システム系論文の読み方と探し方

本多 倫夫
エディンバラ大学

- 論文とは
- 会議と雑誌
- 論文の探し方
- 論文の選び方

論文とは

論文とは？

- 研究成果が述べられた文書
- その研究の意義が書いてある
  - Problem
    - その研究がどのような問題にアタックしているか
    - なぜその問題が重要か
  - Solution
    - その問題をどう解決したか
  - 得られた知見
    - 研究の結論

論文を見てみよう (1/3)

タイトル

著者リスト

論文の概要

第 1 章 はじめに

How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP

Costin Raiciu[†], Christoph Paasch[‡], Sebastien Barre[†], Alan Ford,
Michio Honda[*], Fabien Duchene[‡], Olivier Bonaventure[‡] and Mark Handley[*]
[†]Universitatea Politehnica Bucuresti, [‡]Universite Catholique de Louvain
[*]Keio University, [*]University College London

ABSTRACT

Networks have become multipath: mobile devices have multiple radio interfaces, datacenters have redundant paths and multihoming is the norm for big server farms. Meanwhile, TCP is still only single path.

Is it possible to extend TCP to enable it to support multiple paths for current applications on today's Internet? The answer is positive. We carefully review the constraints—partly due to various types of middleboxes—that influenced the design of Multipath TCP and show how we handled them to achieve its deployability goals.

We report our experience in implementing Multipath TCP in the Linux kernel and we evaluate its performance. Our measurements focus on the algorithms needed to efficiently use paths with different characteristics, notably send and receive buffer tuning and segment reordering. We also compare the performance of our implementation with regular TCP on web servers. Finally, we discuss the lessons learned from designing MPTCP.

1. INTRODUCTION

In today's Internet, servers are often multi-homed to more than one Internet provider, datacenters provide multiple parallel paths between compute nodes, and mobile hosts have multiple radios. Traditionally, it was the role of routing to take advantage of path diversity, but this has limits to responsiveness and scaling. To really gain both robustness and performance advantages, we need transport protocols engineered to utilize multiple paths. Multipath TCP[5] is an attempt to extend the TCP protocol to perform this role. At the time of writing, it is in in the final stage of standardization in the IETF.

Multipath TCP stripes data from a single TCP connection across multiple subflows, each of which may take a different path through the network. A linked congestion control mechanism[23] controls how much data is sent on each subflow, with the goal of explicitly moving traffic off the more congested paths onto the less congested ones. This paper is not about congestion control, but rather it is about the design of the Multipath TCP protocol itself. In principle, extending TCP to use multiple paths is not difficult, and there are a number of obvious ways in which it could be done. Indeed it was first proposed by Christian Huitema in 1995[11]. In practice though, the existence of middleboxes greatly constrains the design choices. The challenge is to make Multipath TCP not only robust to path failures, but also robust to failures in the presence of middleboxes that attempt to optimize single path TCP flows. No previous extension to the core Internet protocols has needed to consider this issue to nearly the same extent.

In the first half of this paper we examine the design options for multipath TCP, with the aim of understanding both the end-to-end problem and the end-to-middle-to end constraints. We use the results of a large Internet study to validate these design choices.

Designing MPTCP turned out to be more difficult than expected. For instance, a key question concerns how MPTCP metadata should be encoded — embed it in the TCP payload, or use the more traditional TCP options, with the potential for interesting interactions with middleboxes. In the IETF opinions were divided, with supporters on both sides. In the end, careful analysis revealed that MPTCP needs explicit connection level acknowledgments for flow control; further, these acknowledgments can create deadlocks if encoded in the payload. In reality, there was only one viable choice.

The second half of this paper concerns the host operating system. To be viable, Multipath TCP must be implementable in modern operating systems and must perform well. We examine the practical limitations the OS poses on MPTCP design and operation. This matters: our experiments show that one slow path can significantly degrade the throughput of the whole connection when MPTCP is underbuffered. We propose novel algorithms that increase throughput even in this case, ensuring MPTCP always matches what TCP would get on the best interface, regardless of buffer size.

