Practical Passive Shared Bottleneck Detection Using Shape Summary Statistics

David A. Hayes*, Simone Ferlin†, Michael Welzl*
*University of Oslo, Norway {davihay,michawe}@ifi.uio.no
†Simula Research Laboratory AS, Norway, ferlin@simula.no

***Abstract*—Practical shared bottleneck detection has proved to be a difficult problem. We present a novel passive approach using efficient estimates of time and frequency domain summary statistics. The approach is not CPU nor network intensive, and has numerous potential applications in the Internet. Simulations and tests over the Internet and 3G cellular network show its efficacy in grouping flows correctly.**

***Index Terms*—Shared bottleneck detection**

I. Introduction

In the Internet, it is not normally known if flows (e.g., TCP connections or UDP data streams) traverse the same bottlenecks. Even flows that have the same sender and receiver may take different paths and share a bottleneck or not. This plays a major role in performance; for example, a congestion control mechanism for scavenger-type traffic like LEDBAT [19] tries to not get in the way of a user's other traffic, but if it does not share a bottleneck with this other traffic, it may well only step out of the way of traffic from other network users, with no benefit whatsoever to the user who utilizes this form of congestion control.

Flows that share a bottleneck link usually compete with one another for their share of the capacity. This competition has the potential to increase packet loss and delays. This is especially relevant for interactive applications that communicate simultaneously with multiple peers (such as multi-party video). Combining the congestion controllers of such flows can not only reduce packet loss and delay [24] but also allow precise honoring of priorities that may be associated with particular flows (cf. [18]). A mechanism for doing this with flows in the new WebRTC standard for inter-browser real-time communication is being proposed for standardization in the RMCAT Working Group of the IETF [24]. Since this mechanism will rely on shared bottleneck detection, it is also planned to propose the shared bottleneck detection method described in this paper for standardization in RMCAT.

Other potential applications of shared bottleneck detection include the ability to better predict the duration of file transfers for the sake of better scheduling in distributed processing [26], better peer selection when downloading from multiple sources in parallel [7], and applying the right form of congestion control for multi-path transfers [6]. To the best of our knowledge, the latter application was also the first context in which shared bottleneck detection was proposed for the Internet in the IETF (for the SCTP protocol), but up to now, such methods have been regarded as being too unreliable and/or computationally intensive for practical use.

This paper presents a novel passive method of detecting if flows share a common bottleneck using packet delay distribution summary statistics, particularly shape. Because the calculations in our method are simple, the amount of feedback required is small, and the accuracy and responsiveness in detecting shared bottlenecks high, it may have the potential to be a practical scheme for online application in the Internet. After a discussion of related work in Section II, we will then discus the signals that can be used for share bottleneck detection in Section III, the reasoning behind our design using summary statistics in Section IV and our mechanism in Section V. We then evaluate our design experimentally through simulation and real network tests in Section VI and conclude in Section VII.

II. Related work

Early work on SBD by [17] suggests host and network locality schemes, but the authors acknowledge that the granularity is not fine enough to be useful for their proposal of *informed transport protocols*. Balakrishnan *et al.* [2] cluster nearby Internet hosts based on similar long term throughput distributions in their analysis of web servers for the Atlanta Summer Olympics. This work is relevant to our work in that it uses a summary statistic, to group flows sharing a bottleneck.

Rubenstein *et al.* [16] conducted the first thorough investigation of the problem, proposing both delay and loss based correlation techniques. Rubenstein *et al.* propose measuring the correlation between pairs of packets within flows, and between flows. If the cross-correlation of packet pairs between flows is greater than the auto-correlation of pairs within each flow, it is concluded that the two flows share a common bottleneck. Although they use Poisson probes, they suggest that in many cases the probes can be incorporated into the underlying data stream. Rubenstein *et al.* find that there are often not enough samples for the loss-correlation technique to be able to conclude one way or the other.

A number of other techniques adopt the delay-based correlation method of Rubenstein *et al.* These include Round Trip Time (RTT) based methods such as Wang *et al.* [22] and Younis and Fahmy [25]. The method in [10] uses One Way Delay (OWD), but applies a wavelet transform to filter the noise by suppressing smaller wavelet coefficients in the signal. Although delay is measured at the receiver, it requires each packet's measurement to be sent back to the sender. However, it should be noted that the Pearson correlation coefficient used by Rubenstein *et al.* [16] and the other papers assumes normally distributed samples, which is not the case during congestion, where the delay distribution can be highly skewed (see Section IV-A), reducing the effectiveness of the correlation coefficient as an estimator of correlation [12].

Yousaf [26] uses a covariance matrix of average OWDs. Singular Value Decomposition (SVD) of the matrix filters the noise. The resulting factors, one per flow, are considered to indicate path sharing if they are similar. Unfortunately the grouping algorithm presumes that there will always be more than one group, which will not always be the case. Also, like the aforementioned correlation techniques it is computationally intensive.

Katabi and Blake [9] investigate shared bottleneck detection using the Rèyis entropy of the packet inter-arrival times. Varga [21] further investigates the work of Katabi and Blake, but limits bottleneck detection to bottlenecks that experience continuous sever queueing. This work investigates the higher order moments of skewness and kurtosis of the packet inter-arrival times. The use of higher order statistics is relevant to our work, however, the use of packet inter-arrival times as a base measure limits its usefulness for our purposes. This requires measurement of a significant proportion of the bottleneck traffic, and the existence of significant bottleneck back-to-back pair measurements that are not overly perturbed by the remaining hops along the path.

All of the above correlation techniques presume that there are no significant differences in lag for the different paths sharing a common bottleneck. In other words, the path samples being compared are presumed to be synchronised. It is not uncommon for different paths that share a common bottleneck to have significantly different end-to-end delays. Under these circumstances the cross correlation techniques will not work well.

Three additional techniques deserve mention. Kanuparthy *et al.* [8] use an active probing method which sends a specially patterned probing packet sequence, which characteristically perturbs other sample traffic that shares a common bottleneck with the probe traffic. Hassayoun *et al.* [6] look at the problem in relation to dynamic window coupling for Multipath TCP or SCTP, but without evaluating the accuracy of the underlying shared bottleneck detection technique, where flows with similar temporal congestion control dynamics are grouped together. Zhu [27] uses the frequency spectrum of active probe packet inter-arrival times to detect TCP path sharing.

