# Estimating Performance of Mobile Services from Comparative Output-Input Analysis of End-to-End Throughput

Fiedler, Markus; Wac, Katarzyna; Bults, Richard; Arlos, Patrik

# Estimating Performance of Mobile Services from Comparative Output-Input Analysis of End-to-End Throughput

Markus Fiedler, *Member, IEEE*, Katarzyna Wac, *Member, IEEE*, Richard Bults, and Patrik Arlos, *Member, IEEE*

**Abstract**—Mobile devices with ever-increasing functionality and the ubiquitous availability of wireless communication networks are driving forces behind innovative mobile applications enriching our daily life. One of the performance measures for a successful application deployment is the ability to support application-data flows by heterogeneous networks within certain delay boundaries. However, the quantitative impact of this measure is unknown and practically infeasible to determine at real time due to the mobile device resource constraints. We research practical methods for measurement-based performance evaluation of heterogeneous data communication networks that support mobile application-data flows. We apply the lightweight Comparative Output-Input Analysis (COIA) method estimating an additional delay based on an observation interval of interest (e.g., 1 second) induced on the flow. An additional delay is the amount of delay that exceeds nonavoidable, minimal end-to-end delay caused by the networks propagation, serialization, and transmission. We propose five COIA methods to estimate additional delay, and we validate their accuracy with measurements obtained from the existing healthcare and multimedia streaming applications. Despite their simplicity, our methods prove to be accurate in relation to an observation interval of interest, and robust under a variety of network conditions. The methods offer novel insights into application-data delays with regards to the performance of heterogeneous data communication networks.

**Index Terms**—Mobile application, additional delay, heterogeneous networks, application level, throughput

## 1 Introduction

Emerging wireless network technologies and miniature personalized networked devices enable provision of new mobile applications that enrich daily activities of their users. These applications aspire to deliver mobile services to users "anywhere-anytime-anyhow" [1] while fulfilling their Quality of Service (*QoS*) and Quality of Experience (*QoE*) requirements [2], [3], e.g., low application response times. The success of the mobile service delivery depends heavily on the performance provided by underlying network infrastructures [1], [2], [4]. While users are on the move, these services operate in heterogeneous networking environments and knowledge of the overall performance at a particular user location and time is required by the service to optimize its user's experience. In particular, for interactive mobile applications that exchange data between spatially dispersed mobile and fixed nodes, the end-to-end data delays and their unexpected increase are critical performance measures [4]. However, these are difficult to measure at application runtime due to limited mobile device resources and clock synchronization issues [5].

To this end, we research practical methods for measurement-based performance evaluation of heterogeneous data communication networks that support mobile application-data flows. Particularly, we propose methods for an application's runtime estimation of additional delay, i.e., a stochastic delay exceeding the nonavoidable end-to-end delay, which considers some allowable data communication network's propagation, serialization and transmission and delays at network nodes [4], [5], [6]. Additional delay consists of those delay elements that increase the end-to-end delay of a message. For example, for a queued and pending transmission message, it is a delay occurring due to the fact that the scheduler has not rescheduled the thread handling its actual transmission.

We propose five different methods for additional delay estimation, that are based on *Comparative Output-Input Analysis* (COIA), which builds upon the comparative analysis of application level throughput at the sender and receiver nodes (e.g., a mobile device and an application server). We assume a "black box" view of the application on the underlying heterogeneous network infrastructure, where the application can just observe network behavior without enforcing it, or exploiting network-level measurements and/or performance feedback. We also assume an absence of strong clock synchronization between the sending and receiving nodes. We use the stochastic fluid flow model [7], [8] to analyze application-data traffic at small observation timescales, e.g., a second, where the

- *M. Fiedler and P. Arlos are with Blekinge Institute of Technology, School of Computing, SE-371 79 Karlskrona, Sweden. E-mail: {markus.fiedler, Patrik.Arlos}@bth.se.*
- *K. Wac is with Quality of Life Group, Institute of Services Science, University of Geneva, Battelle A-205, 7, route de Drize, CH-1227 Carouge, Switzerland. E-mail: katarzyna.wac@unige.ch.*
- *R. Bults is with MobiHealth B.V., Hengelosestraat 525-527, 7521 AG Enschede, The Netherlands. E-mail: richard.bults@mobihealth.com.*

*Manuscript received 15 July 2011; revised 25 Feb. 2012; accepted 30 May 2012; published online 12 June 2012.*
*For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2011-07-0390.*
*Digital Object Identifier no. 10.1109/TMC.2012.141.*

choice of a timescale is determined by the mobile service data delivery requirements.

We evaluate the accuracy of the proposed methods under a variety of network operator conditions, based on data traces from an existing mobile application, a health telemonitoring application provided by the MobiHealth system [9], [10] and a mobile multimedia streaming application. Especially, mobile healthcare applications pose strict application-level QoS requirements, because a patient in an emergency situation may require an immediate system response [11].

This paper is structured as follows: Section 2 provides concepts for modeling of additional delay, while Section 3 introduces the additional delay model and its estimation methods. Sections 4 and 5 provide accuracy evaluation results for these methods with data traces from the existing mobile applications. Section 6 provides implications for methods implementation, while Section 7 concludes upon our research and draws on future work areas.

## 2 Modeling of the Additional Delay

### 2.1 The Concept of Additional Delay

*Additional delay* $T^{\text{add}}$ is the amount of delay that exceeds the minimal end-to-end delay $T^{\text{min}}$, while $T^{\text{min}}$ is dominated by inherent, allowable data communication network's propagation, serialization, and transmission delays at network nodes [4], [5], [6], [12]. $T^{\text{add}}$ obeys a dynamic random process dominated by transmission conditions as well as competition from other traffic and processes [12]. Assuming that data are sent by the sender application entity at time $T^{\text{in}}$ and received by the receiver's entity at time $T^{\text{out}}$, $T^{\text{add}}$ is obtained as follows:

$$T^{\text{add}} = T^{\text{out}} - T^{\text{in}} - T^{\text{min}}. \tag{1}$$

Ideally, the additional delay vanishes. However, as it grows, the more pronounced its negative effect on the aggregated end-to-end delay becomes, in particular if it varies significantly. Most interactive applications suffer from $T^{\text{add}}$ exceeding certain thresholds; i.e., multimedia data can be considered lost if they miss their delivery deadlines, and user-perceived waiting times might grow beyond patience [13]. Ideally, the additional delay of each and every application-level message should be monitored to get a clear view of the disturbances that its delivery process is exposed to. This is not feasible. The mobile node might not be able to trace and time stamp each packet in real time due to scarce processing resources [4]. There are also limitations regarding the exactness of the time stamps, which might be too limited in resolution, incorrect due to processing times [14], and difficult to compare due to clock synchronization issues [15].

A certain amount of $T^{\text{add}}$ variation, also denoted as jitter, is expected for packet-based data delivery. Each link and node leaves its "footprint" on the inter-packet-timing within a packet stream, which means that the timely behavior of traffic at the receiver does not match the one at the sender. For instance, round-trip delays over an empty UMTS network (i.e., when no other traffic exists on the network) usually exhibits jitter of $\pm 10$ ms [16]. These variations are not necessarily problematic from the viewpoint of the application. However, values of $T^{\text{add}}$ in a range of one to several hundred milliseconds are perceptible [13], especially for interactive applications such as gaming [17]. These $T^{\text{add}}$ values stem, e.g., from congestion within access or core networks, or from temporarily bad radio network conditions, implying the need to resend data that were lost or corrupted. From the end-user point of view, the latter looks like a sudden loss of capacity, followed by a burst of data arriving at the receiver with a much smaller spacing in time than they were sent [18], [19]. It is, thus, important to capture and handle additional delays exceeding the expected variation thresholds. In the next section, we present a model that is able to provide this information.

### 2.2 Application Flow Model

A model that has shown to be capable of discerning between less critical delays on a packet level and more critical delays on a burst level is the *fluid flow model* [7], [8]. So far, it has been used for analysis of multiplexing in fast packet-switched networks and the consequences of temporary mismatches between capacity demand and availability, leading to considerable queuing on timescales beyond the packet scale. This model is based on the analysis of instantaneous workload instead of modeling the packet occurrence and length processes. In other words, the packets' intensity flow is quantified by the *data rate* $R(t)$, also denoted as *throughput* and measured in bits per second (bps), Bytes per second (Bps) or packets per second (pps). In general, $R(t)$ is a function of time and can be defined either on a continuous timescale as a derivative of the *cumulative workload* $W(t)$, passing a point of reference, by time:

$$R(t) = \frac{\mathrm{d}}{\mathrm{d}t} W(t), \tag{2}$$

or on a discrete timescale as the workload observed at a point of reference during averaging interval $i$ of duration $\Delta T$, divided by $\Delta T$:

$$R_i = \frac{W(i\Delta T + T_0) - W((i-1)\Delta T + T_0)}{\Delta T}. \tag{3}$$

The time counting is started upon the occurrence of the first packet being observed at the *inlet* (sender) or the *outlet* (receiver) of the network [19]. We denote the corresponding start times as $T_0^{\text{in}}$ and $T_0^{\text{out}}$, respectively. A packet occurring at an arbitrary time $t^{\text{in}}$ at the inlet and at time $t^{\text{out}}$ at the outlet contributes its workload to the overall workload in the intervals with numbers

$$i^{\text{in/out}} = \left\lceil \frac{t^{\text{in/out}} - T_0^{\text{in/out}}}{\Delta T} \right\rceil. \tag{4}$$

Obviously, the same packet can appear in different intervals at inlet and outlet, resulting in a nonvanishing additional delay at the timescale $\Delta T$ as described in the next section. The throughput time series at the network inlet and outlet are denoted by $\{R_i^{\text{in}}\}_{i=1}^{n}$ and $\{R_i^{\text{out}}\}_{i=1}^{n}$, respectively. They can be obtained from bit-, Byte- or packet counting at each $\Delta T$, followed by calculating (3). Compared to the effort related to tracing each and every time stamp of a packet observed at a point of reference, our approach is *lightweight*.

