0RTTBDP drafts
draft-kuhn-quic-bdpframe-extension
draft-kuhn-quic-careful-resume

Nicolas Kuhn (CNES, Thales Alenia Space)
Stephan Emile (Orange)
Gorry Fairhurst (University of Aberdeen)
Tom Jones (University of Aberdeen)
Christian Huitema (Private Octopus Inc.)

0RTTBDP drafts

- [RFC9000]: "Generally, implementations are advised to be cautious when using previous values on a new path."

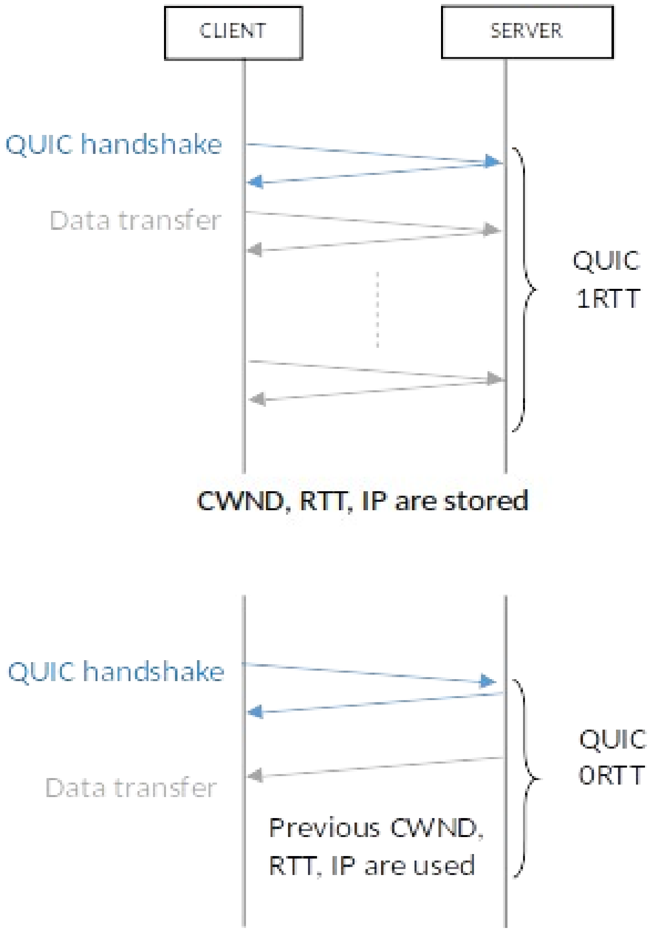
- draft-kuhn-quic-careful-resume-00
  - Carefully resume QUIC session using a recently utilised Internet path
  - How to use previous values in a interoperable manner
  - Integrates implementation recommendations for BBR, NewReno and CUBIC.
  - Safety guidelines

- draft-kuhn-quic-bdpframe-extension-00
  - Implementation description of BDP Frame Extension

Resume QUIC Session

1. During a previous session, store the following at the client and/or server:
   a. current RTT (current_rtt)
   b. CWND (current_cwnd)
   c. client's current IP (current_client_ip)

2. When a session resumes, exploit saved values



Scenarios of Interest

- Large BDP Scenarios
  - Reduces completion time where available capacity is significantly larger than allowed by IW
  - 1 MB transfer could be reduced by 62 %
- Accommodating a Reduction in Capacity
  - There can be reductions of available capacity
  - Quickly and safely resume CC parameter when capacity is known to be again available
    - (e.g. change in the interface used by the local endpoint)
- Optimizing Client Requests
  - (e.g. Requests by DASH-like traffic)
- Sharing Transport Information across Multiple Connections
  - [RFC9040] sharing transport parameters between TCP connections from the same host.
  - Enable the same for QUIC.
- Connection Establishment, Client and Server
  - Both endpoints may be server or client

Need for Safety

- Rationale #1: Variable Network Conditions
  - IP address change
  - RTT change
  - Lifetime of information
  - BB over-estimation
  - Preventing Starvation of New Flows

- Rationale #2: Malicious clients
  - Information sent by a malicious client is not relevant (e.g. to use too high a cwnd)

Phases of CC

1. **Observe** - during a previous connection:

Store current RTT, bottleneck bandwidth and current client IP

2. **Reconnaissance** - resuming a session between the same pair of IP addresses:

Confirm the path appears the same as observed previously (e.g., path with similar RTT)

Seek assurance that initial data is not lost

3. **Unvalidated** - transfer using an unvalidated rate:

Utilise the saved path characteristics to send at a rate higher than allowed by slow start

