MPBond: Efficient Network-level Collaboration Among Personal Mobile Devices

Xiao Zhu    Jiachen Sun    Xumiao Zhang
Y. Ethan Guo†    Feng Qian‡    Z. Morley Mao

University of Michigan
†Uber Technologies, Inc.
‡University of Minnesota – Twin Cities
Ubiquitous Personal Mobile Devices

Multi-device ownership

Group activities
Network-level Collaboration Helps!

- Improve throughput
- Tackle per-device rate limit
- Extend wireless range

Need a software framework to bridge networking hardware 😊
Collaboration Meets Multipath Transport

Simultaneous data transfers over multiple network paths (aka, subflows)

Subflows traverse multiple mobile devices

Traditional multipath transport (e.g., MPTCP)

Need “distributed” multipath transport
Challenges for An Efficient Collaboration Scheme

- Proper management of heterogeneous devices and links
- Efficient leverage of helper devices to improve network performance
- Judicious distribution of data over remote and local paths
- Appropriate interfaces to apps and users
- Transparency of the scheme to client and server apps
Existing Collaboration Schemes Fall Short

• Lack of flexibility
  • Tethering+MPTCP [1], inverse multiplexing [2]
  • Application modifications [3, 4, 5]

• Suboptimal Performance
  • Under fluctuating remote and local network conditions [1, 2]
  • Due to idle times incurred by chunk-based data distribution [3-5]
  • Due to suboptimal scheduling decisions [1-5]

• Excessive energy consumption
  • Long download time [1-5]
  • Prolonged remote or/and local link radio-on time [1-5]

[1] Using cooperation for low power low latency cellular connectivity, CoNEXT 2014
Our Solution: MPBond

A distributed multipath transport system for efficient network-level collaboration
Subflow Management

• Primary-centric pipe establishment to reduce helper-primary hop count
• Handshake is similar to MPTCP but with additional control messages over the pipe
Buffer Management and Connection Split

w/o TCP split and buffering  v.s.  w/ TCP split and buffering

Data download

Bandwidth underutilization

Higher bandwidth utilization

Pipe bandwidth increases after some time
Data Distribution over Multiple Paths

• Realized by a multipath scheduler
• Optimal multipath scheduling requires simultaneous data completion at the receiver
• MinRTT is the default scheduler for MPTCP by selecting the path with available space in congestion window (cwnd) and the minimum RTT
Why Not the Default MinRTT Scheduler?

• MinRTT fails to achieve simultaneous subflow completion in MPBond
  • Due to the lack of consideration of the pipe

How about modifying the subflow availability condition?
• Would lose the capability of buffering
Pipe-aware Multipath Scheduler (PAMS)

- Challenge: enable buffering at the helper while achieving simultaneous subflow completion
- Making packet arrival time estimation pipe-aware
  - Pipe-aware delay: The time it takes for a packet scheduled over a given subflow at a given time to arrive at the receiver
Deriving the Pipe-aware Delay (PAD)

- PAD of the direct subflow ($PAD_1$): $OWD_{ps} + \frac{B_s}{Th_{ps}}$
- For an indirect subflow ($PAD_i$, $i > 1$)

$$PAD_i = \begin{cases} 
OWD_{ps} + \frac{B_s}{Th_{ps}}, & \text{if } i = 1 \\
OWD_{hs} + \frac{B_s + B_p}{Th_p} + OWD_p, & \text{if } i > 1, \frac{B_p}{B_s} + 1 > \frac{Th_p}{Th_{hs}} \\
OWD_{hs} + \frac{B_s}{Th_{hs}} + OWD_p, & \text{if } i > 1, \frac{B_p}{B_s} + 1 \leq \frac{Th_p}{Th_{hs}}
\end{cases}$$
The PAMS Algorithm

• Leverage PAD to make scheduling decisions
  • $minPAD$: select the path with available space in cwnd and minimum $PAD$
    • Useful when there is large amount of remaining data to send (Case 1)
  • When there is only small amount of remaining data (Case 2)?
    • Defer the scheduling!
  • Data Reinjection
User/App Interfaces and Policy Engine

• Users
  • Per-app whitelist, resource usage, prioritization

• Apps
  • Optional APIs for device/pipe monitoring and management

• Dual mode
Evaluation

• MPBond prototype on Android smartphones and smartwatches

• Comparison baselines
  • Single device, kibbutz [1], COMBINE [2]

• Evaluation setup
  • Applications: file download, video streaming
  • Networks: emulated (with `tc`) and real WiFi and LTE
  • Devices: smartphones (Pixel 2, Nexus 6P) and a smartwatch (LG Urbane 2nd)

[1] Using cooperation for low power low latency cellular connectivity, CoNEXT 2014
Evaluation: Stable network conditions

- File download
  - Primary: Pixel 2, Helper 1: Nexus 6P, Helper 2: LG Urbane 2nd
  - PS-Path: 8Mbps, HS-Path: 10Mbps, pipe: 5Mbps

1. Using more devices reduces download time, with reasonable increase of total energy
2. MPBond reduces energy and download time compared to kibbutz and COMBINE
Evaluation: Varying network conditions

Replay real WiFi and LTE bandwidth traces

In-the-wild experiments

MPBond reduces the file download time by 13%-35%, which also translates to lower energy consumption, compared to kibbutz and COMBINE
Evaluation: Video streaming

- Three video sources: Big Buck Bunny w/ 2-sec segments (B2), Tears of Steel w/ 2-sec segments (T2) and 6-sec segments (T6)
- PS-Path: 5Mbps, HS-Path: 10Mbps, pipe: 5Mbps

1. MPBond reduces energy consumption compared to kibbutz with same # of devices
2. With 3 devices, MPBond improves the video bitrate by up to 118% compared to kibbutz
Conclusion

• Mobile devices need network-level collaboration
• Collaboration made efficient & easy by MPBond
  • Distributed multipath transport
  • Device, connection, and buffer management
  • Judicious pipe-aware scheduling
• MPBond prevails over existing collaboration schemes
  • Performance, energy efficiency, and flexibility

Thank you! Questions?