Round-Trip Time Anomaly Detection

Daniel Brahneborg
Infoflex Connect AB
Stockholm, Sweden
firstname.lastname@infoflexconnect.se

Wasif Afzal, Adnan Čaušević,
Daniel Sundmark, Mats Björkman
Mälardalen University
Västerås, Sweden
firstname.lastname@mdh.se

ABSTRACT

Mobile text messages (SMS) are sometimes used for authentication, which requires short and reliable delivery times. The observed round-trip times when sending an SMS message provide valuable information on the quality of the connection.

In this industry paper, we propose a method for detecting round-trip time anomalies, where the exact distribution is unknown, the variance is several orders of magnitude, and there are lots of shorter spikes that should be ignored. In particular, we show that using an adaption of Double Seasonal Exponential Smoothing to reduce the content dependent variations, followed by the Remedian to find short-term and long-term medians, successfully identifies larger groups of outliers. As training data for our method we use log files from a live SMS gateway. In order to verify the effectiveness of our approach, we utilize simulated data. Our contributions are a description on how to isolate content dependent variations, and the sequence of steps to find significant anomalies in big data.

KEYWORDS

log file analysis; round-trip time; exponential smoothing

ACM Reference Format:
Daniel Brahneborg and Wasif Afzal, Adnan Čaušević, Daniel Sundmark, Mats Björkman. 2018. Round-Trip Time Anomaly Detection. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering, April 9–13, 2018, Berlin, Germany*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3184407.3184436

1 INTRODUCTION

Measuring and monitoring round-trip times (RTTs) of data packets in a networked environment is important for at least two reasons: (1) to maintain the negotiated service levels of quality and (2) to minimize operational costs. To better understand the importance of this monitoring, let us consider a scenario where a person wants to login to an Internet bank.

(1) The customer goes to the bank website and enters a personal, unique identification number.
(2) The bank finds this information in its customer database, and sends a verification code as an SMS message to the registered mobile phone for this customer. In most cases, the SMS arrives to the mobile phone well within the negotiated, stipulated time, but in some anomaly situations, the message could be delayed for a considerable amount of time.
(3) The customer enters the verification code, completing the login procedure.

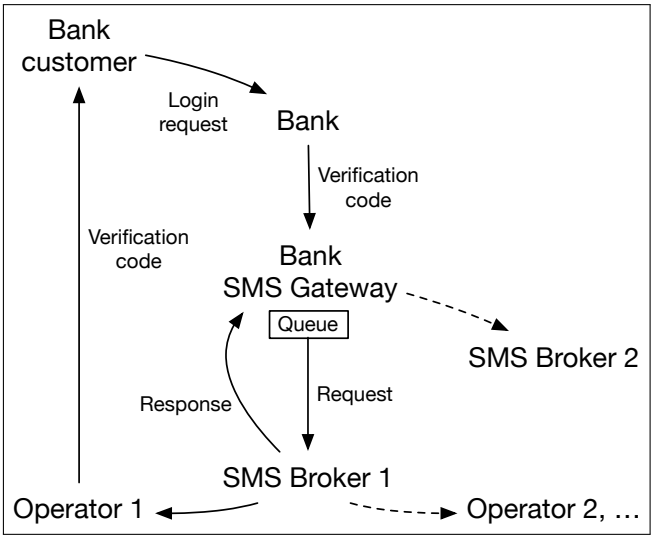


Figure 1: An example scenario emphasizing the importance of measuring and monitoring round-trip times: the bank customer, the bank, two SMS brokers and some operators.

There are many mobile network operators, and as the bank does not want to maintain connections and agreements with all of them, this service is outsourced to one or more "SMS brokers". SMS brokers (SBs) manage the SMS traffic between their customers and the network operators. For increased reliability, the bank connects to two SMS brokers (Figure 1). Assuming the cost for sending messages via SB1 is lower than via SB2, by default the bank sends all messages via SB1. If the connection to SB1 is lost for some reason, the bank quickly switches to SB2 in order to avoid delays in the SMS deliveries. In the worst case it could take hours for the connection to SB1 to be fully functional again, so without this switch, the problem with SB1 would result in dissatisfied customers for the bank as the verification codes would stay in the outgoing queue in the bank's SMS Gateway. The sooner the switch to SB2 can be made, the smaller the delay seen by the bank's customers. Using SB2 all the time would not improve the situation, as SB2 could also become unreachable for any number of reasons, e.g. broken hardware, or problems at their Internet service provider.

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5095-2/18/04…$15.00
https://doi.org/10.1145/3184407.3184436

The bank customer is the only one who knows the exact delivery time, and that is just for their own message. In order to get an overall view, from this point on we will use a simpler measurement: the RTT between the bank and the SMS broker.

In this paper we focus on detecting violations that fall somewhere between a few individual messages being slightly delayed and a fully broken connection. Let us assume the operator normally has two servers handling SMS traffic, and one of them temporarily breaks. With incoming throughput to the SMS broker being constant and outgoing throughput being halved, a queue of messages may form. To prevent this queue from growing without bounds, the SMS broker can throttle incoming traffic by delaying its responses. Clients must implement proper windowing, so these delays will cause them to delay their future requests.

The SMS system behaves much like a train of cars, in that we can draw conclusions on the situation further ahead by observing the car in front of us. If the car slows down, we can assume there is a problem with the traffic in general. Provided the slow speed persists, we might decide to choose an alternate route. Similarly, the RTT towards the SMS broker provides the client (the bank) with valuable feedback on the effective throughput of the entire chain of SMS brokers and operators.