## 2.3 COIA and Equivalent Bottleneck

As outlined in Section 2.2, the comparison of the packet delivery processes at the *outlet* of the network with that at the *inlet*, together with subsequent analysis, abbreviated as COIA, allows for the quantification of the additional delay $T^{\text{add}}$. COIA builds upon the comparative analysis of the application-level throughput as observed at the network inlet and outlet, along a methodology presented in [16] and in absence of perfect clock synchronization of the sender and receiver. Initially, we assume lossless data delivery to highlight the properties and in particular the precision of COIA, and we will relax this assumption later.

We have already applied COIA to the analysis of throughput time series and related summary statistics. Among others, we have shown a classification of bottleneck behavior based on the comparison of throughput averages, standard deviations, and histograms between inlet and outlet [18], [19]. We have illustrated that if the standard deviation of a throughput time series increased between inlet and outlet, the network in-between acts as a shared bottleneck introducing additional delay. Thus, throughput time series and related summary statistics are lightweight descriptions of the data flow at the burst level. As they capture essential properties of the data flow, they can be used as *Reduced Reference Metrics* (RRM). In particular, RRMs can be exchanged between inlet and outlet (i.e., sender and receiver) to allow for runtime classification of bottleneck characteristics. Here, we investigate to which extent COIA based on throughput time series allows for an estimation of the additional delay $T^{\text{add}}$. To this end, we consider the end-to-end path as one *equivalent bottleneck*, whose content at the end of interval $i$ is described by [18]

$$X_i = X_{i-1} + D_i \Delta T, \quad X_0 = 0. \tag{5}$$

$X_i$ describes the amount of data that is still in transit at the end of interval $i$. As synchronization happens on the first packet, there is no content at the beginning of a session, i.e., $X_0 = 0$. $D_i$ is a throughput difference between inlet and outlet, called *drift* and defined as follows:

$$D_i = R_i^{\text{in}} - R_i^{\text{out}}. \tag{6}$$

The drift is a central parameter in fluid flow modeling [8], describing the rate at which the content increases or decreases, e.g., $D_i > 0$ means $X_i > X_{i-1}$ and $D_i < 0$ means $X_i < X_{i-1}$ if $X_{i-1} > 0$, while vanishing drift $D_i = 0$ implies a constant content $X_i = X_{i-1}$. If the time series are equal, i.e., $\{R_i^{\text{in}}\}_{i=1}^n = \{R_i^{\text{out}}\}_{i=1}^n$, there is neither drift ($D_i = 0$) nor content ($X_i = 0$), the equivalent bottleneck remains empty, and the network is considered to be *transparent* at the timescale $\Delta T$. In the special case $\{R_i^{\text{in}}\}_{i=1}^n = \text{const}$, the variations in $\{R_i^{\text{out}}\}_{i=1}^n$ reflect the variations in $D_i$ and, thus, of the buffer content of the equivalent bottleneck as such.

The fluid flow model assumes a fluid particle flow of constant intensity during the averaging interval. As long as a packet that was sent in interval $i$ is received inside the same interval, its additional delay is invisible. However, as soon as a packet belonging to interval $i$ traverses an interval boundary and is received in consecutive interval $j > i$, its additional delay becomes visible. Thus, the proposed methods rely on intervals bounds to quantify how much data from interval $i$ are delayed to consecutive interval(s). Let us illustrate this by assuming $D_i > 0$, $D_{i+1} = -D_i$, and $X_{i-1} = 0$: During interval $i$, less traffic leaves the outlet than what was delivered at the inlet. This particular amount $X_i = D_i \Delta T$ is still in transit on the end of interval $i$ and is, thus, delayed. In the next interval $i+1$, the queue gets empty again ($X_{i+1} = 0$). Intuitively, we estimate an additional delay in the order of $\Delta T$ across the equivalent bottleneck.

Consider now one packet sent at a uniformly distributed time during interval $i$ and received at a uniformly distributed time during interval $i+1$. We arrive at a triangular delay distribution over $](i-1)\Delta T, (i+1)\Delta T]$ with its median and average at $\Delta T$. The latter is consistent with the estimation discussed in the preceding paragraph. However, the uncertainty of our estimation amounts to $\pm\Delta T$, which implies that the choice of the time interval obviously has an impact on the error margins. This will be illustrated in Sections 4 and 5.

So far, we have implicitly assumed that the contents in the equivalent bottleneck have to be *causal* in the sense that—starting from $X_0 = 0$—the accumulated amount of traffic at the outlet cannot exceed the accumulated amount of traffic at the inlet:

$$\sum_i R_i^{\text{in}} \Delta T \geq \sum_i R_i^{\text{out}} \Delta T \ \ \forall i\ . \tag{7}$$

The causility principle (7) prevents the content of the equivalent bottleneck from becoming negative ($X_i \geq 0\ \forall i$).

Any violation of (7) is easily detectable through $X_i < 0$, and it has to be corrected as it implies the risk for erroneous estimations of $T^{\text{add}}$. A potential reason for negative content in the equivalent bottlenecks may be a desynchronization between the measurements on inlet and outlet, entailing $X_0 > 0$. Indeed, a correction is easily performed by

$$X_0 = X_0 + |X_i| \ \ \forall i:\ X_i < 0\ , \tag{8}$$

and estimations of $T^{\text{add}}$ for intervals ahead of $i$ have to be recalculated if necessary.

In case of loss, $X_i$ in (5) includes even the loss $L_i$ encountered in interval $i$. In fact, COIA considers lost traffic to remain in transit forever. Reflecting further on the above example, we now assume that the traffic stemming from $D_i > 0$ is lost. This would imply $D_{i+1} = 0$ and $X_j \geq D_i \Delta T\ \ \forall j > i$. Actually, there remains a residual content of $L_i = D_i \Delta T$ in the equivalent bottleneck, which appears the same way as a permanent delay of $\Delta T$. Equation (5) does not allow to distinguish between these situations; the only indication for loss might be the appearance of a positive trend in the values $X_i$.

In contrast to violations of the causality principle, loss $L_i$ needs to be discovered on a higher layer (e.g., through missing sequence numbers) and corrected through

$$X_i = X_{i-1} + D_i \Delta T - L_i \tag{9}$$

to not to introduce a permanent positive bias of the estimation of $T^{\text{add}}$.

## 2.4 Related Work

Related work areas attempt to model and estimate additional delay for data exchanged in a distributed data communication systems. Namely, with regards to modeling efforts, we recognize that the (5) can be seen as the fluid flow version of Lindley's recursion formula [20]; it expresses the waiting time at the end of an interval as a function of the waiting time at the beginning of the interval and the number of arrivals or departures during that interval. Based on the Lindley's formula, [21] investigated delay in a constant link capacity network, while our observations relate to the user-perceived capacity of mobile links.

The probably most cited paper in the domain of fluid flow modeling and analysis is [8], providing a closed-form analytical description of the buffer content of a fluid buffer of an unlimited size, fed by homogeneous on-off sources. However, Anick et al. [8] do not model the output process of the equivalent bottleneck; this is rudimentarily done in [22]. In [23], we have presented a simple yet effective analysis of the bit rate distribution arising from the Anick-Mitra-Sondhi-type equivalent bottleneck [8] and, thus, describe the effect of the bottleneck in terms of changes of bit rate histograms. The main idea in [23]—deriving information on the bottleneck behavior from a comparison of bit rate histograms at inlet and outlet of a bottleneck—was demonstrated in our previous work through a measurement study of video conferencing traffic [18] and subsequently a measurement study in mobile networks [19]. Both references implicitly used the COIA method.

