

Network Latency Control in Data Centres

Edited by

Mohammad Alizadeh Attar¹, Jon Crowcroft², Lars Eggert³, and Klaus Wehrle⁴

1 MIT – Cambridge, US, alizadeh@csail.mit.edu

2 University of Cambridge, GB, jon.crowcroft@cl.cam.ac.uk

3 NetApp Deutschland GmbH – Kirchheim, DE, lars@netapp.com

4 RWTH Aachen, DE, wehrle@comsys.rwth-aachen.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 16281 “Network Latency Control in Data Centres”. This seminar explored existing and future techniques for controlling data centre latency and thus explores research directions in the new field of data centre latency control in networking research. This need for a new research direction is motivated by the fact that traditional networking equipment and TCP-IP stacks were designed for wide-area networks, where the goal is to maximize throughput, and the control loop between end systems is measured in 10s of milliseconds. Consequently, this seminar discussed new research direction for data center latency control across the entire software and hardware stack, including in-network solutions, end-host solutions, and others.

Seminar July 10–13, 2016 – <http://www.dagstuhl.de/16281>

1998 ACM Subject Classification C.2.1 Network Architecture and Design, C.2.2 Network Protocols, C.2.3 Network Operations, D.4.4 Communications Management

Keywords and phrases data centres, end-to-end transport protocols., latency, network architecture, resource control, scheduling

Digital Object Identifier 10.4230/DagRep.6.7.15

Edited in cooperation with Oliver Hohlfeld

1 Executive Summary

Oliver Hohlfeld

Mohammad Alizadeh Attar

Jon Crowcroft

Lars Eggert

Klaus Wehrle

License © Creative Commons BY 3.0 Unported license

© Oliver Hohlfeld, Mohammad Alizadeh Attar, Jon Crowcroft, Lars Eggert, and Klaus Wehrle

Data centres are at the heart of the modern Internet. They host web services, social networking, cloud computing and are increasingly used by operators to host virtual network functions. All these services have one thing in common: they require extremely low latency communication in the data centre. Consequently we have seen the birth of a new field in networking research – data centre latency control.

Unlike the earlier generation of high-performance computing clusters, data centres have tended to use commodity off-the-shelf servers and switches, and run standard operating systems. However, traditional networking equipment and TCP-IP stacks were designed



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Network Latency Control in Data Centres, *Dagstuhl Reports*, Vol. 6, Issue 7, pp. 15–30

Editors: Mohammad Alizadeh Attar, Jon Crowcroft, Lars Eggert, and Klaus Wehrle



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for wide-area networks, where the goal is to maximize throughput, and the control loop between end systems is measured in 10s of milliseconds. By contrast, data centres operate on timescales that are several orders of magnitude lower. And while throughput is important, the plentiful bandwidth of data centre networks makes throughput a secondary concern to latency.

This seminar explored existing and future techniques for controlling data centre latency across the entire software and hardware stack, including in-network solutions, end-host solutions, and others. The aims of the seminar are to foster closer collaboration between academic researchers, industry, and operators. 38 researchers attended the multidisciplinary seminar. Over the course of the 3-day seminar, seven presentations were given on various aspects of data center networking. Taking the presentations as input, the workshop then broke into six working groups to discuss research aspects of latency control. The seminar was concluded by voting and discussing on possible conclusions from our discussions. Each conclusion was discussed briefly, then voted on. The outcome of the breakout session as well as the concluding statements are summarized in this report.

2 Table of Contents

Executive Summary

Oliver Hohlfeld, Mohammad Alizadeh Attar, Jon Crowcroft, Lars Eggert, and Klaus Wehrle 15

Overview of Talks

Data Centre to the Home; Low Delay for All
Bob Briscoe 19

When does a data plane become an “OS done right”?
Edouard Bugnion 19

Consensus as a Network Service
Marco Canini 20

Recent Advances in Coflow-Based Networking
Mosharaf Chowdhury 21

A New Software Stack for Low-Latency Datacenters
John Ousterhout 22

NDP: New Datacenter Philosophy/Protocol
Costin Raiciu 22

Where has the time gone?
Noa Zilberman 22

Working groups

Low latency outside the DC (5G, open Internet, etc)
Jari Arkko, Bob Briscoe, Jon Crowcroft, Koen De Schepper, Lars Eggert, Michio Honda, Wolfgang Kellerer, Kirill Kogan, Mirja Kühlewind, and Michael Scharf . . . 23

