Trip-time-based Congestion Detection

Julien Chaumond
julien.chaumond@polytechnique.org

July 29, 2006

Abstract

To regulate network traffic and ensure that every user can get a reasonable quality of service whatever the load, networking protocols need to implement some sort of congestion control. To be able to control congestion, one must first know when it occurs and to which extent. This is the scope of congestion detection. Till now, congestion detection has only been based on packet losses. Protocols like TCP assume that a packet loss is a consequence and a signal of the network's overload. Anyway, this detection scheme has a number of flaws[1]. In this report, we discuss the possibility of detecting congestion from the packets' trip times. We first analyse the characteristics and distribution of One-way Trip Time or OTT. We then design a scheme of detection based on OTT measurements, and validate this scheme by simulation over a large range of situations.

1 Distribution of the One-way Trip Time

1.1 Generalities

The Queueing theory provides a useful framework for analyzing the distribution of trip times. Of course, results obtained from the models we use will have to be corrected by real-world experiments. However our analysis will provide some interesting insights on the characteristics and dynamics of trip times, which will be used to design algorithms.

There is obviously no deterministic expression of the One-way Trip Time in the general case. OTT can be viewed as the sum of:

- the physical transmission delay
- a sum of buffering times in routers' queues along the path
- the processing times at the routers

Of these three terms, only the first one may be deterministic. The other two depend on the nature and intensity of the considered source's traffic as well as of the cross-traffic.

What we call transmission delay is the delay of the transmission of one byte of a packet. From now on, we assume that the processing time of a packet is a deterministic function of the packet's size, namely $\alpha(|P|)$.

1.2 Multiple-hop models

The multiple-hop model is a framework for modeling network routes. A network route is modeled as a sequence of n hops. Each hop is a FIFO queue with infinite buffer representing the queue of a store-and-forward router, followed by the corresponding transmission link.

Assume that the path between the sender and the receiver contains n intermediate routers. Then:

$$OTT = \sum_{j=1}^{n+1} D_j + \alpha(|P|) + \sum_{i=1}^{n} B_i$$

where D_j is the physical transmission delay along the j^{th} link and B_i is the buffering time at the i^{th} router. We assume that the physical route does not change in the middle of a connection. Therefore, $\sum_{j=1}^{n+1} D_j$ is a constant value.

We will also assume that the dependence of the processing time on the packet's size is linear, that is to say: $\alpha(|P|) = \alpha.|P|$.

We now have to find a model for the sucessive FIFO queues. The objective is to point out some characteristics of the buffering times B_i.

[1] First, it denies the fact that losses can occur intrinsically, without congestion. With the growing number of wireless links over the Internet, this assumption has to be put into question. Moreover, the TCP-like congestion detection scheme relies on frequent acknowledgements and thus does not apply well to multimedia traffic, which does not want to support the excessive cost of reliability.

1.2.1 M/M/1 model

Assuming that each router queue is a M/M/1 FIFO queue of arrival intensity λ and service intensity μ, a classic result is that the waiting time of a given client follows an exponential law of parameter $\mu - \lambda$.

According to this first basic model[2], the trip-time equals the sum of a constant, of a variable following the packets' size's distribution, and of a sum of independent exponential variables of different parameters[3].

1.2.2 G/G/1 model

Let us release the Markovian character of arrivals and services. We can still define an arrival rate λ_i and a service rate μ_i for each hop i. With this notation, the stability condition is:

$$\forall i \in \{1, 2, \ldots, n\}, \rho_i \equiv \frac{\lambda_i}{\mu_i} < 1$$

where ρ_i denotes the load factor of hop i. For a G/G/1 queue, the probability of being idle at randomly selected instants is given by:

$$\mathbb{P}[\text{queue idle}] = 1 - \rho$$

Thus:

$$\mathbb{P}[\text{route idle}] = \prod_{i=1}^{n}(1 - \rho_i)$$

Consequently, each packet has a strictly positive probability of never having to wait in buffers. If we adopt a fixed-size packets model, OTT distribution should thus contain a discrete component of mass $\mathbb{P}[\text{route idle}]$ at the minimum value $\min \text{OTT}$. In the real-world, the distribution of the packets' size would be reflected on these low-OTT discrete components: for example, if two different packet sizes coexist in the connection, we should observe two peaks of probability near the minimum value of OTT.

1.2.3 M/D/1 model

The M/D/1 queue model is a somewhat more realistic model as the service times of the packets are obviously not exponentially distributed. In fact, we can assume that the service time is deterministic and depends only on the packet's size. If we call μ_i the i^{th} link's bandwidth and $S_i(P)$ the service time for packet P at hop i, we can even assume that:

$$S_i(P) = \frac{|P|}{\mu_i}$$

The Pollaczek-Khintchine formula, reminded in [8], gives the moment-generating-function of the number of clients in a M/GI/1 queue, from the Laplace Transform of the service time S_i. A result from this formula is the explicit expression of the mean number of packets in the queue, N:

$$\mathbb{E}(N) = \frac{\rho}{1-\rho}\left(1 - \frac{\rho}{2}\left(1 - C_S^2\right)\right)$$

where C_S^2 is the service time variation coefficient:

$$C_S^2 = \frac{Var(S)}{(\mathbb{E}S)^2} = \frac{Var|P|}{(\mathbb{E}|P|)^2}$$

We can now explicit the formula giving the mean buffering time of a packet [4].

$$\mathbb{E}(B_i) = \mathbb{E}(N_i).\mathbb{E}(S_i)$$

where $\mathbb{E}(S_i) = \frac{\mathbb{E}(|P|)}{\mu_i}$.