In principle, the COIA method builds upon Little's Law [24], which, however, is very general and estimates the total end-to-end delay spent by a packet in the system (i.e., including transmission, propagation, queuing, and additional delays). The COIA methods use refined versions of Little Law to estimate merely additional delay values for a packet from a sender or receiver viewpoint. The authors of [25], [26], [27] have used Little Law as a base for modeling the total delay of packets in their systems. However, they assume system characteristics that are unattainable in reality, and they do not focus on different delay views, as we propose. Namely, in [25], the authors use Little's Law to estimate the total time packets spend in the system, assuming that the number of nodes in a network, a total number of packets exchanged by nodes, and total bandwidth available for each node are a priori known. They estimate delays for video, audio, voice, and messaging data packets and then use the estimated delay values in their proposal on traffic priority schemata. Similarly, Sarr and Guerin [26] used Little's Law to model delay of MAC-level frames exchanged between nodes connected via a WLAN network. They assume exponential frame interarrivals and service times. Borst [27] model a mean total delay spend by a frame in a flow, assuming a fair scheduling at mobile MAC-layer for different flows and fair capacity sharing at base stations by a mobile network operator.

With regards to modeling or measurements of additional delay, sometimes also denoted by authors as an overall queuing delay, we consider related work on different protocol stack layers for (mobile) nodes, from the MAC-layer, via IP up to the TCP/UDP transport layer. For example, Barry et al. [28] propose an enhanced MAC layer, i.e., a "Virtual MAC," that continuously and passively monitors interference at the MAC layer, interprets frames, and derives estimates for each frame's queuing delay and, based on these estimates, differentiates servicing of frames depending on their application flows, e.g., real-time voice or non real-time data. Graja et al. [29] focus on IP-level, per-datagram delay estimation for 2.5G/3G mobile operator networks, assuming a detailed knowledge on statistical characteristics of the radio channel and parameters of the MAC-layer quality control techniques used in the network, and the size of a transported IP datagram. Based on the network's delay estimation, they propose an adaption of size of the transported IP datagrams.

The authors of [30], [31] focus on TCP-level messages delay estimations. Namely, [30] model the messages' queuing delay distribution (using a finite state machine) for Internet traffic, however, assuming a message being sent only every RTT. Aniba and Aissa [31] propose an estimation of TCP-level RTT for web-based browsing (voice and data traffic) assuming the node in a 3G network knows its link utilization level. Ngamwongwattana and Thompson [32] propose measurements of queuing delays experienced by VoIP UDP-level messages via observing their interpacket time at the receiver. They assume that the first packet in the VoIP application flow has been received without (or with a minimum) queuing delay, and that a baseline interpacket time can be derived from it. However, this interpacket monitoring method is resource intense and error prone in mobile devices, due to clock's synchronization and resolution issues, especially for intense flows, where many packets are being sent or received in a second. In contrary, our methods quantify system behavior at the given time interval, balancing these factors. Moreover, we propose methods for receivers, as well as the sender's view.

As a related work, we also distinguish the proposal of Cola et al. [33], aiming to use a covert channel, where unused bits of the IP datagram would transfer a coded time stamp from sender to receiver, from which one-way delays and, thus, additional delays can be estimated. While this approach is passive in the sense that no extra traffic is created, it requires the modification of each and every data packet, which is infeasible given the limited API to the protocol stack, and limited computational resources of a mobile device. On the other hand, Kogel [34] proposes to derive one-way delays from flow data available in routers. Both papers address the issue of time stamp accuracy that emerges from different kinds of quantification issues as a key challenge for one-way delay estimations. The work of [6] supports a motivation for our proposal, as it provides a measurement study that compares the performance of 3G and 3.5G networks in both stationary and mobile settings, where it underlines the critical issue of delay spikes, observed in one-way delay measurements in a testbed, and presents their related statistics. Estimation of additional delays is not addressed.

In summary, literature conveys the picture that the challenges of delay spikes, the issue of correct time stamping, and the necessity of estimating additional delays are recognized; however, they are not treated in combination for mobile applications. Thus, this paper and its predecessor [35] close this gap.

## 3 Estimation Methods

This section provides methods for estimations of the additional delay $T^{\text{add}}$ based on throughput information at the inlet and outlet of the equivalent bottleneck representing the end-to-end network path. Given the limited communication, processing and storage capabilities of mobile devices, and a need of timely estimations, *simplicity* of the estimators is a major point of our concern. In particular, the parameters used for the estimation shall be considered in close time proximity to the current interval to make the approach as *stateless* as possible. For interval $i$, the latter constraint limits the scope to $X_{i-1}$, $X_i$, $R_i^{\text{in}}$, $R_i^{\text{out}}$, and $D_i$.

The proposed methods implement the COIA principle and, thus, require the exchange information about throughput ($R_i^{\text{in}}$, $R_i^{\text{out}}$ or $D_i$) or of buffer level $X_i$ between inlet and outlet (the implications for the implementation of the methods are discussed in Section 6). Therefore, we assume that either

1. $R_i^{\text{in}}$ can be sent from the sender toward the receiver, where (6) and (5) are calculated; or
2. $R_i^{\text{out}}$ can be sent from the receiver toward the sender, where (6) and (5) are calculated; or
3. the receiver can observe $X_i$, e.g., from the deviation of a jitter buffer from its reference value, and then calculate $D_i$ from (5) and $R_i^{\text{in}}$ from (6); or
4. the receiver might estimate the sender's average rate $\mathbf{E}[R_i^{\text{in}}]$, e.g., through the mean of $R_i^{\text{out}}$, which converges to $\mathbf{E}[R_i^{\text{in}}]$ in case of negligible loss, or use a-priori-known rates, e.g., from earlier measurements.

### 3.1 Sender View Ahead (SVA)

The sender is concerned about whether the data sent in the current interval $i$ have been received without experiencing additional delay. This means that at an arbitrary time, there should not be any (bottleneck) content left in the network. If there would be any content left, it would be visible at the end of the current averaging interval $i$ as $X_i > 0$. Seen from the sender point of view, which is actually expecting a throughput of $R_i^{\text{in}}$, this amount $X_i > 0$ is considered to be late by a time of

$$T_i^{\text{add}} = \frac{X_i}{R_i^{\text{in}}}. \tag{10}$$

As in this method, the sender considers its "left over" for the next interval $i+1$, we call this method *SVA*. If the current throughput vanishes, i.e., $R_i^{\text{in}} = 0$, the estimation is undefined.

### 3.2 Sender View Backwards (SVB)

In another but similar view, the sender is concerned about the impact of "left-over" data from the most recent interval $i-1$ onto the current interval $i$, while its current expected throughput is $R_i^{\text{in}}$. The *SVB* method for the estimation of additional delay is, thus, defined as follows:

$$T_i^{\text{add}} = \frac{X_{i-1}}{R_i^{\text{in}}}. \tag{11}$$

For constant sender throughput, SVB produces same estimation series as SVA, however moved by one interval, i.e., $\{\hat{T}_{\text{SVB},i}^{\text{add}}\}_i = \{\hat{T}_{\text{SVA,i-1}}^{\text{add}}\}_i$. $R_i^{\text{in}} = 0$ leads to an undefined estimation.

### 3.3 Receiver View Backwards (RVB)

The receiver is concerned about whether the data received in the current interval $i$ have experienced queuing. Such queuing is seen from a nonempty bottleneck at the end of the previous interval $i-1$, i.e., $X_i > 0$. Observing a leftover at the end of the interval $i-1$, the receiver can estimate the transport time needed for this outstanding data based on the current receiver throughput $R_i^{\text{out}}$. The *RVB* method for the estimation of $T^{\text{add}}$ is, thus, defined as follows:

$$T_i^{\text{add}} = \frac{X_{i-1}}{R_i^{\text{out}}}. \tag{12}$$

Replacing $R_i^{\text{out}}$ by the nominal link capacity, this method becomes the one used in [20]. Vanishing output $R_i^{\text{out}} = 0$ yields an undefined estimation.

### 3.4 Maximum of SVA, SVB, and RVB (MAX)

Due to the different points of view and depending on throughput and buffering content values, the methods SVA, SVB, and RVB are likely to provide different estimations of the additional delay. A pragmatic approach consists of using the most pessimistic estimation, which is the maximum of the three estimations obtained with SVA, SVB, and RVB, given as follows:

$$\hat{T}_i^{\text{add}} = \max\{\hat{T}_{\text{SVA,i}}^{\text{add}}, \hat{T}_{\text{SVB,i}}^{\text{add}}, \hat{T}_{\text{RVB,i}}^{\text{add}}\}. \tag{13}$$

Undefined values in the argument of the maximum operator are ignored; in the worst case, (13) cannot provide any estimation.

### 3.5 Mean Sender View Ahead (MSVA)

As a variant of SVA method, this method is designed for a receiver that may estimate the sender's average rate $\mathbf{E}[R_i^{\text{in}}]$ instead of the actual value of $R_i^{\text{in}}$ as described above, as a basis for estimation of the bottleneck content $X_i$. The *MSVA* method for the estimation of $T^{\text{add}}$ is defined as follows:

$$T_i^{\text{add}} = \frac{X_i}{\mathbf{E}[R_i^{\text{in}}]}. \tag{14}$$

## 4 Validation: Health Telemonitoring

In this section, we investigate to which extent the proposed methods are able to estimate $T^{\text{add}}$ based on time-synchronized, lossless data traces collected along the execution of a mobile health application. In addition, we evaluate the methods' accuracy under a variety of network conditions.