How to make datacenter research more reproducible
Marinho Barcellos, Olivier Bonaventure, Edouard Bugnion, Oliver Hohlfeld, Andrew W. Moore, Hakim Weatherspoon, and Noa Zilberman 23

Switch programmability: How much functionality should be handled by switches?
Matthew P. Grosvenor 24

Congestion Control for 100 G networks
Dongsu Han, Alexandru Agache, Mohammad Alizadeh Attar, Marco Canini, Nandita Dukkhipati, Matthew P. Grosvenor, Patrick Jahnke, Lavanya Jose, Mike Marty, John Ousterhout, Rong Pan, Ihsan Ayyub Qazi, Costin Raiciu, and Michael Scharf 25

Role of the Operating System (Group 2)
Mike Marty, Jari Arkko, Patrick Thomas Eugster, Oliver Hohlfeld, and Andrew W. Moore 26

Role of the Operating System (Group 1)
John Ousterhout, Edouard Bugnion, Marco Canini, Nandita Dukkhipati, and Jitendra Padhye 26

Panel discussions

Concluding Statements
John Ousterhout 27

Open problems

Questions about flow aggregation	
<i>Michael Welzl</i>	29
Participants	30

3 Overview of Talks

3.1 Data Centre to the Home; Low Delay for All

Bob Briscoe (Simula Research Laboratory – Lysaker, NO)

License © Creative Commons BY 3.0 Unported license
© Bob Briscoe

This talk is about extending the applicability of technology for reducing queuing delay from private data centres to heterogenous data centres and to the wider Internet. Data Centre TCP (DCTCP) can be thought of as an example of how we would now design TCP congestion control given our improvements in understanding since TCP was originally implemented in 1988. As flow rates have increased, the variance of the well-known sawtooth rate of TCP has grown proportionately, causing large fluctuations in queuing delay. DCTCP solves this problem, so it can consistently keep queuing delay extremely low, whatever the flow rate. DCTCP is therefore categorised as a ‘Scalable’ congestion control. However, DCTCP requires changes to all senders, all receivers and all switches in a network, therefore (until now) it has not been feasible to deploy it on the public Internet, particularly because it is very aggressive, so existing ‘Classic’ TCP algorithms tend to starve themselves when they occupy the same queue.

This talk is about a simple technology called the Coupled Dual Queue that solves this coexistence problem without inspecting flows or anything deeper than the outer headers of packets or frames. It allows Scalable congestion controls to coexist with existing Classic controls. It is like a semi-permeable membrane that isolates the new scalable flows from the queuing delay of classic traffic, but the two share the capacity of a link roughly equally as if they were both the same type of TCP sharing a single queue. With this advance, we show that all traffic on the public Internet could enjoy consistently low queuing delay. This should enable new applications that require natural interaction over public networks, such as offloading interactive video processing to cloud data centres. The technology also enables the low queuing delay of scalable congestion controls like DCTCP to be deployed in multi-tenant data centres, or networks of data centres, where there are too many independent administrators to arrange a simultaneous ‘flag-day’ deployment.

3.2 When does a data plane become an “OS done right”?

Edouard Bugnion (EPFL Lausanne, CH)

License © Creative Commons BY 3.0 Unported license
© Edouard Bugnion

Main reference G. Prekas, M. Primorac, A. Belay, C. Kozyrakis, E. Bugnion, “Energy proportionality and workload consolidation for latency-critical applications,” in Proc. of the 6th ACM Symp. on Cloud Computing (SoCC’15), pp. 342–355, ACM, 2015.

URL <http://dx.doi.org/10.1145/2806777.2806848>

Main reference A. Belay, G. Prekas, A. Klimovic, S. Grossman, C. Kozyrakis, E. Bugnion, “IX: A Protected Dataplane Operating System for High Throughput and Low Latency,” in Proc. of the 11th USENIX Symp. on Operating Systems Design and Implementation (OSDI’14), pp. 49–65, USENIX Association, 2014.

URL <https://www.usenix.org/conference/osdi14/technical-sessions/presentation/belay>

In a (software) data plane architecture, the application bypasses the OS for most interactions and in particular I/O, e.g. using DPDK or some comparable user level networking framework. Data plane increase throughput and reduce jitter by first statically provisioning resources

(cores and network queues) and then combining networking polling, bounded batching, and run-to-completion techniques. They can be build within an application [4, 5], by multi-threading networking with application [3], on top of research operating systems [1], or using hardware virtualization [2].