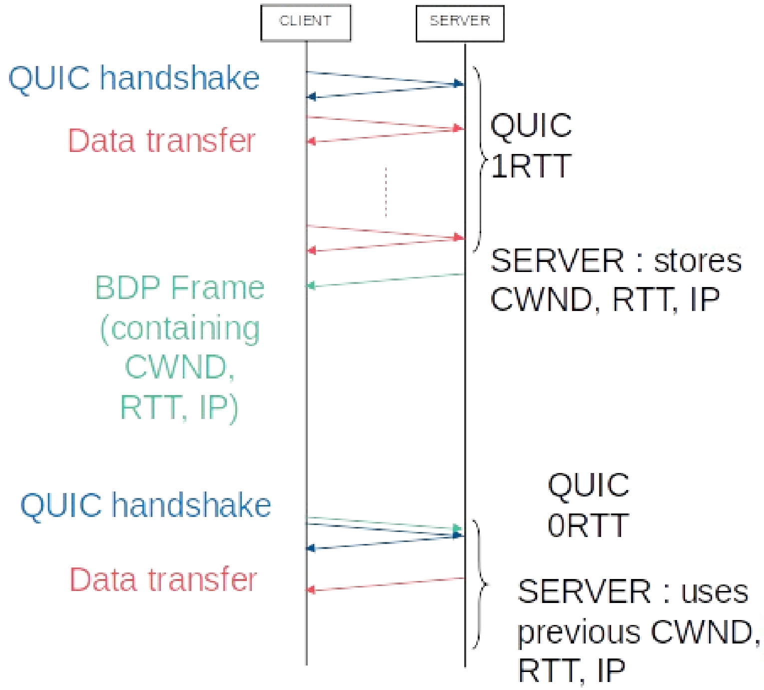
Should not be instantaneous, to avoid adding congestion to a congested bottleneck

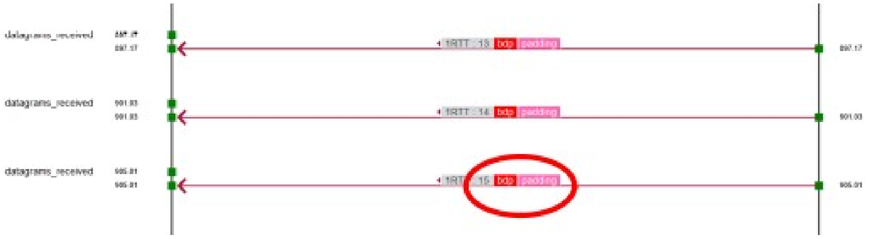
   1. If no noticeable congestion with saved rate is measured, continue sending at this rate in the 'Normal' phase
   2. If noticeable congestion, (i.e. parameters are not valid) withdraw rapidly to a safe rate, before 'Normal' phase.

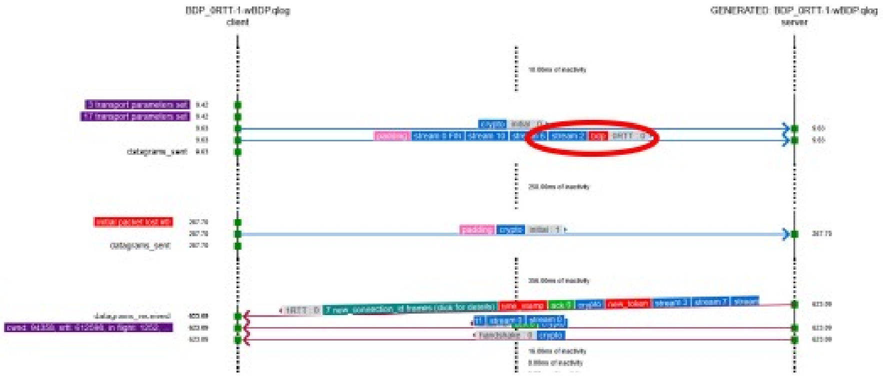
4. **Normal** - Resume using the normal CC method.

Further safety guidelines described : https://datatracker.ietf.org/doc/draft-kuhn-quic-careful-resume/

BDP Frame Extension







BDP Frame Extension

- Implemented prototype in picoquic :
  - https://github.com/private-octopus/picoquic/pull/1209
- Evaluated using a public SATCOM access :
  - Gain for a full 500kB or 1MB transfer (including HTTP GET)
  - 0-RTT vs 1-RTT
    - up to 33% gain for 500 kB
    - gain due to arrival of HTTP request one RTT in advance (and satellite RTT !)
    - up to 45 % gain for 1 MB
  - 0-RTT-BDP vs 1-RTT
    - up to 67% gain for 500 kB
    - up to 62% gain for 1 MB
- More info : https://arxiv.org/abs/2112.05450

Next Steps

- Any WG interest in the use-cases ?
- Any WG interest in implementing BDPFRAME in other QUIC stacks ?

Additional Slides

Safe Jump

Rationale	Solution	Advantage	Drawback
#1 : Variable network	#1 : set_current_* to saved_*	Ingress optimization	Risks of adding congestion
	#2 : implement safety check	Reduce risks of adding congestion	Negative impact on ingress optimization
#2 : Malicious client	#1 : Local storage	Enforced security	Client can not decide to reject Malicious server coulf fill client's buffer Limited use-cases
	#2 : NEW_TOKEN	Save resource at server Opaque token protected	Malicious client may change token even if protected Malicious server could fill client's buffer Server may not trust client
	#3 : BDP extension	Extended use-cases Save resource at server Client can read and decide to reject BDP extension protected	Malicious client may change BDP even it protected Server may not trust client