### 4.1 Setup

The MobiHealth system [9], [10] has been developed and used in the EU-FP6 MobiHealth project, and it enables real-time telemonitoring of vital signs (e.g., ECG) and context (e.g., location) of the mobile patients. A patient wears a Body Area Network (*BAN*) with an a Mobile Base Unit (*MBU*) as a central unit, acquiring data from wireless sensor system(s), processing it (e.g., deriving heart rate) and sending to a back-end-system (*BEsys*) in, e.g., a hospital. The end-to-end communication path is heterogeneous and

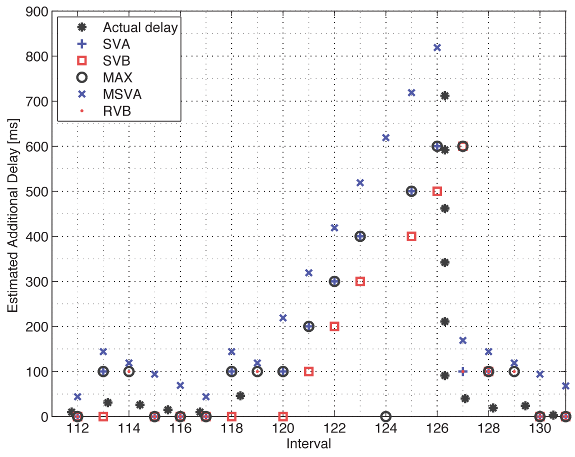

Fig. 1. Estimated additional delays for the sender and receiver view for $\Delta T = 100$ ms.

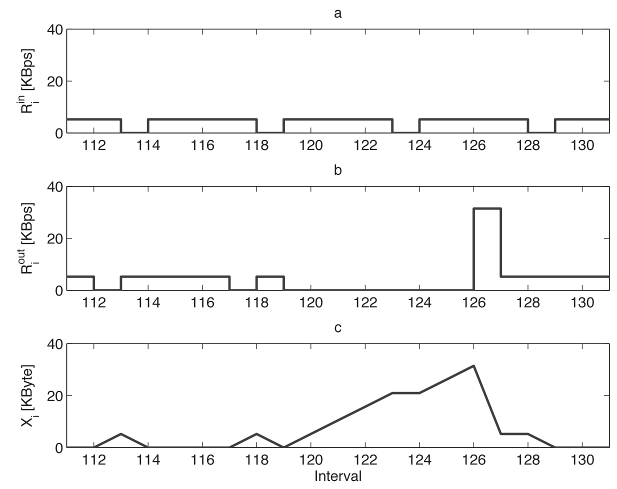

Fig. 2. Equivalent bottleneck: (a) inlet, (b) outlet and (c) content.

includes, e.g., 2.5G or 3G access network provided to a patient. Telemonitoring service is supported by the proprietary TCP/IP-based *MSP-Interconnect Protocol* (MSP-IP) [36] and conforms to the Jini Surrogate specification [37]. The application-data delivery is both lossless and in order but may suffer from additional delays. For performance evaluation of the MobiHealth system please refer to [16], [38].

For our studies, we exploit data traces derived from telemonitoring of a (hypothetical) cardiac patient living at the campus of the University of Twente (the Netherlands), using MBU[1] with 3G-UMTS access network of Vodafone [16] and BEsys being a high-performance server in the university network. We encapsulated application-data in fixed size (524 B) TCP message payload, and controlled the send rate to 7, 8, 9, 11 or 12 packets/second; i.e., resulting in an uplink rate of 32.6 to 55.9 kbps (given the overhead of 58 B per packet[2]). We have also controlled the application and the TCP buffer sizes, along the combinations: 64/64, 32/64, and 32/32 KB. The MBU and BEsys clocks were synchronized using Simple Network Time Protocol (SNTP)[3] over an external, dedicated Ethernet connection. We time stamp each sent and received application-data packet with a potential inaccuracy of $\pm 20$ ms [14]. At the sender side (i.e., MBU), we collect throughput time series $\{R_i^{\text{in}}\}_{i=1}^n$ at the ingress boundary of the TCP socket (i.e., the network inlet, after the "send" function), and collect $\{R_i^{\text{out}}\}_{i=1}^n$ at the egress boundary of the TCP socket (i.e., network outlet, after the "receive" function) at the receiver side (i.e., BEsys). Five different data rates, three combinations of buffer sizes, five replications of each experiment, and one trace affected by a hardware crash left us with 74 traces on which we conduced a validation of the proposed additional delay estimation methods.

## 4.2 Illustrative Example

We present an example of the causal relation between network (i.e., the equivalent bottleneck) behavior and the telemonitoring application-data traces for 8 pps (125-ms interpacket time), and application and TCP buffer sizes of 32 KB, respectively. In this example, the equivalent bottleneck content increases over a period of 2 seconds and then rapidly decreases (i.e., the queue releases). We investigated how the proposed methods estimate the additional delay. First, we present in Fig. 1 the behavior of sender and receiver and its relation to the value of $T^{\text{add}}$, captured by the data series "Actual delay." The $x$-axis displays the time intervals under consideration., and the $y$-axis shows its corresponding $T^{\text{add}}$ value as derived from the time stamps in the sender (MBU) and the receiver (BEsys) application-data traces. Each data point represents a packet as registered on the receiver side.

The packet sent in interval 119 experiences the maximal additional delay of $T^{\text{add}} = 712$ ms, which amounts to almost six nominal interpacket times (Fig. 1). We can see that this packet is suddenly released at the receiver together with five subsequent packets being queued. This behavior, which is quite common for mobile links, can be explained as follows: The first packet was corrupted or lost while being in transit and is retransmitted, while the subsequent packets are held until the retransmission of the first packet was successful.

Fig. 2 presents the data rate at (a) inlet (MBU) and (b) outlet (BEsys), and (c) the bottleneck content for intervals as in Fig. 1. The sender has a regular pattern of sending data in four out of five intervals, which matches the ratio between $\Delta T$ and the interpacket time. The reception is, however, quite bursty with no data being received during intervals 119 to 125. During the interval 126, the receiver gets hold of all six outstanding packets. Afterward, it continues receiving a data stream during intervals 127 to 129. The content of the equivalent bottleneck shows the increase and a release of the data.

Fig. 1 presents estimated additional delay values for the five proposed methods. The methods SVA and SVB provide increasing estimations of $T^{\text{add}}$, reaching values of 600 ms as they react upon the growing bottleneck content. SVB with its backwards point of view is typically one interval "late" with its estimations. A vanishing input in intervals 114, 119, 124, and 129 makes the results undefined, *cf.* (10) and (11). When the bottleneck grows, the RVB method performs very

1. Asus laptop (MPIII 1-GHz proc., 640-MB RAM, WinXP OS), using a Nokia 6650 phone as a USB modem to 3G-UMTS network.
2. MSP-IP (10 B), TCP/IP (40 B) and PPP (8 B) overhead.
3. SNTP: http://www.ntp.org/ntpfaq/NTP-s-def.htm.

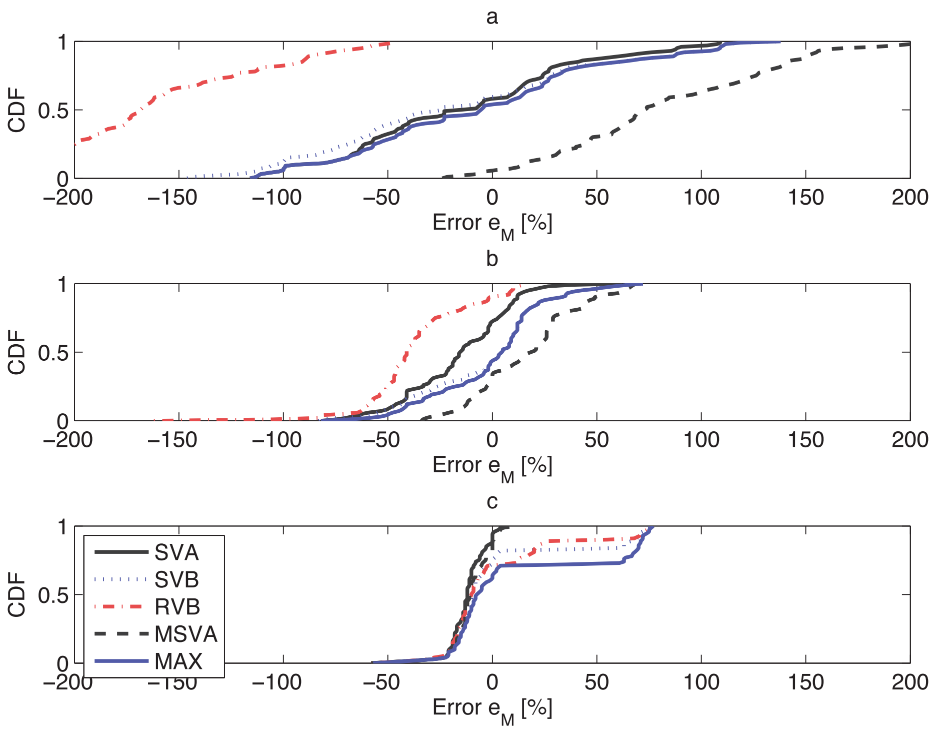

Fig. 3. TCP: Empirical CDF of the relative estimation error $e_M$ for $\Delta T \in \{$(a) 100 ms, (b) 300 ms, (c) 1 s$\}$.

poorly if there is no observed data at the receiver, e.g., in intervals 119 to 125; thus, the result of (12) is undefined. Only in the interval 127, method RVB provides $T^{\text{add}}$ estimations that relate to the measured values.

