Pantheon: the training ground for Internet congestion-control research
https://pantheon.stanford.edu

Francis Y. Yan†, Jestin Ma†, Greg Hill†, Deepti Raghavan¶, Riad S. Wahby†, Philip Levis†, Keith Winstein†

†Stanford University, ¶Massachusetts Institute of Technology

May 15, 2018
Congestion control

Most important topic in computer networking

- Avoid congestion collapse
- Allocate resources among users
- Affect every application using TCP socket
By all accounts, today’s Internet is not moving data as well as it should. Most of the world’s cellular users experience delays of seconds to minutes; public Wi-Fi in airports and conference venues is often worse. Physics and climate researchers need to exchange petabytes of data with global collaborators but find their carefully engineered multi-Gbps infrastructure often delivers at only a few Mbps over intercontinental distances. These problems result from a design choice made when TCP congestion control was created in the 1980s—interpreting packet loss as "congestion." This equivalence was true at the time but was because of technology limitations, not first principles. As NICs (network interface controllers) evolved from Mbps to Gbps and memory chips from KB to GB, the relationship between packet loss and congestion became more tenuous.

Today TCP’s loss-based congestion control—even with the current best of breed, CUBIC—is the primary cause of these problems. When bottleneck buffers are large,
Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks

Keith Weinstein, Anirudh Sivaraman, and Hari Balakrishnan

MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, Mass.

{keithw, anirudh, hari}@mit.edu

Abstract

Sprout is an end-to-end transport protocol for interactive applications that desire high throughput and low delay. Sprout works well over cellular wireless networks, where link speeds change dramatically with time, and current protocols build up multi-second queues in network gateways. Sprout does not use TCP-style reactive congestion control; instead the receiver observes the packet arrival times to infer the uncertain dynamics of the network path. This inference is used to forecast how many bytes may be sent by the sender, while bounding the risk that packets will be delayed inside the network for too long.

In evaluations on traces from four commercial LTE and 3G networks, Sprout, compared with Skype, reduced self-inflicted end-to-end delay by a factor of 7.9 and achieved 2.2 × the transmitted bit rate on average. Compared with Google’s Hangout, Sprout reduced delay by a factor of 7.2 while achieving 4.4 × the bit rate, and compared with Apple’s Facetime, Sprout reduced delay by a factor of 8.7 with 1.9 × the bit rate.

Although it is end-to-end, Sprout matched or outperformed TCP Cubic running over the CoDel active queue management algorithm, which requires changes to cellular carrier equipment to deploy. We also tested Sprout as

Figure 1: Skype and Sprout on the Verizon LTE downlink trace. For Skype, overshoots in throughput lead to large standing queues. Sprout tries to keep each packet’s delay less than 100 ms with 95% probability.
Adaptive Congestion Control for Unpredictable Cellular Networks

Yasir Zaki
NYU Abu Dhabi
Abu Dhabi, UAE
yasir.zaki@nyu.edu

Thomas Pötsch
University of Bremen
Bremen, Germany
tpoetsch@uni-bremen.de

Jay Chen
NYU Abu Dhabi
Abu Dhabi, UAE
jchen@cs.nyu.edu

Lakshminarayanan Subramanian
NYU and GTED
New York, USA
lakshmi@cs.nyu.edu

Carmelita Görg
University of Bremen
Bremen, Germany
cg@cornnets.uni-bremen.de

ABSTRACT
Legacy congestion controls including TCP and its variants are known to perform poorly over cellular networks due to highly variable capacities over short time scales, self-inflicted packet delays, and packet losses unrelated to congestion. To cope with these challenges, we present Verus, an end-to-end congestion control protocol that uses delay measurements to react quickly to the capacity changes in cellular networks without explicitly attempting to predict the cellular channel dynamics. The key idea of Verus is to continuously learn a delay profile that captures the relationship between end-to-end packet delay and outstanding window size over short epochs and uses this relationship to increment or decrement the window size based on the observed short-term packet delay variations. While the delay-based control is primarily for congestion avoidance, Verus uses standard TCP features including multiplicative decrease upon packet loss and slow start.