This paper addresses the situation when the absolute values of the delivery time are not known. We know from earlier results [4] that the RTT has very few anomalies, but when they happen, we want to know as soon as possible. We have seen that there are several shorter spikes in these RTTs, so our research objective is to develop a method of automatically finding longer periods of outliers in RTTs while ignoring these short uninteresting spikes. In particular, we examine the variation of the RTTs in a production system of an SMS broker between their own system and several external operators.

Section 2 describes the context in more detail and Section 3 describes related work for RTT measurement and anomaly detection. Our approach is described in Section 4. We then describe our case study in Section 5, and the results in Section 6. Section 7 discusses these results, and the paper ends with conclusions and future work in Section 8.

2 BACKGROUND AND TERMINOLOGY

Figure 2 shows the simple base scenario of the network traffic as seen from the SMS Gateway software used by the SMS brokers. The filled arrows represent SMS messages, the unfilled arrows are responses, and A, B etc. are points in time. The SMS Gateway only knows about the times B, C, E and H. The arrow from J to K is dashed, as we do not know when this event occurs. The difference between B and C shows the processing time required for an incoming message, while the difference between E and H shows the full RTT to the operator. We will examine both these differences, as anomalies between B and C reveal problems in the local environment and anomalies between E and H reveal problems in the network or with the remote node. The difference between C and E is how long the message sits in the outgoing queue, waiting to be sent. From the bank customer's point of view, the login request starts at some point before A, and the verification code arrives to the phone at K. The delivery of the message to the mobile phone and the response sent back to the SMS Gateway happens in parallel, so the relationship between H and K is undefined.

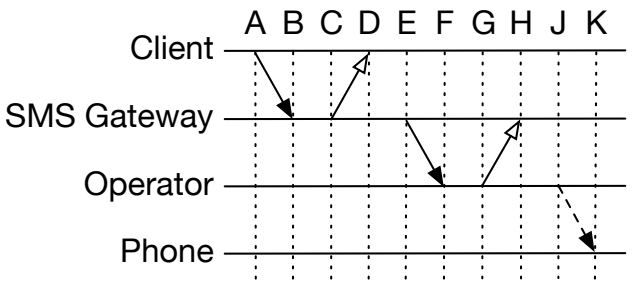


Figure 2: The network traffic between a client, an SMS Gateway, an operator and a mobile phone.

In many cases, monitoring of response times is required as a way of making sure the system works as expected. According to our industrial experience, one of two methods are commonly used for this monitoring. 1) Visualize selected measurements on a display, which is simple to implement but requires a human to look at the display. This is easily forgotten if anomalies are rare. 2) Utilize tools based on Simple Network Management Protocol (SNMP), reporting detected anomalies without requiring human interaction. A drawback is that those checks are usually trivial with static tolerance levels, e.g. whether a single RTT is longer than 1 second or whether the processing queue contains more than 1000 elements.

Once an anomaly has been detected, some fault localization technique [22] should be applied to find the root cause of the problem. This is however outside the scope of this paper.

While SMS brokers reduce the number of accounts needed, the combined network traffic becomes more difficult to analyze. Some brokers specialize in operators in a particular region, decreasing the number of accounts but increasing the number of intermediate nodes. Broker handling of messages varies, e.g. they may store the messages on disk for safety, or wait for acknowledgment from the next node before responding back to the previous. These factors incur variability in response times, even between the same nodes. We assume that if a node uses a server cluster, all these servers are homogeneous, giving consistent RTTs.

2.1 Terminology

We will now define the concepts discussed in this paper:

Node: Common term for clients, operators and SMS brokers.

Downstream/Upstream: Downstream is as ordered in Figure 2, i.e. client to SMS broker to operator to mobile phone. Upstream is, obviously, the reverse direction.

Request: A data packet containing an SMS message, including the sender, recipient and message body, or a delivery report.

Response: Acknowledgement of a received request.

PDU: Protocol Data Unit, refers to both requests and responses.

Delivery report: A data packet sent as confirmation of successful message delivery to the recipient or rejection by a node.

Round-trip time (RTT): For outgoing traffic, RTT is the interval between when the request is sent and the response comes back. For incoming traffic, RTT is the interval after receiving

a request until the response is sent. In Figure 2, these are the intervals from E to H and from B to C respectively.

Throughput: The number of messages received and forwarded by a node, per some specified time unit.

Window size: The number of requests the client sends before waiting for a response.

Outlier: A single RTT measurement significantly higher than usual for a specific connection. Responses arriving earlier than usual is both very rare and typically not a problem.

Anomaly: A larger cluster of outliers. This is defined in more detail in Section 4.4.

3 RELATED WORK

Earlier studies have focused on either RTT measurement or anomaly detection, so we will describe these groups of papers separately.

3.1 RTT measurement

For RTT measurement, existing work can be structured according to what protocol they analyze. A relatively common layer for RTT measurements is TCP, as it is used for many applications and therefore enables analysis of large amounts of data. Here we find an examination of several different TCP implementations [18], and a description of the experiences using the tool Tstat [16].

TCP includes an ACK packet which, similarly to our response PDUs, provides an easy way to calculate the RTT. The RTT can then be either approximated using just the SYN/ACK pair used to initiate the connection [12] or more correctly using also the data packets and their responses [26]. Martin et al. [15] took this further by using the minimum and average values of the RTT for both the SYN and data packets to separate the physical latency from the server side processing time. The packet-pair strategy was then generalized for raw IP traffic [27].