For interval 124, as there is data neither sent nor received, none of the methods SVA, SVB or RVB are able to estimate the additional delay. In the other intervals, the MAX method provides estimations of an increasing value, as it uses the maximum value of SVA, SVB, and RVB, where especially the first two react upon the growing bottleneck content.

The MSVA method delivers quite conservative estimations. It takes the estimated sender behavior instead of the real one into account. Hence, it assumes that 420 B are sent in each interval of $\Delta T = 100$ ms. The method accounts this data for bottleneck content, which results in an over-estimation of additional delays. Similarly to SVA, this method reacts upon the growing bottleneck content. Just before releasing the queue, MSVA estimates $\hat{T}^{\text{add}}_{\text{MSVA},126} = 819$ ms, which is to be compared to real value of $T^{\text{add}} = 712$ ms and the SVA-based estimation of $\hat{T}^{\text{add}}_{\text{SVA},126} = 600$ ms. For the interval 124, MSVA is the sole method able to provide an estimation of $\hat{T}^{\text{add}}_{\text{MSVA},124} = 619$ ms.

### 4.3 Accuracy of the Estimations

Having examined in details an illustrative example in the previous section, we now present the cumulative results for the accuracy of estimations of additional delays along all 74 application-level traces collected by the telemonitoring application that serve as points of reference. From the sender (MBU) and receiver (BEsys) time stamps, we derive for each trace $T^{\min}$ which is defined as the minimum delay value ever occurred in that trace. Then, for each packet $p$ of this trace, we derive its (measured) additional delay $T^{\text{add}}_p$. The *maximum additional delay* ever occurred in the trace is denoted as follows:

$$T^{\text{add}}_{\max} = \max_p \left\{T^{\text{add}}_p\right\} = \max_p \left\{t^{\text{out}}_p - t^{\text{in}}_p - T^{\min}\right\}. \tag{15}$$

Then, for each method $M$, we calculate its maximal estimated additional delay in the trace as follows:

$$\hat{T}^{\text{add}}_{M,\max} = \max_i \left\{\hat{T}^{\text{add}}_{M,i}\right\}. \tag{16}$$

The relative estimation error is defined as follows:

$$e_M = \frac{\hat{T}^{\text{add}}_{M,\max} - T^{\text{add}}_{\max}}{\Delta T}. \tag{17}$$

We have observed in all traces that a typical value of $T^{\text{add}}_{\max}$ is found in the order of magnitude of 300 ms, with some exceptions reaching up to 712 ms as shown before. Fig. 3 presents the cumulative distribution function (CDF) of the relative estimation error $e_M$ of all five methods SVA, SVB, RVB, MAX, and MSVA for $\Delta T = 100$ ms, 300 ms and 1 s, respectively. The $x$-axis displays the estimation error in percent of $\Delta T$.

For $\Delta T = 100$ ms (Fig. 3a), the estimation error for SVA ranges from $-1.2\Delta T$ to $1.1\Delta T$, for SVB from $-1.5\Delta T$ to $1.4\Delta T$ and for MAX from $-1.2\Delta T$ to $1.4\Delta T$. In most cases, the error is bounded by $\pm\Delta T$. RVB underestimates the additional delay by $-5\Delta T$ to $-0.5\Delta T$. This is due to its inability to trace growing content in the equivalent bottleneck, if there is no data being received (*see* Section 4.2). The MSVA displays error values between $-0.3\Delta T$ to $2.2\Delta T$; it has a clear tendency to overestimate the additional delay (*see* also Section 4.2).

For $\Delta T = 300$ ms (Fig. 3b), the estimation error for SVA ranges from $-0.8\Delta T$ to $0.7\Delta T$, for SVB from $-0.9\Delta T$ to

$0.8\Delta T$ and for MAX from $-0.9\Delta T$ to $0.8\Delta T$. The majority of absolute errors is found in the range of $-\frac{1}{2}\Delta T \leq e_M \leq \frac{1}{4}\Delta T$. RVB exhibits errors in the range of $-1.7\Delta T$ to $0.2\Delta T$; it again shows a tendency to underestimate the additional delay. The MSVA has an error range of $-0.4\Delta T$ to $0.7\Delta T$, with a less pronounced tendency to overestimate the value because $\mathbf{E}[R^{\text{in}}]$ is closer to the actual $R^{\text{in}}$ values on this timescale. For any of the methods, the relative error has decreased significantly at this timescale, whose choice coincides with the typical value of $T^{\text{add}}_{\max}$ in the order of 300 ms.

For $\Delta T = 1$ s (Fig. 3c), the most precise estimation methods are SVA and MSVA, with relative estimation errors ranging from $-0.6\Delta T$ to $0.04\Delta T$ (SVA) and to $0.12\Delta T$ (MSVA). Yet, 99 percent of the $e_M$ values for both methods are found in range of $-10$ to 4 percent (SVA) and to 12 percent (MSVA). The reason for such a high accuracy of the latter method is as follows: In the telemonitoring application, the sender is precisely timing the one-second time intervals during which it sends an integer number of packets. Hence, $\mathbf{E}[R^{\text{in}}] = R^{\text{in}}$ and that results in a precise estimation of the additional delays. This particular case shows that the MSVA method can be beneficial if $\Delta T$ corresponds to an integer multiple of the nominal inter-packet time. From Fig. 3, we also conclude that SVB and RVB and, hence, MAX have a tendency to overestimate the additional delays. The estimation error ranges from $-0.6\Delta T$ to $0.8\Delta T$.

In general, the best estimations are delivered by the SVA method, with a tendency to underestimate the additional delay, and by the MSVA method if the averaging interval matches the periodicity interval of the traffic. On the timescale $\Delta T = 100$ ms, which is considerably smaller than the majority of additional delays in our traces, the estimation error is more or less limited by this timescale, with exception of the method RVB. Indeed, in most cases, we perceive errors that are smaller than the timescale itself. Obviously, the methods even allow for observation of additional delays that are smaller than the time scale on which the measurements are carried out. This is observed because packets are occasionally "pushed" from one interval to the subsequent one, thus the methods can detect it. If $\Delta T$ is small, changes in $T^{\text{add}}$ are significant, and the shift of a single packet by a few milliseconds can easily yield a much larger predicted additional delay and, thus, a significant relative error. When the interval size increase, it is still possible to detect a number of packets being delayed from the one interval to the other and estimate an additional delay, which is a fraction of the timescale $\Delta T$ itself. At the same time, the longer averaging time also helps to reduce the relative contribution of a potentially shifted packet to the estimation error.

### 4.4 Discovery of Additional Delay Spikes

In this section, we investigate the accuracy of the proposed estimation methods for $T^{\text{add}}$ to indicate *additional delay spikes*, in a trace. A spike occurs when $T^{\text{add}}$ is higher than a certain threshold defined based on application's requirements for delays. For example, as illustrated in Fig. 1, the value of 712 ms is a $T^{\text{add}}$ spike, for the threshold of 300 ms, derived from the health telemonitoring application-level delay requirements.

TABLE 1
TCP: Performance of Additional Delay Spikes Indication by Different Estimation Methods ($\sigma^{\text{real}} = 45$)

| | $\Delta T$ | SVA | SVB | RVB | MAX | MSVA |
|---|---|---|---|---|---|---|
| $\sigma^{\text{ind}}_M$ | 100 ms | 125 | 129 | 4 | 147 | 176 |
| | 300 ms | 51 | 91 | 21 | 111 | 79 |
| | 1 s | 6 | 19 | 23 | 27 | 13 |
| $\sigma^{\text{FP}}_M$ | 100 ms | 85 | 94 | 0 | 107 | 131 |
| | 300 ms | 23 | 61 | 9 | 72 | 45 |
| | 1 s | 2 | 9 | 13 | 15 | 4 |
| $\sigma^{\text{FN}}_M$ | 100 ms | 5 | 10 | 41 | 5 | 0 |
| | 300 ms | 17 | 15 | 33 | 6 | 11 |
| | 1 s | 41 | 35 | 35 | 33 | 36 |
| $\gamma_M$ | 100 ms | 89 % | 78 % | 9 % | 89 % | 100 % |
| | 300 ms | 62 % | 67 % | 27 % | 87 % | 76 % |
| | 1 s | 9 % | 22 % | 22 % | 27 % | 20 % |

In that particular example, the spike extended over several consecutive observation intervals $\Delta T$ and was, thus, easily located. However, a spike might also occur inside an arbitrary interval. Consequently, $X_i$ is not affected, and because of this, the spike goes unnoticed. Such an uncaught spike is henceforth called *false negative*. On the other hand, due to the approximation nature of the proposed methods, their additional delay value might overstep the predefined threshold for a spike, while the actual additional value delay might not. We denote this case as *false positive*.

