DAPS: Intelligent Delay-Aware Packet Scheduling For Multipath Transport

Nicolas Kuhn [1,2,3] Emmanuel Lochin [2] Ahlem Mifdaoui [2]
Golam Sarwar [3] Olivier Mehani [3] Roksana Boreli [3]

[1]IMT Telecom Bretagne, France

[2]Université de Toulouse, ISAE, United States

[3]NICTA, University of New South Wales, Australia

Table of content

Motivation

- Increasing population relying on smart-phones and tablets
- Heterogeneous wireless network access
- **Motivation for research and industry in multipath transport protocols :**
  - discussions at IETF for multipath capable versions of TCP (MPTCP included in Apple iOS7)
  - adoption of an enhancement of SCTP, CMT-SCTP, for WebRTC
- **Smart-phones and tablets, Wi-Fi and 3G/4G links :**
  - asymmetry between the links
  - receiver's buffer block : out-of-order packets occupying the entire receiver's buffer and eventually stalling the whole transmission
- **Existing solution for this issue :**
  - buffer management (increasing its size or splitting it)
  - specific retransmission policies
- **Our solution :**
  - scheduling algorithm, based on delay measurements, for in-order reception in CMT-SCTP

Our contribution

- **Model for maximum blocking time :**
  - model of the maximum time a packet will wait in the receiver's buffer for in-order delivery
  - validation of the model with NS-2 simulations
- **Delay-Aware Packet Scheduling (DAPS)**
  - our scheduling algorithm to reduce the blocking time
  - description of the algorithm behind DAPS
  - implementation and evaluation of DAPS in NS-2

Table of content

Receiver blocking on asymmetric links

- **Objectives of this section :**
  - source of receiver blocking
  - model of the maximum blocking time
  - validation of the model
- **Notations and topology :**

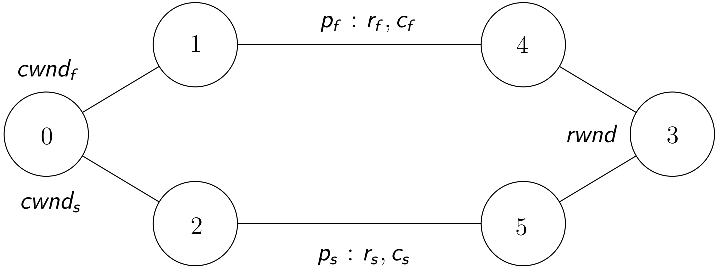


FIGURE : Two paths p_i (where i is s for "slow", or f for "fast")

Source of receiver blocking

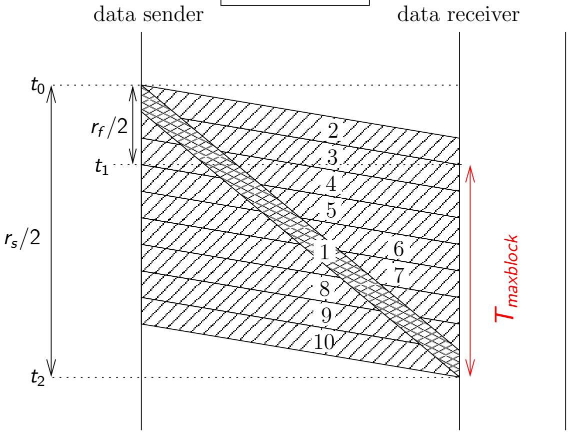
- for each path, the “blind” round robin sends as much data as the congestion window allows :

$$\min(cwnd_i - unack_i, rwnd - \sum_i unack_i) \tag{1}$$

- if $rwnd - \sum_i unack_i = 0$, no more data can be sent
- the receiver's buffer is blocked until the needed packet arrives
- this is more frequent with asymmetric links

Source of receiver blocking - example

$cwnd_1 = 1$
$cwnd_2 \geq 9$
$rwnd = 10$



2
3,2
4,3,2
5,4,3,2
6,5,4,3,2
7,6,5,4,3,2
8,7,6,5,4,3,2
9,8,7,6,5,4,3,2
deliver to app. 1 to 10

Model of the maximum blocking time

- **Notations :**
  - TSN_j : sequence number of each packet
  - L : size of a data packet
  - $8L/c_i$: time to place one packet on the physical medium
- **Sequence triggering maximum blocking time :**
  - TSN_1 on slow path p_s, TSN_2–TSN_{10} on fast path p_f
  - $t_1 = t_0 + r_f/2 + 8L/c_f$: reception of TSN_2
  - $t2 = t_0 + r_s/2 + 8L/c_s$: reception of TSN_1

$$T_{maxblock} = t_2 - t_1 = \frac{r_s}{2} + \frac{8L}{c_s} - \frac{r_f}{2} - \frac{8L}{c_f}. \qquad (2)$$