At the application layer, which is most similar to our work, we have studies on HTTP traffic by Mosberger and Jin [17] using their tool `httperf`, and Halepovic et al. [9] who examined the RTTs from mobile clients to web servers. The throughput values given by `httperf` had an average close to the maximum, which corresponds to an average RTT being close to the minimum.

In some cases, the minimum and maximum RTT values are the most interesting [8], in which case there is no need to examine the distribution in more detail. Papers that have analyzed the data deeper, have found variances in RTT for TCP traffic between 1 millisecond and 200 seconds [2, 11]. In an analysis for Controller Area Networks, the type of network used in real-time environments, the data had a good fit with the Gamma distribution [28]. Taken together, most papers that have examined the distribution of RTT values, explicitly or implicitly describe it as exponential in some way. This is consistent with our findings.

3.2 Anomaly detection

Shanbhag and Wolf suggest using multiple anomaly detection algorithms in parallel [21], and using the combined result as the trigger. Even though we do not use multiple algorithms, we use all relevant data fields in the PDU to calculate the expected values with as much precision as possible.

E2EProf, as described by Agarwala et al. [1], is similar to our approach as it also uses time-series analysis, of which exponential smoothing is one of the methods, to analyze the performance of each subsystem of an application. They define a "spike" as a local maximum, exceeding a threshold of the mean plus three times the standard deviation. For testing, they used `httperf`.

Bayesian Principal Anomaly Detection (BPAD) warns for individual outliers [10], and because these occur too frequently, it does not suit our context.

Between the years 2000 and 2010, there were several papers [13, 14, 19] on using Principal Component Analysis (PCA) for anomaly detection. Even though the method worked fine, it was difficult to find the right sensitivity [5].

Wang et al. [25] stress that anomaly detection methods must in some circumstances be "lightweight", both in terms of the number of metrics they require to run (the volume of data continuously captured and used), and in terms of their runtime complexity. They suggest smoothing the data, just as we use exponential smoothing (Section 4.2), and detect anomalies using the Tukey method based on "fences" and "hinges" [24]. This method splits the data into quartiles separated at Q1, Q2 and Q3, and classifies anomalies in multiples of the difference between Q1 and Q3. While different from our method, it also uses the median instead of the mean.

4 APPROACH

In order to understand the RTT values, we first calculated the mean and standard deviation of a few collections of RTTs (Section 4.1). We then used exponential smoothing to get a mean value that gave higher importance to newer RTTs (Section 4.2). Some parts of the variance turned out to be related to specific aspects of the message data, so the exponential smoothing was further refined to isolate these as adjustment factors (Section 4.3). Finally we calculated the median of smaller and larger groups of RTTs as a way of identifying outlier clusters (Section 4.4).

4.1 Mean and standard deviation

As mentioned in Section 2.1, the time spent by a node processing a request can vary significantly, so the RTT varies from fractions of a millisecond to multiple seconds. Calculating the mean from such data does not give meaningful results.

The exact distribution of the RTTs is not known to us. However, earlier work shows that it resembles a log-normal distribution, so we calculate the mean and variance of the logarithms of the RTTs.

For efficiency, we use formulas based on those described by Finch [7]. The formula used for the incrementally calculated mean is shown in Equation 1. Here, x_n is the new value, and n is the number of values so far. We use Equation 2 to get the variance S_n, and Equation 3 for the standard deviation σ_n.

$$\mu_n = \mu_{n-1} + \frac{1}{n}(\ln x_n - \mu_{n-1}) \tag{1}$$

$$S_n = S_{n-1} + (\ln x_n - \mu_{n-1})(\ln x_n - \mu_n) \tag{2}$$

$$\sigma_n = \sqrt{S_n/n} \tag{3}$$

4.2 Exponential smoothing

Over time, the effect of new values added to Equation 1 shown in Section 4.1 will diminish. By instead using exponential smoothing, we are able to analyze an endless series of data.

We calculate the expected value E_n using the well-known Equation 4, where n is the number of observations, and V_n is the nth value. Or rather, V_n is the logarithm of the measured RTT, and E_n is the logarithm of the expected value. The new value is the sum of two terms based on the current observation and on the previously expected value, respectively. The constant α is used to select the scaling factor between them, where a lower value of α gives a more stable E_n, as the effect from individual values of V_n is smaller. We set E_1 to V_1.

$$E_n = \alpha V_n + (1 - \alpha)E_{n-1},\ n > 1 \tag{4}$$

4.3 Adjustment factors

As the traffic between SMS brokers uses Internet, network related RTTs can vary both by time of day and day of week. While grouping the data by hour gives a lower variance and therefore improved anomaly detection, it also gives less data in each group, resulting in reduced stability. Moreover, it disregards the similarities of RTTs during consecutive hours.

Communication protocols for SMS consist of fields with key-value pairs which specify how the SMS should be handled, so we assume their values might affect the RTT. To minimize the variance, each unique combination of fields should be analyzed separately. This strategy leads to a combinatorial explosion, and requires large amounts of data for satisfactory stability of E_n. In the financial domain Double Seasonal Exponential Smoothing [23] is sometimes used, basing the result on time values, e.g. day of month and month of year. The idea is to get a single average value for the entire dataset, with a small number of adjustment factors. Similar approaches have also been used in network contexts [6]. We use a variation of this method, but with field values instead of time values.