In general the works have some common shortcomings: 1) simulation tests have been conducted with scenarios that make shared bottleneck detection easy; 2) real network tests often do not have a proper *ground truth* with which to compare; and 3) quantitative statistics, although valuable, often hide the dynamics of bottlenecks carrying real traffic. Apart from proposing a novel mechanism for detecting shared bottlenecks, this paper attempts to overcome these shortcomings.

III. Signals

Unless the network directly signals that flows share a common link, say by marking packets, end hosts need to infer this from the packets they receive. The raw information that Internet end-hosts have is: packet delay, packet loss, and ECN (Explicit Congestion Notification) marking. Both packet loss and traditional ECN marks are relatively rare, representing a small percentage of packets. The rarity makes it difficult to use for shared bottleneck detection, though it can be used as a supplement. Packet delay is then the primary signal.

A. Packet Delay

Packet delay tends to be a very noisy signal. The delay a packet experiences along an end-to-end path can be summarised as:

$$T_{\text{path}} = \sum_l T_{\text{prop}}^{(l)} + \sum_l T_{\text{queue}}^{(l)} + \sum_l T_{\text{misc}}^{(l)}\ ,\ \forall l \in L, \qquad (1)$$

where $T_{\text{prop}}^{(l)}$ is the propagation delay along link l, $T_{\text{queue}}^{(l)}$ is the queueing delay at link l, and $T_{\text{misc}}^{(l)}$ are miscellaneous other delays experienced on link l, such as switching delays and media access delays, and L is the set of links along the path. If a Round Trip Time (RTT) measurement is used the reverse path delay is added to this as well.

For detecting shared bottlenecks the delay characteristics of the bottleneck link $T_{\text{b\_signal}}$ (at link $l = b$) are the key elements, but they are masked by other sources of delay along the path:

$$T_{\text{b\_signal}} = T_{\text{queue}}^{(b)} + T_{\text{misc}}^{(b)} \qquad (2)$$

and the rest of the delay is noise

$$T_{\text{path\_noise}} = \sum_l T_{\text{prop}}^{(l)} + \sum_{l \neq b} T_{\text{queue}}^{(l)} + \sum_{l \neq b} T_{\text{misc}}^{(l)}\ ,\ \forall l \in L, \quad (3)$$

For wired bottleneck links this is dominated by $T_{\text{queue}}^{(b)}$; for wireless bottleneck links the media access component of $T_{\text{misc}}^{(b)}$, in some sense also a form of queueing delay, is also of interest and may even dominate the delay measurement.

B. Dealing with Noise

For the delay signal to be useful in determining shared bottlenecks, the delay characteristics of the bottleneck, $T_{\text{b\_signal}}$, must be able to be distinguished from the path noise, $T_{\text{path\_noise}}$. However, without special knowledge of the path characteristics, we require $T_{\text{b\_signal}} \gg T_{\text{path\_noise}}$.

If One Way Delay (OWD) rather than RTT is used, up to half of the potential sources of noise are removed, providing a much cleaner signal. For this reason we measure the delay signal at the receiver. In the simplest scenario coupled congestion control will be implemented in the sender, requiring a mechanism to provide the necessary feedback. This mechanism is an important element in a coupled congestion control system, but is beyond the scope of this paper and an item for further study.

Even if we can perfectly measure $T_{\text{b\_signal}}$, it in itself is a difficult signal to deal with. Queueing delay is very dynamic, for example, two flows sharing the same queue will have packets that experience widely different and seemingly uncorrelated delays at similar epochs in time (see [4]). As a result, some sort of filtering is still required, as well as some means of coping with the differing lags of each path.

C. Unsynchronised sender and receiver clocks

Lack of synchronisation between different senders' and receivers' clocks means that raw OWD measurements cannot be directly compared. However, provided both sender and receiver use the same clock resolution and the offset between the clocks remains constant over the time window used for calculations, statistics that are based on differences in OWD remove the constant offset, C:

$$T_1 = t_1^{(\text{rx})} - t_1^{(\text{tx})} + C \text{ and } T_2 = t_2^{(\text{rx})} - t_2^{(\text{tx})} + C \quad (4)$$

$$\Delta T_{1,2} = t_1^{(\text{rx})} - t_2^{(\text{rx})} + t_2^{(\text{tx})} - t_1^{(\text{tx})} \quad (5)$$

This means that although $\overline{T}_{\text{path}}$ cannot be directly compared, higher order central moments can – as their difference operation removes the constant offset. This is also how OWD is applied in some delay-based congestion controls, e.g. LEDBAT [19]. Clock drift and corrections to clocks made by NTP still add noise to the delay measurement. However, over the relatively small measurement intervals, the clock drift can be considered to be negligible [19] and provided corrections to clocks made by NTP perturb a minority of measurements they too can be ignored.

IV. Summary Statistics

A key contribution of this paper is the use of summary statistics[1] to group flows sharing a bottleneck. Summary statistics filter the delay signal to make it more manageable. They also help cope with path lag, provided the period of time over which the summary is made is significantly longer than the lag. A summary statistic is smaller than the original data, and only needs to be fed back to the senders at intervals equivalent to the summary period.

Higher order summary statistics describe the shape of the data, in this case T_{path}. We observe that the shape of the delay signal often differs according to the load at the bottleneck and the dynamics of the traffic traversing it. In this respect we base our mechanism on the second and third central moments of the T_{path}. We also observe that the delay signal has a shape in time. We use the *key frequency* of the delay signal as a summary statistic to describe this shape.

A. Insight from an M/M/1/K queue

The mechanism is based on the observation that the shape of the one way delay distribution is dominated by the bottleneck queue. We use a simple M/M/1/K queue to gain insight into the value of the time based summary statistics we use to describe the shape. We postulate that the shape is usually distinct enough to distinguish between flows sharing small numbers of different bottlenecks. Although the M/M/1/K queue is a simplistic model for Internet traffic, it suffices for this purpose.

