Coupled Congestion Control for MPTCP

Mark Handley,
Damon Wischik,
Costin Raiciu
Resource pooling means the network is better able to accommodate a surge in traffic by shifting traffic and thereby “diffusing” congestion across the network.

or a loss of capacity
Linking the subflows can reduce the traffic sent on the more congested path, and so balance load.

**Regular TCP:**

- Each round trip, increase window \( w \) by 1.
- Each loss, decrease window \( w \) by \( \frac{w}{2} \)

**Multipath TCP:** for subflow \( r \):

- Each round trip, increase window \( w_r \) by \( \frac{w_r}{w_{total}} \).
- Each loss, decrease window \( w_r \) by \( \frac{w_{total}}{2} \)

The subflow that has the smaller window increases less and decreases more.
In our initial experiments with multipath congestion control, we investigated a simple setup with two paths, each with the same packet drop probability. Two separate (uncoupled) TCP congestion controllers use the two paths equally. A naive coupled congestion controller, inspired by Kelly+Voice (2005), flaps from one path to the other.
The noisy nature of congestion feedback makes it difficult to estimate congestion levels.

Random drops

loss rate 1%  
loss rate 1%  
random drops  
random drops

The top link is so congested!

I better switch to the bottom link.

Now the bottom link is more congested!
We could alleviate flappiness by averaging over a longer timescale

- But the longer we average, the worse we are at reacting promptly when congestion truly does change.

(This is why TCP probes continuously, rather than accumulating an ever more accurate average over its entire run-time.)
The Zen of resource pooling

In order for multipath congestion control to pool resources effectively, it should not try too hard to pool resources — instead of using only the paths that currently look least-congested it should instead maintain equipoise, i.e. it should balance its traffic equally between paths to the extent necessary to smooth out transient fluctuations in congestion and to be ready to adapt to persistent changes.
We devised a parameterized family of multipath congestion control algorithms, indexed by \( \phi \in [0,2] \), to investigate the tradeoff between load balancing and equipoise.
We devised a parameterized family of multipath congestion control algorithms, indexed by $\phi \in [0, 2]$, to investigate the tradeoff between load balancing and equipoise.

- Each ACK, increase $w_r$ by $a \left( \frac{w_r}{\frac{1}{n} w_{total}} \right)^\alpha w_{total}^\beta$

- Each loss, decrease $w_r$ by $b \left( \frac{w_r}{\frac{1}{n} w_{total}} \right)^{\alpha + \phi} w_{total}^{\beta + \delta}$
Decreasing $\phi$ improves resource pooling and effectively moves traffic away from congestion.
With varying background traffic, low values of $\phi$ move so much traffic off the more congested path that we miss opportunities to send.

On/off flows

Long lived flows
Neither extreme for $\phi$ seems desirable. How about intermediate values?

$\phi = 1$ is an interesting case.

- Reasonable load balancing, good equipoise.
- Very simple algorithm.
φ = 1 is an algorithm that just links the increases.

Recall: **Fully Coupled Algorithm:** for subflow \( r \):

- Each round trip, increase window \( w_r \) by \( \frac{w_r}{w_{\text{total}}} \).
- Each loss, decrease window \( w_r \) by \( \frac{w_{\text{total}}}{2} \).

**Linked Increases Algorithm:**

- Each round trip, increase window \( w_r \) by \( \frac{w_r}{w_{\text{total}}} \).
- Each loss, decrease window \( w_r \) by \( \frac{w}{2} \).
We tweaked the $\phi=1$ algorithm, to ensure fairness with TCP.

Assign a weight to each link, and run a weighted version of the $\phi=1$ algorithm. We have an adaptive algorithm for choosing the weights:

- the multipath flow gets as least as much throughput as if it used the best single path
- the multipath flow takes no more bandwidth on any link than a single-path TCP would.
We have a working implementation: mobile host

laptop with multipath TCP → wifi → 3G → server with multipath TCP

wifi throughput [Mb/s]

3G throughput [Mb/s]

time [min]
We have a working implementation: Resource pooling with a multihomed server.