We need one adjustment factor per field value, and use the syntax F_n^v for the nth value of the adjustment factor for field value v. The value of F_0^v is set to 0, representing the case when the RTT is identical for all values. The adjustment factor can be either additive or multiplicative, and because of the exponential nature of the RTT distribution, multiplicative adjustments seem to make the most sense. However, as the values of E_n and V_n are logarithms, the actual adjustment needs to use addition. The calculation of the effect from a specific field value is shown in Equation 5. We want the expected value E_n to be free from these variations, so Equation 4 is modified to instead use the adjusted value of V_n, as shown in Equation 6.

$$F_n^v = \alpha(V_n - E_n) + (1 - \alpha)F_{n-1}^v \tag{5}$$

$$E_n = \alpha(V_n - F_{n-1}^v) + (1 - \alpha)E_{n-1} \tag{6}$$

For the more general case, we see the difference between the expected value E_n and the measured value V_n as the sum of all adjustment factors for all fields. We can then update the adjustment factors using the same exponential smoothing as in Equation 4. This is shown in detail in Algorithm 1, lines 11 to 19. For simplicity we use the same scaling factor α as for the expected value in Equation 4, but it is possible to use different scaling factors for each adjustment factor.

4.4 Medians

Even with α as low as 0.0001, the wide range of values in the input data renders E_n too unstable to be useful in detecting anomalies. A more reliable reference point is given by the median, in our case calculated using the Remedian [20] method. The algorithm is simple but effective, using k arrays A_i, each of length b.

(1) Store the first b values in A_0, where typically $b < 10$.
(2) Calculate the median of A_0 and append the result to A_1.
(3) Repeat steps 1 and 2 until A_1 contains b values. Calculate the median of them, and append the result to A_2.
(4) Repeat the previous steps up to A_k for all i less than some k, appending the median of A_{i-1} to A_i.

The median of A_k is now an estimate of the median of the full series of values. The number of operations required to find the median of b values is fixed for each b, giving an execution time complexity of $O(n)$ for n values. We can think of it as a software version of multiple connected Geneva drives [3].

The value we append to A_0 is E_n, the most recent measurement with all adjustment factors removed. Using arrays with $b = 5$ values each achieves a good balance between stability, which requires more values in each array, and sensitivity, which requires fewer values. This way A_0 has the median of the most recent 5 values, A_1 of 25 values, A_2 of 125, etc.

We can now define an anomaly as a cluster of outlier measurements that increase the median of A_3 above twice the median of A_5. To avoid repetitive notifications, each notification suspends further ones until the median of A_3 goes below the median of A_5. A period of large values that is long enough to affect the median of A_3 this way occurs sufficiently seldom, as shown in Section 6.3.

4.5 Summary

Algorithm 1 combines the steps described earlier in this section. The algorithm is implemented as an extension to our existing tool called ELFA (EMG Log File Analyser - initially introduced in [4]). The method described here has several benefits.

(1) All calculations are done in constant time, depending only on the number of adjustment factors. This is necessary as we need to be able to handle continuous traffic with up to 1000 measurements per second.
(2) The sensitivity is easily adjusted, even online.
(3) It is independent of the frequency of values.
(4) The expected value is calculated from all observations, not just from an artificial subset.
(5) Adjustment factors can be added and removed online as needed.
(6) For each connection we need to persist only the base value E_n and the non-zero adjustment factors F_n^v to be able to resume a paused analysis.
(7) It is self-adapting, using the most recent RTT values for each individual connection as the basis for detecting outliers.

ALGORITHM 1: Find Outlier Clusters

input : A list of data points, each one consisting of a list of key-value pairs and a measured value V_n.
output : A list of start and end points for anomalies.

```
// Initialization.
forall A do A ⟵ ∅                          // Clear all Remedian arrays.
haveReported ⟵ false
outliers ⟵ ∅
expected ⟵ 0
foreach possible key do
    foreach value used by key do
        adjustments[key,value] ⟵ 0
    end
end
foreach data point dp do
    // Update the expected value from measurement(dp), minus
       the current adjustment factors.
    b ⟵ 0
    foreach (key,value) in dp do
        b ⟵ b + adjustments[key,value]
    end
    // Normal exponential smoothing.
    expected ⟵ α ∗ (measurement(dp) − b) + (1 − α) ∗ expected
    // Update the adjustment factors.
    foreach (key,value) in dp do
        // Assume all other adjustment factors are correct,
           and calculate what is left.
        diff ⟵
          measurement(dp) − (expected + b − adjustments[key,value])
        // Update the adjustment factor for this key-value
           pair.
        adjustments[key,value] ⟵
          α ∗ diff + (1 − α) ∗ adjustments[key,value]
    end
    // Update the Remedian arrays.
    i ⟵ 0
    Append expected to A[0]
    while A[i] is full and i + 1 < 6          // We have 6 arrays
     do
        Append median(A[i]) to A[i + 1]
        A[i] ⟵ ∅
        i ⟵ i + 1
    end
    // Find start and end points for anomalies.
    if not haveReported and A[5] has been filled at least once and
      median(A[3]) > 2 ∗ median(A[5]) then
        Add ('start', timestamp(dp)) to outliers
        haveReported ⟵ true
    end
    if haveReported and median(A[3]) < median(A[5]) then
        Add ('end', timestamp(dp)) to outliers
        haveReported ⟵ false
    end
end
return outliers
```

5 CASE STUDY DESIGN

To evaluate our approach for detecting anomalies in the RTTs, we undertook an industrial case study. Specifically, we wanted to investigate and exemplify how log files generated by the production system of an SMS broker can be utilized to identify anomalies in RTTs between itself and several external operators.