We investigate the second and third central moment of the delays introduced by the queue. The i^{th} central moment can be calculated from the response time density as follows [20]:

$$m_i = \int_{-\infty}^{\infty} f_r(t)(t - \overline{T})^i dt \quad (6)$$

where

$$f_r(t) = \sum_{k=0}^{K-1} \frac{p_k}{1 - p_K} f_{E_{k+1}}(t) \quad (7)$$

and $f_{E_k}(t)$, the Erlang density function, is

$$f_{E_k}(t) = \frac{(\mu t)^k}{tk!} e^{-\mu t} \quad (8)$$

where k is the number of packets in the queue, p_k is the probability of having k packets in the queue, and $f_{E_k}(t)$, the k packet Erlang density function where μ is the service rate of the queue.

Fig. 1a shows the normalised second, and third central moments of the response time of an M/M/1/50 queue versus load. If different bottlenecks experience different loads at different times, then grouping flows according to differences in their summary statistics should be feasible (in practice the different bottlenecks can also have different queue sizes).

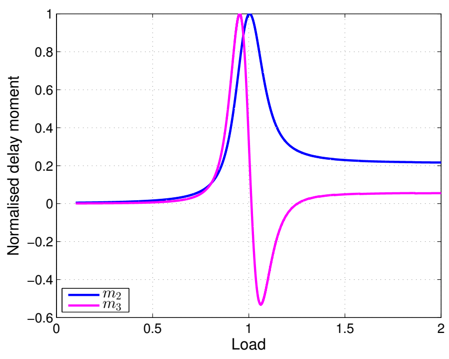
Attempting to group flows that are not experiencing congestion is impractical, since the delay signal is small relative to the noise. Without prior knowledge of maximum queue depths along the path, the magnitude of $\max(T_{\text{path}}) - \min(T_{\text{path}})$ cannot robustly be used for this. Fig. 1a shows that the skewness, m_3, changes from positive to negative near 100% load. This makes it a very good metric for this purpose, as congestion occurs when the load exceeds 100% during the measurement period. However, the skewness does become positive again at extreme sustained loads of above 125%, where the loss rate exceeds 25%. Therefore flows can be grouped into the F_{cong} group if their skewness is negative, or their loss rate is very high.

The variance, m_2, is a monotonically decreasing function for flows experiencing congestion, making it a useful candidate for distinguishing between flows based on relative bottleneck load. Unfortunately it changes the least for load levels close to 100%. The skewness, m_3, changes most when the variance, m_2, is flat, hence it can help group flows in this area. Skewness on its own is not a good metric for two reasons: it is hard to measure with precision due to its susceptibility to extreme sample values, and it is not either monotonically increasing or decreasing at congestion loads, meaning two identical skewness values can result from two very different loads.

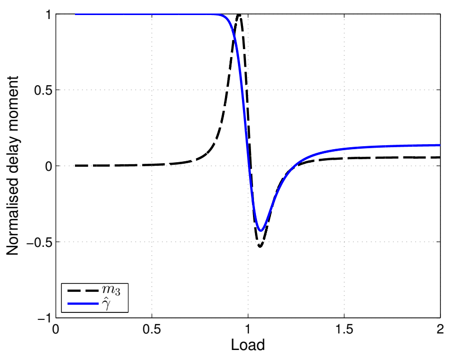
B. Making it practical

Variance and skewness can be difficult to calculate. Skewness, m_3, is particularly difficult to estimate accurately without a large sample size. With errors in estimation being magnified

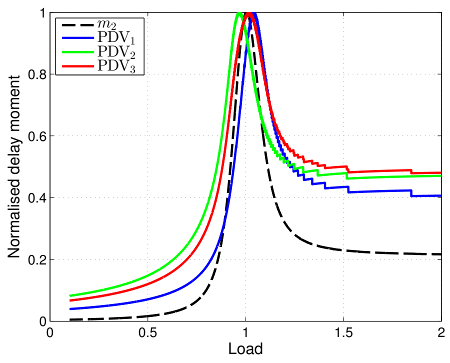
[1]descriptive statistics that summarise a set of observations



(a) Variance (m_2), and skewness ($\gamma = m_3$)



(b) Comparison of skewness estimate ($\hat{\gamma}$) with $\gamma = m_3$



(c) Comparison of the different estimates of variance ($\hat{v}$) – PDV$_1$, PDV$_2$, and PDV$_3$.

Fig. 1. Graphs of normalised delay moment versus load for the response time of the M/M/1/50 queue

by its cubic nature—which also makes it CPU intensive, it is sensitive to extreme values. However, for the purposes of this algorithm, we are most concerned as to whether m_3 is positive or negative, and its shape when m_2 is flat. An estimate of m_3 can be made by counting the proportion of samples above and below the mean. This type of shape estimate is not influenced by extreme values in T_{path} measurement since extreme values carry no more weight than non-extreme values:

$$\hat{\gamma}_i = \frac{\sum_{n \in N_i}(T_n < \overline{T}_i) - \sum_{n \in N_i}(T_n > \overline{T}_i)}{N_i} \tag{9}$$

where N is the number of samples and $-1 \le \hat{\gamma} \le 1$. Fig. 1b compares m_3 with its estimate $\hat{\gamma}$, showing that it still has the properties we require. If the mean from the previous sample interval, $\overline{T}_{i-1}$, is used to proportion delay samples there is no need to store samples for the calculation. This also has the effect of making the metric more sensitive to change, potentially allowing earlier detection of bottlenecks.

The Packet Delay Variation (PDV) [13] can be used as an estimate of the variance, $\hat{v}$. There are three possible PDV measures (see Fig. 1c):

$$\text{PDV}_1 = \overline{T}_{\text{path}} - \min(T_{\text{path}}) \ , \tag{10}$$
$$\text{PDV}_2 = \max(T_{\text{path}}) - \overline{T}_{\text{path}} \ , \text{ and} \tag{11}$$
$$\text{PDV}_3 = \max(T_{\text{path}}) - \min(T_{\text{path}}) \ . \tag{12}$$

Note that $\min(T_{\text{path}})$ and $\max(T_{\text{path}})$ are time interval dependent (Fig. 1c uses a 95% percentile of the probabilities). Measurement with respect to $\min(T_{\text{path}})$ is more stable than measurement with respect to $\max(T_{\text{path}})$ as $\min(T_{\text{path}})$ has a hard limit of the propagation delay, and is generally the OWD measure with the least noise. Unfortunately (10) is shifted to the right of the true variance making it possible to have two identical PDV measurements for two very different bottleneck load conditions (for F_{cong}); the estimate of skewness, $\hat{\gamma}$, is unable to aid as it has a similar issue at the same loads. $\max(T_{\text{path}})$ is the noisiest measurement, since it is due to not simply the maximum queueing delay at the bottleneck link, but is likely to include the maximum amount of noise from other queues along the path. Despite this, it is better to measure with respect to $\max(T_{\text{path}})$ to avoid the aforementioned problem. In view of this (11) provides the best characteristics to use for $\hat{v}$ in the proposed mechanism.

