March 27, 2019

Model-based Network Congestion Control

Neal Cardwell

Van Jacobson

Yuchung Cheng

Soheil Hassas Yeganeh

Victor Vasiliev

See next page for additional authors

Inventor(s)
Neal Cardwell, Van Jacobson, Yuchung Cheng, Soheil Hassas Yeganeh, Victor Vasiliev, Ian Swett, Priyaranjan Jha, Yousuk Seung, and David Wetherall

Model-based Network Congestion Control

ABSTRACT

Computer network congestion control algorithms control the sending rate for flows of data from sender network nodes to receiver network nodes. These algorithms attempt to utilize network bandwidth capacity efficiently, while keeping network data loss rates low, and allocating network capacity between different flows sharing the network in an approximately fair manner, or at least avoiding starvation of some flows. In addition, some congestion control algorithms attempt to keep network queues short, to reduce queuing delays and further reduce loss rates.

This disclosure describes model-based congestion control, a technique that explicitly models the network conditions along the path(s) between senders and receivers. The algorithm updates the model using measurements obtained from the packets in the flow it is controlling. The algorithm uses those measurements as inputs to update the model and then uses that model to control its sending process. This model-based approach can allow the congestion control algorithm to achieve higher throughputs and/or lower delays and/or lower data packet loss rates than would be achievable by other techniques.

KEYWORDS

- Congestion control
- Rate adaptation
- BBR
- TCP
- QUIC
- RTP/RTCP

- Bandwidth
- Queuing delay
- Packet loss
- Explicit Congestion Notification (ECN)

BACKGROUND

Computer network congestion control algorithms control the sending rate for flows of data from sender network nodes to receiver network nodes, by deciding whether it is permissible to send more data at a given time, and if so, how much data may be sent, and (for some algorithms) how fast that data may be sent. These algorithms may run on end host sender nodes, inside or in cooperation with end-to-end transport protocols like TCP or QUIC; or they may run on switches, routers, or other nodes inside the network, to shape the rate of traffic flow.

Congestion control algorithms attempt to utilize network bandwidth capacity efficiently, while keeping network data loss rates low, and allocating network capacity between different flows sharing the network in an approximately fair manner, or at least avoiding starvation of some flows. In addition, some congestion control algorithms attempt to keep network queues short, to reduce delays and further reduce loss rates.

Congestion control algorithms generally use signals implicitly or explicitly provided by the nodes in the network path. When network packets arrive at a node, e.g. switch or router, faster than the node can send the packets over the appropriate next hop link, the node may place the packets in a queue if there is free space in its buffer memory, or if there is insufficient free space then it may discard ("drop") the packet, resulting in packet loss. The longer such a rate mismatch lasts, the deeper the queue may grow, and the longer packets wait in the queue. If the queue or the wait is sufficiently long, some nodes mark packets with an Explicit Congestion Notification ("ECN") mark, indicating the presence or extent of queuing.

Network congestion control algorithms generally use as input signals some subset of the information that is produced by that network node queuing and forwarding behavior, including one or more of:

- Packet loss
- ECN marks
- Delays caused by time network packets spend in queues and in transit in the network
- The throughput (or "goodput" or "bandwidth" or "delivery rate") of the data packet flow

Previous widely-used congestion control algorithms were "model-free," in the sense of having no explicit model of the underlying processes in the network. Instead, they predominantly relied on a single signal from the list above and had a direct mapping from the value of that signal to a change in sending behavior. Most previous algorithms conservatively cut the volume of data they allowed in the network to a fraction of the previous level (a "multiplicative decrease") during any time interval, e.g. a network round-trip time, during which the signal indicated possible excess, e.g. the signal exceeded some target threshold. However, given the diversity of physical network paths and network traffic workloads, no single signal or threshold reliably corresponds to the point at which the sending behavior of a flow exceeds a sensible rate. As a result, model-free congestion control algorithms often tend to overshoot, producing high delays and/or packet loss rates, or undershoot, producing low throughput.

DESCRIPTION

This disclosure describes model-based congestion control, a technique for computer network congestion control algorithms to control the traffic a sender transmits over a network by having the algorithm maintain an explicit model of the processes inside the network path (or paths) between a sender node and a receiver node.