Of course, the focus on certain metrics comes at the expense of others, and points to two key limitations not found in conventional operating systems. (1) The static allocation of core that poll network queues leads to poor resource efficiency, e.g. energy proportionality or workload consolidation; (2) the run-to-completion model leads to a FCFC scheduling disciplines, which works fine in some cases (e.g., when the service time of packets is centrally distributed), but not for long-tailed or bi-modal situations.

The open question is whether #1 and #2 can be addressed without reducing the performance of the data plane model, or whether the feature creep will turn the data plane model into a regular operating system architecture. My immediate published work addresses problem #1 through a dynamic resource controls and low-level flow group migration mechanisms. Our current (very raw) research agenda looks at problem #2 through a data plane scheduler that combines FCFS with processor sharing.

References

- 1 Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, Thomas Anderson, and Timothy Roscoe. *Arrakis: The Operating System is the Control Plane*. OSDI, 2014.
- 2 Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, and Edouard Bugnion. *IX: A Protected Dataplane Operating System for High Throughput and Low Latency*. OSDI, 2014.
- 3 EunYoung Jeong, Shinae Wood, Muhammad Jamshed, Haewon Jeong, Sunghwan Ihm, Dongsu Han, and KyoungSoo Park. *mTCP: a Highly Scalable User-level TCP Stack for Multicore Systems*. NSDI, 2014.
- 4 Hyeontaek Lim, Dongsu Han, David G. Andersen, and Michael Kaminsky. *MICA: A Holistic Approach to Fast In-Memory Key-Value Storage*. NSDI, 2014.
- 5 Ilias Marinos, Robert Watson, and Mark Handley. *Network stack specialization for performance*. ACM SIGCOMM, 2014.

3.3 Consensus as a Network Service

Marco Canini (University of Lowain, BE)

License  Creative Commons BY 3.0 Unported license
© Marco Canini

Joint work of Marco Canini, Tu Dang, Pietro Bressana, Han Wang, Ki Suh Lee, Hakim Weatherspoon, Fernando Pedone, Robert Soulé

The Paxos protocol is the foundation for building many fault-tolerant distributed systems and services. This talk posits that there are significant performance benefits to be gained by implementing Paxos logic in network devices. Until recently, the notion of a switch-based implementation of Paxos would be a daydream. However, new flexible hardware and expressive data plane programming languages are on the horizon and will provide customizable packet processing pipelines needed to implement Paxos. This talk describes an implementation of Paxos in one of those languages, P4, as well as our on-going efforts to evaluate the implementation on a variety of hardware devices. Implementing Paxos provides a critical use case for P4, and will help drive the requirements for data plane languages in

general. In the long term, we imagine that consensus could someday be offered as a network service, just as point-to-point communication is provided today.

References

- 1 Huynh Tu Dang, Daniele Sciascia, Marco Canini, Fernando Pedone, and Robert Soulé. *NetPaxos: Consensus at Network Speed*. SOSR, 2015.
- 2 Huynh Tu Dang, Marco Canini, Fernando Pedone, and Robert Soulé. *Paxos Made Switch-y*. SIGCOMM Comput. Commun. Rev., Apr., 2016.
- 3 Huynh Tu Dang, Pietro Bressana, Han Wang, Ki Suh Lee, Hakim Weatherspoon, Marco Canini, Fernando Pedone, and Robert Soulé. *Network Hardware-Accelerated Consensus*. abs/1305.7429, 2016.

3.4 Recent Advances in Coflow-Based Networking

Mosharaf Chowdhury (University of Michigan – Ann Arbor, US)

License © Creative Commons BY 3.0 Unported license
© Mosharaf Chowdhury

Joint work of Mosharaf Chowdhury, Ion Stoica, Yuan Zhong, Zhenhua Liu, Matei Zaharia

Main reference M. Chowdhury, Y. Zhong, I. Stoica, “Efficient coflow scheduling with Varys,” in Proc. of the 2014 ACM Conf. of the Special Interest Group on Data Communication (SIGCOMM’14), pp. 443–454, ACM, 2014.