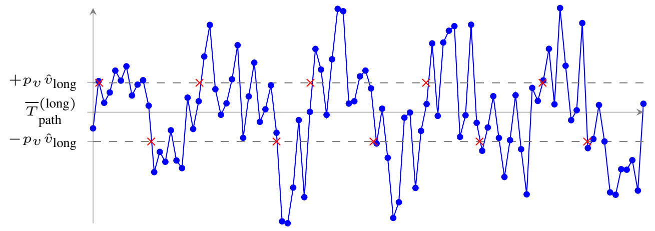


Fig. 2. Illustration of key frequency of bottleneck estimation ($\hat{f}$)

C. Estimating the key frequency of the bottleneck link

The key frequency of the bottleneck could be thought of as its fundamental frequency[2] of oscillation over a certain period of time. This could be calculated from the Fourier transform of T_{path}, however, the mechanism only requires a consistent measurement of a key frequency characteristic in order to be able to make comparisons. We propose a measure of the number of *significant* crossings (normalised by the number of samples being considered), of the short term mean of the OWD signal, $\overline{T}^{(\text{short})}_{\text{path}}$, about a longer term mean of the OWD signal, $\overline{T}^{(\text{long})}_{\text{path}}$. We define *significant* crossings as those that cross $\overline{T}^{(\text{long})}_{\text{path}} \pm p_v \hat{v}_{\text{long}}$, where $0 < p_v \le 1$. In practice a value of $0.1 \le p_v \le 0.2$ works well.

Fig. 2 illustrates estimation of the key frequency of the bottleneck link, $\hat{f}$. The blue dot points are $\overline{T}^{(\text{short})}_{\text{path}}$ estimates, with the red cross points indicating countable crossings. In this example the normalised key bottleneck frequency is $\hat{f} = 10/100 = 0.1$.

V. Mechanism Design

For practical purposes such as coupled congestion control in RMCAT, the mechanism is designed to work with:

- a relatively small numbers of bottlenecks (up to about 10)

[2]lowest frequency of a periodic waveform

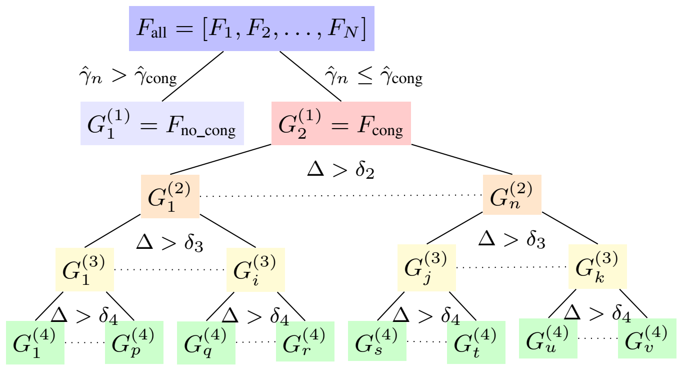


Fig. 3. Grouping Algorithm. At each step, flows are placed in the same group when the difference in their measured summary statistic Δ exceeds a threshold δ.

- a relatively small numbers of flows in the order of two to tens
- provide grouping decisions in hundreds of milliseconds to seconds, and
- work with multiple senders and receivers (though the protocol to distribute summary statistics and grouping decisions to the various senders and receivers is beyond the scope of this paper).

The mechanism presented has two parts: summary statistic generation and flow grouping. Both packet delay and loss are used as base measures, however delay is the main measure with loss only used when the bottleneck is under extreme loads (see Section IV-A).

A. Summary statistics

We use estimates of three summary statistics of OWD at the bottleneck link. Each has key attributes, which we exploit:

- An estimate of the skewness in OWD (3rd central moment), $\hat{\gamma}$. Estimated over T_i, and averaged over N samples.
- An estimate of the variance in OWD (2nd central moment), $\hat{v}$. Estimated over T_i, and averaged over N samples.
- An estimate of the key frequency component of OWD at the bottleneck link, $\hat{f}$.

Bottlenecks can be very dynamic. The time interval that these are gathered over depends on how an application wants to deal with these dynamics. Generally, $\hat{\gamma}$ and $\hat{v}$ should not be gathered over an interval less than the queue length of the bottleneck queue. $\hat{f}$ requires a longer time interval, as it is especially looking at the lower frequency component of the bottleneck (see Section VI for examples).

B. Grouping

Grouping flows that share a common bottleneck is a clustering problem. In this case, we are trying to identify clusters in the three dimensional space $(\hat{\gamma}/P_{\text{loss}},\ \hat{v}/[\hat{\gamma}/P_{\text{loss}}],\ \hat{f})$.There are many solutions to the clustering problem [3] which vary in how they define clusters and in the processing power required to determine them. We adopt a simple divide-and-conquer approach, extending the technique in Yousaf [26].