Validation of the model

- **Simulation parameters :**
  - $r_f = 20\,\text{ms}$, $rwnd = 65\,\text{kB}$, $L = 1500\,\text{B}$
  - $(c_f\,;c_s/r_s)$

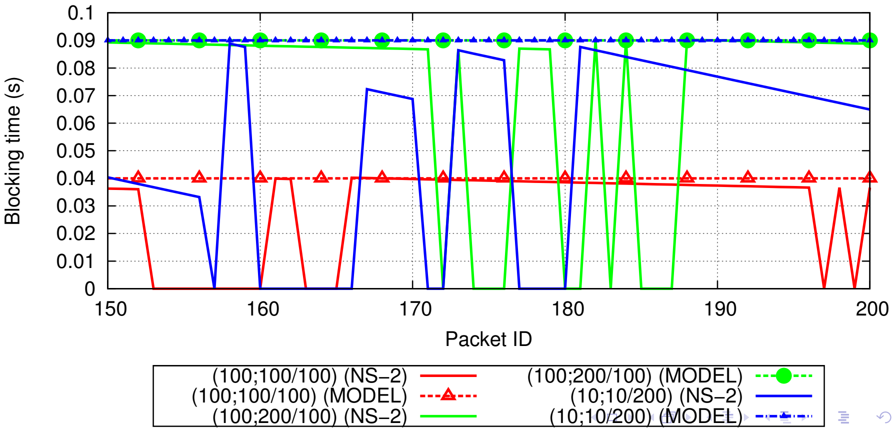


Table of content

Delay-Aware Packet Scheduling (DAPS)

- **Objectives of DAPS :**
  - in-order arrival at the receiver to avoid receiver's buffer blocking
- **DAPS pseudo-code :**
  - if capacity is available on multiple paths :
    - DAPS generates $S = \{s_1, \ldots, s_m\}$
    - each element $s_j = (TSN_j, p_j)$ represents that packet TSN_j is to be transmitted on path p_j
    - DAPS sends as much packets as possible, considering the *rwnd*
  - if capacity is available on the fast path only :
    - DAPS sends as much packets as possible, considering the *rwnd*
  - if capacity is available on the slow path only :
    - DAPS sends as much packets as possible, considering the *rwnd* and the last generated S

Table of content

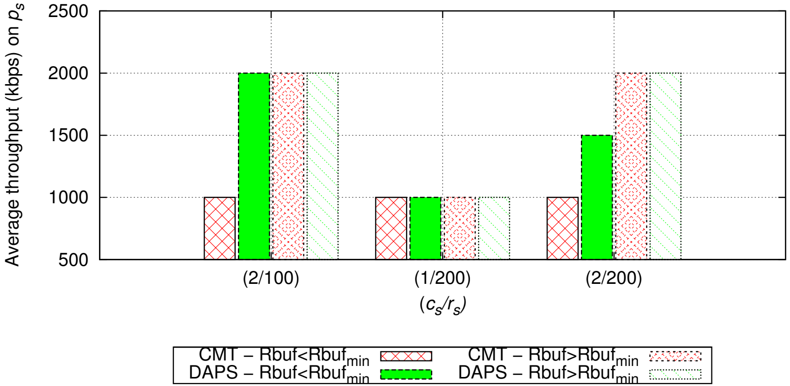
Simulation parameters

- dumbbel network topology
- various receiver's buffer size, capacities and RTTs
- $Rbuf_{min}$: minimum receiver's buffer size to address buffer blocking

$$Rbuf_{min} = \sum_{i \in \{p_1,\dots,p_n\}} c_i \times \max_{i \in \{p_1,\dots,p_n\}} r_i . \quad (3)$$

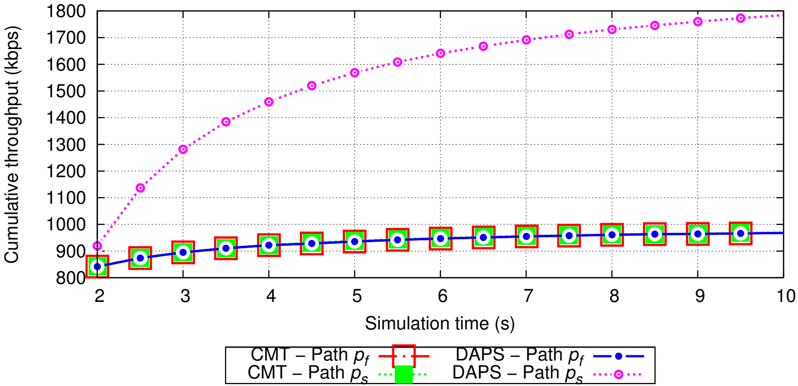
Label (c_s ; r_s)	c_s [Mbps]	r_s [ms]	$Rbuf$ [kB]		$Rbuf_{min}$ [kB]
(2/100)	2	100	35	$<$	37.5
(2/100)	2	100	500	$>$	37.5
(1/200)	1	200	45	$<$	50
(1/200)	1	200	500	$>$	50
(2/200)	2	200	45	$<$	75
(2/200)	2	200	500	$>$	75