Through a combination of simulations, empirical evaluations using cellular network traces, and real-world evaluations against standard TCP flavors and state of the art protocols like Sprout, we show that Verus outperforms these protocols in cellular channels. In comparison to TCP Cubic, Verus achieves an order of magnitude (> 10x) reduction in delay over 3G and LTE networks while achieving comparable throughput (sometimes marginally higher). In comparison to Sprout, Verus achieves up to 30% higher throughput in rapidly changing cellular networks.

CCS Concepts
- Networks → Network protocol design; Transport protocols; Network performance analysis;

Keywords
Congestion control, Cellular network, Transport protocol, Delay-based

1. INTRODUCTION
Cellular network channels are highly variable and users often experience fluctuations in their radio link rates over short time scales due to scarce radio resources making these channels hard to predict [26, 20, 7]. TCP and its variants are known to perform poorly over cellular networks due to high capacity variability, self-inflicted queuing delays, stochastic packet losses that are not linked to congestion, and large bandwidth-delay products [15, 32, 33].

Three specific characteristics directly impact the unpredictability of cellular channels. First, the state of a cellular channel between a mobile device and a base station undergoes several complex state transitions that
PCC: Re-architecting Congestion Control for Consistent High Performance

Mo Dong*, Qingxi Li*, Doron Zarchy**, P. Brighten Godfrey*, and Michael Schapira**

*University of Illinois at Urbana-Champaign
**Hebrew University of Jerusalem

Abstract

TCP and its variants have suffered from surprisingly poor performance for decades. We argue the TCP family has little hope of achieving consistent high performance due to a fundamental architectural deficiency: hardwiring packet-level events to control responses. We propose Performance-oriented Congestion Control (PCC), a new congestion control architecture in which each sender continuously observes the connection between its actions and empirically experienced performance, enabling it to consistently adopt actions that result in high performance. We prove that PCC converges to a stable and fair equilibrium. Across many real-world and challenging environments, PCC shows consistent and often 10× performance improvement, with better fairness and stability than TCP. PCC requires no router hardware support or new packet format.

A very difficult task within TCP’s rate control architecture, which we refer to as hardwired mapping: certain predefined packet-level events are hardwired to certain predefined control responses. TCP reacts to events that can be as simple as “one packet loss” (TCP New Reno) or can involve multiple signals like “one packet loss and RTT increased by x%” (TCP Illinois). Similarly, the control response might be “halve the rate” (New Reno) or a more complex action like “reduce the window size w to f(ΔRTT)w” (Illinois). The defining feature is that the control action is a direct function of packet-level events.

A hardwired mapping has to make assumptions about the network. Take a textbook event-control pair: a packet loss halves the congestion window. TCP assumes that the loss indicates congestion in the network. When the assumption is violated, halving the window size can severely degrade performance (e.g. if loss is random, rate should stay the same or increase). It is fundamentally
Inconsistent behaviors

Figure: AWS Brazil to Colombia (cellular, 1 flow, 3 trials, P1392)
Inconsistent behaviors

**Figure:** a figure presented in the paper (flows share the bandwidth fairly)

**Figure:** between two real nodes on Pantheon (throughput ratio 32:4:1)
Challenges and problems

- Every emerging algorithm claims to be the “state-of-the-art”
- ... compared with other algorithms that they picked
- ... evaluated on their own testbeds in real world
- ... and/or on simulators/emulators with their settings
- ... based on the specific results that they collected
Challenges and problems

... compared with other algorithms that they picked

... evaluated on their own testbeds in real world

... and/or on simulators/emulators with their settings

... based on the specific results that they collected
Challenges and problems

... compared with other algorithms that they picked
\[\rightarrow\] must acquire, compile, and execute prior algorithms
... evaluated on their own testbeds in real world

... and/or on simulators/emulators with their settings

... based on the specific results that they collected
Challenges and problems

... compared with other algorithms that they picked
⇒ must acquire, compile, and execute prior algorithms
... evaluated on their own testbeds in real world
⇒ large service operators: risky to deploy, long turnaround time

... and/or on simulators/emulators with their settings

... based on the specific results that they collected
Challenges and problems

... compared with other algorithms that they picked
⇒ must acquire, compile, and execute prior algorithms
... evaluated on their own testbeds in real world
⇒ large service operators: risky to deploy, long turnaround time
⇒ researchers: at much smaller scale, results may not generalize
... and/or on simulators/emulators with their settings

