QUIC and Satellite Open Stakeholder Meeting

Accelerating Start-up

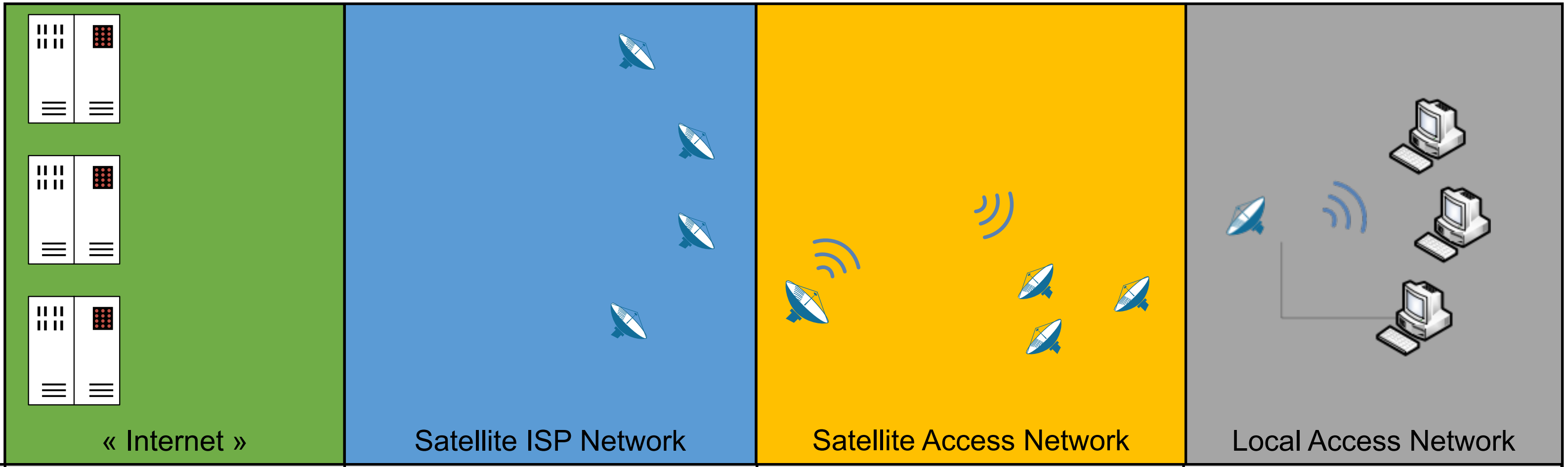
Nicolas KUHN

nicolas.kuhn@cnes.fr

cnes
CENTRE NATIONAL
D'ÉTUDES SPATIALES

VIVERIS
Innover. Simplifier. Partager.

Typical GEO satellite-based Internet access



	« Internet »	Satellite ISP Network	Satellite Access Network	Local Access Network
Data rate	High	High	Variable	Average
Latency	Low	Low	High	Low
Loss	No loss	Congestion losses	No loss	Loss if Wi-Fi

- **Solution #1 : adapt the end-to-end protocols**
- **Solution #2 : inform end point of the path characteristics**

Solution #1 : adapt the end-to-end protocols

Utilised platform for tests

QUIC SERVER

Delay / capacity limitation
Losses

QUIC CLIENT

picoquic/picoquic (h3-25/24/23)
or
h2o/quicly(h3-25)

picoquic/picoquic (h3-25/24/23)
or
curl/ngtcp2 (h3-25)