Thus:

$$\mathbb{E}\left(\sum_{i=1}^{n} B_i\right) = \mathbb{E}|P| . \sum_{i=1}^{n}\left(\frac{1}{\mu_i}\frac{\rho_i}{1-\rho_i}\left(1 - \frac{\rho_i}{2}\left(1 - C_S^2\right)\right)\right) \tag{1}$$

Although no explicit analytical expression for the sum of buffering times can be given, we have figured out a relation between the non-deterministic part of the OTT, and load-related characteristics of the route[5]. In section 2, we will use this relation to design a tentative scheme of congestion detection.

However, these theoretical models have to be confronted to the experience. Especially, the Poisson modeling of the arrivals at the queues does not correspond to the real-world traffic, as was shown in [7]. We have to see how that fact affects the OTT characteristics we pointed out until now.

[2] Since service times constitute a Poisson process, the dependence of the processing time on the packet sizes is irrelevant here.

[3] There is no analytical expression for this sum, unless all exponential parameters are equal, in which case the sum follows a gamma distribution.

[4] As $B = \sum_{k=1}^{N} S_k$, $u^B = \prod_{k=1}^{N} u^{S_k}$. Assuming that the packet sizes are independent and identically distributed, so are the service times, so $\mathbb{E}(u^B) = \left(\mathbb{E}(u^S)\right)^N$. The final result comes from the fact that $\mathbb{E}(B) = \left.\frac{d\mathbb{E}(u^B)}{du}\right|_{u=1}$.

[5] The set of routers' load factors ρ_i.

1.3 Experimental data

1.3.1 Distribution of packet sizes

Packet size distribution is relevant because it impacts directly on the buffering times at the routers.

To get statistically relevant data, measures have to be done on a backbone link. The data we use comes from the IP Monitoring Project (IPMON) Research Group at Sprint[6].

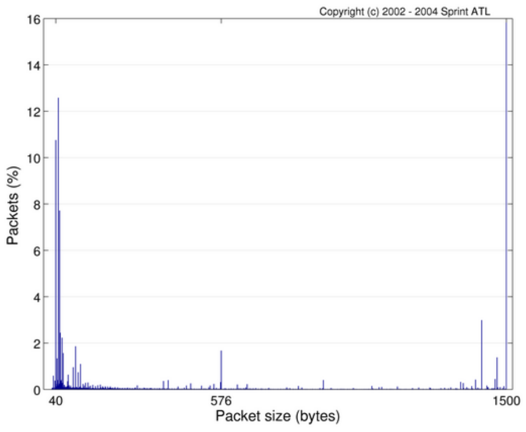

Figure 1: Typical packet size distribution

According to the data contained in Figure 1, the approximate packet size mean and variance's orders of magnitude would be:

$$\mathbb{E}|P| \propto 100B \qquad \text{and} \qquad Var|P| \propto 100{,}000B^2$$

To get a more precise – though older – estimation, we use a 18,929,141-packets trace file recorded from a link on MCI's backbone in June 1997. The result is:

$$\mathbb{E}|P| \approx 354B \qquad \text{and} \qquad Var|P| \approx 231{,}717B^2$$

As the Internet traffic has changed since 1997[7], we looked for an up-to-date estimation, using data from the National Laboratory for Applied Network Research (NLANR). The result is consistent with the former one as:

$$\mathbb{E}|P| \approx 377B \qquad \text{and} \qquad Var|P| \approx 320{,}163B^2$$

[6]Sprint Nextel is one of the biggest US backbone service provider.

[7]The pre-2000 tri-modal packet sizes distribution around 40, 576, and 1500B has shifted towards a bimodal one at 40B and 1500B.

1.3.2 Influence of the packet size on the processing time

The most reasonable model for the evaluation of the processing time $S_i(P)$ of packet P at router i is:

$$S_i(P) = \frac{|P|}{\mu_{i+1}}$$

where μ_{i+1} is the outgoing link's bandwidth.

Thus, the expression of $\alpha(|P|)$ would be:

$$\alpha(|P|) = \alpha.|P| = |P|.\left(\sum_{j=1}^{n+1} \frac{1}{\mu_j}\right)$$

i.e.

$$\alpha = \frac{1}{\mu_1} + \frac{1}{\mu_2} + \ldots + \frac{1}{\mu_{n+1}}$$

Let us confront this result to the experience, first to simulations, and then to real-world experiments. Using `ns-2`, we were able to simulate the dependence of the processing time on the packet size. The result is that the processing time is actually implemented following this model.

Then, let us have a look at some real-world experiments described in [6]. The result is that a protocol encapsulation at the routers leads to an increase in the real transmitted packet size. Moreover, a minimum frame size exists, which means that a packet whose size is smaller than a certain threshold is padded with arbitrary characters to reach its full size.

However, for the robustness of our methods, we will stick to the first simple model.

1.3.3 Experimental OTT distributions

We now have a look at some experimental OTT distributions, also found on Sprint IPMON's website[10]. A typical example can be seen on Figure 2.

These observations validate the qualitative results of our theoretical analysis: the OTT distribution is the sum of a appoximately-continuous, exponential-looking law, and of some discrete peaks indirectly reflecting the most common packet sizes.

2 A tentative to compute a global charge rate

In this part, we will try to use the analytical expression (1) obtained in the first part to design a congestion detection scheme.

