A Vegas-based Approach for MPTCP Congestion Control

MPTCP – 84. IETF Vancouver – July 31, 2012

Mirja Kühlewind, Régel González Usach
mirja.kuehlewind@ikr.uni-stuttgart.de
Semi-coupled Reno-Based Congestion Control

**RFC6356**

Increase \( cwnd_i \) by \( \min( \alpha / cwnd_{total}, 1 / cwnd_i ) \) for each ACK on subflow \( i \)

Decrease \( cwnd_i \) to \( \max( cwnd_i - cwnd_i / 2, 1 ) \) for each loss event on subflow \( i \)

- Increase of \( cwnd_i \) of each subflow \( i \) coupled by a factor \( \alpha \) (same for all subflows)
- Sum throughput of all subflows at least as much as a single TCP would get on the best path
- Always slower increase rate on each subflow than single TCP (\( \alpha < 1 \))
  - Each subflow gets a smaller share than one TCP Reno flow would get on the same link

→ To change the increase in TCP Vegas would not lead to a different share
→ TCP Vegas always aims to share the available capacity equally
→ A smaller increase rate will only take longer to converge
Vegas-based MPTCP Congestion Control

**Principle of TCP Vegas**
- Compare an expected sending rate with the actual sending rate
- Calculate increase/decrease (once per RTT) if threshold is reached
  
  if \( \frac{cwnd}{RTT_{\text{min}}} - \frac{cwnd}{RTT} < \alpha \) then increase \( cwnd \) by 1
  
  if \( \frac{cwnd}{RTT_{\text{min}}} - \frac{cwnd}{RTT} > \beta \) then decrease \( cwnd \) by 1

- Halve congestion window on loss: \( cwnd = \max( cwnd - \frac{cwnd}{2}, 1 ) \)

**Vegas-based Approach for MPTCP congestion control (MPVegas)**
- Adjust thresholds to achieve different shares of capacity on one link
- Calculate increase/decrease (once per RTT) with thresholds scaled by \( k_i \) on subflow \( i \)
  
  if \( \frac{cwnd}{RTT_{\text{min}}} - \frac{cwnd}{RTT} < k_i \cdot \alpha \) then increase \( cwnd \) by 1
  
  if \( \frac{cwnd}{RTT_{\text{min}}} - \frac{cwnd}{RTT} > k_i \cdot \beta \) then decrease \( cwnd \) by 1

with \( k_i = \frac{\text{expected\_throughput}_i}{\text{sum\_throughput}} \)
Implementation

- MPTCP protocol extension not implemented
- Only congestion control algorithm
  - Linux kernel module
  - Small number of state variable that can be accessed by all congestion control procedures of all subflows
  - Approximation when calculating the maximum rate from all subflow to simplify implementation

→ In simulations all TCP connections initiated by one network stack are regarded as subflows of one MPTCP connection
Simulation Scenario

- **Scenario 1**: Two paths with same capacity and TCP Vegas Cross Traffic
  - Path A **5 Mbps**, path B **5 Mbps**, **50 ms** One-Way-Delay (OWD) for all subflows
  - One TCP Vegas flow on path A and two TCP Vegas flows on path B
- **Scenario 2**: Three paths with different capacity and TCP Vegas Cross Traffic
  - Path A **10 Mbps**, path B **5 Mbps**, path C **2.5 Mbps**, **50 ms** OWD for all subflows
  - One TCP Vegas flow on each path

→ We measure the cwnd and throughput at sender side
First Look on MPVegas Congestion Control

Scenario 1: Two Path with Same Capacity and TCP Vegas Traffic
First Look on MPVegas Congestion Control

Path A (10 Mbps)

Path B (5 Mbps)

Path C (2.5 Mbps)

The graphs show the throughput over time for different paths with varying bandwidths. The lines with markers represent the Vegas path, and the dashed lines represent the multipath path.
Questions?
Example on the Resource Pooling Principle

Sum capacity (5 Mbps + 5 Mbps): 10 Mbps

Each flow should get 1/3 of sum capacity (10 Mbps / 3): 3.33 Mbps

→ Reno flow A gets 2/3 of path A
→ Reno flow B gets 2/3 of path B
→ Each MPTCP subflow gets 1/3 of each path (A and B)

→ Equal share over all resources
Example on the Resource Pooling Principle

Sum capacity (5 Mbps + 5 Mbps): 10 Mbps
Each flow should get 1/3 of sum capacity (10 Mbps / 3): 3.33 Mbps

→ Reno flow A gets 2/3 of path A
→ Reno flow B gets 2/3 of path B
→ Each MPTCP subflow gets 1/3 of each path (A and B)

→ Equal share over all resources
Evaluation of Reno-based MPTCP

Scenario: Same Capacity and Same Base Delay for All Flows

MPTCP sum: 3.54 Mbps  Reno path A: 3.13 Mbps  Reno path B: 3.24 Mbps