September 27, 2019

Estimation Of Network Link Capacity Using Particle Filtering

Sebastian Jansson

Estimation Of Network Link Capacity Using Particle Filtering

ABSTRACT

In real-time communications (RTC) over the internet (WebRTC), an optimal trade-off is sought between the quality and the latency of transmission. For example, a high-resolution video call requires a large throughput, which can cause congestion at network nodes, in turn resulting in unacceptable call delay. Current RTC protocols use congestion control mechanisms to try to achieve an acceptable quality-versus-latency trade-off. Under certain circumstances, these protocols are sensitive to delay spikes, compensate inadequately for large data packets, lack robustness in situations where media encoders over-produce data, and use test probes that often return inaccurate estimates of network capacity.

The techniques of this disclosure address the problems of congestion control mechanisms by estimating the current network state using particle filtering. The network state is used to set the bit-rates of media encoders such that the quality-latency trade-off is optimized.

KEYWORDS

- Network state
- Network capacity
- Capacity estimation
- Particle filtering
- Network modeling
- WebRTC
- Quality-latency trade-off
- Encoder bit-rate

BACKGROUND

In real-time communications (RTC) over the internet (WebRTC), an optimal trade-off is sought between the quality and the latency of transmission. For example, a high-resolution video call requires a large throughput, which can cause congestion at network nodes, in turn resulting in unacceptable call delay. Current RTC protocols use congestion control mechanisms to try to achieve an acceptable quality-versus-latency trade-off. Some such mechanisms are limited due in that they are not designed for real-time media, and hence do not advantageously control the media encoder bit-rate. Others try to measure the network state via self-induced delays, which limits link utilization.

Other techniques for congestion control, even those specifically designed for real-time communications, assume a static model of the network, which makes performance sensitive to unexpected variations in delay. These techniques also assume that the media encoder can perform well at a targeted bit-rate, but if the encoder fails to do so, e.g., it over-produces data (e.g., as in screenshare scenarios), congestion control fails.

Other problems with current congestion control mechanisms include their dependence on sending test probes to determine the present network capacity. If the test probes coincide with delay spikes, it results in an underestimation of network capacity which in turn prevents the ramp-up of data throughput; on networks that allow traffic bursts, the test probes overestimate network capacity, leading to congestion and high latency.

DESCRIPTION

Per the techniques of this disclosure, the network state is estimated by the application of particle filtering on a model of the network. The determination of the network state leads to a real-time estimate of network capacity and in turn, optimal loading of the network, e.g., the network is neither underloaded (which manifests as inadequate throughput, e.g., video-call resolution) nor is the network overloaded (which manifests as call latency). The estimate of network state or capacity can be performed continuously or periodically, since the network itself is subject to changes in topology or traffic conditions.

Per the principles of particle filtering, the estimate of network capacity is obtained by considering several hypotheses of the network state and pruning those hypotheses that don't match with the observed behavior, e.g., end-to-end latency, of the network. When a network state that optimally fits observed network behavior is identified, media encoders are tuned to match the network state without the need for test probes or gradual throughput ramp-ups.

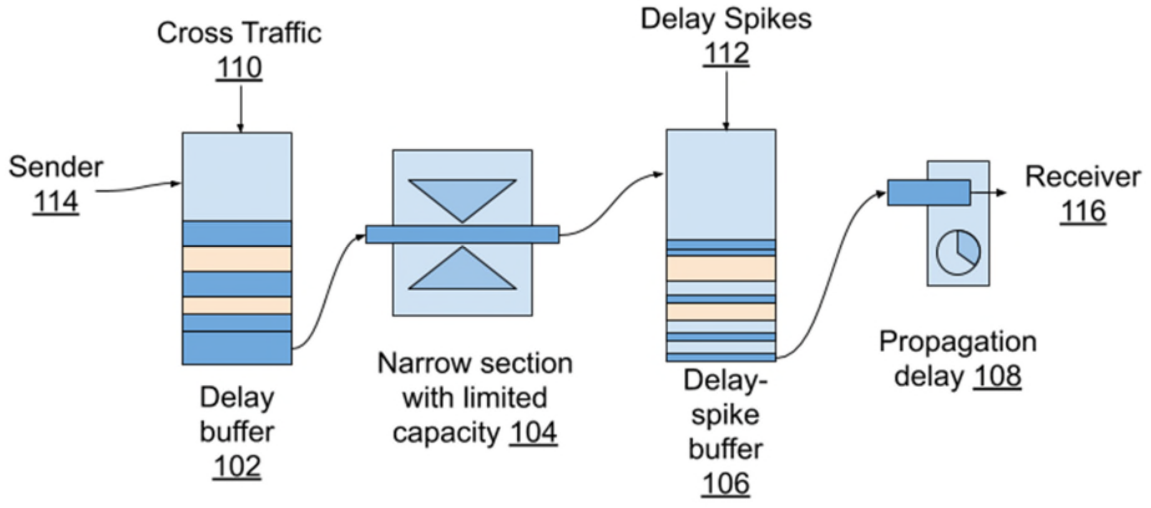

Fig. 1: The network link model

As shown in Fig. 1, the network link between a sending node (sender 114) and a receiving node (receiver 116) is modeled as comprising a delay buffer (102), a narrow section with limited capacity (104), a delay-spike buffer (106), and a fixed propagation delay (108). A network comprises many such links, but it is generally the weakest links that most affect network behavior, such that a limited number of links, even just one, adequately models the network. In addition to the sending-node data filling the delay buffer, randomly generated cross-traffic (110) is also inserted into the delay buffer. Random delay spikes (112) are inserted into the delay-spike buffer. The following state variables characterize the network link model:

- Buffer delay before the narrow section (`pre_link_buffer_delay`)
- Bandwidth capacity of the narrow section (`link_capacity`)
- Buffer delay after the narrow section (`post_link_buffer_delay`)
- Propagation delay (`propagation_delay`)
- An output state used to track cumulative random cross-traffic (`cross_traffic_sum`)
- An output state used to track cumulative random delay spikes (`delay_spike_sum`)

Particle filtering is a technique to estimate internal states of a dynamical system based on partial and noisy observations of the behavior of the system. Per the techniques, a particle filter estimates the states of the above network link model as follows.

- State update, based on information about packet send times.
- Measurement update, based on measured end-to-end latency.

The particle filter works by evaluating the fit of a set of state hypotheses to the actually measured data, e.g., the per-packet end-to-end latency. In this context, hypotheses are referred to as particles. Hypotheses (particles) that don't fit with the data are removed, and hypotheses that fit well with the data are duplicated. Over time, this procedure provides an increasingly accurate picture of the state of the network.

```
time_delta = packet.send_time - last_send_time
last_send_time = packet.send_time

link_capacity += random_link_capacity_change(time_delta)
propagation_delay += random_propagation_delay_change(time_delta)

cross_traffic = random_spike(time_delta)
cross_traffic_sum += cross_traffic
pre_delay = max(pre_delay+cross_traffic-time_delta, propagation_delay)
pre_delay += packet.size/link_capacity

delay_spike = random_spike(time_delta)
delay_spike_sum += delay_spike
post_delay = max(post_delay+delay_spike-time_delta, 0)
```

Fig. 2: Pseudo-code to perform state update based on packets transmitted

Fig. 2 illustrates pseudo-code that performs a state update based on packets transmitted across the network. For every packet transmitted across the network, each particle is updated with the expected state change that that particular packet would cause. As illustrated in Fig. 2, random components are added to represent the stochasticity of the network.

```
measured_latency = packet.receive_time - packet.send_time
predicted_latency = pre_delay + post_delay
error = predicted_latency-measured_latency
weight *= exp(-error²/(LATENCY_MEASURE_ERROR*2))
```

Fig. 3: Pseudo-code to perform state update based on feedback

Fig. 3 illustrates pseudo-code that performs a state update where feedback is available. For packets where feedback is available, the latency prediction of the model is compared with the measured latency to update the weight of the particle.

To provide link estimates, a weighted mean of each state variable over all particles is obtained. Note that this isn't guaranteed to be a valid state, as outliers can bias the mean. This is generally not a problem, as the particles tend to cluster together, but sometimes unrealistic edge cases have to be specially handled.

As mentioned before, particles with low weight, e.g., that don't fit the observed data, are removed in a resampling step. The resampled particle set is based on a weighted sample of the previous particle state. In effect, this means that each previous particle is repeated based on its weight in the output set.

As mentioned before, the network state, as determined by the particle filter, is used to control the target bit-rate for media encoders. The model-based estimation enables the prediction of the effect of using a certain target bit-rate which in turn enables the optimization of the quality-latency trade-off. In particular, the network state estimate can restrict the increase in target bit-rate and can be used as a reference for back-offs in target bit-rate that are initiated due to network re-use. The network state estimate can be used in conjunction with other control schemes. For example, the estimated variance of the network state estimate can be used to control the target bit-rate to enable quicker throughput increase; buffer delay estimates can be used to directly control the target bit rate by backing off encoder bit-rate based on estimated delay; cross-traffic estimates can be used to achieve precise control over fairness; etc.

The network model can be extended to include more complex descriptions of the network, e.g., by the inclusion of more capacity-limited sections and delay buffers; by the use of non-Gaussian (long-tail) noise to model uncertainty in link capacity and propagation delay; by the use of advanced cross-traffic models that better predict cross-traffic behavior, etc. Filters other than particle filters can be used to model the network, e.g., Kalman filters, extended or modified Kalman filters, etc. The techniques apply generally to network transmission schemes, e.g., TCP, QUIC, real-time streaming applications, etc. and can improve throughput and optimize network load.

In this manner, the techniques of this disclosure use particle filtering to obtain an estimate the state of a network, in turn enabling the precise prediction of the delay build-up that would result at various encoder bit-rates. Real-time communication across the network is optimized such that it is not so slow as to affect communications quality, nor is it so fast that it suffers from excessive latency.

CONCLUSION

The techniques of this disclosure control network congestion by matching media encoder bit-rates to a real-time estimate of the capacity of the network. The capacity of the network is estimated by applying particle filtering on a model of the network.