... based on the specific results that they collected
Challenges and problems

... compared with other algorithms that they picked
⇒ must acquire, compile, and execute prior algorithms
... evaluated on their own testbeds in real world
⇒ large service operators: risky to deploy, long turnaround time
⇒ researchers: at much smaller scale, results may not generalize
... and/or on simulators/emulators with their settings
⇒ how to configure the settings?
... based on the specific results that they collected
Challenges and problems

... compared with other algorithms that they picked
⇒ must acquire, compile, and execute prior algorithms
... evaluated on their own testbeds in real world
⇒ large service operators: risky to deploy, long turnaround time
⇒ researchers: at much smaller scale, results may not generalize
... and/or on simulators/emulators with their settings
⇒ how to configure the settings?
... based on the specific results that they collected
⇒ but the Internet is diverse and evolving
ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures. Click here to learn more about ImageNet, Click here to join the ImageNet mailing list.

What do these images have in common? Find out!

Check out the ImageNet Challenge on Kaggle!
TPC-C

TPC-C is an On-Line Transaction Processing Benchmark

Approved in July of 1992, TPC Benchmark C is an on-line transaction processing (OLTP) benchmark. TPC-C involves a mix of five concurrent database transactions, each of which may involve one or more database operations. The database is comprised of nine types of tables with a wide range of record and population sizes. TPC-C is not limited to the activity of any particular business segment, but rather represents a general model of transaction processing.

Specification

- The current TPC-C specification can be found on the TPC Documentation Webpage.

More Information

- Detailed TPC-C Description
- Frequently Asked TPC-C Questions
- Sigmod HTML Presentation
- Order-Of-Magnitude Advantage on TPC-C Through Massive Parallelism
- Thousands of Debit/Credit Transactions-Per-Second: Easy and Inexpensive
- Transaction Performance vs. Moore’s Law: A Trend Analysis

Results

- Advanced TPC-C Sorting and Filtering Options
- Results Spreadsheet (downloadable file)
Pantheon: a community evaluation platform for congestion control research

- a common reference set of benchmark algorithms
- a diverse testbed of network nodes
  - Cellular and wired: U.S., Mexico, Brazil, Colombia, India, China
  - Wired networks only: U.K., Australia, Japan, Korea
- a collection of calibrated emulators and pathological emulated networks
- a continuous-testing system and a public archive of results at https://pantheon.stanford.edu
Pantheon: a community evaluation platform for congestion control research
Our vision

- XXX outperforms 15 algorithms on Pantheon
- XXX shows 10x performance improvement on Pantheon
- results (P1392, P950, ...) are available on Pantheon
Pantheon: a community resource

- A common language in congestion control
  - shared testbeds
  - benchmark algorithms
  - public data
Pantheon: a community resource

- A common language in congestion control
  - shared testbeds
  - benchmark algorithms
  - public data
- A training ground for congestion control
  - enables faster innovation and more reproducible research
  - e.g., Vivace, Copa (NSDI ’18), and Indigo: data-driven congestion control based on neural networks
Outline

1. Introduction

2. Pantheon: a community evaluation platform for congestion control

3. Calibrated Emulators

4. Ongoing projects
   - Vivace, Copa, and more
   - Indigo

5. Conclusion
A collection of congestion-control algorithms, each exposing the same interface

15+ algorithms and designed to be easily extended
- TCP Cubic, TCP Vegas, TCP BBR, QUIC Cubic, LEDBAT, WebRTC (media), Sprout, Remy, Verus, PCC, SCReAM, FillP, Copa, Vivace, Indigo, ...

