A UDP-based multipath application: mpmosh

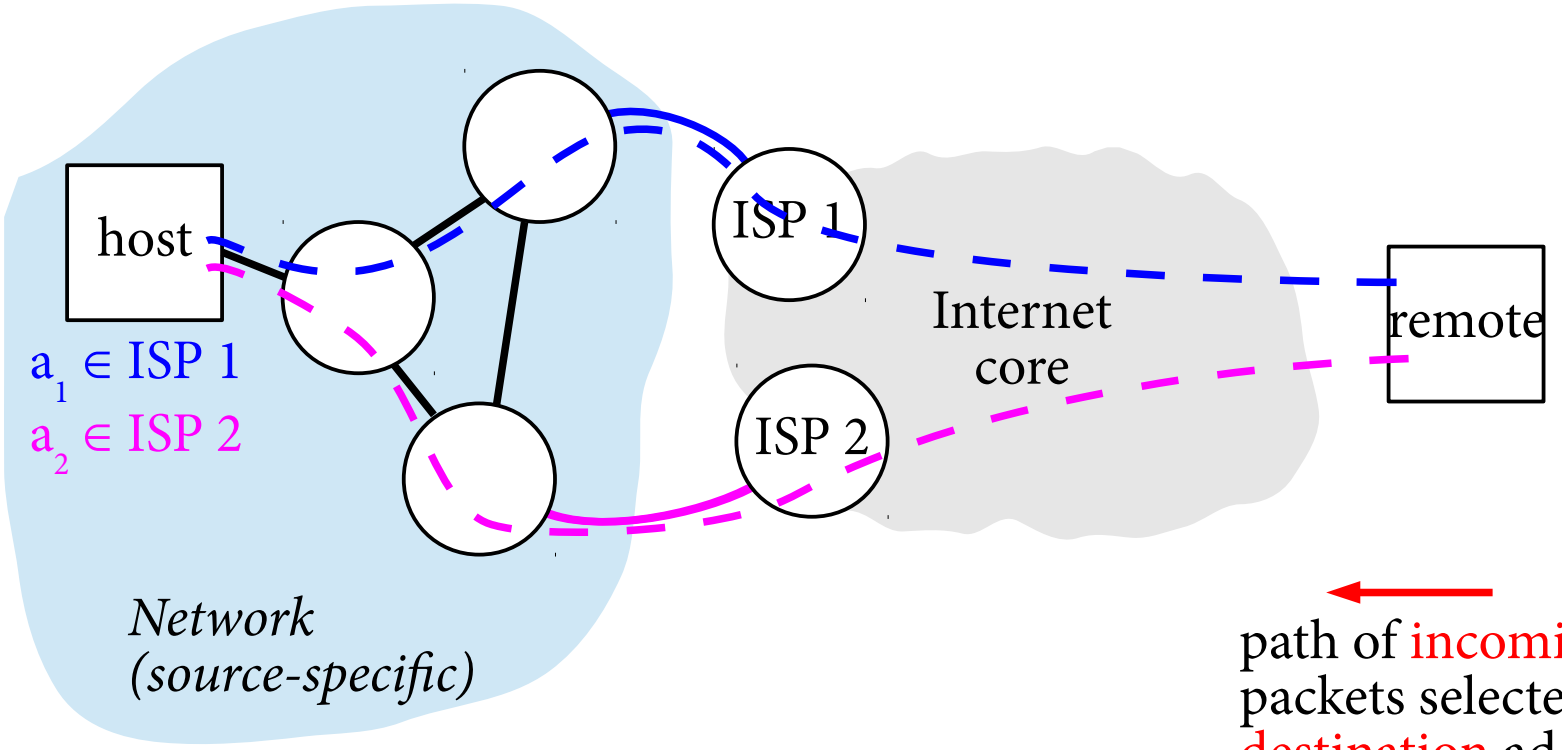
Matthieu Boutier
Juliusz Chroboczek

Laboratoire PPS - Université Paris Diderot
boutier@pps.univ-paris-diderot.fr

6 August 2015

Wireless Battle of the Mesh v8

Multihoming, multipath



path of outgoing packets selected by source address (local host choice)

path of incoming packets selected by destination address (remote host choice)

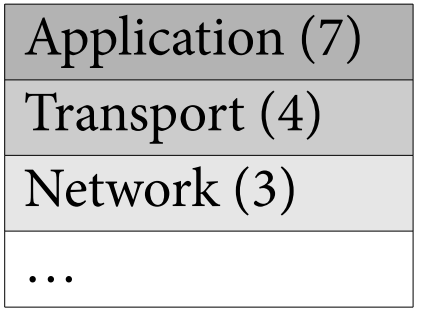
→ higher layers can do multipath:

- transport (MPTCP)
- application

Multipath TCP

- Compatible multipath replacement of TCP
  - provides reliability,
  - provides performances (load balancing).

→



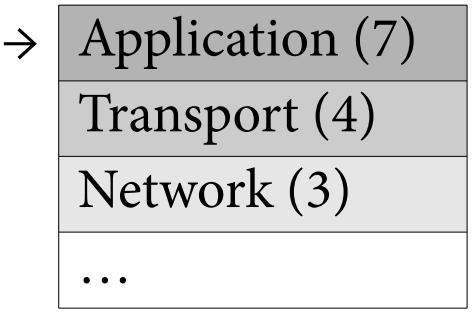
Everything works out-of-the box
with source-specific routing:

→ *(just rebuild your kernel…)*

Multipath at application layer: motivations

But (MP)TCP will be clever at your place:

- retransmissions,
- sends keepalives, and
  may timeout the connection,
- optimizes throughput,
  can't be tweaked for a particular application.