5.1 Data collection

We examined data from the Enterprise Messaging Gateway (EMG), an Infoflex Connect AB product used by many SMS brokers. The data was taken from existing log files as they contained the data we needed without requiring modifications to the core product with a risk of introducing bugs. The amount of data per operator varied between 33 and 497 MB.

In this study we selected one of the most commonly used protocols for SMS traffic, SMPP (Short Message Peer to Peer). Each PDU starts with a header, comprised of the operation number, a transaction number, a status and the length of the data section. Following the header is the data section, consisting of a sequence of key-value pairs, where the keys and their order depend on the operation. As responses can arrive in an undefined order, the transaction number from the request must be exactly duplicated in the response.

The EMG log files contain information on whether each PDU was sent or received, the timestamp, which connection was used, the operation name, the transaction number, and all key-value fields from the data section.

6 CASE STUDY RESULTS

This section presents the results of the industrial case study. In particular, we first discuss how different characteristics (keys) of the data and messages sent affect the RTT (Section 6.1). Second, we discuss how using certain adjustment factors enabled higher accuracy in the outlier detection (Section 6.2). Third, we detail the results of applying the anomaly detection algorithm to a large dataset of network traffic (Section 6.3).

In the presented results, data is analyzed for three different operators, referred to as "O1", "O2" and "O3".

6.1 RTT for selected keys

To explore how individual keys affected the RTTs, we counted the number of unique values used by each key. This revealed three distinct categories.

Message specific: 11 keys, e.g. destination numbers and message bodies. We assume these values are unique for each message.

Groups: 11 keys, e.g. whether a delivery report is requested, the character encoding, and similar keys with a very limited set of values. We identified "data coding", "esm class" and "registered delivery" as having the largest effect on the RTTs.

Constants: 4 keys that are either not supported by EMG, or ignored by most recipients, and therefore always sent with the same value.

Next we describe the key "data coding" in more detail, and how its value affects the RTT. All RTT values in this section are shown with their mean and one standard deviation up and down, to give an indication of their relative positions and spread.

Table 1 shows the RTT grouped by data coding. The values in the first column have the following meaning.

0 Text message, using the GSM character encoding IA5.
8 Text message, using the character encoding UCS-2.
240 Special messages, e.g. configuration settings.
245 8 bit data, e.g. ring tones.

With the exception of the values 240 and 245 to operator "O1", the RTT distributions for different values are clearly separated. The operators seem to perform some time consuming processing of UCS-2 texts, as those RTTs are significantly longer than for IA5 texts. "N/A" means the value was not used with that operator.

Value	O1	O2	O3
0	9.3/9.7/10	8.1/8.2/8.4	440/466/493
8	23/25/28	21/24/27	651/673/697
240	3.6/3.7/3.7	3.5/3.5/3.5	N/A
245	3.2/4.9/7.7	N/A	329/358/390

Table 1: RTT in milliseconds, grouped by data coding. The three values are $\mu - \sigma$, μ and $\mu + \sigma$, respectively.

The RTTs when grouped by the "esm class" and "registered delivery" keys showed similar patterns, with a ratio of up to 3 for some values. This motivates us to show the results with a deeper analysis using the adjustment factors.

6.2 Adjustment factors

The adjustment factors for the message key values were mostly consistent with the results in Section 6.1. The "data coding" adjustment factors are shown in Table 2. As the values represent the difference in exponent, a value of 1 corresponds to a ratio between the RTTs equal to e.

For O1, whether data coding is 0 or 8 gives an RTT that varies by a factor of $e^{0.92-(-0.46)} \approx 3.97$. UCS-2 data requires twice as much space as IA5, but even if we adjust for this, there is still a remaining factor of $3.97/2 \approx 1.99$. We see a similar pattern for O2, with adjustment factors -0.13 and 1.30 for data codings 8 and 240. The "esm class" and "registered delivery" keys also showed a clear correlation between the RTTs and the adjustment factors.

Value	O1	O2	O3
0	**-0.46** (9.7)	-0.13 (8.2)	-0.10 (466)
8	**0.92** (25)	1.30 (24)	0.11 (673)
240	-1.12 (3.7)	-0.87 (3.5)	N/A
245	-0.03 (4.9)	N/A	-0.25 (358)

Table 2: Adjustment factors, by data coding. The value inside parentheses is μ from Table 1.

6.3 Anomaly frequencies

Figure 3 uses blue circles to show the RTTs for 288,515 outgoing requests to O1, over a period of approximately two months. Most measurements are around 10 milliseconds (1e+04 microseconds), but RTTs of up to several seconds are common enough that they are not considered outliers. The black, green and red lines show the medians from A_1, A_3 and A_5, respectively, as described in Section 4.4. The black line shows the median of the $5^2 = 25$ most recent measurements. Even with the large number of measurements above 1e+06 microseconds at Index 240,000, there is still enough data with lower values to keep the median below 1e+05 microseconds. The green line shows the median of 25 values from the black line, i.e. $25^2 = 625$ measurements. It stays significantly calmer, peaking only for indices 18,000, 190,000 and around 240,000, all corresponding to wider peaks of the black line. The red line shows the median yet another factor of 25 up, for $25^3 = 15,625$ measurements. Although some noise remains, the values shown by the red line (A_5) can be used for comparisons with those shown by the green line (A_3).