End-to-end path without split

Multiple paths with split

IPERF3 SERVER

PEPSal

Delay / capacity limitation

PEPSal

Losses

IPERF3 CLIENT

Default
Kernel 4.15
Ubuntu 16.04
iperf3 v3.6

Default ; Kernel 4.4
Same configuration on both PEP client and PEP server

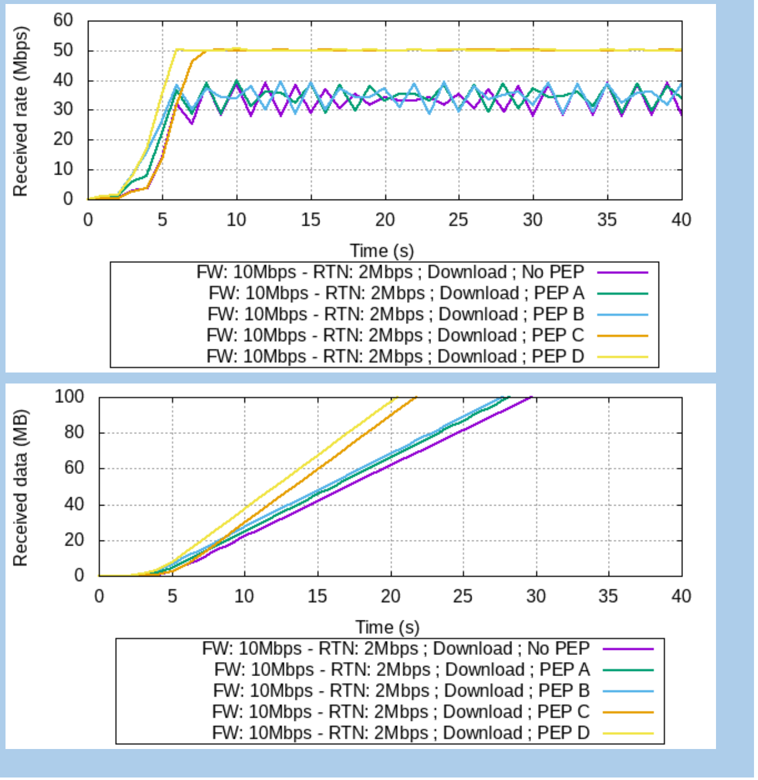
	TCP_WMEM_MAX (MB)	TCP_RMEM_MAX (MB)	CORE_WMEM_MAX CORE_RMEM_MAX (MB)	ICWND IRWND (packets)
NO PEP	4	6	0,2	10
PEP A	4	6	0,2	10
PEP B	4	6	0,2	100
PEP C	33	33	33	10
PEP D	33	33	33	100

Default
Kernel 4.15
Ubuntu 16.04
iperf3 v3.6

Solution #1 : adapt the end-to-end protocols

Focus on the 50 Mbps / 10 Mbps use-case

Multiple paths with split



With TCP-Proxy:

- Capacity to reach channel capacity
- Reduced transmission time

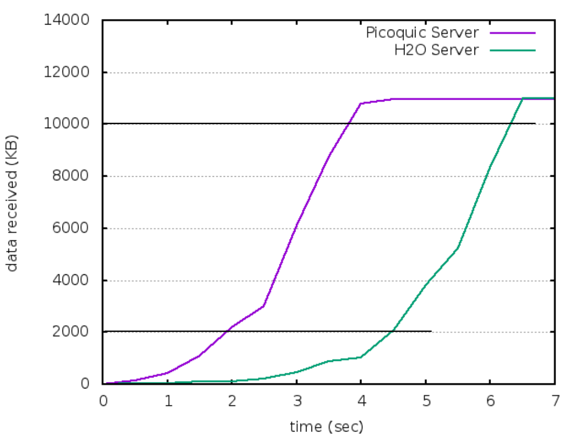
Proposed objectives :

- 2MB: 3 sec
- 10 MB: 5 sec
- 100MB: 20 sec

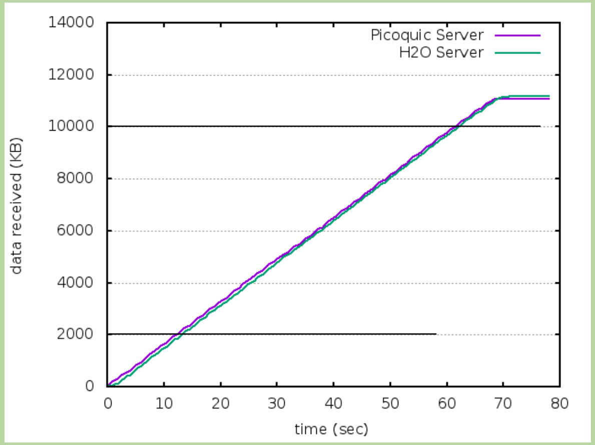
Solution #1 : adapt the end-to-end protocols