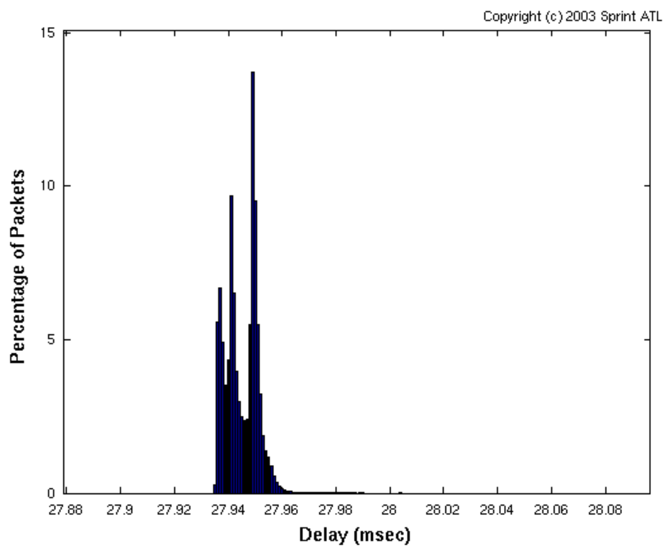

Figure 2: A typical delay distribution

The goal is to compute a quantifier for the congestion on a network path, that is to say a value x ranging from 0 to 1 and describing the utilization of the route. Provided we have such a quantifier x, implementing an efficient congestion control scheme would be really easy: the network agents would aim at a value $x_{optimal}$ of the quantifier and would iteratively change their sending rates to adjust x to $x_{optimal}$.

2.1 Estimation of the transmission delay and the processing time

Each trip-time value verifies the following equation:

$$OTT = D + \alpha . |P| + \sum_{i=1}^{n} B_i$$

From a sample of trip-time values, we want to evaluate the transmission delay D and the processing time coefficient α.

The objective is to evaluate the total buffering time B defined by:

$$B = OTT - D - \alpha . |P|$$

To evaluate α and D, we have to consider packets that are unlikely to have waited a significant time in buffers. Obviously, packets with a small OTT are more likely not to have spent a long buffering time[8].

Given a time T, we thus define this set of OTT values:

$$O(T) = \{o_1, o_2, \ldots, o_n\}$$

where $o_1, o_2, \ldots, o_n$ are the n smallest values of OTT measured from $t = 0$ to $t = T$. Hence we can assume that:

$$\begin{cases} o_1 = D + \alpha . p_1 + \varepsilon_1 \\ o_2 = D + \alpha . p_2 + \varepsilon_2 \\ \vdots \\ o_n = D + \alpha . p_n + \varepsilon_n \end{cases}$$

where $p_1, p_2, \ldots, p_n$ are the packet sizes and $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ are negligible compared to the two other terms.

A linear regression thus leads to the following α and D estimates:

$$\tilde{\alpha} = \frac{cov(o, p)}{Var(p)}$$

$$\tilde{D} = \bar{o} - \tilde{\alpha} . \bar{p}$$

In our implementation of this algorithm, we will use $n = 5$. Our simulations show these estimates converge towards the real values of α and D.

2.2 Estimation of the global charge rate

For each packet received, we now have an estimate of its total buffering time B:

$$\tilde{B}_t = OTT_t - \tilde{D} - \tilde{\alpha} . |P_t|$$

From this estimate, we can compute the mean buffering time $\hat{B}_t$ using an exponential averaging.

$$\hat{B}_{t+1} = \lambda . \hat{B}_t + (1 - \lambda) . \tilde{B}_t$$

Thus we deduce from formula (1):

$$\hat{B}_t \cong \mathbb{E}|P| . \sum_{i=1}^{n} \left(\frac{1}{\mu_i} \frac{\rho_i}{1 - \rho_i} \left(1 + \frac{\rho_i}{2} \left(C_S^2 - 1 \right) \right) \right)$$

Furthermore:

$$\tilde{\alpha} \cong \sum_{j=1}^{n+1} \frac{1}{\mu_j}$$

[8]Of course the topology of the network has to be taken into account. However for a typical network path, the mean processing time of a packet is of the same order of magnitude as the transmission delay, and the mean buffering time is an order of magnitude higher.

Let us define a function f:

$$f : \rho \mapsto \frac{\rho}{1-\rho}\left(1+\frac{\rho}{2}\left(C_S^2-1\right)\right)$$

Thus:

$$\hat{B}_t \cong \mathbb{E}|P| \cdot \sum_{i=1}^{n}\left(\frac{1}{\mu_i}.f(\rho_i)\right)$$

where ρ_i is the charge rate of the i^{th} hop. We define the global charge rate $\bar{\rho}$ as:

$$\hat{B}_t \cong \mathbb{E}|P|.\tilde{\alpha}.f(\bar{\rho})$$

$f : [0,1] \longrightarrow [0,+\infty]$ is bijective so the last definition is possible and univoque. Using the experimental values obtained in the first part, we find that $C_S^2 \approx 1.85$. The profile of function f with this value of C_S^2 can be seen on Figure 3.

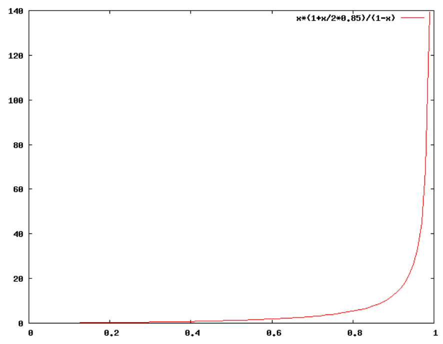

Figure 3: Profile of function f

Finally:

$$\bar{\rho}_t = f^{-1}\left(\frac{\hat{B}_t}{\mathbb{E}|P|.\tilde{\alpha}}\right)$$

The explicit expression of f^{-1} being:

$$f^{-1}(x) = \frac{-x-1+\sqrt{2(C_S^2-1)x+x^2+2x+1}}{C_S^2-1}$$

The profile of function f^{-1} for the same value of C_S^2 can be seen on Figure 4.