URL <http://dx.doi.org/10.1145/2619239.2626315>


Over the past decade, the confluence of an unprecedented growth in data volumes and the rapid rise of cloud computing has fundamentally transformed systems software and corresponding infrastructure. To deal with massive datasets, more and more applications today are scaling out to large datacenters. Communication between the distributed computation tasks of these applications often result in massive data transfers over the network. Consequently, concentrated efforts in both industry and academia have gone into building high-capacity, low-latency datacenter networks at scale. At the same time, researchers and practitioners have proposed a wide variety of solutions to minimize flow completion times or to ensure per-flow fairness based on the point-to-point flow abstraction that forms the basis of the TCP/IP stack.

We observe that despite rapid innovations in both applications and infrastructure, application- and network-level goals are moving further apart. Data-parallel applications care about all their flows, but today’s networks treat each point-to-point flow independently. This fundamental mismatch has resulted in complex point solutions for application developers, a myriad of configuration options for end users, and an overall loss of performance.

The recently proposed coflow abstraction bridges the gap between application-level performance and network-level optimizations. Each multipoint-to-multipoint coflow represents a collection of flows with a common application-level performance objective, enabling application-aware decision making in the network. We describe complete solutions including architectures, algorithms, and implementations that apply coflows to multiple scenarios using central coordination, and we demonstrate through large-scale cloud deployments and trace-driven simulations that simply knowing how flows relate to each other is enough for better network scheduling, meeting more deadlines, and providing higher performance isolation than what is otherwise possible using today’s application-agnostic solutions.

3.5 A New Software Stack for Low-Latency Datacenters

John Ousterhout (Stanford University, US)

License  Creative Commons BY 3.0 Unported license
© John Ousterhout

Hardware latencies for communication and storage access in datacenters are dropping dramatically, from milliseconds to (soon) microseconds. Unfortunately, today’s highly-layered software stacks have too much overhead to capitalize on these improvements. If applications are to take advantage of the hardware improvements, we will need to make radical changes to the software stack. In this talk I will describe two projects underway at Stanford to create elements of a low-latency software stack. The first project is defining a new transport protocol for low-latency remote procedure calls; the second project is creating a new core-aware thread scheduling mechanism that moves most of the scheduler to user-level. This work, and other recent developments such as the rise of virtual machines, suggest that operating systems of the future may be “hollowed out”: the data plane will largely bypass the operating system, leaving it mostly as a control plane.

3.6 NDP: New Datacenter Philosophy/Protocol

Costin Raiciu (University Politehnica of Bucharest, RO)

License  Creative Commons BY 3.0 Unported license
© Costin Raiciu

Joint work of Mark Handley, Alexandru Agache, Marcin Wojcik, Gianni Antiki, Costin Raiciu

Modern datacenter networks provide very high capacity and low switch latency, but transport protocols rarely manage to deliver performance matching the underlying hardware. We present NDP, a novel datacenter transport architecture that achieves both near-optimal file completion times and throughput in a wide range of scenarios including incast. NDP achieves this through a novel and synergistic combination of techniques including packet spraying, extremely small switch buffers, a randomized variant of packet trimming when queues fill, a priority queuing mechanism and a highly tuned receiver-driven ack clocking mechanism that can trigger retransmissions so rapidly that lost packets have very little impact on performance.

3.7 Where has the time gone?

Noa Zilberman (University of Cambridge, GB)

License  Creative Commons BY 3.0 Unported license
© Noa Zilberman

Joint work of Matthew Grosvenor, Neelakandan Manihatty-Bojan, Diana Andreea Popescu, Gianni Antichi, Salvator Galea, Andrew Moore, Robert Watson, Marcin Wojcik, Noa Zilberman

Latency control is mandatory to the operation of data centres. In this talk we present a study of the different latency components and show the breakdown of end-to-end latency, all the way from the application level to the wire. This breakdown provides insights on the challenges for latency control and the importance of different latency components. We present several trajectories taken in Cambridge to improve latency control, and discuss the instrumentation required to this end.