It is not our goal to convince the reader that multipath transport protocols in general are a good idea. There has been a wealth of work that motivates the use of multipath transport for robustness[24], the use of linked congestion control across multiple paths for load balancing[23, 14, 4] and the ability of multi-path transport protocols

https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final125.pdf

論文を見てみよう (2/3)

第 2 章 研究のゴール

第 3 章 設計

to find and utilize unused network capacity in redundant topologies[10, 19]. Rather, the main contribution of this paper is the exploration of the design space for MPTCP confined by the many constraints imposed by TCP's original design, today's networks which embed TCP knowledge, and the need to perform well within the limitations imposed by the operating system.

2. GOALS

As many researchers have lamented, changing the behavior of the core Internet protocols is very difficult [7]. An idea may have great merit, but without a clear deployment path whereby the cost/benefit tradeoff for early adopters is positive, widespread adoption is unlikely.

We wish to move from a single-path Internet to one where the robustness, performance and load-balancing benefits of multipath transport are available to all applications, the majority of which use TCP for transport. To support such unmodified applications we must work below the sockets API, providing the same service as TCP: byte-oriented, reliable and in-order delivery. In theory we could use different protocols to implement this functionality as long as fallback to TCP is possible when one end does not support multipath. In practice there is no widely deployed signaling mechanism to select between transport protocols, so we have to use options in TCP's SYN exchange to negotiate new functionality.

The goal is for an unmodified application to start (what it believes to be) a TCP connection with the regular API. When both endpoints support MPTCP and multiple paths are available, MPTCP can set up additional subflows and stripe the connection's data across these subflows, sending most data on the least congested paths.

The potential benefits are clear, but there may be costs too. If negotiating MPTCP can cause connections to fail when regular TCP would have succeeded, then deployment is unlikely. The second goal, then, is for MPTCP to work in all scenarios where TCP currently works. If a subflow fails for any reason, the connection must be able to continue as long as another subflow has connectivity.

Third, MPTCP must be able to utilize the network at least as well as regular TCP, but without starving TCP. The congestion control scheme described in [23] meets this requirement, but congestion control is not the only factor that can limit throughput.

Finally MPTCP must be implementable in operating systems without using excessive memory or processing. As we will see, this requires careful consideration of both fast-path processing and overload scenarios.

3. DESIGN

The five main mechanisms in TCP are:

- Connection setup handshake and state machine.
- Reliable transmission & acknowledgment of data.
- Congestion control.
- Flow control.
- Connection teardown handshake and state machine.

The simplest possible way to implement Multipath TCP would be to take segments coming out of the regular stack and "stripe" them across the available paths *somehow*[1]. For this to work well, the sender would need to know which paths perform well and which don't: it would need to measure per path RTTs to quickly and accurately detect losses. To achieve these goals, the sender must remember which segments it sent on each path and use TCP Selective Acknowledgements to learn which segments arrive. Using this information, the sender could drive retransmissions independently on each path and maintain congestion control state.

This simple design has one fatal flaw: on each path, Multipath TCP would appear as a discontinuous TCP bytestream, which will upset many middleboxes (our study shows that a third of paths will break such connections). To achieve robust, high performance multipath operation, we need more substantial changes to TCP, touching all the components listed above. Congestion control has been described elsewhere[23] so we will not discuss it further in this paper.

In brief, here is how MPTCP works. MPTCP is negotiated via new TCP options in SYN packets, and the endpoints exchange connection identifiers; these are used later to add new paths—subflows—to an existing connection. Subflows resemble TCP flows on the wire, but they all share a single send and receive buffer at the endpoints. MPTCP uses per subflow sequence numbers to detect losses and drive retransmissions, and connection-level sequence numbers to allow reordering at the receiver. Connection-level acknowledgements are used to implement proper flow control. We discuss the rationale behind these design choices below.

3.1 Connection setup

The TCP three-way handshake serves to synchronize state between the client and server[2]. In particular, initial sequence numbers are exchanged and acknowledged, and TCP options carried in the SYN and SYN/ACK packets are used to negotiate optional functionality.