The intervals that satisfy our condition for anomalies, i.e. when the median of A_3 is more than twice the median of A_5 as described in Section 4.4, are marked with red lines at the bottom of the graph, surrounded by grey dotted rectangles. These lines perfectly mark the sections with many slow responses.

Despite the large variance shown in Section 6.1, using adjustment factors and medians provides a base level that is relatively stable. The area containing outliers for O2 is shown in Figure 4(b), where the blue dots have been removed for clarity. The end point of the marked area is quite far away from the starting point, indicating low precision of our method. This is the trade-off for high recall and avoiding multiple adjacent groups of outliers. There are no round-trips at 196,000 shorter than 5000 microseconds, causing A_3 (shown by the green line) to increase from 4267 microseconds to 8229. This makes A_3 more than twice the value of A_5 (shown by the red line), i.e. $8229 > 2 * 3964$, satisfying our condition for outliers.

The effect of the adjustment factors is illustrated in Figure 4(a) and Figure 4(b). Both figures show the same data, without and with adjustment factors, respectively. The black and green lines in Figure 4(b) are more stable, reporting one anomaly instead of three.

The algorithm detected no anomalies in the traffic towards O3.

For validation, we created simulated log files. The RTTs were randomized with a log normal distribution and a minimum value of 1000 microseconds. After at least 20,000 roundtrips, a group of up to 4095 entries with up to half a second slower responses was added. The results from the analysis on one such file are shown in Figure 5. There were three groups with slow responses, one at 48,515 with 2488 entries, one at 82,120 with 1222 entries, and one at 9133 with 192 entries, corresponding to the three blue peaks. Given there must be at least $625/2 = 313$ entries for our algorithm to report an anomaly, only the first two peaks are reported.

The red line is almost perfectly flat, showing the Remedian [20] is stable.

7 VALIDITY THREATS

Below we discuss the threats to the validity of our study.

Internal: We see two possible internal validity threats. First, although the 8 option keys we discarded in Section 6.1 showed no significance in the RTTs in our preliminary results, a more advanced analysis might show an effect. Second, any implementation errors were mitigated by carefully examining the program output, manually comparing it with the raw data in the log files.

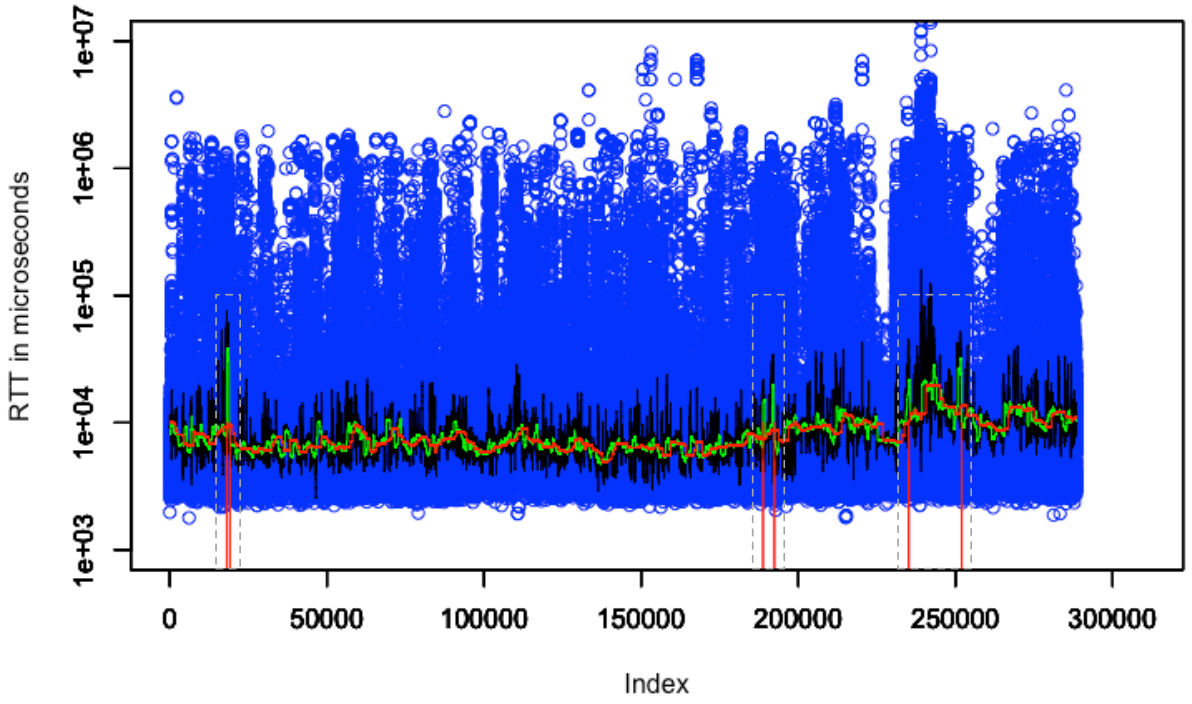
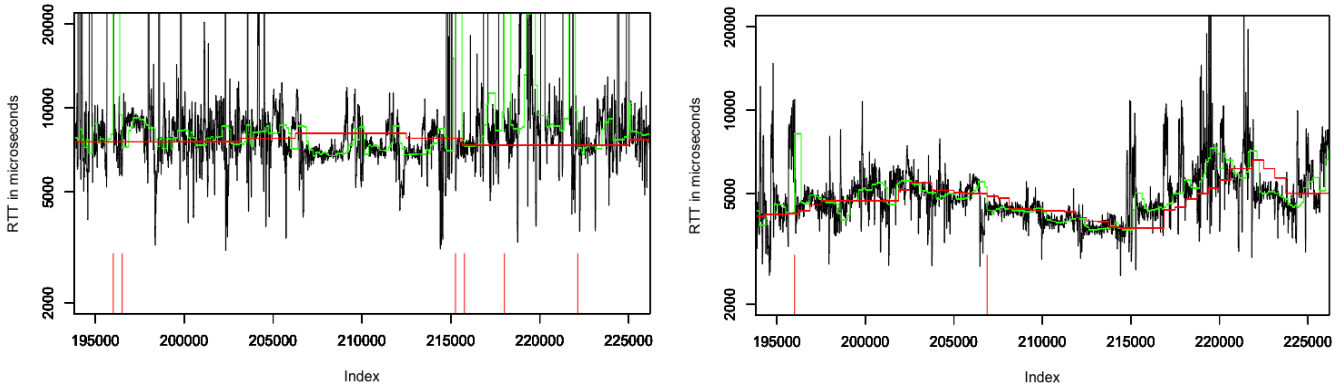