4 Working groups

4.1 Low latency outside the DC (5G, open Internet, etc)

Jari Arkko (Ericsson – Jorvas, FI), Bob Briscoe (Simula Research Laboratory – Lysaker, NO), Jon Crowcroft (University of Cambridge, GB), Koen De Schepper (NOKIA Bell Labs – Antwerp, BE), Lars Eggert (NetApp Deutschland GmbH – Kirchheim, DE), Michio Honda (NetApp Deutschland GmbH – Kirchheim, DE), Wolfgang Kellerer (TU München, DE), Kirill Kogan (IMDEA Networks – Madrid, ES), Mirja Kühlewind (ETH Zürich, CH), and Michael Scharf (NOKIA – Stuttgart, DE)

License © Creative Commons BY 3.0 Unported license
 © Jari Arkko, Bob Briscoe, Jon Crowcroft, Koen De Schepper, Lars Eggert, Michio Honda, Wolfgang Kellerer, Kirill Kogan, Mirja Kühlewind, and Michael Scharf

Discussion points:

- Clarification: use case providers need to clearly distinguish between guaranteed latency and low latency.
- Localized computing applications can potentially benefit from low-latency.
- 5G and DC networking have common aspects and can be considered similar in some respects: routing and addressing are challenges addressed by both, but do not scale to both cases. Capacity is a key concern in both cases but has fundamental differences: capacity comes from multi-paths in DC and frequency sharing in 5G. Can lessons learned in one domain be carried over to another? E.g., can multi-path routing be carried over to 5G to optimize the capacity problem?
- Limitations: Internet is a signaling plane. DC mechanisms can not be deployed on Internet.
- There is no DC position. In IETF, DC things do not run in Routing area, DC proposals got pushed to transport. Perhaps change IETF to take in IP layer work.
 - routing area in IETF, transport area more focused
 - Specification is not regarded as publication, a motivation issue for academics
- Internet needs a framework to bring in DC technologies.

4.2 How to make datacenter research more reproducible

Marinho Barcellos (Federal University of Rio Grande do Sul, BR), Olivier Bonaventure (University of Louvain, BE), Edouard Bugnion (EPFL Lausanne, CH), Oliver Hohlfeld (RWTH Aachen, DE), Andrew W. Moore (University of Cambridge, GB), Hakim Weatherspoon (Cornell University – Ithaca, US), and Noa Zilberman (University of Cambridge, GB)

License © Creative Commons BY 3.0 Unported license
 © Marinho Barcellos, Olivier Bonaventure, Edouard Bugnion, Oliver Hohlfeld, Andrew W. Moore, Hakim Weatherspoon, and Noa Zilberman


Question: How do we make artifacts reproducible in the context of data centers?

Summary/Takeaways:

1. whatever we do have to have incentives and has to scale
2. Back end: longer repeatable papers journal; specifically CCR
3. Front end: Track for reproducibility at SIGCOMM
4. Community award for reproducibility
5. Badging: Use ACM policy (see Results and Artifact Review and Badgin – <http://www.acm.org/publications/policies/artifact-review-badging/>)

4.3 Switch programmability: How much functionality should be handled by switches?

Matthew P. Grosvenor (University of Cambridge, GB)

License  Creative Commons BY 3.0 Unported license
© Matthew P. Grosvenor

- Starting point: Innovators dilemma – building components means that you already have a model of how the components might go together which means they need to envisage how other innovators might innovate creates constrictions on what they can do. Should really only do things in HW that become stable over de
- Current SDN concentrates on efficient representation of packet processing, but not buffer management. Desire to make buffer/queue management programmable.
- P4 building blocks: P4 pipeline, match packet → place into buffer → (drop, send, modify) P4 may be able represent some states, but may not be able to implement these states efficiently? If we consider a single queue, there are only two things we are doing: ingress and egress and we want to be able to program that.
- But we may want to do more than just simple “old” “boring” “classic” “legacy” switch operations. We want to do packet rewriting, paxos, content based load balancing etc. etc. What are the primitives to do this. For NetPaxos primitives: state/register, compare, broadcast.
- What are the sorts of applications outside of networks that we can use to implement building blocks. E.g. stream processing, aggregation etc. that we can use to steer design decisions.
- Cisco what are the applications? We want to use from the switches? What sort of functionality do we need? What sort of monitoring? Statistics? Average queuing delay. Monitoring may be at a huge cost e.g. 25,000 queues counters, costs energy etc.
- Straw man: In a datacenter we only need two things: source routing and priority queueing Counter argument: You need some kind of feedback from the network (not packet drop?) not ECN? But what? Perhaps packet.
- NIC features / Switch and NIC vs. Switch Are they the same? The NIC may want to offload more complex things into the CPU Or maybe the division of labour between NIC and switch.
- How are NICs different to switches. Application offload (TCP segmentation), encryption, control groups NICs have a more information. More generic building blocks necessary in the NIC for other protocols. Experience in NICs suggests that NICs are slow, and offloading into the NIC from a high speed CPU usually has no net positive gain.
- What about C# → gates? Is that enough? Application programmer wants to know what the template engine is that does C# → gates? E.g. template languages that were used in GPUs. Do we really want the application program to run on the NIC, or do we want some generic functions in the NIC, that many applications do want to do? What about application sharing? How do we share the resources in the NIC.