$\bar{\rho}_t$ ranges from 0 to 1 and describes the state of congestion on the considered network path: a value of 0 means that there is no congestion, whereas a value near 1 means the path is overloaded.

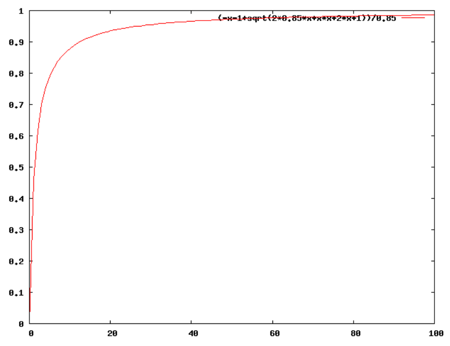

Figure 4: Profile of function f^{-1}

2.3 Pseudo-code algorithm

```
float[] OTT_set;
int[] pktsize_set;

updateLinearRegression(OTT, pktsize){
    if (OTT<max(OTT_set)){
        OTT_set[argmax(OTT_set)]:=OTT;
        pktsize_set[argmax(OTT_set)]:=pktsize;

        alpha:=cov(OTT_set,pktsize_set)
               /var(pktsize_set);
        D:=mean(OTT_set)-alpha*mean(pktsize_set);
    }
}

while(newPacket p){
    updateLinearRegression(p.OTT, p.pktsize);

    B:=p.OTT-D-alpha*p.pktsize;

    avg:=lambda*avg+(1-lambda)*B;

    rho := f_inverse(avg/(mean_pktsize*alpha));
}
```

Obviously, the estimating process of α and D takes some time at the beginning of the algorithm. Thus the accuracy of the computed $\bar{\rho}_t$ would not be too high at the beginning[9]. That means that a congestion control scheme based on this congestion detection algorithm should wait some time before actually increasing or decreasing the sending rate according to $\bar{\rho}_t$.

[9]Obviously, especially the first 5 estimations should not be considered as they constitute the initialization of the process.

3 A simple OTT-based congestion detection scheme

3.1 Limits of the theoretical approach

3.1.1 Flaws of the model

The model described in section 2 has been developed based on strong statistical assumptions. The method to which the model leads is rather simple, but in the end it relies only on the current averaged mean of OTT. This leads to the question of whether or not other statistical characteristics can be used.

However this model led to insightful considerations, especially on the estimation of transmission delay and processing time coefficient. In this section we are going to look for a simple, heuristic method which will prove easy to implement and yet efficient to detect congestion.

Relevant measurements The formula based on the model presented in the last section depends only on the exponential average of the trip time. In the scheme we are about to present, we also use the second moment of the trip time: its exponentially-averaged variance.

Variance of OTT is clearly linked to the congestion of the link: in the most extreme case, if there is no cross-traffic at all, the OTTs of the packets sent are constant, i.e. the variance is null.

Simple experiments presented in Figures 5 and 6 show that as the traffic rises, the variance of the OTT increases along with its mean.

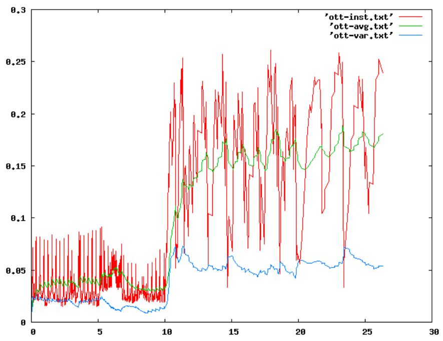

Figure 5: Influence of the traffic growth on $Var(OTT)$

Jitter vs. OTT One could think that jitter – the difference between successive packets' Inter-Arrival-Time and Inter-Departure-Time – would convey more information on the state of congestion on a network path than OTT.

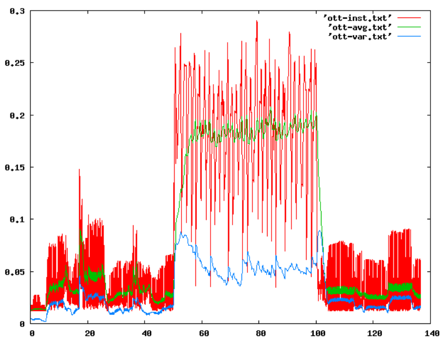

Figure 6: Influence of the traffic variation on $Var(OTT)$

For example, if the variance of the jitter is high, that should mean that trip times vary a lot because of congestion.

However we prove that in the first order, variance of the jitter J does not convey more information than variance of the OTT:

$$J = IAT - IDT$$
$$J = (R_1 - R_0) - (S_1 - S_0)$$
$$J = (R_1 - S_1) - (R_0 - S_0)$$
$$J = OTT_1 - OTT_0$$

Thus:

$$Var(J) = 2\,Var(OTT)\ (1 - \rho_{OTT_1,OTT_2})$$

where ρ_{OTT_1,OTT_2} is the correlation between OTT_1 and OTT_2.

In the first order, two succesive OTTs can be assumed to be uncorrelated[10], so the variance of the jitter is equal to two times the variance of OTT.

$$Var(J) = 2Var(OTT) + \Theta(OTT)$$

3.1.2 Related work

Getting properties of the traffic from the study of trip times is not a new idea. However computing a direct quantification of the congestion from OTTs is an original tentative.

A tentative to decide if there is congestion or not, based on the increase of the RTT is done in Wang and Crowcroft's DUAL algorithm [3], also cited in Brakmo and O'Malley's paper on TCP Vegas [4]. Congestion is tested positive if the current RTT is greater than the

[10] which is a less strong assumption than their independence.

average of the minimum and maximum RTTs seen so far. This scheme does not appear to be statistically robust as it relies on outliers of the observed trip times.