Figure 3: RTT and medians for O1.



(a) RTT and medians for O2, without adjustment factors.

(b) RTT and medians for O2, with adjustment factors.

Figure 4: RTT and medians for O2 with and without adjustment factors.

External: The analyzed log files in this paper all contain SMS traffic over SMPP, but the approach with exponential smoothing and the equations in Section 4.2 should be usable in any system where parameter values affect which parts of the code are executed, and therefore also the response time. When calculating the median, the sensitivity can easily be changed by adjusting the array length, and selecting which arrays to compare.

Reliability: We consider the reliability threat to be small, as we have seen similar RTT distributions for connections to several operators around the world.

Construct: The system model used in this paper is somewhat simplified, abstracting the network traffic into logged "send" and "receive" events. In reality, an outgoing PDU requires multiple steps:

(1) The data structure with the information to be sent is created.
(2) The data is packed into a byte array that can be transmitted on a socket.
(3) The data in the data structure is logged.
(4) The byte array is sent to the operating system kernel.
(5) The operating system sends the byte array to the network device.
(6) The network device transmits the byte array to the network.

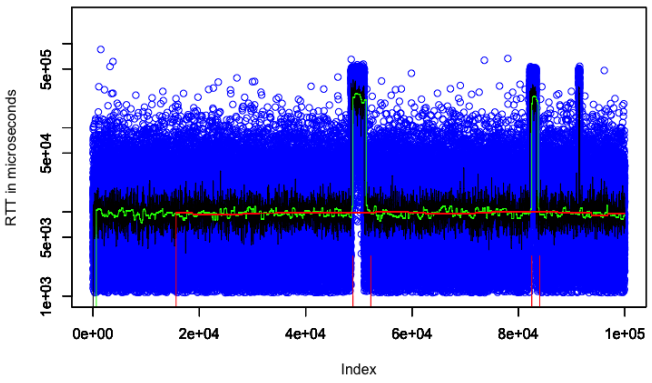


Figure 5: Simulated RTTs, with medians and outliers.

The timestamp used for the PDU comes from step 3, ignoring any delays caused by the subsequent steps. The operating system used is Unix, which does not provide a simple way to find the exact time for step 6. Instead we assume that delays are small compared to the network transmission and application processing times.

A limitation of the Remedian (described in Section 4.4) is that only value sequences that start on multiples of 5^n are considered, so the number of outliers required to trigger an anomaly notification varies. We do not consider this a problem, as the algorithm must always be adapted in order to achieve the desired sensitivity.

8 CONCLUSIONS AND FUTURE WORK

Anomaly detection in production systems is valuable for ensuring service levels towards customers. Making use of our domain knowledge, we developed an algorithm that reduces noise, enabling the detection of larger clusters of outliers. The algorithm is implemented as an extension to our tool ELFA which calculates RTTs between different communicating systems.

Even when the average RTT is within acceptable limits when analyzing data from the live production environment of an SMS broker, our approach can be used to identify conditions which have an unreasonable effect. A relevant example would be the UCS-2 handling (see Table 2, the factor when the parameter is 8) by O1 and O2. Whilst being functionally correct, it suggests the UCS-2 handling could possibly be made more efficient in those systems.

RTT anomalies may also be possible to detect by observing their side effects, such as a queue of outgoing messages being formed. The higher the throughput normally is, the longer the queue can be while maintaining an acceptable delivery time, so such an algorithm would have to take the current throughput into account. The throughput is not logged by EMG, so we could not use this method.

ACKNOWLEDGMENTS

This work was sponsored by The Knowledge Foundation industrial PhD school ITS ESS-H, 20160139 (TestMine), 20130085 (TOCSYC) and Infoflex Connect AB.