4.4 Congestion Control for 100 G networks

Dongsu Han (KAIST – Daejeon, KR), Alexandru Agache (University Politehnica of Bucharest, RO), Mohammad Alizadeh Attar (MIT – Cambridge, US), Marco Canini (University of Louvain, BE), Nandita Dukkhipati (Google Inc. – Mountain View, US), Matthew P. Grosvenor (University of Cambridge, GB), Patrick Jahnke (TU Darmstadt, DE), Lavanya Jose (Stanford University, US), Mike Marty (Google – Madison, US), John Ousterhout (Stanford University, US), Rong Pan (CISCO Systems – San Jose, US), Ihsan Ayyub Qazi (LUMS – Lahore, PK), Costin Raiciu (University Politehnica of Bucharest, RO), and Michael Scharf (NOKIA – Stuttgart, DE)

License © Creative Commons BY 3.0 Unported license

© Dongsu Han, Alexandru Agache, Mohammad Alizadeh Attar, Marco Canini, Nandita Dukkhipati, Matthew P. Grosvenor, Patrick Jahnke, Lavanya Jose, Mike Marty, John Ousterhout, Rong Pan, Ihsan Ayyub Qazi, Costin Raiciu, and Michael Scharf

Topic: What is different in 100 Gbps?

1. Everything may look like mice?
 - Argument: everything look like mice. ↔ counter: some apps will find way to fill up.
 - CPU (a single thread, single flow) may not be that fast for a single flow too fill up 100 Gbps ↔ RDMA (cpu offload) is there to fill up.
 - Today’s traffic distribution comes from today’s limitation. If b/w is plenty, some apps will find way to fill the pipe.
 - Everything looking like a mice → What if they arrive altogether?
 - Solving incast very hard at 100 Gbps. (no time to control)
 - Counter argument: But because it requires tighter synchronization at 100 G, it may not matter if there enough randomness in the host (assuming flow size is the same).
 - Even a small randomness at the sender will break synchronization from the network standpoint.
 - Counter argument: tomorrow flow size will increase (as b/w will become plenty)
 - Conclusion: We bet on that there will be apps that send more flows. If we have short incast, congestion control can’t do much in at 100 G. Or we don’t have problem b/c of plenty b/w and less synchronized flows.
2. Core oversubscription
 - Even today, core of data center is oversubscribed due to cost. Full bisection B/W is a myth. With 100 G at the host, oversubscription will become worse because it so expensive to provision for full bisection.
 - Note, 1G → 10G transition was the opposite.
 - Potentially, this will increase core congestion.
3. Buffers
 - Buffers are getting scares per port per gbps. Sometimes even absolute amount of buffers is going down.
 - How do you cope with incast, large fan-in without large buffers?
 - Some high-end 100 G switches have no shared buffer.
 - Packets are scattered throughout non-shared buffers. Difficult to count the number of packets, which is required for ECN. Timing also difficult. So delay feedback won’t work.
 - Congestion control alternatives:
 - Loss as congestion signal always our friend.

- If you have very low buffers, credit-based flow control looks more attractive, which means IB.
- On-chip networks do credit-based flow control.

4.5 Role of the Operating System (Group 2)

Mike Marty (Google – Madison, US), Jari Arkko (Ericsson – Jorvas, FI), Patrick Thomas Eugster (TU Darmstadt, DE), Oliver Hohlfeld (RWTH Aachen, DE), and Andrew W. Moore (University of Cambridge, GB)

License © Creative Commons BY 3.0 Unported license
© Mike Marty, Jari Arkko, Patrick Thomas Eugster, Oliver Hohlfeld, and Andrew W. Moore

Slogan: OS job is to allocate, map, and multiplex if necessary.