Example systems using this technique can be structured as depicted in Fig 1. The algorithm uses inputs gathered from measurements derived from network traffic to maintain an explicit model of the network path. The algorithm then implements a state machine to use the model to inform its decisions about how to set one or more control parameters that govern the sending process.

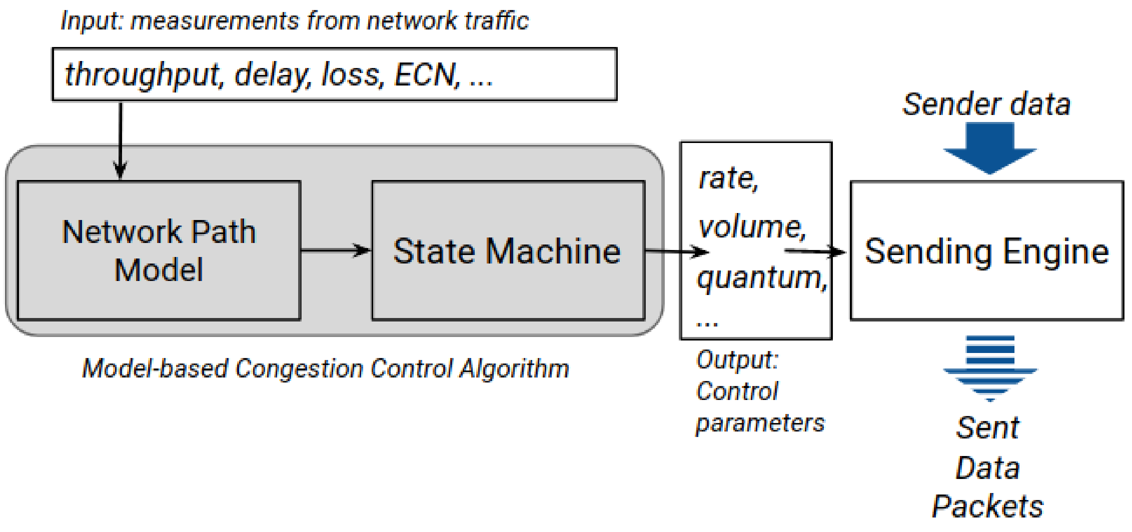
Input: measurements from network traffic
throughput, delay, loss, ECN, ...
Network Path Model
State Machine
Model-based Congestion Control Algorithm
rate, volume, quantum, ...
Output: Control parameters
Sender data
Sending Engine
Sent Data Packets

Fig. 1: Structure of an example model-based network congestion control

Algorithm Inputs: Network Traffic Measurements

A model-based congestion control algorithm updates its explicit model of the network path using as inputs measurements derived from the network traffic flow it is controlling. These measurements can include, but are not necessarily limited to, one or more of the following:

1. Network data delivery rate (achieved throughput, "goodput," or "bandwidth").
2. Round-trip time ("RTT") delay.
3. One-way delay from sender to receiver.
4. Data packet loss.

5. Explicit congestion notification (ECN) marks on data packets. (ECN marks are set by routers, switches, or other nodes along the network path when they experience congestion, e.g., a queue size over a configured threshold. The mark is echoed by the data receiver in acknowledgment packets it sends to the sender.)

The algorithm collects and processes these measurements periodically to update its model. The algorithm may process the updates at one or more of the following times: on every incoming packet, on some multiple or fraction of a network round trip, on time scales measured by elapsed time, or triggered by network, traffic, or state machine events.

Network Path Model

A model-based congestion control algorithm uses the aforementioned network measurements as inputs to dynamically update an explicit model of the network path, and then uses that model to control the sender's network sending process. Using the model of the network path can allow the congestion control algorithm to achieve higher throughputs and/or lower delays and/or lower data packet loss rates than would be achievable by other techniques.

The dynamic model parameters estimating characteristics of the network path(s) traveled by the flow can include one or more of the following, as well as potentially other parameters:

1. *Bottleneck bandwidth available to the flow*: the estimated maximum rate at which the network can deliver data for the flow without incurring any additional queuing or loss of data packets. In an example method to implement this technique, the model estimates this parameter by taking bandwidth samples computed as the average delivery rate over the time scale of one network round-trip, and using the maximum of all such bandwidth samples over the last R round trips.