A good example of getting properties of the traffic from the distribution of OTT (and not only its extrema) is presented in [5]. Although the goal is different (differentiating between losses due to wireless link errors and losses due to congestion), this paper shows that studying the distribution of OTT can be an efficient way to get characteristics of the traffic.

Pathload [9], developed by Manish Jain and Constantinos Dovrolis, would be the most related work as it aims at estimating available bandwidth on a network path, from the distribution of OTT.

An essential difference with our present work is that Pathload, as several other bandwidth estimation tools, is based on streams of probes, whereas our scheme is based on the stream of data itself.

Also, Pathload has got a number of limitations: it is long (about 20s) and needs rather complex calculus, which makes it more a measurement tool than a real-time traffic monitor.

But more importantly, Pathload is a complex congestion-control-enabling scheme, implying two-ends cooperation. Thus it does not quantify the congestion but the available bandwidth of a network path. While it can be argued that the available bandwidth is more directly usable in a congestion control protocol, this kind of two-ends protocol goes beyond the objective of this work.

3.1.3 Two-dimensional classification of the traffic

Due to the complexity of modeling the OTT distribution, it is wise to restrict ourselves to the first two moments of it, i.e. its mean and variance. This leads to a two-dimensional classification of the traffic as in Figure 7, one variable being the mean $\mathbb{E}(OTT)$ and the other being the variance $Var(OTT)$.

Of course, the metrics used to evaluate mean and variance have to be made explicit. Given one particular profile of traffic, one cannot say if it represents a congested state from its mean and variance alone. Given one connection, we have to remind the past values of mean and variance and infer the current state of congestion from those past values. That is why any scheme of congestion detection based on the mean and variance will not be truly accurate in the first seconds of the connection, but it will have to adapt itself over a short period of time – exactly as the TCP flow control scheme.

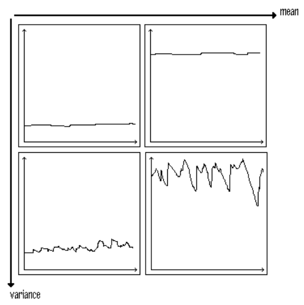

Figure 7: Two-dimensional classification

The sketch of the two-dimensional differentiation is the following:

- if the mean of OTT is low as well as its variance, compared to the past history of the connection, then most certainly the path is experiencing no congestion.
- whereas if the mean and variance are high relatively, the buffers on the path must be rather strongly-occupied, which means that the path is likely to experience congestion.
- the two other cases are more problematic. If the mean is rather low but the variance is high, it should mean that the buffers' occupancy changes a lot over short periods of time, but that congestion is not too important. If the mean is high but variance is low, it could mean that congestion causes the buffers to be full but that packets always wait for the same amount of time in them. Or it could mean that the routing has changed and the packets now follow a different, longer path.

3.2 Description of the algorithm

3.2.1 Sketch of the method

The goal of the algorithm is to give, as an output, a quantification of the congestion on the considered path of the network. The algorithm we describe here outputs a value between 0 and 1. A value of 0 means that there is no congestion at all, whereas a value of 1 indicates a strong overload of the traffic.

To get this output, we use two values describing the current relative values of the mean and variance of the *total buffering time*. That is to say, we subtract to the OTT the constant transmission delay and the estimate of the packet-size-related processing delay. Then we compute two quantifiers for congestion, the first one based on mean, the second one based on variance. These quantifiers range from 0 to 1 and we combine them to get the quantification of the congestion, as explained in the last paragraph.

To combine the two different quantifiers into a single relevant one, we have to choose a relevant function $f : [0;1] \times [0;1] \rightarrow [0;1]$. This function has to rise against both of its variables. In this report, we will use $f : (x, y) \mapsto \sqrt{x.y}$.

3.2.2 Pseudo-code algorithm

```
while(newPacket){
    updateLinearRegression(p.OTT, p.pktsize);

    B:=p.OTT-D-alpha*p.pktsize;
    avg:=lambda*avg+(1-lambda)*B;
    var:=lambda*var+(1-lambda)*(B-avg)*(B-avg);

    avg_min:=min(avg,avg_min);
    avg_max:=max(avg,avg_max);
    var_min:=min(var,var_min);
    var_max:=max(var,var_max);

    x_avg:=(avg-avg_min)/(avg_max-avg_min);
    x_var:=(var-var_min)/(var_max-var_min);

    x_congestion := sqrt(x_avg * x_var);
}
```

4 Experimental results

We ran a series of simulations to evaluate the two presented schemes. Experiments are carried out in `ns-2`, a popular Wide Area Network simulator developed by the VINT project [11].

The agent performing the quantification of the congestion is a `mpeg-udp` sink. Thus packet sizes are variable, and the packet-size-dependence of the algorithms is highlighted. `mpeg-udp` packets, whose implementation is based on [1], are timestamped, which permits computation of OTT[11].

We tested our two schemes on two different network topologies, I and II. In both topologies, cross-traffic comes from `ftp-tcp` sources turned on and off during the simulation pattern. Each link that connects a source or destination node to a network node has 10 Mbps link capacity and 1 ms of delay. Links connecting two network nodes have 1 Mbps or 400 kbps link capacity and 10 ms of delay.

[11]The possibility of a clock offset between the two end-hosts does not matter as we are only interested in the variations of OTT.