Problems we have right now:

- low productivity dev environment (old time shared machine perception leads to the current OS concept)
- control plane done wrong or data plane done right?
- hard to test and debug if there are no other ways to do isolation
- unclear picture: API? or privileges?
- multi-core scheduling (e.g., Linux)
- OS is naive in resource sharing beyond CPU and process management, particular for networking – hard to evolve the OS

Discussion points:

- Is the core OS role to aid the development. or then aid for debugging?
- OS is good to do fine-grain sharing
- If hardware can multiple, then OS only does the map, then scheduling becomes a allocation.
- DC OS and stand along OS?
- Accounting and auditing role for OS?
- Open questions: applications may need know the need, but express such needs in a less-defined abstraction way. The needs from application should be abstracted, otherwise people will do things for each of them

4.6 Role of the Operating System (Group 1)

John Ousterhout (Stanford University, US), Edouard Bugnion (EPFL Lausanne, CH), Marco Canini (University of Louvain, BE), Nandita Dukkipati (Google Inc. – Mountain View, US), and Jitendra Padhye (Microsoft Corporation – Redmond, US)

License © Creative Commons BY 3.0 Unported license
© John Ousterhout, Edouard Bugnion, Marco Canini, Nandita Dukkipati, and Jitendra Padhye

- Docker style containers are popular but inter-container communication is not the same as IPC. Can we have isolation features without the communication overhead of going through all the layers to communicate b/n two containers?
Unikernels try to solve the problem by compiling a custom OS for the application at runtime with just the necessary layers.

- Why is kernel networking so slow?
 - Guest VMs in clouds have to go through two network stacks. Hypervisor bypass at least allows them to skip the hypervisor and (think they can) run directly on NICs. Only Amazon has done this successfully- maybe using bump in the wire (i.e., CPU on other side of NIC modifying packets at line rate...)
 - Most of the time is spent in the socket layer, not the TCP/IP processing?
 - Sockets as an API b/n the application and networking stack doesn't lend itself to scaling with multicores. There is typically a lot of churn in sockets. In fact Google's SPDY was just a way for user to open socket once and have a permanent connection to the mothership. Sockets is the wrong API.
- What is the right API then? Maybe RDMA?
 - RDMA is better than RPC at the tail (it's one sided, while RPC could suffer cu of interrupts at destination etc.) Heated discussion on whether RDMA was better for median. J.O.: RDMA memory model bad for complex applications, if multiple sources want to update a key, value they'll have to get locks, look up data structures to find new memory for a bigger value etc..
- Rack scale computing vs tighter NIC-CPU integration for super-fast networks
 - No interesting application that can fit in a rack yet but hardware will evolve. Why do we need different interconnects within a rack? Can go CPU to CPU, avoiding PHY, learning fabric etc. If it's latency may be we can just avoid the half us PCIe latency with having the NIC right next to the fastest cache.
- Hardware offloading gives some performance saving and orders of magnitude power savings compared to function virtualization, is the way to go. What about ossification? Re-configurable smart NICs that can be programmed using P4 enable new features to be offloaded (yet to characterize the kind of features).

Suggested topics for an extended discussion:

- Thread scheduling
- Role of the NIC and the OS
- Storage class memory / NVRAM
- Impact of Isolation on Performance: e.g., Containers are Processes Isolated form each other, when they want to talk to each other through network., overhead
- What is the right API between applications and OS bypass?
- OSes and distributed application, OS : what's the boundary between what OS schedules for one machine and what scheduler schedules for a cluster of machines
- Disaggregated computing: Disaggregation enabled by low latency interconnects "Rack scale computing"

5 Panel discussions

5.1 Concluding Statements

John Ousterhout (Stanford University, US)

License © Creative Commons BY 3.0 Unported license
© John Ousterhout

In the Wednesday morning session, seminar participants proposed possible conclusions from our discussions. Each conclusion was discussed briefly, then voted on. The proposed conclusions and votes follow below.