Average use of the slow path



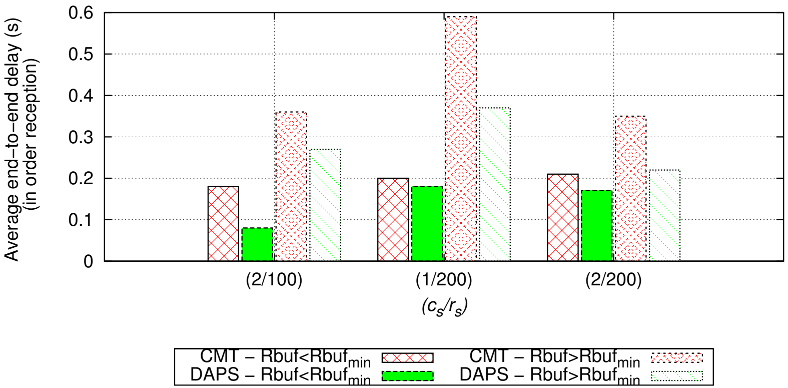
- $c_s = 2\,\text{Mbps}$ and $Rbuf < Rbuf_{min}$: the capacity of path p_s cannot be fully exploited with CMT-SCTP
- DAPS enables improved use of this capacity

Cumulative throughput



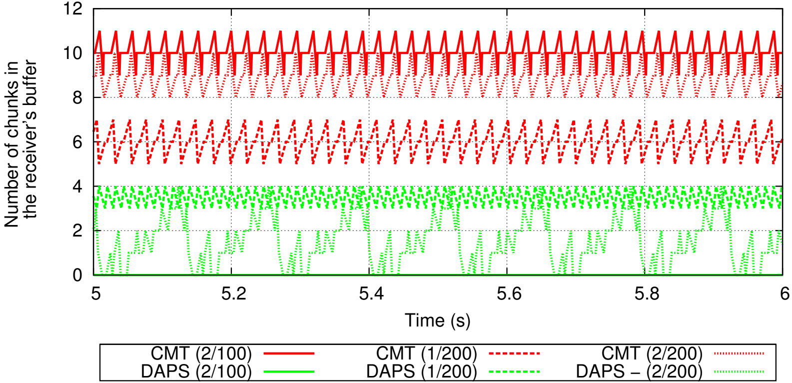
- ($c_f = 1\,\text{Mbps},\ r_f = 20\,\text{ms}$) ($c_f = 2\,\text{Mbps},\ r_f = 200\,\text{ms}$) $Rbuf < Rbuf_{min}$
- DAPS increases by 42% the cumulative throughput of both paths

Average application level transmission delay



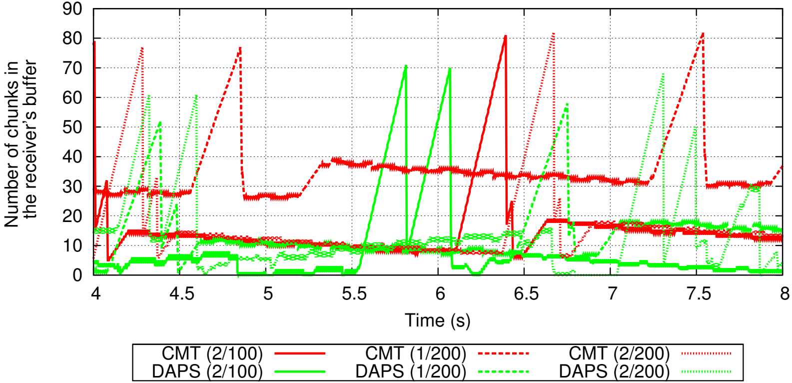
- DAPS allows for chunks to be delivered in-order earlier to the application

Occupancy of a small receiver's buffer



- DAPS reduces the occupancy of the receiver's buffer

Occupancy of a large receiver's buffer



- DAPS reduces the occupancy of the receiver's buffer

Table of content

Conclusion

- We argued for enhanced packet scheduling mechanisms
- We present Delay-Aware Packet Scheduling to improve the performance of multipath transport
- When there is asymmetry between the path, with our solution :
  - the buffer occupancy can be reduced by up to 77%
  - the delivery-to-application delay is consistently shorter
- Future work :
  - implement DAPS in FreeBSD's CMT SCTP and Linux implementation of MPTCP