Focus on the 50 Mbps / 10 Mbps use-case

End-to-end path without split



PICO-QUIC CLIENT



CURL CLIENT

- PICO-QUIC (Version v0.24d) client
  - PICO-QUIC server (Version v0.24d) : the objectives are met
  - H20 server (2.3.0-DEV@9f65c27) : the objectives are not met
- CURL client (Version 7.69.0-DEV (x86_64-pc-linux-gnu) libcurl/7.69.0-DEV OpenSSL/1.1.1d zlib/1.2.8 brotli/1.0.4 ngtcp2/0.1.0-DEV nghttp3/0.1.0-DEV)(any server)
  - The objectives are not met

Solution #1 : adapt the end-to-end protocols

- **Designing a CC that is relevant for all deployment cases may not be relevant**
- **Knowing about the path characteristics can help in adapting the CC in specific deployment scenarios**
  - Tuning RTT_INIT
  - Tuning flow control parameters (MAX_STREAM_DATA)
- **Issue in CC convergence in SATCOM use cases**

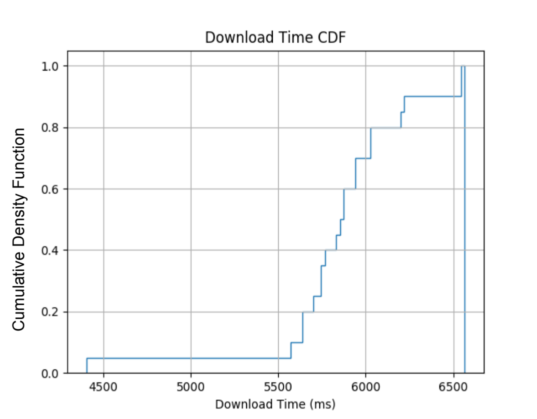
Solution #2 : inform end point of the path characteristics

- The objective: inform end point of the path characteristics
  - Based on previous connections
  - During a connection:
    - Both peers measure BDP (rtt * bytes_in_flight)
    - The server sends to the client the information that has been measured
  - When reconnecting:
    - Client sends a 0-RTT token for faster connection establishment
    - The idea: add the information from previous BDP to the server
- See draft-kuhn-quic-0rtt-bdp-07 for how it could be done in QUIC
  - There is also a strawman algorithm in the draft on how to safely jump to the available capacity
  - The results showed here :
    - No safe jump
    - The server uses directly previously measured BDP

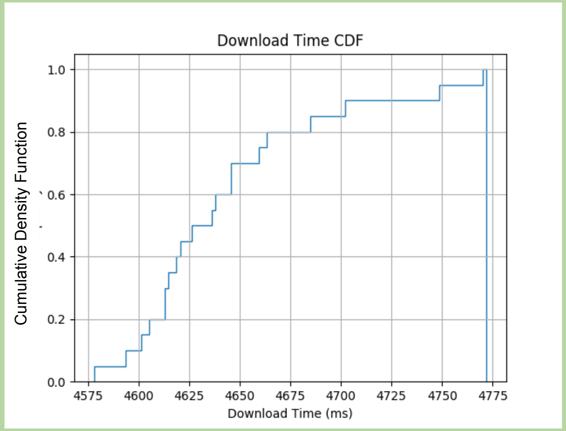
Solution #2 : inform end point of the path characteristics