The algorithm is shown diagrammatically in Fig. 3. The steps are as follows:

1) First the flows are divided into flows experiencing no congestion, $F_{\text{no\_cong}}$, and flows experiencing congestion, F_{cong}. Generally, when the estimate of the skewness is less than a threshold, $\hat{\gamma}_n < \hat{\gamma}_{\text{cong}}$, for flow n it is grouped with the flows experiencing congestion, F_{cong}. During extended periods of extreme congestion with high loss this relationship does not hold, so loss is used instead. Typically $\hat{\gamma}_{\text{cong}} = 0$, however this parameter can be adjusted to tune how much congestion a flow must experience to be grouped with F_{cong}. Really this is a decision as to whether grouping should be attempted or not, with grouping only attempted when $\hat{\gamma}_n < \hat{\gamma}_{\text{cong}}$.
2) Flows determined to be experiencing congestion (i.e. flows traversing a bottleneck) from step 1, $G_2^{(1)}$ are further divided using the key frequency estimate, $\hat{f}$, of the different flows. $G_2^{(1)}$ is sorted in descending order by $\hat{f}$. Flows that differ in $\hat{f}$ by less than $\hat{f}_{\text{th}} = \delta_2$ are grouped together.
3) Flows are then further divided using the estimated variance $\hat{v}$. $G^{(2)}$ are sorted by $\hat{v}$ in descending order. Flows that differ in $\hat{v}$ by less than $\hat{v}_{\text{th}} = \delta_3$ are grouped together.
4) The resulting groups from step 3,$G^{(3)}$, are now divided using either $\hat{\gamma}$, or P_{loss} if the loss rate is very high. $G^{(3)}$ are sorted in descending order by $\hat{\gamma}$. Flows that differ in $\hat{\gamma}$ by less than $\hat{\gamma}_{\text{th}} = \delta_3$ are grouped together. Steps 3 and 4 work together (see Section IV-A).

VI. Experimental verification

Experimentally verifying SBD on a real network is difficult as we need to know the "ground truth" concerning bottlenecks to confirm correct grouping by the algorithm. Therefore we have taken a twofold approach: verifying the algorithm's operation through NS2 simulations, and through a specially constructed real network test using the NorNet test bed (https://www.nntb.no). Both tests use the same SBD mechanism parameters (see Table I). This highlights the robustness of the mechanism under very different network conditions.

A. Simulations

NS2 [14] simulations are conducted using the topology depicted in Fig. 4. The potential bottlenecks are shown with a thick blue line. Each of the first six (from the left) bottlenecks carry traffic from a different set of two test sources, with the last bottleneck carrying traffic from all four test sources. Bottlenecks have variable buffer sizes (100–500 pkts, averaging about 100 ms in length) and are instantiated by reducing the capacity of the nominated link.

This topology allows a range of test scenarios, including "worse than real" scenarios. The results included in this paper use the parameters outlined in Table II. Link delays are fixed for the duration of the experiment, but are generated randomly according to a normal distribution.

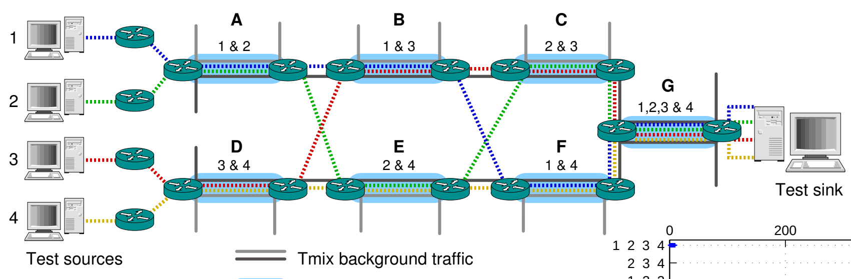


Fig. 4. Shared bottleneck detection NS2 simulation set up.

TABLE I
SHARED BOTTLENECK DETECTION MECHANISM PARAMETERS (SEE V). THE MECHANISM MAKES GROUPING DECISIONS EVERY T_I MS.

T_I	N used in estimates		Thresholds				
(ms)	μ, $\hat{v}$ & $\hat{\gamma}$	$\hat{f}$	p_v	δ_v	δ_γ	δ_f	$\hat{\gamma}_{\text{cong}}$
350	50	50	0.2	0.3	0.2	0.2	0.0

TABLE II
SIMULATION PARAMETERS (BN – BOTTLENECK, BG – CONNECTING TMIX TRAFFIC GENERATORS,AND OTHER – ALL OTHER LINKS)

Link rates 10^6bps				Test src rates (pps)		link delay	
BN		BG	Other	1 & 2	3 & 4	(ms)	
high	low					μ	σ
$110^\dagger$	$30^\ddagger$	55	110	400	200	10	2.5

† 220×10^6bps for the last bottleneck
‡ 45×10^6bps for the last bottleneck

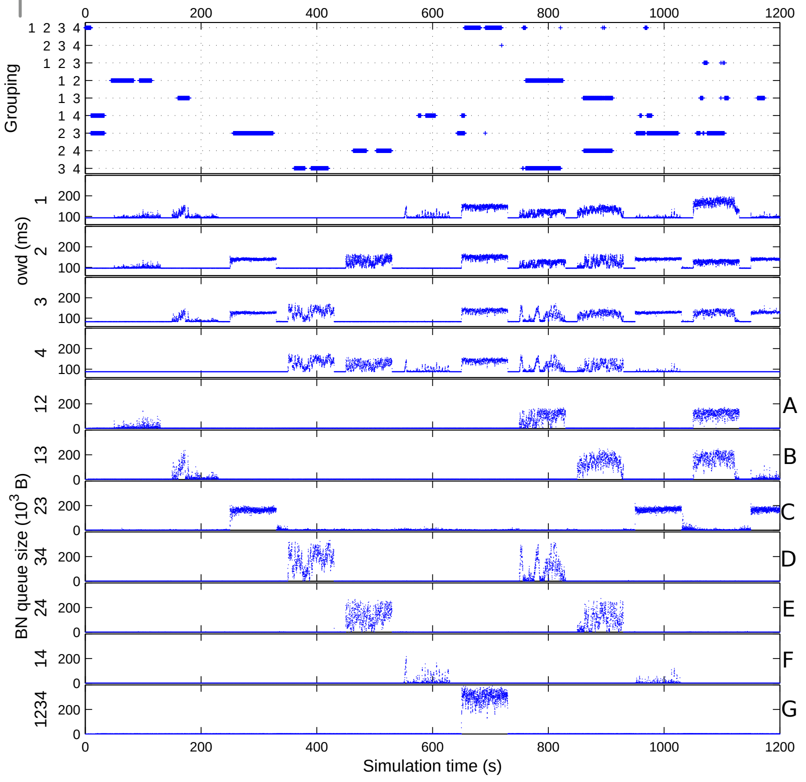


Fig. 5. Three part graph showing grouping decisions, OWD measurements, and actual bottleneck queue sizes over a 1200 s simulation run. Bottleneck queue sizes have been subsampled (1:350) and OWDs (1:20)

The majority of traffic ($> 90\%$) traversing the bottlenecks is the background traffic. It consists of thousands of TCP sessions of varying lengths generated by Tmix [23] with standard TCP using the TCP evaluation suite traces which were generated from real traffic traces. (available at http://hosting.riteproject.eu/tcpevaltmixtraces.tgz).