- 5–15 us μ s round-trip latencies will soon be common in datacenter applications
Yes: 18 , No: 3
 - Depends if we localize computation. In Google DCs, the speed of light is higher than that (100's of meters fibers). Todo: do a better job in taking hierarchy into account
 - Every DC is different. Googles DC is just orders of magnitude larger than others.
- Packet scheduling (clock-out) should be done by the receiver (or its TOR), as much as possible
Yes: 9, No: 15
- Network oversubscription will always be present in the datacenter (apps will be bandwidth-constrained by core bandwidth)
Yes: 20, No: 5
 - At what time-scale? You can't assume that you'll never run into congestion etc.
 - Question is more on which apps are bandwidth constrained or not?
- Future datacenters will be flat (not pod-structured)
Yes: 3.5, No: 16
 - Key observation: if we would have this seminar 4 years ago, we wouldn't have this discussion. The common thinking at the time was to take layer 3 all the way down. Is this just an artifact of current thinking?
- Link-level flow control will become a primary solution at datacenter scale (i.e. no congestion drops)
Yes: 5, No: 14
 - Long-term: source aggressive senders down – could not become a solution. Short-term is possible by link-level flow control
 - Building a zero loss networks is expensive
 - Remind lessons learned on ATM control loops failed: ABR (available bit rate) operated edge to edge across the VC through all the switches- problem was that the rate adjustment cycle was close to TCP's AIMD scheme, but not quite, so the two, nested control loops of TCP & ATM would be out of phase, and so the capacity/demand would oscillate, between underutilized, and overusing/ with higher delay. Because there was an IP layer between TCP and the ATM VC, there wasn't an easy way to get cross layer info from the lower layer up to TCP (always assuming we could have had a way to inform TCPs cwnd adjustment algorithm with something more sophisticated than just delay or loss or ECN, which we didn't have).

The following conclusions were voted on with no discussion:

- The OS should not be involved in in-host dataplane networking ops (read and write calls)
Yes: 18, No: 3
- The OS should allocate, map, and multiplex only when necessary
Yes: 21, No: 0
- DC and rest of the Internet are different (speed-of-light)
Yes: 18, No: 4
- Link-level flow control will become a primary solution at rack scale
Yes: 16, No: 4

6 Open problems

6.1 Questions about flow aggregation

Michael Welzl (University of Oslo, NO)

License © Creative Commons BY 3.0 Unported license
© Michael Welzl

In a practical datacenter setting, how can we apply flow (actually, congestion control) aggregation in data centers? By doing congestion control between hypervisors? Would this have to be carrying TCP traffic, or can we develop our own specialized congestion control? Can we aggregate at TOR switches?

Participants

- Alexandru Agache
University Politehnica of
Bucharest, RO
- Mohammad Alizadeh Attar
MIT – Cambridge, US
- Jari Arkko
Ericsson – Jorvas, FI
- Marinho Barcellos
Federal University of Rio Grande
do Sul, BR
- Olivier Bonaventure
University of Louvain, BE
- Bob Briscoe
Simula Research Laboratory –
Lysaker, NO
- Edouard Bugnion
EPFL Lausanne, CH
- Marco Canini
University of Louvain, BE
- Julian Chesterfield
University of Cambridge, GB
- Mosharaf Chowdhury
University of Michigan – Ann
Arbor, US
- Jon Crowcroft
University of Cambridge, GB
- Koen De Schepper
NOKIA Bell Labs – Antwerp, BE
- Aaron Yi Ding
TU München, DE
- Nandita Dukkupati
Google Inc. –
Mountain View, US
- Lars Eggert
NetApp Deutschland GmbH –
Kirchheim, DE
- Patrick Thomas Eugster
TU Darmstadt, DE
- Matthew P. Grosvenor
University of Cambridge, GB
- Dongsu Han
KAIST – Daejeon, KR
- Oliver Hohlfeld
RWTH Aachen, DE
- Michio Honda
NetApp Deutschland GmbH –
Kirchheim, DE
- Patrick Jahnke
TU Darmstadt, DE
- Lavanya Jose
Stanford University, US
- Wolfgang Kellerer
TU München, DE
- Kirill Kogan
IMDEA Networks – Madrid, ES
- Mirja Kühlewind
ETH Zürich, CH
- Mike Marty
Google – Madison, US
- Andrew W. Moore
University of Cambridge, GB
- John Ousterhout
Stanford University, US
- Jitendra Padhye
Microsoft Corporation –
Redmond, US
- Rong Pan
CISCO Systems – San Jose, US
- Max Plauth
Hasso-Plattner-Institut –
Potsdam, DE
- Ihsan Ayyub Qazi
LUMS – Lahore, PK
- Costin Raiciu
University Politehnica of
Bucharest, RO
- Michael Scharf
NOKIA – Stuttgart, DE
- Hakim Weatherspoon
Cornell University – Ithaca, US
- Klaus Wehrle
RWTH Aachen, DE
- Michael Welzl
University of Oslo, NO
- Noa Zilberman
University of Cambridge, GB