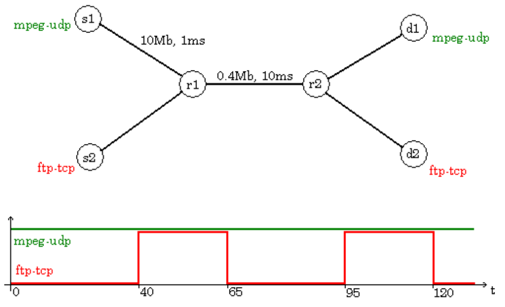

Figure 8: First simulation pattern

Congestion Control Scheme The congestion control scheme we used is derived from the one used in [1]. When MPEG encodes video, it generates a stream of frame types I, P and B, associated with a typical frame pattern called Group of Pictures (GOP), such as IBBPBBPBBPBB. Among the three frame types, only I-frames (intra-coding frames) can be decoded on their own. P-frames and B-frames are respectively, forward prediction, and bidirectional prediction frames.

The congestion control scheme is the following: the stream's destination agent computes the congestion quantifier from the packets' timestamps. According to certain thresholds, it then maps the quantifier to a scale value, an integer ranging from 0 to 4. If the quantifier is almost 1, which means that congestion is high, the scale will be set to 0. If the quantifier is low, the scale will be set to 4. This scale value is sent to the source in an acknowledgement packet, and the `mpeg-udp` source regulates its sending rate accordingly: a scale of 4 means all frames are transmitted, whereas a scale of 0 means only I-frames are transmitted. A scale of 1, 2 or 3 means that I-frames, P-frames, and a gradually growing number of B-frames, are transmitted.

We tested the efficiency of this congestion control scheme with `evalvid`, which is a tool for evaluating MPEG video transmission quality.

4.1 First topology

The first network topology tested is the simplest possible, with only one backbone link and a TCP source being turned on and off during the experiment The detailed pattern is showed on Fig. 8.

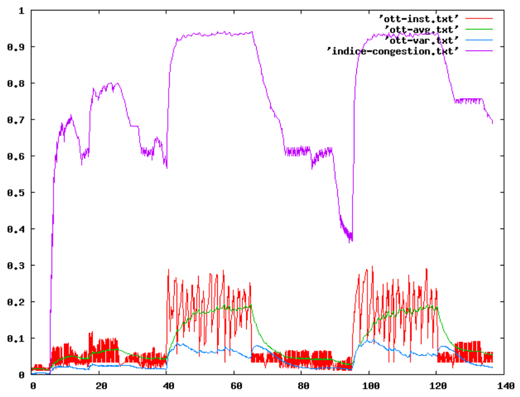

Figure 9: Topology I - Scheme 1

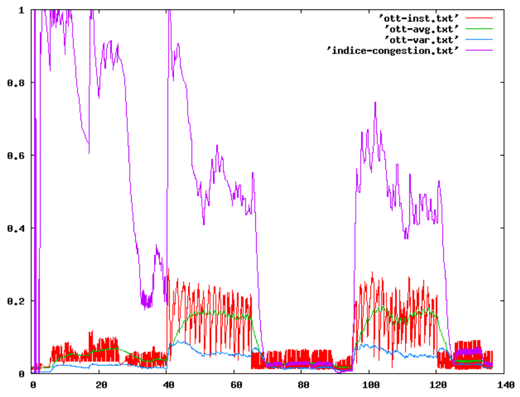

Figure 10: Topology I - Scheme 2

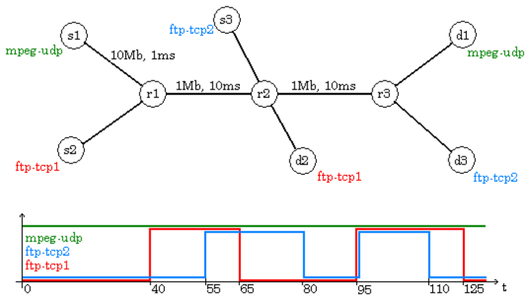

Figure 11: Second simulation pattern

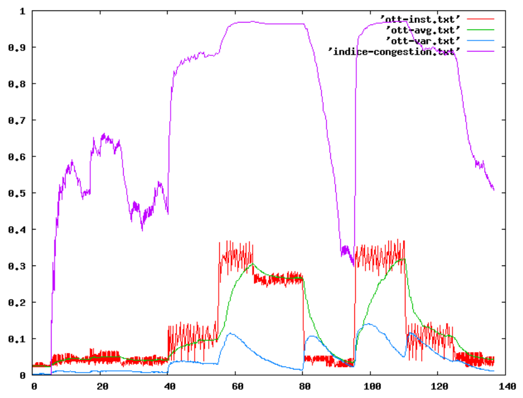

Figure 12: First scheme

Figures 9 and 10 show the result of these experiments. The congestion quantifier called `indice-congestion.txt` appears to be relevant. As of Scheme 2, the beginning of the quantification is predictably overestimated, due to the iterative nature of the algorithm.

4.2 Second topology

The second network topology is more complex, with cross-traffic occuring on different successive links of the backbone (cf. Fig. 11). This leads to a broader scale of possible traffic: the tested path can experience congestion on a particular segment or on all the links at a time. Thus our schemes should be able to differentiate between levels of congestion better than in the first case tested.

Once again, Figure 12 seems to show Scheme 1 is performing well. However Scheme 2 should be more accurate in the long term, as it incorporates data on the past mean and variance of OTT as opposed to Scheme 1 which only relies on the current mean OTT.