Let us define a set of approximation performance parameters for an observation timescale $\Delta T$ as follows:

- $\sigma^{\text{real}}$ as the total number of spikes (so-called true positives) existing in a set of traces;
- $\sigma^{\text{ind}}_M$ as the number of spikes indicated by method $M$;
- $\sigma^{\text{FP}}_M$ as the number of false positives indicated by method $M$;
- $\sigma^{\text{FN}}_M$ as the number of false negatives indicated by method $M$;
- $\sigma^{\text{TP}}_M = \sigma^{\text{real}} - \sigma^{\text{FN}}_M = \sigma^{\text{ind}}_M - \sigma^{\text{FP}}_M$ as the number of true positives indicated by method $M$;
- $\gamma_M = \sigma^{\text{TP}}_M / \sigma^{\text{real}}$ as the spike hit ratio for method $M$.

We consider a threshold for spikes of 300 ms as derived from health-telemonitoring application requirements and we find $\sigma^{\text{real}} = 45$ spikes in the investigated set of 74 traces. For $\Delta T = 100$ ms, $\Delta T = 300$ ms, and $\Delta T = 1$ s, Table 1 presents the total number of spikes indicated by the methods; the numbers of false positives; the numbers of false negatives; and the hit rate for each combination of method and timescale, respectively.

The SVA, SVB, MAX, and MSVA methods indicate many spikes and false positives for $\Delta T = 100$ ms, while those numbers decrease for $\Delta T = 300$ ms and 1 s. On the other hand, the number of false negatives rises for these four methods as $\Delta T$ grows. Despite those general trends, the numbers for the different methods and timescales differ. As compared to SVA, the three methods SVB, MAX, and MSVA indicate more spikes and false positives. For the short time interval $\Delta T = 100$ ms, SVB misses most spikes, while MSVA does not miss any.

The method RVB behaves differently, having a few indications, but no false positives on the short timescale, and rising numbers of those on the longer timescales. This behavior stems from the difficulty that RVB has with estimating additional delays when the bottleneck is increasing and nothing is received, *cf.* Fig. 1. The $T^{\mathrm{add}}$ for data inside the bottleneck content cannot be estimated by this method. The method gets more accurate as $\Delta T$ grows, because there are fewer intervals with vanishing receiver throughput. Still, the number of false negatives is high almost independently of $\Delta T$.

From the methods' hit ratios on the different timescales, we observe the same trends as already indicated. The sender-based methods show large hit rates between 78 percent (SVA) to 100 percent (MSVA) for $\Delta T = 100$ ms. For the time interval matching the delay threshold $\Delta T = 300$ ms, between 62 percent (SVA) and 87 percent (MAX) of the real peaks are discovered. For this $\Delta T$, also RVB shows its largest hit ratio of 27 percent. On the timescale of $\Delta T = 1$s, all the hit ratios are found between 9 percent (SVA) and 27 percent (MAX).

Summarizing the findings, we can see that the sender-based methods SVA, SVB, and MSVA are superior to the receiver-based method RVB unless the time interval $\Delta T$ clearly exceeds the duration of the spike. SVB and MAX are more indicative than SVA. While for small $\Delta T$, the sender-based methods have a tendency to overestimate the delay values (resulting in false positives), all methods tend to not notice spikes in case of the large $\Delta T$, which is somehow expected given the fact that the threshold for spikes is merely 30 percent of the observation interval. This behavior stems from the fact that, to accurately detect a spike of a given value, $\Delta T$ must be of a smaller size than this spike. The larger $\Delta T$, the more spikes smaller than $\Delta T$ go unnoticed. The choice of $\Delta T$ shall be related to the designated levels of delay spikes to be discovered, when using the methods.

## 5 Validation II: Multimedia Streaming

In this section, we provide evaluation results along the same goals and approach as already presented in Section 4, but for UDP-based application traces. Therefore, in this section, we investigate to which extent the proposed methods are able to correctly estimate $T^{\mathrm{add}}$ and delay spikes. We evaluate the accuracy of each method against time-synchronized data traces collected at sender and receiver along the execution of a multimedia streaming application provided to a mobile user.

### 5.1 Setup

The system used for the evaluation of methods provides a multimedia streaming to and from a mobile user with use of a UDP-based application protocol; the user downloads or uploads some multimedia content. For our studies, we exploit the traces obtained from an application used on a mobile laptop, by a user located in Karlskrona (Sweden) and using his application in one specific location (Blekinge Institute of Technology). The laptop is connected to 3G-UMTS networks of three different mobile operators, two of which share the radio part of their radio access networks.

The application was configured to send data streams using different multimedia coding; resulting in 1 up to 25 packets per second, with packet sizes of 64, 256, 512 or 750 B, being sent separately to the network at interpacket times from 4 to 256 ms. The protocol stack overhead[4] is 36 B and the overall data rate uploaded by the mobile laptop to the application server ranges from 8 to 360 kbps, while data rate downloaded to the mobile laptop ranges from 8 kbps to 2 Mbps. Instead of collecting the data in the laptop and server, we opted to collect the from the data link layer using Endace DAG cards. The setup is similar to the one shown in [39]. This perfectly synchronised system, with an accuracy of less than 60 ns, provides an ideal basis for our validation endeavour. At the ingress boundary of the UDP socket (i.e., at heterogeneous network infrastructure inlet) at the laptop side, we collect throughput time series $\{R_i^{\mathrm{in}}\}_{i=1}^n$, while $\{R_i^{\mathrm{out}}\}_{i=1}^n$ is collected at the egress boundary of the UDP socket (i.e.,at the network outlet) at the application server side. All the packets included a sequence number that enables us to trace losses and packet reordering. We consider 33 traces as a base for the accuracy validation for the proposed methods. Traces exhibited occasional, random packet losses (0.2 percent), and no packet reordering. Similarly to the TCP case, we observed the typical value of $T_{\max}^{\mathrm{add}}$ around 300 ms.

As for the TCP-based traces evaluation, we present the methods' accuracies of the estimated maximal delay at different timescales (Section 5.2) as well as the evaluation of spikes of additional delays (Section 5.3).

### 5.2 Accuracy of the Estimations

Fig. 4 presents the CDF of the relative estimation error $e_M$ of all five methods SVA, SVB, RVB, MAX, and MSVA for $\Delta T = 100$ ms, 300 ms, and 1 s, respectively. The $x$-axis displays the estimation error in percent of $\Delta T$.

For $\Delta T = 100$ ms (Fig. 4a), the relative estimation error for SVA is bounded by $-1.5\Delta T$ to $0.65\Delta T$ and for SVB by $-2.4\Delta T$ to $1.5\Delta T$. For RVB, the error stretches from $-5\Delta T$ to $10\Delta T$. This is due to its inability to trace growing content in the equivalent bottleneck, if there is no data being received, and due to overestimating delays if very little traffic is received compared to what has been sent. The MSVA has a tendency to overestimate the additional delay. It is caused by two factors, both resulting in an effective sender throughput being smaller than the estimated sender's average throughput. First, the sender does not follow a very regular sending pattern (even at the timescale of 1 s); hence, the estimated sender's throughput may be significantly lower or higher than the instant one. Second, the traces exhibit occasional losses; for such events, the estimated sender's throughput is higher than the instant one.

For $\Delta T = 300$ ms (Fig. 4b), the relative estimation error for SVA is bounded by $\pm\Delta T$ and for SVB by $-1.1\Delta T$ to $0.2\Delta T$. The RVB and MSVA methods show a tendency to overestimate the additional delay for the reasons mentioned in previous paragraphs. In general, for all of the methods, the relative estimation error has decreased significantly at this timescale.