MPTCP must use this initial handshake to negotiate multipath capability. An MP_CAPABLE option is sent in the SYN and echoed in the SYN/ACK if the server

[1]such *striping* needs additional mechanisms because both destination-based forwarding and network ECMP try hard not to stripe packets belonging to the same TCP connection

[2]The correct terms should be *active opener* and *passive opener*. For conciseness, we use the terms client and server, but we do not imply any additional limitations on TCP usage.

https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final125.pdf

論文を見てみよう (3/3)

the semantics of the window field of the TCP header and extends it beyond 16 bits. Nearly 20 years after the publication of RFC1323, there are still stateful firewalls that do not understand this option in SYNs but block data packets that are sent in the RFC1323 extended window. A TCP extension that changes the semantics of parts of the packet header must include mechanisms to cope with middleboxes that do not understand the new semantics.

In an end-to-end Internet, all the information carried inside TCP packets is immutable. Today this is no longer true: the entire TCP header and the payload must be considered as mutable fields. If a TCP extension needs to rely on a particular field, it must check its value in a way that cannot be circumvented by middleboxes that do not understand this extension. The DSM checksum is an example of a solution to deal with these problems.

Most importantly, deployable TCP extensions must necessarily include techniques that enable them to fall-back to regular TCP when something wrong happens. If a middlebox interferes badly with a TCP extension, the problem must be detected and the extension automatically disabled to preserve the data transfer. A TCP extension will only be deployed if it guarantees that it will transfer data correctly (and hopefully better) in all the cases where a regular TCP is able to transfer data.

8. CONCLUSIONS

TCP was designed when the Internet strictly obeyed the end-to-end principle and each host had a single IP address. Single-homing is disappearing and a growing fraction of hosts have multiple interfaces/addresses. In this paper we evaluated whether TCP can be extended to efficiently support such hosts.

We explored whether it was possible to design Multipath TCP in a way that is still deployable in today's Internet. The answer is positive, but any major change to TCP must take into account the various types of middleboxes that have proliferated. In fact, they influenced most of the design choices in Multipath TCP besides the congestion control. Our experiments show that MPTCP safely operates through all the middleboxes we've identified in our previous study [9].

From an implementation viewpoint, we proposed new algorithms to solve practical but important problems such as sharing a limited receive buffer between multiple flows on a smartphone, or optimizing the MPTCP receive code. Experiments show that our techniques are effective, making MPTCP ready for adoption.

This work highlights once again the fact that hidden middleboxes increase the complexity of the Internet, making evolution difficult. We should revisit the Internet architecture to recognize explicitly their role. The big challenge, however, is to build a solution that is deployable in today's Internet.

Acknowledgments

We would like to thank all the members of the Trilogy project and of the MPTCP IETF working-group who have helped us to shape the MPTCP protocol. Xinxing Hu wrote the first version of the Click elements that model middlebox behaviours.

This work has been supported partially by Google through a University Research project and by FP7 projects ECODE and CHANGE. Christoph Paasch has partially been supported by Nokia Research.

9. REFERENCES

[1] S. Barré. *Implementation and Assessment of Modern Host-based Multipath Solutions*. PhD thesis, Université catholique de Louvain, November 2011.

[2] M. Becke et al. Load sharing for the stream control transmission protocol (sctp). Internet draft, draft-tuexen-tsvwg-sctp-multipath-02.txt, work in progress, July 2011.

[3] A. Bittau, M. Hamburg, M. Handley, D. Mazieres, and D. Boneh. The case for ubiquitous transport-level encryption. In *USENIX Security'10*, pages 26–26, Berkeley, CA, USA, 2010. USENIX Association.

[4] Y. Dong, D. Wang, N. Pissinou, and J. Wang. Multi-path load balancing in transport layer. In *EuroNGI Conference*, 2007.

[5] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP extensions for multipath operation with multiple addresses, Jan 2012. IETF draft (work in progress).

[6] FreeBSD Project. tcp – internet transmission control protocol. *FreeBSD Kernel Interfaces Manual*.