Common testing interface
- A full-throttle flow that runs until killed
Operation and testing methods

- Runs multiple benchmarks per week on each wired and wireless paths in both directions
- Each benchmark runs all schemes in round-robin for multiple times
- ... with single flow for 30 seconds or multiple flows
- Calculates mean throughput, 95th-percentile one-way delay, loss rate
Findings

- Measurement study from more than a year of data
- Comparative analysis: which scheme an end host should run
Key finding 1: scheme performance varies by path

**Figure**: AWS Brazil to Colombia (cellular, 1 flow, 3 trials, P1392)

**Figure**: Stanford to AWS California (cellular, 1 flow, 3 trials, P950)
Key finding 2: scheme performance varies by path direction

**Figure:** AWS Brazil to Colombia
(cellular, 1 flow, 3 trials, P1392)

**Figure:** Colombia to AWS Brazil
(cellular, 1 flow, 3 trials, P1391)
Key finding 3: scheme performance varies in time

Figure: AWS Brazil to Colombia (cellular, 1 flow, 3 trials, filled dots show performance after 2 days)
Limitations

- Only tests schemes at full throttle
- Nodes are not necessarily representative
- No available Wi-Fi connections currently
- Does not measure interactions between different schemes (ongoing collaboration with CMU)
Outline

1. Introduction

2. Pantheon: a community evaluation platform for congestion control

3. Calibrated Emulators

4. Ongoing projects
   - Vivace, Copa, and more
   - Indigo

5. Conclusion
Motivations

Simulation/emulation: reproducible and allow rapid experimentation
- ns-2/ns-3, Mininet, Mahimahi, etc.
- fine-grained and detailed, providing a number of parameters
- do not necessarily capture real-world paths
Calibrated Emulators

Motivations

Simulation/emulation: reproducible and allow rapid experimentation
- ns-2/ns-3, Mininet, Mahimahi, etc.
- fine-grained and detailed, providing a number of parameters
- do not necessarily capture real-world paths

Open problem
What is the choice of parameter values to faithfully emulate a particular target network?
New figure of merit for network emulators

Replication error

Average difference of the performance of a set of transport algorithms run over the emulator compared with over the target real network path.
New figure of merit for network emulators

Replication error

Average difference of the performance of a set of transport algorithms run over the emulator compared with over the target real network path.

Figure: Filled dots: real results over a network path; open dots: results over an emulator.
Emulator characteristics

Five parameters

- a bottleneck link rate
- a constant propagation delay
- a DropTail threshold for the sender’s queue
- a loss rate (per-packet, i.i.d)
- a bit that selects constant rate or Poisson-governed rate
Automatically calibrate emulators to match a network path

Collect a set of results over a particular network path on Pantheon

- average throughput and 95th percentile delay of a dozen algorithms
Automatically calibrate emulators to match a network path

Collect a set of results over a particular network path on Pantheon
- average throughput and 95th percentile delay of a dozen algorithms

Run Bayesian optimization
- standard black-box optimization approach
- designed for optimizing a noisy function
- when observations are expensive to obtain
Bayesian optimization details

- Input $x$: <rate, propagation delay, queue size, loss rate>
- Run twice: constant rate and Poisson-governed rate
- Objective function $f(x)$: mean replication error
- Other common settings:
  - Prior: Gaussian process
  - Acquisition function: expected improvement
Results of calibrated emulators

- Trained emulators calibrated to 6 of Pantheon’s paths
  - Nepal (Wi-Fi), Colombia (cellular), Mexico (cellular), China (wired), India (wired), and Mexico (wired)
  - including single flow and three flows for Mexico (wired)
- Each for about 2 hours on 30 machines with 4 cores each
Results of calibrated emulators

- Trained emulators calibrated to 6 of Pantheon’s paths
  - Nepal (Wi-Fi), Colombia (cellular), Mexico (cellular), China (wired), India (wired), and Mexico (wired)
  - including single flow and three flows for Mexico (wired)

- Each for about 2 hours on 30 machines with 4 cores each

- Replication error is within 17% on average
Figure: AWS California to Mexico (wired, 3 flows, 10 trials, P1237).
Mean replication error: 14.4%.
Outline

1. Introduction

2. Pantheon: a community evaluation platform for congestion control

3. Calibrated Emulators

4. Ongoing projects
   - Vivace, Copa, and more
   - Indigo

5. Conclusion
Pantheon’s contribution to Vivace, Copa, ...

- Vivace (NSDI ’18): validating a new scheme in the real world
- Copa (NSDI ’18): iterative design with measurements
- Other ongoing projects:
  - Mixed-scheme multi-flow measurements (CMU)
  - FillIP (Huawei)
Indigo: a machine-learned congestion-control algorithm