For $\Delta T = 1$s (Fig. 4c), all the estimation methods but MSVA sufficiently accurate the $e_M$ values for the SVA, SVB,

4. It includes the UDP, IP and PPP overheads.

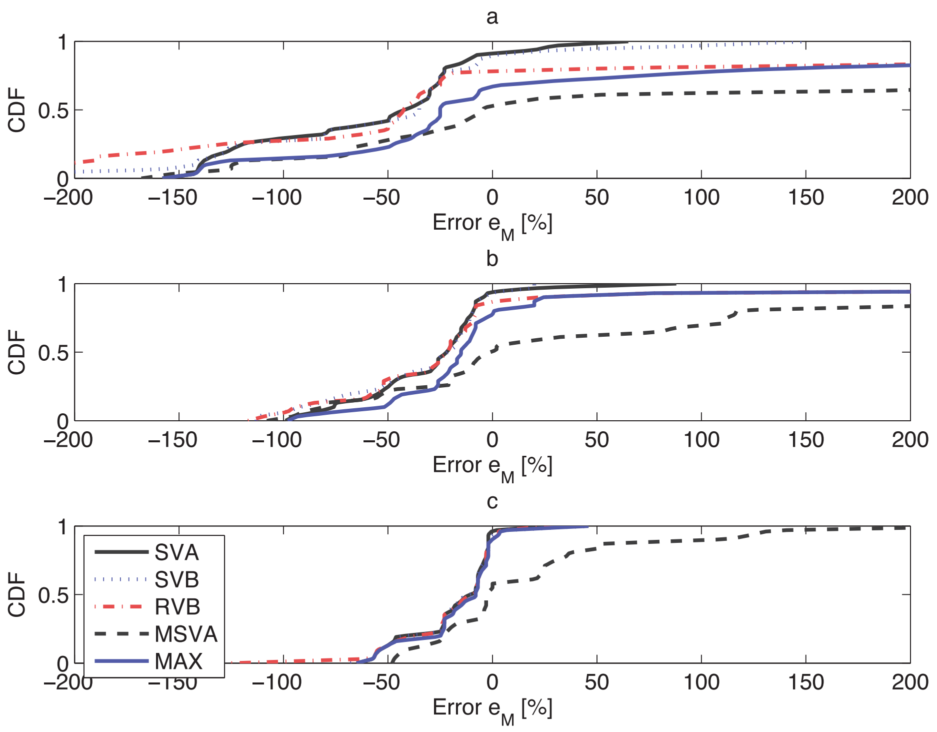

Fig. 4. UDP: Empirical CDF of the relative estimation error $e_M$ for $\Delta T \in \{$(a) 100 ms, (b) 300 ms, (c) 1 s$\}$.

RVB, and MAX methods are in range of $-1.0\Delta T$ to $0.5\Delta T$. At this timescale, the phenomenon of a growing content in the equivalent bottleneck is not as pronounced as at smaller timescales; hence, the RVB method exhibits a high accuracy. Moreover, at this timescale, the phenomenon of the sender's imprecise behavior influences less the accuracy of the MSVA method.

From all the three figures, we conclude that the relative accuracy of SVA, SVB, and RVB and, hence, even MAX methods increases with the increasing timescale. Yet, especially at small timescales, all these methods have a slight tendency to overestimate the additional delays. This overestimation is in many cases related to the high variability of the throughput being received. Namely, there exist intervals in which very small amount of traffic is received with comparison to what has been sent, but the traffic being in transit does not necessarily experience delays peaks, as the methods attempt to indicate. The latter behavior relates to the fact that methods relay their estimation of $T^{\mathrm{add}}$ on counting packets that are pushed from one interval to the subsequent one. The smaller the intervals, the greater the potential errors become, while larger intervals help to average out some issues, however at the price of an increased risk of missing delay spikes.

### 5.3 Discovery of Additional Delay Spikes

Considering a threshold for spikes of 300 ms, we find 10 spikes in the investigated set of 33 application-level traces. For $\Delta T = 100$ ms, $\Delta T = 300$ ms and $\Delta T = 1$ s, Table 2 presents the total number of spikes indicated by the methods; the numbers of false positives; the numbers of false negatives, and the hit rates for each combination of method and timescale, as introduced in Section 4.4.

All methods indicate many spikes for $\Delta T = 100$ ms, while the numbers decrease for $\Delta T = 300$ ms and 1 s. The number of false negatives rises as $\Delta T$ grows for these methods. The methods do not indicate spikes as the timescale grows, as these peaks happen inside the observation interval. Despite those general trends, the numbers for the different methods and timescales differ. The methods SVA and SVB exhibit the highest accuracy.

The method MSVA behaves in a different way. We observe a lot of spikes, many of which are false positives, especially on the short timescale. In general, the accuracies decrease on the longer timescales. This behavior stems from the inaccuracy of MSVA throughput estimations when the sender is not adhering to regular sending patterns and when data are occasionally lost.

Looking at the methods' hit ratios on the different timescales, we observe the same trends as indicated in the previous sections. The SVA, SVB, RVB, and MAX methods show high hit rates, especially on a scale $\Delta T = 100$ ms. The MAX method exhibits an accuracy of 100 percent.

TABLE 2
UDP: Performance of Additional Delay Spikes Indication by Different Estimation Methods ($\sigma^{\mathrm{real}} = 10$)

| | $\Delta T$ | SVA | SVB | RVB | MAX | MSVA |
|---|---|---|---|---|---|---|
| $\sigma_M^{\mathrm{ind}}$ | 100 ms | 11 | 12 | 10 | 17 | 166 |
| | 300 ms | 14 | 8 | 11 | 15 | 45 |
| | 1 s | 5 | 5 | 5 | 5 | 27 |
| $\sigma_M^{\mathrm{FP}}$ | 100 ms | 2 | 3 | 3 | 7 | 157 |
| | 300 ms | 7 | 2 | 6 | 8 | 38 |
| | 1 s | 2 | 2 | 2 | 2 | 24 |
| $\sigma_M^{\mathrm{FN}}$ | 100 ms | 1 | 1 | 3 | 0 | 1 |
| | 300 ms | 3 | 4 | 5 | 3 | 3 |
| | 1 s | 7 | 7 | 7 | 7 | 7 |
| $\gamma_M$ | 100 ms | 90 % | 90 % | 70 % | 100 % | 90 % |
| | 300 ms | 70 % | 60 % | 50 % | 70 % | 70 % |
| | 1 s | 30 % | 30 % | 30 % | 30 % | 30 % |

For the time interval matching the predefined threshold $\Delta T = 300$ ms, between 50 percent (RVB) and 70 percent (SVA, MAX) of the real peaks are discovered. Finally, on the timescale of $\Delta T = 1$ s, all hit ratios of the methods are 30 percent.

As already indicated in Section 4.4, this behavior stems from the fact that, to detect a delay spike of a given value, the interval $\Delta T$ must be shorter that this spike, hence making it possible to discover packets that are pushed from one interval to the subsequent. This indicates that the choice of $\Delta T$ shall be related to the designated levels of delay spikes to be discovered using the proposed methods.

Summarizing the findings, we can see that the MAX method is the most indicative as it combines strengths of all the underlying methods: SVA, SVB, and RVB. While for small $\Delta T$, the sender-based methods have a tendency to under- or overestimate the delay values (and produce false negatives and positives), all methods tend to overlook spikes in case of large $\Delta T$, which is somehow natural given the fact that the threshold for spikes is merely 30 percent of the observation interval. As already indicated in the Section 4.4, this behavior stems from the fact, that to detect a delay spike of a given value, the interval $\Delta T$ must be of a smaller length that this spike, hence enabling to discover packets that are "pushed" from one interval to the subsequent one, forming a spike. This indicates that the choice of $\Delta T$ shall be related to the designated levels of delay spikes, to be discovered when using the methods.

## 6 Implications for Implementation

In this section, we draw conclusions and recommendations for a practical application of the proposed and evaluated additional delay estimation methods for real mobile application scenarios.

### 6.1 Impact of the Observation Interval

One important practical implication relates to the choice of an appropriate observation interval $\Delta T$ at which additional delays are estimated. The choice is limited by a tradeoff between feasibility and effort required to instrument the application for throughput measurements at the sender and receiver side, and the requirements of discovering application-data additional delay at the given timescale. In many cases, this additional delay will be related to interpacket times. We propose that an optimal $\Delta T$ shall be defined as a percentage of the required minimal end-to-end delay and, if possible, related to a integer multiple of the designated interpacket time, to improve the accuracy of the proposed methods. If $\Delta T$ is too long, the methods loose possibilities to discover additional delays, shown by many false negatives. If $\Delta T$ is too short, the system becomes too sensitive, discovering many nonexisting additional delay spikes, i.e., many false positives. Such a situation can destabilize a system whose application control loops depend upon these estimations. The most important question one needs to answer when choosing the observation interval $\Delta T$ is how much additional delay is tolerated by the application before a control action, such as application adaptation, or a warning toward the user would be required.

### 6.2 Particularities of Specific Methods

The RVB method exhibited a low performance, i.e., frequent underestimations of the additional delay, especially for a growing bottleneck content, where the receiver does not receive any data. To improve this method's accuracy in practice, one can combine it with other methods and assume that if the receiver does not receive any data, the bottleneck content and, thus, the additional delay grows. Then, one can use the MSVA method at the receiver to estimate the delay values.

The MSVA method exhibited a low performance, i.e., overestimation of the additional delay, especially for cases when the sender's current behavior deviated significantly from its average behavior, or in case of traces with data loss. The estimations provided by this method can be treated as a worst case scenario; the other methods can be used for more precise estimations of additional delay.

### 6.3 Missing Measurement Data

COIA is relying upon comparison of performance data. In this section, we consider cases in which exchange of the data does not work as expected. We follow the classification presented in Section 3.