The simulations are run for 1200 s (20 min), with each test source sending exponentially distributed 500 B packets. Test sources 3 & 4 send packets at half the rate of test sources 1 & 2.

Fig. 5 shows the simulation results. Time is along the x-axis, with the y-axis divided into three parts: the bottom showing the queue sizes of the bottleneck links in 10^3 B, above this the OWD (ms) measured at the sink, and on top the grouping decisions made by the mechanism. The grouping part y-axis tick labels show flows grouped for the point plotted. Grouping decisions occur every 350 ms (with more than 3000 possible decisions in this run), but only flows which have a $\hat{\gamma} < 0$ are grouped. The point size is large for visibility, but this tends to magnify the significance of isolated points. The test has three main sections: 1) a single bottleneck ($0 < t \leq 740$s) 2) multiple bottlenecks – flows only share one bottleneck ($750 < t \leq 1040$s) 3) multiple bottlenecks – flows can share more than one bottleneck ($t > 1050$s).

1) single bottleneck: Bottleneck A is activated at $t =$ 50s. It is shared by links 1 & 2 and although the congestion level is low, grouping is correct. At $t =$ 150s bottleneck B is activated with flows 1 & 3 grouped correctly during the congestion episode. Similarly, bottlenecks C, D, E, and F are activated in turn. At $t =$ 650 bottleneck G carrying all 4 flows is activated. The congestion level is quite high over the activation period. Just before this, flows 2 & 3 are grouped due to a small amount of congestion on bottleneck C. However once congestion is strong on bottleneck G all 4 flows are grouped together. The queueing delay starts to drop near the middle of this congestion episode yielding $\hat{\gamma} > 0$, resulting in no grouping decision being made and a gap on the graph.

2) multiple bottlenecks – flows only share one bottleneck: This is a more interesting, but difficult test. It relies on the statistical differences between the bottlenecks. At $t =$ 750s both bottlenecks A & D are activated at once. At first this simultaneous onset of congestion for all flows causes them to

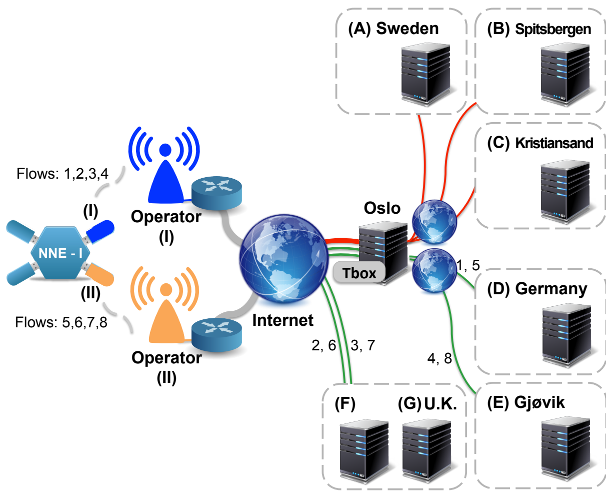


Fig. 6. NorNet set up. Our sink is the multi-homed NNE-1 node. The main bottlenecks are on the 3G wireless links, with a secondary bottleneck possible at the Oslo Tunnel Box (TBox)

be grouped together, but as the statistical differences become measurable they are grouped separately. Similarly, flows are only very briefly wrongly grouped together as first B & E, and then C & F are activated.

3) multiple bottlenecks – flows can share more than one bottleneck: This is the most difficult problem for shared bottleneck detection since a flow passing through more than one bottleneck has the characteristics of both bottlenecks. At $t =$ 950s both bottlenecks A and B are simultaneously activated. Unless the measured statistics of one bottleneck are significantly larger than the other, it is difficult for the mechanism to group flows. Reliably grouping flows that share multiple bottlenecks is beyond the capabilities of this mechanism.

B. Network tests

In order to justify the effectiveness and robustness of the proposed algorithm in operational networks, we conduct real-world experiments. The experiments are run on the NorNet testbed, consisting of wire connected NorNet Core (NNC) [5] and the wireless NorNet Edge (NNE) parts[11]. Having our test traffic traverse the Internet and mobile wireless links has the effect of making the delays on which the mechanism bases its decisions realistic.

1) Test Setup: Fig. 6 illustrates the measurement setup using both NNC and NNE nodes connected across the Internet. On the right-hand side, we show the NNC sites and their locations. The green solid lines indicate the NNC sites that generate application traffic, while the red solid lines indicate the NNC sites that generate the background traffic used to create the bottlenecks. Note that all traffic generated at each NNC site (except for (F) and (G)) will first go through the Tbox in Oslo, before going out again on the Internet towards the NNE node. NNE-1 is connected to the Internet via two different 3^{rd} Generation (3G) Universal Mobile Telecommunications System (UMTS) UM (NNE) service providers.

The *ground truth* concerning bottlenecks is not 100% known in a real operational network. However we use `traceroute` and `STAB` [15] to identify the tight common links along the paths from the NNC nodes to NNE-1 (the wireless links, and on occasion the link through the Oslo Tbox). We then generate background traffic from NNC nodes (A),(B) and (C) in order to create realistic bottleneck conditions on the 3G links.

Operators (I) and (II) have very different 3G links each with a different data rate. The queue on (I) appears to use some sort of Active Queue Management (AQM), while the queue on (II) appears to rarely drop packets.

Background and application modelling traffic are generated with UDP using `D-ITG` [1] from all NNC sites to NNE-1 via operators (I) and (II):

Background: (A) generates one exponentially distributed flow with mean rate of 92 packets per second (pps) via (I), and another 460 pps flow via (II). (B) and (C) generate more complex long range dependent traffic (Hurst=0.8). (B) and (C) each generate 8 exponentially distributed flows with Pareto distributed on times, and exponentially distributed off times: 4 by 2s/2s and 4 by 5s/5s on/off intervals at 72 pps via (I), and 8 flows with 1s/1s, 2s/2s, ..., to 8s/8s on/off intervals at 90 pps via (II). Background flows have variable packet sizes with an average 1000 B per packet.