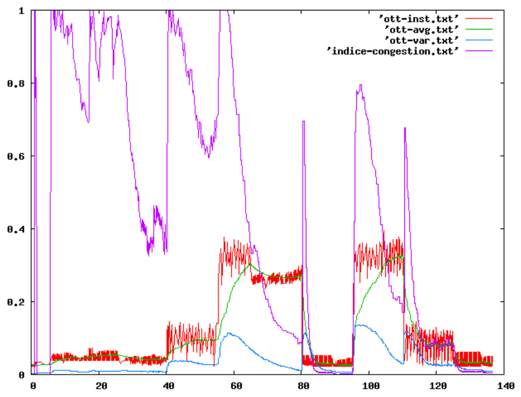

Figure 13: Second scheme - Version a

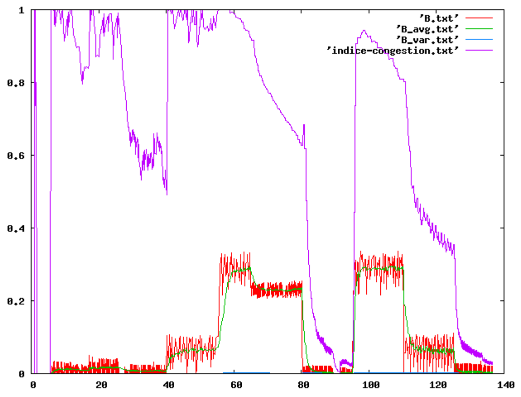

Figure 14: Second scheme - Version b

Figure 13 shows the results of the experiment for Scheme 2 with some particular parameters. The result is disappointing as the quantifier does not seem to be relevant, even after the first few seconds.

By looking in more detail at the mean-based quantifier `x_avg` and the variance-based one `x_var`, one can see that the variance-based quantifier's behavior is wrong: every time the traffic pattern changes suddenly, the typical OTT changes suddenly as well, which causes the exponential-averaged variance to be highly overestimated for a while. Thus `x_var` is almost equal to 1 just after the change, and then decreases gradually. This leads to a bias in the evaluation of the congestion quantifier. The solution we found is to use two different averaging constants for the mean and for the variance. Instead of using $\lambda = 0.95$ for both `B_avg` and `B_var`, Version b of the algorithm uses $\lambda_{avg} = 0.90$ for `B_avg` and $\lambda_{var} = 0.995$ for `B_var`. Thus variance variations are smoother, whereas mean variations follow the current OTT more closely.

Figure 14 shows the results of Version b. This time, the quantifier is relevant. One could add that in the real world, traffic conditions changes are less sudden and more continuous, which means that the algorithm should be less sensitive to the variance over-estimation bias at the transitions.

5 Influence of the Inter-Departure-Time

5.1 General Framework

The scheme proposed in the last part is legitimate, if congestion caused by the considered source itself is negligible compared to the external congestion. On the contrary, if the source's traffic is important compared to the cross-traffic, then a refinement of the method is needed to take this into account.

The source can have an influence on the global path congestion if it sends a high volume of data over a short period of time. As the current packet's size has already been taken into account, the main remaining factor is the frequency of the sendings, which can be described by the Inter-Departure-Time or IDT. Another relevant factor is the size of the preceding packets.

The point of this refinement is to compute the IDT every time a packet arrives, by subtracting timestamps with the preceding packet. If IDT is low, it means that the packets have been sent almost at the same time and that the second one has probably waited longer in

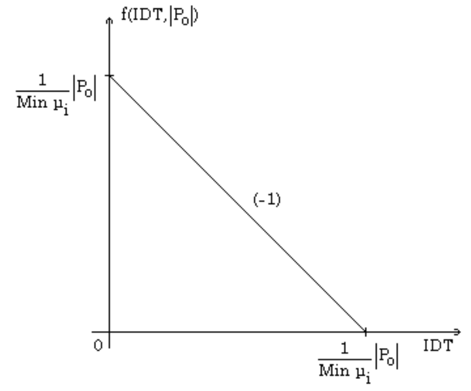

Figure 15: Assumed influence of IDT on the buffering time

queues because of the presence of the first one right before[12]. If IDT is high, the influence of the preceding packet on the second one is supposed to be negligible.

Thus, let us define a modified buffering time approximate or *unbiased buffering time*, with $|P_0|$ being the preceding packet's size:

$$\dot{B} = OTT - \tilde{D} - \tilde{\alpha}.\,|P| - f(IDT, |P_0|)$$

Function f models the influence of IDT on the buffering time. The model we use is a simple linear model: when $|P_0|$ is fixed, $f(IDT) = F$ if $IDT = 0$ and $f(IDT) = 0$ if $IDT \geq IDT_{max}$. IDT_{max} is the IDT value after which the preceding packet does not have an influence anymore.

Now let us find relevant values for F and IDT_{max}. When two packets are sent back-to-back[13], as in the packet pair methods, the Inter-Arrival-Time is determined by the first packet's size and the bottleneck bandwidth $\min \mu_i$:

$$IAT = \frac{|P_0|}{\min \mu_i}$$

Thus, we assume that $f(IDT, |P_0|) = \frac{1}{\min \mu_i}.|P_0| - IDT$. That is to say, the influence of IDT decreases linearly until the corresponding packet pair's IAT, where it equals 0, as shown in Fig. 15.

A small calculation shows that a typical back-to-back packets' IAT is far from being negligible, which gives legitimacy to this refinement. For example in Topology I, the bottleneck bandwidth was 400 kbps and the standard batch packet size was 1000 B, so $IAT \approx 20ms$.

[12]Routers are still supposed to behave like FIFO queues.

[13]i.e., with the minimum intra-pair distance, determined by the sender's bandwidth.

5.2 First method

From now on, we note $\beta = \frac{1}{\min \mu_i}$. Thus:

$$\dot{B} = OTT + IDT - \tilde{D} - \tilde{\alpha}.\,|P| - \tilde{\beta}.\,|P_0|$$

We now have to evaluate β in addition to α and D. The goal is once again to have a good approximate of the *unbiased* buffering time of a packet on a network path.