- Trained on data gathered by Pantheon
  \[\Rightarrow\] Pantheon enables entirely new designs
Indigo: a machine-learned congestion-control algorithm

- Trained on data gathered by Pantheon
  - Pantheon enables entirely new designs
- Uses a recurrent neural network (LSTM) to encode the algorithm
- Imitates a *congestion-control oracle*
Model and imitation learning

- Agent: sender
- Environment: network
- State: congestion signals
- Action: congestion window adjustment
Model and imitation learning

- Agent: sender
- Environment: network
- State: congestion signals
- Action: congestion window adjustment

Indigo’s goal

Learn to imitate an expert’s behavior: state \( \rightarrow \) action
Expert policy

Congestion-control oracle

- Brings congestion window size closer to the ideal size
- Does not know ideal size in real world — only exists in emulators
Expert policy

Congestion-control oracle
- Brings congestion window size closer to the ideal size
- Does not know ideal size in real world — only exists in emulators

What is the ideal size?
- BDP: simple links with a fixed bandwidth and min RTT
- Search around BDP otherwise
Indigo: design and training

Model

- Input: a state
- → 1-layer LSTM network with 32 hidden units per layer
- → Softmax classifier with cross-entropy loss
- → Output: an action
Indigo: design and training

Step

- 10 ms (congestion window is adjusted every 10 ms)
- but observations occur each time an ACK is received
Indigo: design and training

Step
- 10 ms (congestion window is adjusted every 10 ms)
- but observations occur each time an ACK is received

State
- An EWMA of the queuing delay
- An EWMA of the sending rate
- An EWMA of the receiving rate
- The current congestion window size
- The previous action taken
Indigo: design and training

Step

- 10 ms (congestion window is adjusted every 10 ms)
- but observations occur each time an ACK is received

State

- An EWMA of the queuing delay
- An EWMA of the sending rate
- An EWMA of the receiving rate
- The current congestion window size
- The previous action taken

Action

- ÷2, −10 (packets), +0, +10 (packets), ×2
Indigo: design and training

Training

- State-of-the-art algorithm: DAgger
- Like supervised learning but in a repeated fashion
- Distributed
Indigo: design and training

1. sync local parameters
2. start flow
3.a get state
3.b get expert action
3.c get action
4. enqueue data, finish iteration
5. train on data

1 repeat step 3 every 10ms
Method

- Trained on 6 calibrated emulators
  - help mitigate the “distribution mismatch”
- and on 24 synthetic emulators
  - all combinations of (5, 10, 20, 50, 100, 200 Mbps) link rate with (10, 20, 40, 80 ms) min one-way delay
  - infinite queues and no loss
**Compute cost**

- **Training**
  - 30 worker machines with 4 cores each
  - the training server has 8 cores and an Nvidia Tesla K80 GPU
  - takes less than 12 hours

- **Inference**
  - each action evaluation takes about 0.5 ms
  - slightly higher than TCP Cubic in CPU usage
Real-world results

Figure: AWS Brazil to Colombia (wired, 1 flow, 10 trials, P1439)
Real-world results

Figure: India to AWS India (wired, 3 flows, 10 trials, P1476)
Real-world results

Figure: Time-domain three-flow test (one trial in P1476)
Outline

1. Introduction

2. Pantheon: a community evaluation platform for congestion control

3. Calibrated Emulators

4. Ongoing projects
   - Vivace, Copa, and more
   - Indigo

5. Conclusion
Conclusion

Pantheon

- A common language in congestion control
  - shared testbeds, benchmark algorithms and public data
- A training ground for congestion control
  - enables faster innovation and more reproducible research
  - e.g., Vivace, Copa (NSDI ’18), and Indigo: data-driven congestion control based on neural networks
Conclusion

Pantheon

- A common language in congestion control
  - shared testbeds, benchmark algorithms and public data
- A training ground for congestion control
  - enables faster innovation and more reproducible research
  - e.g., Vivace, Copa (NSDI ’18), and Indigo: data-driven congestion control based on neural networks

Calibrated emulators

- Replication error — new figure of merit for network emulators
- Automatically calibrate an emulator to accurately model real networks
Visit https://pantheon.stanford.edu for more results and the paper!