.core.wmem_max {max_value}
net.ipv4.tcp_wmem {min_value default_value max_value}

Conclusions

4***) For RTTs > 100ms, or/and links > 10/40/100Gbit links you need huge buffers***, even 30x or 40x bigger than the default kernel values
Example: 131070 KByte TCP window, 256 Mbyte buffer, 148ms RTT
GRNET DC - ESnet New York <= 7254.90 Mbit/sec

5) ***TCP rx buffer max_value defines the max window size of the rx throttling the max throughput of the TCP over the path & the TCP send buffer max_value defines the max number of bytes on flight*** (you need window scaling and receive-buffer autotuning on, see RFC-7323)

6***) A larger MTU size can support larger throughput <u>when</u>***:

(a) link congestion arise somewhere to the end-to-end path
(b) the bandwidth-delay product is not a bottleneck
(c) there is no packet loss (see the red/blue line for intra-DC tests)

Conclusions

7) ***MTU size affects the bandwidth performance of a single TCP flow when this TCP flow consumes all the available bandwidth & the window size is not a bottleneck***

8) ***Kernel's window-size auto-tuning, with unchanged read buffer default_value, is importing delays to the desired high bitrate build-up in large RTT TCP connections***

Backup

Provided that you do not change the net.ipv4.tcp_rmem default_value the tcp connection is starting with a small window importing a delay to the desired high bitrate building up . Obviously, the 3-way TCP handshake over large RTTs paths is also an inevitable “a priori” worsening factor.

150ms RTT

```
[ 15] local xxxx port 5165 connected with yyyy port 5165
[ ID] Interval Transfer Bandwidth
[ 15] 0.0- 1.0 sec 13.9 MBytes 117 Mbits/sec
[ 15] 1.0- 2.0 sec 593 MBytes 4979 Mbits/sec
[ 15] 2.0- 3.0 sec 891 MBytes 7472 Mbits/sec
```

0.15ms RTT

```
[ 3] local xxxx port 55414 connected with yyyy port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0- 1.0 sec 1.15 GBytes 9.92 Gbits/sec
[ 3] 1.0- 2.0 sec 1.15 GBytes 9.90 Gbits/sec
[ 3] 2.0- 3.0 sec 1.15 GBytes 9.90 Gbits/sec
```