[7] M. Handley. Why the internet only just works. *BT Technology Journal*, 24:119–129, 2006.

[8] M. Handley, V. Paxson, and C. Kreibich. Network intrusion detection: evasion, traffic normalization, and end-to-end protocol semantics. In *Proc. USENIX Security Symposium*, pages 9–9, 2001.

[9] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is it still possible to extend TCP? In *IMC 2011, 11th Internet Measurement Conference*, Nov. 2011.

[10] H-Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. In *Proc. MobiCom '02*, pages 83–94, New York, NY, USA, 2002. ACM.

[11] C. Huitema. Multi-homed TCP. Internet draft, IETF, 1995.

[12] J. Iyengar, P. Amer, and R. Stewart. Performance implications of a bounded receive buffer in concurrent multipath transfer. *Computer Communications*, 30(4), February 2007.

[13] J. R. Iyengar, P. D. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *IEEE/ACM Trans. Netw.*, 14(5):951–964, 2006.

[14] P. Key, L. Massoulié, and D. Towsley. Path selection and multipath congestion control. In *Proc. IEEE Infocom*, May 2007.

[15] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18:263–297, August 2000.

[16] I. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. *ICNP*, page 0165, 2001.

[17] F. Mirani, M. Khemraj, and N. Boukhatem. Forward prediction scheduling: Implementation and performance evaluation. In *ICT 2011*, pages 321–326, may 2011.

[18] C. Pluntke, L. Eggert, and N. Kiukkonen. Saving mobile device energy with Multipath TCP. In *MobiArch*, 2011.

[19] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with Multipath TCP. In *Proc ACM Sigcomm*, 2011.

[20] C. Raiciu, D. Niculescu, M. Bagnulo, and M. Handley. Opportunistic mobility with Multipath TCP. In *MobiArch*, 2011.

[21] K. Rojviboonchai and H. Aida. An evaluation of multi-path transmission control protocol (M/TCP) with robust acknowledgement schemes. *IEICE Trans. Communications*, 2004.

[22] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) terminology and considerations. RFC 2663, August 1999.

[23] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI'11*, Berkeley, CA, USA, 2011. USENIX Association.

[24] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *Proc. USENIX '04*, 2004.

第 8 章
結論/まとめ

参考文献 [3] A. Bittau, M. Hamburg, M. Handley, D. Mazie`res, and D. Boneh. The case for ubiquitous transport-level encryption. In *USENIX Security'10*, pages 26–26, Berkeley, CA, USA, 2010. USENIX Association.

著者リスト　タイトル
発表された会議/ジャーナル
論文集内でのページ番号
出版社 (学会) 等

https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final125.pdf

論文の読み方

- 以下がとても参考になる
  - http://ccr.sigcomm.org/online/files/p83-keshavA.pdf
- The Three-Pass Approach
  - (The first pass) まずはざっと読む
    - タイトル、アブスト、イントロ、セクションのタイトル etc.
  - (The second pass) 次にちょっと深く読む
    - 証明等の詳細はまだ無視
  - (The third pass) 最後に深く読む

論文の種類

論文の種類

- Extended abstract – アイデア段階の研究
  - Double-column 2 枚か single-column 4 枚程度で、ポスター発表
- Workshop paper – 途中結果段階の研究
  - Double-column 4-6 枚程度か、single-column 8-12 枚程度
  - Workshop と名前がついていても、量が extended abstract と変わらないものは実質的には extended abstract と同等
  - Conference paper の中には short paper というカテゴリがある場合があり完成された研究の場合と workshop paper とほぼ同じ場合があるのでその workshop の Call for Papers を要参照
- **Conference** or Journal paper – 完成している研究
  - Double-column 5-16 枚程度か、single-column 10-24 枚程度

会議と雑誌 (1/2)

- 論文は、会議 (**Conference**) の論文集 (Proceedings) として、あるいは雑誌 (**Journal**) にて出版される
  - Journal は transaction または論文誌とも呼ばれる
  - 論文を参照したりする時には、タイトル、著者と共に会議名や雑誌名が併記される
  - 論文中の参考文献の中の表記をみてみよう

[3] A. Bittau, M. Hamburg, M. Handley, D. Mazie`res, and D. Boneh. The case for ubiquitous transport-level encryption. In *USENIX Security'10*, pages 26–26, Berkeley, CA, USA, 2010. USENIX Association.