Application: (D),(E),(F) and (G) generate the traffic we measure in order to test our mechanism (flows labelled 1,2,. . . ,8 in Fig. 6) . In a real system, this will just be the application traffic. For this test, we ensure that it comprises of a minority of the traffic traversing the bottlenecks, and that it does not have characteristics that on its own would make it easy to group. Sites (D), (E), and (F) generate an exponentially distributed flow of 100 pps with 50 B packets. (G) generates a constant rate of 100 pps with 50 B packets.

Using OWDs and packet loss observed at NNE-1, the mechanism attempts to group flows according to the current bottleneck conditions.

The experiments were run for 1200 s (20min), with the mechanism using the same parameters that were used in the simulation experiment (see Table I). This shows the robustness of the mechanism's operation as this network has very different characteristics to the simulated network (see Fig. 4).

2) Results: Fig. 7 shows the results in a similar form to those in Section VI-A with grouping decisions for flows 1,2,. . . ,8 made every 350 ms (more than 3000 possible decisions in this experiment). The bottom half of the graph shows the OWDs for each of the 8 application modelling flows versus test time. Note that flows exhibit sizeable different clock offsets, and clock drift is especially noticeable in the OWD measurements of flows 2 and 6. The OWDs show a highly fluctuating load. The top half shows the grouping decisions with the y-axis tick labels showing flows grouped for the point plotted.

There are 247 possible non-single grouping arrangements with 8 flows. The mechanism is able to correctly group flows

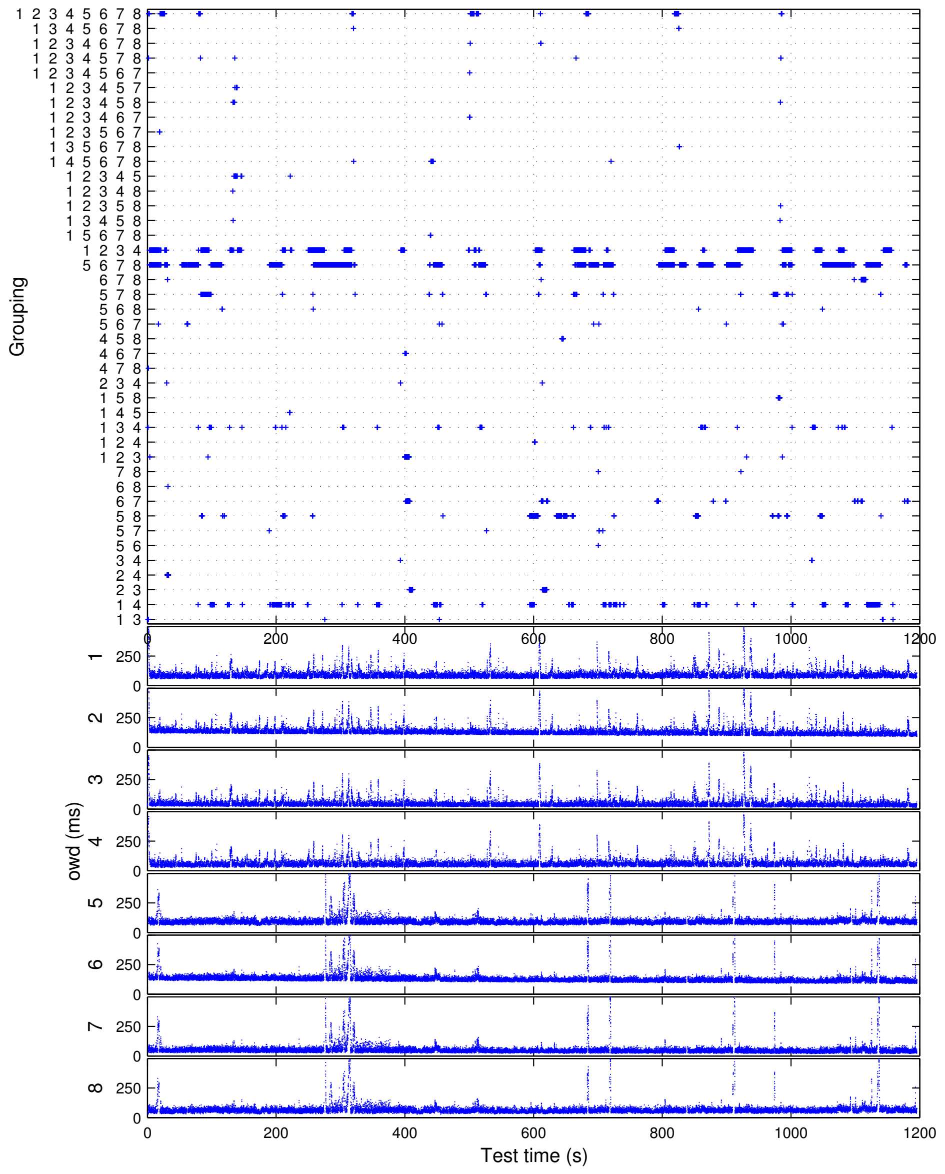


Fig. 7. Two part graph showing grouping decisions and OWD measurements for the NorNet tests. OWDs are sub-sampled (1:5)

1,2,3,4 (sharing 3G link (I)) and flows 5,6,7,8 (sharing 3G link (II)) when there is significant shared congestion, for example at times around $t = 300\,$s.

Since both wireless links are in close proximity to each other, they sometimes share common disturbances. An example of this at $t =$ 500s shows the mechanism grouping all 8 flows together.

Flows 1, 4, 5 & 8 pass through a common node in Oslo, called a Tunnel box (Tbox – see Fig. 6). This at times is also a minor bottleneck causing these flows to be sometimes grouped separately often as 1 & 4 and 5 & 8 as they also share the 3G link. As discussed in Section VI-A3, multiple bottlenecks make correct grouping very difficult.