The first method we apply to evaluate the unbiased buffering time is a linear regression similar to the one used to evaluate α and D, but this time simultaneously evaluating β. That is to say, we perform a 3-dimensional linear regression.

Again, we have to consider a set of packets that are unlikely to have spent a significant time in buffers. For these packets $\dot{B} \approx 0$ so:

$$\begin{cases} o_1 + i_1 = D + \alpha.p_1 + \beta.p_1^0 + \varepsilon_1 \\ o_2 + i_2 = D + \alpha.p_2 + \beta.p_2^0 + \varepsilon_2 \\ \vdots \\ o_n + i_n = D + \alpha.p_n + \beta.p_n^0 + \varepsilon_n \end{cases}$$

where $p_1, p_2, \ldots, p_n$ are the current packets' sizes, $p_1^0, p_2^0, \ldots, p_n^0$ are the preceding packets' sizes, and $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ are negligible compared to the other terms.

However the design of this scheme raises some questions. In particular, which samples should be used to perform the linear regression? We have to discriminate between packets' likelihood of having spent a short time in buffers, based not only on their OTT, but also on their IDT.

Obviously, packets with a small OTT are more likely not to have spent a long buffering time. But between two packets whose OTT, size and preceding packet's size are equal, we know that the one with the smallest IDT has spent less unbiased buffering time, because of the influence of its preceding packet. Thus we have to choose samples of low OTT and IDT. An example of a simple way to choose such samples is to order them by increasing value of $OTT + IDT$ and select the first ones.

5.3 Second method

The second idea is to evaluate firstly α and D using the same scheme as before[14], and then to evaluate β when $\tilde{\alpha}$ and $\tilde{D}$ are supposed to be relevant enough. Again, the question of which samples to use is raised.

[14]a two-dimensional linear regression, without taking any IDT into account.

We choose to select once again the packets of lowest $(OTT + IDT)$. For these packets $\dot{B} \approx 0$ so $\tilde{\beta}$ is given directly by:

$$\tilde{\beta} \approx \frac{OTT + IDT - \tilde{D} - \tilde{\alpha}.\,|P|}{|P_0|}$$

By averaging the values obtained, we can get a good approximate value of β. Thus this second method is easier to implement and simpler in terms of complexity, and yet efficient.

5.4 Extension to multi-packet batches

So far, we only took into account the influence of *one* preceding packet. However the generalization to several packets is easy: assuming that we keep track of a small number $n + 1$ of the preceding packets, we can, for each of them, subtract $f(IDT, |P_0|) = \tilde{\beta}.|P_0| - IDT$ from $\dot{B}$ if they have an influence on $\dot{B}$ according to the scheme, i.e. if $IDT \geq 0$ and $\frac{IDT}{|P_0|} \leq \tilde{\beta}$.

Thus, if we call $P_0, P_{-1}, \dots, P_{-n}$ the $n + 1$ preceding packets:

$$\dot{B} = OTT - \tilde{D} - \tilde{\alpha}.|P| \\ - \sum_{m=0}^{n} \left(\tilde{\beta}.|P_{-m}| - IDT\right).\mathbb{1}_{\{IDT \geq 0\} \cap \{\frac{IDT}{|P_{-m}|} \leq \tilde{\beta}\}}$$

6 Conclusion

We presented a simple stochastic model for the one-way trip time. This model is based on the differentiation of the packet's trip's parts and provides us with interesting insights on the transmission of data over a network.

Two estimation methods for a quantifier of the congestion on a particular path were then designed. The initial experiments we performed using simulation tools gave encouraging results. In particular the second scheme appeared to be robust and relevant. However further work is needed to test and improve the accuracy of the method in real-world conditions.

If the improved method proves to be relevant, several ways of using it are conceivable. It could be used as a measurement tool, i.e. as a punctually-used way to evaluate the state of the buffers on a route, or it could be part of a real-time detection scheme actually implemented in a networking protocol.

Acknowledgements

I would like to gratefully thank Karthikeyan Balaji Dhanapal, Sakib Mondal and Sungsoon Cho from Infosys' SETLabs for helpful discussions and support on this work.

References

[1] Jae Chung, Mark Claypool: *Better-behaved, better-performing Multimedia Networking*, SCS Euromedia, May 2000

[2] Eddie Kohler, Mark Handley, Sally Floyd: *Designing DCCP: Congestion Control Without Reliability*, ICSI Center for Internet Research

[3] Zheng Wang and Jon Crowcroft: *Eliminating periodic Packet Losses in the 4.3-Tahoe BSD TCP Congestion Control Algorithm*, ACM Computer Communications Review, April 1992

[4] Lawrence Brakmo, Sean O'Malley, Larry Peterson: *TCP Vegas: New Techniques for Congestion Detection and Avoidance*, University of Arizona, February 1994

[5] Jun Liu, Ibrahim Matta, Mark Crovella: *End-to-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment*, Boston University

[6] Vivien Tran-Thien: *Bandwidth estimation through active probing*, Ecole Polytechnique, July 2002

[7] Vern Paxson, Sally Floyd: *Wide-Area Traffic: The Failure of Poisson Modeling*, IEEE/ACM Transactions on Networking, June 1995

[8] Jean-Marc Vincent: *Memo on M/GI/1 queues*, Université Joseph Fourier, Grenoble

[9] Manish Jain, Constantinos Dovrolis: *Pathload: a measurement tool for end-to-end available bandwidth*, University of Delaware, August 2002

[10] Sprint IPMON Group's website: `http://ipmon.sprint.com`

[11] The Network Simulator - `ns-2`: `http://www.isi.edu/nsnam/ns/`