著者リスト　　タイトル
発表された会議/ジャーナル
論文集内でのページ番号
出版社 (学会) 等

会議と雑誌 (2/2)

- 会議や雑誌は、学会によって運営される
  - IEEE, ACM, USENIX など
  - 日本独自の、情報処理学会 (IPSJ)、電子情報通信学会 (IEICE)なども存在する
  - なので、会議名を表すときには、
    IEEE INFOCOM, ACM SIGCOMM のように、学会名 + Conference 名または Journal/transaction 名で表すことが多い

論文が出版されるまで (1/3)

- Conference 論文
  - Conference がホテルや会議場、大学等で数日～1週間程度開催される
  - 研究者が自分達の論文に対するプレゼンを行う
    - 開催前に論文が募集される (Call for Papers)
    - 投稿された論文が審査される (査読)
      - 審査プロセスの詳細: https://note.com/michiohjp/n/nd2608d3483fe
    - 会議の web ページに論文が掲載される
      - それに加え、学会のオンライン図書館 (後述) に置かれたりも
      - ある会議の同じ年の論文集は Proceedings とも呼ばれる

論文が出版されるまで (2/3)

- Journal 論文
  - 論文募集、査読を経て学会の定期刊行物である Journal/Transaction に載る (プレゼンテーションはなし)

会議/雑誌とテーマについて

- 会議や雑誌毎に扱う**テーマ**や**広さ**が異なる
- ネットワーク系
  - 全般: CoNext, INFOCOM, NSDI, SIGCOMM
  - 特定テーマ: IMC (計測), SOSR (SDN/NFV)
- システム系
  - 全般: ASPLOS, ATC, EuroSys, NSDI, OSDI, SOSP
  - 特定テーマ: FAST (ストレージ), SoCC (クラウド)
- モバイル系
  - 全般: Mobicom, Mobisys
  - 特定テーマ: SenSys (センサー)

Conference/Journalと論文のレベル (1/2)

- 様々なレベル (要求される研究のクオリティ) のConference/Journalがある
- レベルが高い所で論文を発表するためには、質の高い研究が要求される
  - 会議のレベルの指標としてインパクトファクター (被引用数/発行された論文数) は相関がある程度であまりあてにならない
  - 投稿される論文全体のレベルが低い会議もたくさんあるので論文の採択率 (採択数/投稿数) もあてにならない
- よい論文はインパクトが大きく他の**ハイレベルなカンファレンスやワークショップの**論文から引用される
  - 数は多くても微妙な論文からしか引用されてない論文もある

Conference/Journalと論文のレベル (2/2)

- Journal 論文の位置づけ
  - 昔は Conference 論文より発展したレベルの高い研究という位置づけだったらしいが、今のコンピュータサイエンス分野ではよい論文は集まりにくい傾向
  - 現在の一流大学では、PhD 取得やファカルティ職の獲得においてジャーナル論文ではだめでトップカンファレンス論文が要求されることも多い
- トップカンファレンスとは
  - 具体的な会議名は人によって異なるが、意味のある基準は、国際社会における上位校*でファカルティ職を得るための主要論文となりうるかどうか

*アメリカ型の公募時期 (冬) /応募書類/選考基準/プロセスがとられるためほぼ同時期に応募可能で、世界中の卒業間近の博士やポスドク、ファカルティがポジションを競い合う。英米加星などに多い。選考プロセスの詳細は以下:
https://note.com/michiohjp/n/n86f330adf666

演習: 会議の web ページをみてみよう

- USENIX NSDI 2019
  - https://www.usenix.org/conference/nsdi19/
  - トップページを見てみよう
- Call for Papers を見てみよう
  - 画面上部の PARTICIPATE タブの中
- Program を見てみよう