2. *Round-trip propagation delay*: the estimated minimum round-trip time delay for packets traveling over the network path. This is the sum of (a) the time required for a data packet to travel from a sender to a receiver, and (b) the time required for a packet acknowledging that data packet to travel from the receiver to the sender. The minimum of such delays would reflect the minimal (e.g., 0) queuing delay experienced at each network node along the path. In an example method to implement this technique, the model estimates this parameter by taking round-trip time ("RTT") samples for each data packet by computing each RTT sample as the time elapsed between the transmission of a data packet and its acknowledgment, and then using the minimum of all such RTT samples over the last S seconds.
3. *In-flight capacity available to the flow*: the estimated maximum volume of unacknowledged in-flight data from the flow that should be held in transit in the network between the sender and receiver. Above this amount, data packet queues or data packet loss rates are estimated by the algorithm to be unacceptably high. In an example method to implement this technique, the model estimates this parameter by examining the time interval between when each data packet is sent and an acknowledgment for the data packet is received. The sender records with each sent data packet the amount of data, D, that it estimates was in flight at the start of that interval, and computes the data packet loss rate and data packet ECN mark rate over that interval. When it finds an interval where the loss rate and ECN mark rate were both below the corresponding configured thresholds for each rate, it ensures its estimate of this capacity parameter is at least as high as D. When it is raising the amount of data in flight, and finds either the loss rate or

ECN mark rate exceeds the configured threshold, it lowers its estimate of this capacity parameter to D.

4. *Degree of network aggregation*: the estimated degree to which data and/or acknowledgment packets have recently been aggregated or batched in the network path. In an example method to implement this technique, the model estimates this by examining at each point in time the amount of excess data recently delivered that was beyond the amount that was expected to be delivered given the current bandwidth estimate (1), and then using the maximum of all such excess data calculations over the last A round trips.

The model may maintain a value for each parameter over some time scale, where the time scales may be different for each parameter. In addition, for one or more parameters, the model may maintain an estimate over multiple time scales simultaneously, e.g., both a short-term summary of recent values and a longer-term estimate composed using a longer history.

The algorithm updates the model to adapt to changes in the network, the routes used by the network, or the traffic flowing over the network. The algorithm updates the model by adjusting model parameters over time, using mathematical and/or computational functions, adapting to events, e.g., including network traffic measurement inputs; the passing of time; transitions in the state machine; etc.

From the model parameters above, the algorithm can derive its key targets and bounds for the amount of unacknowledged data to maintain in-flight in the network. In an example method to implement this technique, the algorithm may compute the estimated bandwidth-delay product ("BDP") for the flow as the product of (1) the estimated bandwidth available to this flow, and (2) the round-trip propagation delay. Then the algorithm may calculate the target amount of data to maintain ("in flight") inside the network as a function of the BDP that is close to 1.0*BDP; similarly the maximum amount of in-flight data may be calculated as a mathematical function that is at least as large as the sum of the BDP and (4) the estimated degree of aggregation. Using these targets and bounds allows high utilization of network bandwidth while bounding the amount of data in flight, which reduces queuing latencies. The algorithm may place an additional bound on the maximum amount of in-flight data by bounding this to be no higher than the estimated network capacity available to the flow (3). This further reduces queuing latencies and packet loss rates.

State Machine

A model-based congestion control algorithm implements a state machine to use its model to inform its decisions about how to evolve the sending process over time. The state machine varies the sending behavior with several goals. First, it attempts to keep the amount of in-flight data high enough to achieve efficient utilization of the network path(s), but low enough to achieve low queuing delays and packet loss rates. Second, it attempts to obtain measurement samples to feed into the network model, to refresh the model and keep it up-to-date, allowing the algorithm to adapt to changes in the network, network routes, or traffic.