- Focus on the 50 Mbps / 10 Mbps use-case
- Time needed to download 10 MB

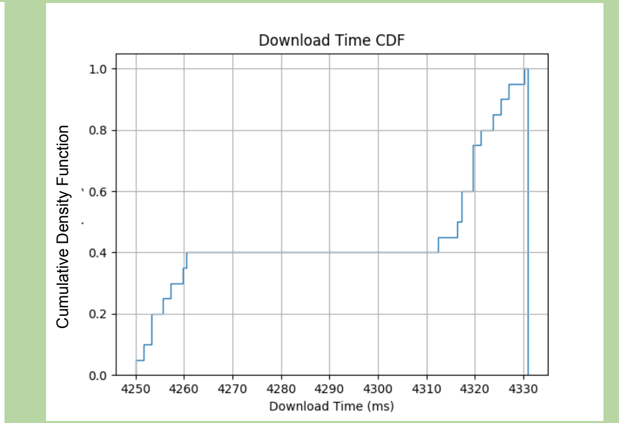
End-to-end path without split



No 0-RTT



0-RTT

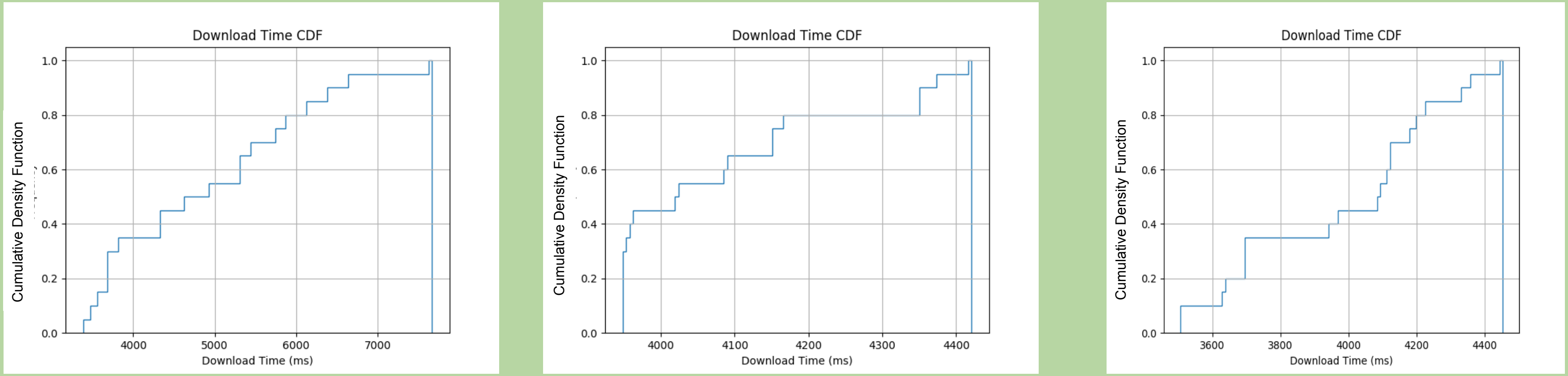


0-RTT-BDP

Solution #2 : inform end point of the path characteristics

- Focus on the 250 Mbps / 3 Mbps use-case
- Time needed to download 10 MB

End-to-end path without split



No 0-RTT

0-RTT

0-RTT-BDP

Summary

- **We are working at IETF on this**
  - If you are interested, please join the ETOSAT list: https://www.ietf.org/mailman/listinfo/Etosat
  - Or discuss our drafts
    - draft-kuhn-quic-0rtt-bdp-07
    - draft-kuhn-quic-4-sat-06
- **On the median case**
  - 0-RTT-BDP improves by approx. 25% cases with no 0-RTT
  - 0-RTT-BDP improves by approx. 10% cases with *just* 0-RTT
- **The CC has been simply modified**
  - More specific optimization could be imagined
- **What is going on ?**
  - What are the impacts of RRM mechanisms or various satellite-based (variable delay in mega constellation, etc.) or losses, etc on the end-to-end performance of QUIC ?
  - What are the performance of application level designs ? Do they need specific tuning or implementations ?