VII. Conclusions and Future Work

We have presented a novel shared bottleneck detection mechanism based on computationally simple estimates of time and frequency domain summary statistics. It has been shown to be robust in both simulation and real network tests. The mechanism has a wide range of applications from WebRTC to possible implementation in kernel protocol stacks.

In our continuing work, we plan to investigate thoroughly the effect that changes to the different parameters have and their robustness in different scenarios. In looking at different scenarios we plan to define and generate useful quantitative statistics concerning the mechanisms efficacy. The mechanism has been designed to be practical, but this will be further tested as we implement the mechanism, and as a first example, integrate it with the coupled congestion control efforts proposed for standardization in the RMCAT Working Group of the IETF.

Acknowledgements

The authors are funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the authors.

The authors would like to thank Thomas Dreibholz for helping with the `NorNet` testbed configuration, and Özgü Alay for her advice concerning these experiments.

References

[1] S. Avallone, S. Guadagno, D. Emma, A. Pescapè, and G. Ventre, "D-ITG distributed internet traffic generator.," in *1st International Conference on Quantitative Evaluation of Systems (QEST)*, IEEE Computer Society, Jan. 31, 2005, pp. 316–317.

[2] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz, "Analyzing stability in wide-area network performance," in *Proceedings of ACM SIGMETRICS*, 1997, pp. 2–12.

[3] V. Estivill-Castro, "Why so many clustering algorithms: a position paper," *SIGKDD Explorations Newsletter*, vol. 4, no. 1, pp. 65–75, Jun. 2002.

[4] G.McCullagh and D.J.Leith, "Delay-based congestion control: sampling and correlation issues revisited," Hamilton Institute, National University of Ireland Maynooth, Tech. Rep., 2008.

[5] E. G. Gran, T. Dreibholz, and A. Kvalbein, "NorNet Core - A Multi-Homed Research Testbed," *Computer Networks*, Jan. 2014.

[6] S. Hassayoun, J. Iyengar, and D. Ros, "Dynamic window coupling for multipath congestion control," in *Proceedings of IEEE ICNP*, 2011, pp. 341–352.

[7] H. Ijaz, S. Saleem, and M. Welzl, "Fewest common hops (FCH): an improved peer selection approach for P2P applications," in *21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2013, pp. 449–453.

[8] P. Kanuparthy, C. Dovrolis, and M. Ammar, "Spectral probing, crosstalk and frequency multiplexing in internet paths," in *Proceedings of ACM SIGCOMM IMC*, 2008, pp. 291–304.

[9] D. Katabi and C. Blake, "Inferring congestion sharing and path characteristics for packet interarrival times," Massachusetts Institute of Technology, Tech. Rep. TR-828, Dec. 2001.

[10] M. S. Kim, T. Kim, Y.-J. Shin, S. S. Lam, and E. J. Powers, "A wavelet-based approach to detect shared congestion," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 763–776, Aug. 2008.

[11] A. Kvalbein *et al.*, "The NorNet Edge Platform for Mobile Broadband Measurements," *Computer Networks*, Jan. 2014.

[12] C. D. Lai, J. C. W. Rayner, and T. P. Hutchinson, "Robustness of the sample correlation - the bivariate lognormal case," *J. Appl. Math. Decis. Sci.*, vol. 3, no. 1, pp. 7–19, 1999.

[13] A. Morton and B. Claise, *Packet Delay Variation Applicability Statement*, RFC 5481 (Informational), Internet Engineering Task Force, Mar. 2009.

[14] *Network simulator (ns-2.35)*, 2011. [Online]. Available: http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[15] V. J. Ribeiro, R. H. Riedi, and R. G. Baraniuk, "Locating available bandwidth bottlenecks.," *IEEE Internet Comput.*, vol. 8, no. 5, pp. 34–41, Oct. 26, 2004.

[16] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," *IEEE/ACM Trans. Netw.*, vol. 10, no. 3, pp. 381–395, Jun. 2002.

[17] S. Savage, N. Cardwell, and T. Anderson, "The case for informed transport protocols," in *Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on*, 1999, pp. 58–63.

[18] M. Savorić, H. Karl, M. Schläger, T. Poschwatta, and A. Wolisz, "Analysis and performance evaluation of the EFCM common congestion controller for TCP connections," *Comput. Netw.*, vol. 49, no. 2, pp. 269–294, 2005.

[19] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, *Low Extra Delay Background Transport (LEDBAT)*, RFC 6817 (Experimental), Internet Engineering Task Force, Dec. 2012.

[20] J. Sztrik, *Basic queueing theory*, Lecture Notes, University of Debrecen, Faculty of Informatics, Hungary, 2012. [Online]. Available: http://irh.inf.unideb.hu/~jsztrik/education/16/SOR_Main_Angol.pdf.

[21] P. Varga, "Analyzing packet interarrival times distribution to detect network bottlenecks," in *Proceedings of IFIP EUNICE*, C. D. Kloos, A. Marìn, and D. Larrabeiti, Eds., vol. 196, Springer US, 2006, pp. 17–29.

[22] L. Wang, J. N. Griffioen, K. L. Calvert, and S. Shi, "Passive inference of path correlation," in *Proceedings of ACM NOSSDAV*, 2004, pp. 36–41.

[23] M. C. Weigle, P. Adurthi, F. Hernández-Campos, K. Jeffay, and F. D. Smith, "Tmix: a tool for generating realistic TCP application workloads in ns-2," *ACM SIGCOMM Computer Communications Review*, vol. 36, no. 3, pp. 65–76, Jul. 2006.

[24] M. Welzl, S. Islam, and S. Gjessing, *Coupled congestion control for RTP media*, Internet-draft draft-welzl-rmcat-coupled-cc-02 (work in progress), Oct. 2013.

[25] O. Younis and S. Fahmy, "FlowMate: scalable on-line flow clustering," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 288–301, Apr. 2005.

[26] M. M. Yousaf, "Accurate shared bottleneck detection and file transfer delay prediction for grid scheduling," PhD thesis, University of Innsbruck, Austria, Dec. 2008. [Online]. Available: http://heim.ifi.uio.no/michawe/research/students/murtaza_yousaf-diss_final.pdf.

[27] W. Zhu, "TCP path sharing detection," in *Proceedings IEEE Malaysia International Conferenc of Communications (MICC)*, Nov. 2013, pp. 244–249.