REFERENCES

[1] Sandip Agarwala, Fernando Alegre, Karsten Schwan, and Jegannathan Mehalingham. 2007. E2EProf: Automated end-to-end performance management for enterprise systems. In *Proceedings - IEEE International Conference on Dependable Systems and Networks (DSN'07)*.
[2] Jay Aikat, Jasleen Kaur, F. Donelson Smith, and Kevin Jeffay. 2003. Variability in TCP round-trip times. In *Proceedings - ACM SIGCOMM Conference on Internet Measurement (IMC'03)*.
[3] John H Bickford. 1972. *Mechanisms for intermittent motion.* Krieger Pub Co.
[4] Daniel Brahneborg, Wasif Afzal, and Adnan Čaušević. 2017. A Black-Box Approach to Latency and Throughput Analysis. In *Proceedings - IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C'17)*.
[5] Daniela Brauckhoff, Kave Salamatian, and Martin May. 2009. Applying PCA for traffic anomaly detection: Problems and solutions. In *Proceedings - IEEE International Conference on Computer Communications (INFOCOM'09)*.
[6] Jake D Brutlag. 2000. Aberrant Behavior Detection in Time Series for Network Monitoring. In *LISA*, Vol. 14. 139–146.
[7] Tony Finch. 2009. Incremental calculation of weighted mean and variance. (2009).
[8] Marina Gutiérrez, Wilfried Steiner, Radu Dobrin, and Sasikumar Punnekkat. 2015. Learning the parameters of periodic traffic based on network measurements. In *Measurements & Networking (M&N), 2015 IEEE International Workshop on*. IEEE.
[9] Emir Halepovic, Jeffrey Pang, and Oliver Spatscheck. 2012. Can you GET me now? Estimating the time-to-first-byte of HTTP transactions with passive measurements. In *Proceedings - ACM SIGCOMM conference on Internet measurement*.
[10] Anders Holst and Bjorn Bjurling. 2013. A bayesian parametric statistical anomaly detection method for finding trends and patterns in criminal behavior. In *Proceedings - European Intelligence and Security Informatics Conference*.
[11] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, Jim Kurose, and Don Towsley. 2004. Inferring TCP Connection Characteristics Through Passive Measurements. In *Joint Conf. of IEEE Computer and Communications Societies*.
[12] Hao Jiang and Constantinos Dovrolis. 2002. Passive estimation of TCP round-trip times. *ACM SIGCOMM Computer Communication Review* 32, 3 (2002), 75–88.
[13] Anukool Lakhina, Mark Crovella, and Christophe Diot. 2005. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Comp. Comm. Rev.* 35, 4 (2005).
[14] Yang Liu, Linfeng Zhang, and Yong Guan. 2010. Sketch-based streaming PCA algorithm for network-wide traffic anomaly detection. In *Proceedings - IEEE International Conference on Distributed Computing Systems (ICDCS'10)*.
[15] H Martin, A McGregor, and J Cleary. 2000. *Analysis of internet delay times.* Technical Report.
[16] A Finamore M Mellia and M Meo M M Munaf. 2011. Experiences of Internet Traffic Monitoring with Tstat. *IEEE Network* June (2011), 8–14.
[17] David Mosberger and Tai Jin. 1998. httperf - A Tool for Measuring Web Server Performance. *SIGMETRICS Performance Evaluation Review* 26, 3 (1998), 31–37.
[18] Vern Paxson. 1997. Automated packet trace analysis of TCP implementations. *ACM SIGCOMM Computer Communication Review* 27, 4 (1997), 167–179.
[19] Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. 2007. Sensitivity of PCA for traffic anomaly detection. *ACM SIGMETRICS Performance Evaluation Review* 35, 1 (2007).
[20] Peter J Rousseeuw and Gilbert W Bassett. 1990. The Remedian: A Robust Averaging Method for Large Data Sets. *J. Amer. Statist. Assoc.* 85, 409 (1990), 97–104.
[21] Shashank Shanbhag and Tilman Wolf. 2009. Accurate anomaly detection through parallelism. *IEEE Network* 23, 1 (2009), 22–28.
[22] Małgorzata Steinder and Adarshpal S. Sethi. 2004. A survey of fault localization techniques in computer networks. *Sci. of Comp. Prog.* 53, 2 (2004), 165–194.
[23] James W Taylor. 2003. Short-Term Electricity Demand Forecasting Using Double Seasonal Exponential Smoothing. *J. of Oper. Res. Society* 54, 8 (2003), 799–805.
[24] John W Tukey. 1977. Exploratory data analysis. (1977), 43–45.
[25] Chengwei Wang, Krishnamurthy Viswanathan, Lakshminarayan Choudur, Vanish Talwar, Wade Satterfield, and Karsten Schwan. 2011. Statistical techniques for online anomaly detection in data centers. In *IFIP/IEEE International Symposium on Integrated Network Management (IM'11) and Workshops*.
[26] Haijin Yan, Kang Li, Scott Watterson, and David Lowenthal. 2004. Improving passive estimation of TCP round-trip times using TCP timestamps. *IEEE International Workshop on IP Operations and Management* (2004), 181–185.
[27] Sebastian Zander and Grenville Armitage. 2013. Minimally-intrusive frequent round trip time measurements using synthetic packet-pairs. In *Proceedings - Conference on Local Computer Networks (LCN'13)*.
[28] Haibo Zeng, M Di Natale, Paolo Giusto, and A Sangiovanni-Vincentelli. 2010. Using Statistical Methods to Compute the Probability Distribution of Message Response Time in Controller Area Network. *IEEE Transactions on Industrial Informatics* 6, 4 (2010), 678–691.

Received October 2017; final version February 2018