Advantages of the application layer:

- More flexibility: choose what to retransmit,
- Keep control on the traffic sent,
- Be smarter: optimize delay, throughput,…
  (application dependant problem)
- Experimenting new stuff,

- *(don't need to rebuild your kernel!)*

Multipath application with UDP

It is possible with:

`sendmsg`

Now, a few details…

The mobile shell (mosh)

Keith Winstein

Mosh is a « replacement for SSH » *(mosh.mit.edu)*

Mosh is a lightweight interactive application.

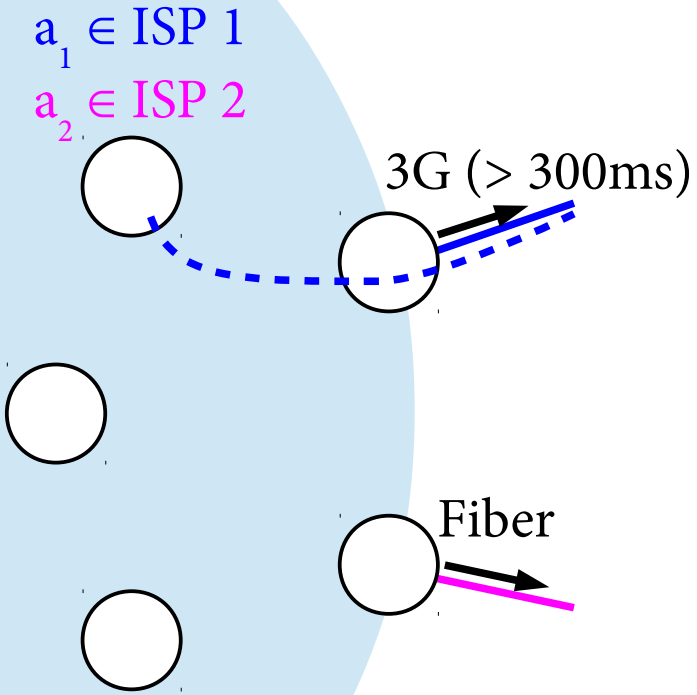
Mosh is robust:

→ allows roaming,
(client address or port switching)

→ supports intermittent connectivity,
(will not timeout without your consent)

→ resilient to packet loss
(mosh doesn't care)

Mosh in a multipath environment



Mosh makes no differences between paths:
→ 3G is as good as the fiber (eh!)

Mosh will not roam if the address is not gone
→ even if there is no more connection

Mosh uses one socket, and one remote address: it will not rebind its socket to a different server's address.
→ no IPv4 / IPv6 roaming

Mosh will not try to increase its performances with multiple paths
→ two bad paths may be combined

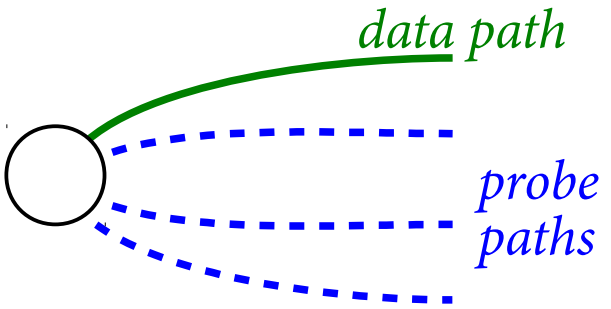
Mosh: an interactive application

Objective: minimize latency (RTT).

MP-mosh measures the RTT of the different paths using probes.

MP-mosh distinguish 2 kinds of packets:

- data packets: contains actual mosh data,
  (transit on The selected path)
- probes: only used for path estimation.
  (transit on other paths)



Mosh already provides classical RTT computation. MP-mosh uses the same mechanism with probes.

The RTT never decreases !!!

The RTT doesn't decrease when the connection is lost: if the best path break, all data packets will be lost!

- evaluate the RTO (Retransmission TimeOut) as in TCP,
- based on this, evaluate the idle time,
- the remote may delayed acknowledgements,
  (takes this into account)
- integrate this to the event-loop.

Probes have a little overhead: 9 B/s for idle paths, 180 B/s max on active paths.

Mosh: a lightweight application

« Why not just duplicating data on all paths? »

→ increase performances on lossy paths,

→ increase useless overhead on good paths,
(I disagree to pay for nothing)

MP-mosh evaluates path loss ratio:

- uses a slicing window (64 packets),
- differentiates loss from reordering,
- sends back loss ratio to the remote peer.

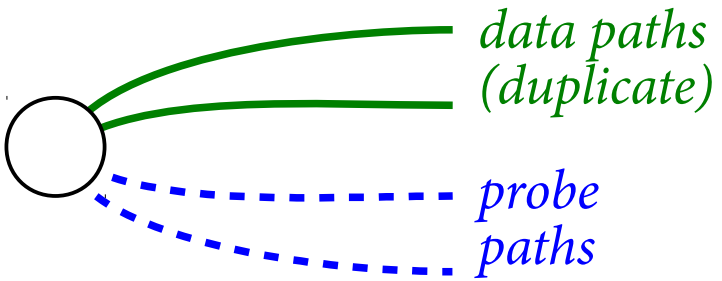
Then duplicates on "some" paths.

MP-mosh sending procedure

SRTT + idle time	10ms	200ms	200ms	…
outgoing loss ratio	42%	12%	19%	…
expected loss ratio	42%	5%	0%	…

sending data

sending probes



Conclusion

MP-mosh is an application designed for
host-centric multihomed networks,
with source-specific routing.

MP-mosh uses (very) lightweight probes estimate
the RTT and loss ratio of each paths
and duplicates to achieve a minimum reliability.

Perhaps a good basis for a library, but:
what if duplicated packets goes to the same bottleneck?
what should we change for peer to peer applications?