論文の探し方

論文はどこから入手できるか？

- 会議の web ページのプログラム（一番よく使う）
- 著者のホームページ（次によく使う）
- 各学会（ACM, IEEE など）のオンライン図書館
  - ACM Digital Library（dl.acm.org）や ieeeExplore（http://ieeexplore.ieee.org/）
  - 学会の会員にならないとアクセスできない論文もある
    - 大学が学会と契約を結んでいて学内のネットワークからなら会員にならなくてもそういう論文にアクセスできることも多い
    - ただし昨今では一流の論文は、お金を払ったり会員にならないと論文にアクセスできない会議やジャーナルでは殆ど発表されない

論文の探し方

- Google Scholar
  - http://scholar.google.com/
  - Google image とかの論文版だと思えばいい
  - タイトルとかキーワードとか著者とかを検索窓に入れる
- Microsoft Academic Search というのもあるが、情報が古かったりでおすすめしない

演習: Google Scholar

- 身近な人の論文を探してみよう
  - Google Scholar (http://scholar.google.com/) の検索窓に名前を入力

論文の選び方

- 論文の質は玉石混合
- いい論文の例
  - たくさん引用されている論文 *1
    - Google scholar の検索結果に表示されている
  - 実システムに適用されている論文
    - 例) Windows の TCP 輻輳制御アルゴリズムの論文*2
- 直近の論文は次のスライドへ

*1 但しどこから引用されているかが大事 (スライド 21 を参照)

*2 Tan et al, A compound TCP approach for high-speed and long distance networks, IEEE INFOCOM 2006

論文の探し方 (1/3)

- **ぼんやりとある研究分野の最近の動向などを知りたい時**
  - 研究ネタを探したい時とかに有効
- その分野のトップカンファレンスのプログラムを見る
  - スライド 20 枚目を参照
- その分野のトップワークショップのプログラムを見る
  - カンファレンス論文は研究が始まってから時間が経ったものが多いがワークショップは初期から中期の研究が集まる。
  - 一部のワークショップ (HotNets, HotOS, HotStorage) は非常に高難易度で有力研究者からも高く評価される
- 著名研究者のキーノートスピーチも参考になることがある
  - トランスポート技術の歴史: https://dl.acm.org/doi/10.1145/3341302.3359768

論文の探し方 (2/3)

- **探したいテーマ等がある程度判っている場合**
  - ある程度具体的なネタを思いついたりして、それに対する関連研究を調べたい時等
  - キーワード、その分野での著名人の名前などを駆使して、一つ、関連の深そうな論文を見つける
    - 業界での著名人=研究者としての著名人ではないことに注意
  - その論文の参考文献を見る
  - それぞれがどう参照されているかの情報などを元に、それをたどる
  - また逆に、その論文がどんな論文から参照されているかを見るのも有効 (Google scholar でできる)
  - 関係のある論文の、関連研究の章を読む
  - そのうちその問題や技術の変遷や全体像、本質がだんだん見えてくる

論文の探し方 (3/3)

- **どんなテーマがあるかもわからない！どの論文を読めばいいかわからない！そもそも自分が何に興味があるのかわからない！**
  -> リーディングリストを読もう！
  - 星の数ほどある論文からエキスパートが厳選してくれている
  - UC Berkeley のネットワーク系論文読みの授業
    - https://www.eecs.berkeley.edu/~sylvia/cs268-2014/syllabus.html
    - 昔の論文から新しい論文まで、ネットワーク分野全体をカバー
  - 他にもいろいろあるけど古い論文しかカバーしてなかったり分野が偏っていたりするので最初は上記のやつがおすすめ

まとめ

- 論文を読もう
- 一流の論文に触れよう
  - 研究に限らず、一流に触れることは大事
  - 一流の論文は、その内容以上のものを教えてくれます
  - 一流のカンファレンスやワークショップに出ている論文を読むべし
    - 日本の学会の論文は読む/書く必要ないです
- 論文を書こう
  - 国際社会に出れるチャンスが増える！