The state machine explores the path to see if more bandwidth can be achieved by placing more data in flight, or if lower queuing delays and loss rates can be achieved by placing less data in flight. Inherently, these cannot be achieved simultaneously, so the state machine necessarily alternates between increasing the amount of in-flight data by sending faster, and decreasing the amount of in-flight data by sending slower. It increases the amount of in-flight data to probe for available bandwidth, to inform the estimated available bandwidth (1) and allow higher throughputs. It reduces the amount of in-flight data to reduce queues, to inform the estimated round-trip propagation delay (2) and allow lower queuing delays and packet loss rates.

The state machine may trigger transitions based on a statically pre-configured time schedule, potentially scaled by the round-trip, and/or it may trigger transitions based on the estimated amount of data in flight in the network relative to the estimated BDP, computed as described above. In addition, it may trigger transitions based on recent measurements, such as measuring loss rates and/or ECN mark rates cross above configured upper and/or lower tolerance thresholds.

The state machine may also attempt to coordinate some state machine phase transitions between flows sharing a network bottleneck. In an example method to implement this technique, the state machine may try to periodically reduce the amount of in-flight data to try to empty the flow's bottleneck queue, to increase the chance that flows traversing that queue may measure a delay sample that more accurately reflects the round-trip propagation delay of their paths.

Algorithm Outputs: Network Sender Control Parameters

A model-based congestion control algorithm computes, as a function of its network model parameters and state machine variables, a set of outputs that comprise one or more control parameters governing the sending process of the sending engine. The sending engine may be implemented by the software or hardware implementation of an end-to-end transport protocol (e.g. TCP or QUIC); or it may be implemented by the software or hardware of traffic rate control mechanisms (traffic shaping or policing) inside any node of the network, including end hosts, switches, routers, or other nodes.

A model-based congestion control algorithm may export as output control parameters that include one or more of the following, as well as potentially other parameters:

1. The maximum amount of data (or "quantum") that the algorithm should schedule for transmission at a single time, in a single transmission decision made by the network sender software or hardware.
2. The maximum sending rate (or "pacing rate") that the sender may currently use. This rate specifies a minimum time separation that the sending engine should impose between data packet "quanta" (of one or more data packets scheduled in a single decision).
3. The maximum amount of unacknowledged data to allow "in flight" in the network path (or "congestion window").

In addition, the algorithm may output, export, or log model parameters for the purposes of debugging, troubleshooting, benchmarking, or telemetry for characterizing the performance of the network traffic or the network path(s).

CONCLUSION

This disclosure describes model-based congestion control, a technique which uses network measurements as inputs to maintain an explicit model of the network path between a sender node and a receiver node, and then uses that model to generate output parameters that control the sender's network sending process. This model-based approach can allow the congestion control algorithm to achieve higher throughputs and/or lower delays and/or lower data packet loss rates than would be achievable by other congestion control techniques.

REFERENCES

1. Transmission Control Protocol, DARPA Internet Program Protocol Specification September 1981, available online at https://tools.ietf.org/html/rfc793 accessed Mar 18, 2019.

2. QUIC: A UDP-Based Multiplexed and Secure Transport, available online at https://tools.ietf.org/html/draft-ietf-quic-transport accessed Mar 18, 2019.
3. Cardwell, Neal, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. "BBR: congestion-based congestion control." Communications of the ACM, Volume 60, Issue 2, February 2017, available online at https://cacm.acm.org/magazines/2017/2/212428-bbr-congestion-based-congestion-control/pdf accessed Mar 18, 2019
4. Cardwell, Neal, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. "BBR: Congestion-Based Congestion Control - Measuring bottleneck bandwidth and round-trip propagation time" available online at https://queue.acm.org/detail.cfm?id=3022184 accessed Mar 18, 2019.
5. Cardwell, Neal, Yuchung Cheng, Soheil Hassas Yeganeh, and Van Jacobson. "BBR Congestion Control" available online at https://tools.ietf.org/html/draft-cardwell-iccrg-bbr-congestion-control accessed Mar 18 2019.
6. Cheng, Yuchung, Neal Cardwell, Soheil Hassas Yeganeh, and Van Jacobson. "Delivery Rate Estimation" available online at https://tools.ietf.org/html/draft-cheng-iccrg-delivery-rate-estimation-00 accessed Mar 18 2019.