In cases 1 and 2, lacking throughput data should not be interpreted as zero values in the first hand due to the recursive nature of (5). $R_i^{\text{in}}$ values set to zero would drive the system toward incausality and underestimations of $T^{\text{add}}$, while $R_i^{\text{out}}$ values set to zero would suggest persistent bottleneck content, loss and overestimations of $T^{\text{add}}$. A rather simple but straightforward mitigation strategy in case 1, which is also supported by the examples shown in Figs. 3 and 4, is to increase the estimation of $T^{\text{add}}$ by the time that $R_i^{\text{in}}$ is overdue, as the latter might be strongly correlated with the fact that the data path is blocked. In case 2, such a practice might yield overestimations when the delay occurs on the return path. Indeed, complete loss of monitoring data puts COIA as such at stake. Eventually, data loss discovery on some higher level and the causality condition (7) can help to reconstruct the buffer content estimation. It is important to note that (5) and (6) require data from the same interval to be compared. For this reason, it is advisable to include sequence numbers $i$ in the messages containing the $R_i$ values.

In case 3, the content $X_i$ can be estimated locally, e.g., from drain from a jitter buffer, which should be unproblematic in terms of availability of data. In case 4, the sender's average rate may be estimated in an incorrect way. However, given fact that the MSVA method works best for streaming-type services in view of vanishing loss, the risk for crude receiver-side estimations appears limited.

### 6.4 Desynchronization

Incorrect synchronization of sender and receiver can imply residual buffer contents or (noncausal) buffer under runs and thus biased (over- and under-) estimations of $T^{\text{add}}$. Eventual *time synchronization* problems are typically seen through a bias of the content of the equivalent bottleneck $X_i$, and thus of $T^{\text{add}}$, that is constant over time and that might be corrected by modifying $X_0$ as described in Section 2.3. *Frequency synchronization* problems manifest themselves in $\Delta T^{\text{in}} \neq \Delta T^{\text{out}}$, and eventually in values of $\Delta T$ that vary over time. Assume a relative deviation between $\Delta T$ values of $\varepsilon(\ll 1)$, then a drift of approximately $\varepsilon T^{\text{add}} \ll T^{\text{add}}$ would be observed per time interval. A mitigation strategy consists in estimating the $\Delta T$ value used by the sender of the measurements, and to adapt the $\Delta T$ of the receiver of the measurements. To help with the latter, the messages containing the $R_i$ values might contain time stamps that allow for estimating the duration of interval $i$ on the sender side.

A detailed investigation of the issues described in Sections 6.3 and 6.4 would motivate a quantitative study of its own.

### 6.5 Mobility

The mobility of a mobile user may result in horizontal or vertical network handovers. Such handovers may cause significant impacts on the performance for data communication by increasing the additional delay. Naturally, the latter is revealed by all five methods, which means that COIA allows to quantify the performance impact of mobility, both regarding coverage and control. Furthermore, during handovers, both data and measurement traffic might get lost; the latter situation needs to be dealt with as described in Section 6.3. Finally, a failed handover might imply a connection loss and, thus, naturally a detectable termination of the COIA measurements.

## 7 Conclusions and Outlook

In this paper, we propose five simple, yet powerful methods for the estimation of additional delay for an application-level data induced by heterogeneous data communication network behavior; e.g., radio link problems, protocol stack processing or queuing phenomena. The methods are based on both sender and receiver observed application-level data rates and rather simple calculations of additional delay. Both sender and receiver implement the COIA methods. The methods are based on lightweight exchange and comparison of throughput time series between sender and receiver. Using traces of four different network providers under a variety of traffic conditions for two different mobile applications, in particular transfer of health-care and multimedia data, each using different transport protocol for uploading or downloading of the data, we show that the most precise methods for additional delay estimation are based on the sender view on the content of the equivalent bottleneck and the current throughput of the sender. Furthermore, the pragmatic method that is assuming the maximum value of three sender- or receiver-view-based methods helps to yield conservative estimations of additional delays, especially for the TCP-based application. For the UDP-based applications that exhibit occasional packet loss, the receiver-view-based estimation relying on the expected throughput of the sender yields conservative estimation of additional delay. The conservative estimations for additional delay may be of particular interest for selected safety- or mission-critical mobile systems.

The estimation error of all methods depends on the size of the observation interval for the sender and receiver throughput time series. This is determined by the granularity (i.e., time resolution) of the events that need to be discovered, application instrumentation possibilities and capabilities of the mobile device to support both the application and its COIA instrumentation. Our results show that "oversampling," in terms of using an observation time interval smaller than the major additional delays to be observed, does not necessarily improve the quality of the estimation. Indeed, significant additional delay values can be observed and approximately quantified on timescales that are significantly larger than those of the additional delay phenomena themselves. This speaks in favor of the methods to be implemented in a constrained mobile environment, where the time interval choice and, hence, the frequency of such observations are limited by the resources of the mobile device. Here, the methods can help to quickly discover extraordinary additional delays and inform the application how late the data is. Thus, the methods can facilitate an implementation of lightweight monitoring and control loops for adapting an application's data stream to volatile network conditions, e.g., by compressing or even suppressing low-priority data.

Furthermore, as we have shown in this paper, all methods have different but complementary profiles with regards to their sensitivity and failure or success regarding the estimation of additional delays at different timescales. To maximize the accuracy of additional delay estimation, all methods could be run in parallel and be combined in a hybrid method. This hybrid method could continuously track the estimations provided by all methods and implement a type of voting mechanism to decide upon the value of additional delay and decide whether there is a peak or not, based on knowledge of the strengths and weaknesses of the methods with respect to their tendency to indicate peaks (i.e., their false positive or negative ratios).

Future work will address detailed investigations of the capability of the methods to detect arbitrary peaks of additional delay in other application domains. In this context, we will further analyze the impact of the observation interval size in relation to timescales used by the application. We will also address issues of systematic under- or overestimations of additional delays, undefined estimations, sender jitter, data loss time synchronization, and clock drifts between sender and receiver.

## Acknowledgments

Support of the EU-COST Action IC0703, the Swedish KKS project QoEMoS (Dnr 2008/0520), and Swiss National Science Founadation (SER-C08.0025, PBGEP2-125917) is acknowledged.

## References

[1] U. Hansmann, L. Merk, M. Nicklous, and T. Stober, *Pervasive Computing: The Mobile World.* Springer, 2003.

[2] "Definitions of Terms Related to QoS and Network Performance Including Dependability," ITU-T Recommendation E.800, 2008.

[3] "Vocabulary for Performance and QoS," ITU-T Recommendation P.10/G.100, 2007.

[4] D. Chalmers and M. Sloman, "A Survey of QoS in Mobile Computing Environments," *IEEE Comm. Surveys and Tutorials,* vol. 2, no. 2 pp. 2-10, Second Quarter 1999.

[5] L. Zhang, Z. Liu, and C. Xia, "Clock Synchronization Algorithms for Network Measurements," *Proc. IEEE INFOCOM,* 2002.

[6] J. Prokkola, P.H.J. Perala, M. Hanski, and E. Piri, "3G/HSPA Performance in Live Networks from the End User Perspective," *Proc. IEEE Int'l Conf. Comm. (ICC),* pp. 1-6, 2009.

[7] L. Kosten, "Stochastic Theory of a Multi-Entry Buffer," *Delft Progress Report*, vol. 1, pp. 10-18, 1974.

[8] D. Anick et al., "Stochastic Theory of a Data Handling System with Multiple Sources," *Bell System Technical J.*, vol. 61, pp. 1871-1894, 1982.

[9] A. van Halteren et al., "Mobile Patient Monitoring: The Mobihealth System," *J. IT in Healthcare*, vol. 2, no. 5 pp. 365-373, 2004.

[10] MobiHealth, http://www.mobihealth.com, 2012.

[11] S. Tachakra, X. Wang, R. Istepanian, and Y. Song, "Mobile E-Health: The Unwired Evolution of Telemedicine," *Telemedicine J. and E-Health*, vol. 9, no. 3 pp. 247-257, 2003.

[12] P. Carlsson, D. Constantinescu, A. Popescu, M. Fiedler, and A. Nilsson, "Delay Performance in IP Routers," *Proc. Int'l Working Conf. Performance Modelling Evaluation Heterogeneous Networks (HET-NETs)*, 2004.

[13] J. Nielsen, *Usability Engineering*. Morgan Kaufman, 1994.

[14] K. Wac, P. Arlos, M. Fiedler, S. Chevul, L. Isaksson, and R. Bults, "Accuracy Evaluation of Application-Level Performance Measurements," *Proc. EURO-NGI Conf. Next Generation Internet*, 2007.

[15] P. Arlos, "On the Quality of Computer Network Measurements," PhD thesis, Blekinge Inst. of Technology, School of Eng., 2005.

[16] K. Wac and R. Bults, "Performance Evaluation of a Transport System Supporting the MobiHealth BANip: Methodology and Assessment," MSc thesis, Univ. of Twente, the Netherlands, 2004.

[17] M. Claypool, "The Effect of Latency on User Performance in Real-Time Strategy Games," *Elsevier Computer Networks - Networking Issues in Entertainment Computing*, vol. 49, no. 1 pp. 52-70, 2005.

[18] M. Fiedler, K. Tutschku, P. Carlsson, and A. Nilsson, "Identification of Performance Degradation in IP Networks Using Throughput Statistics," *Proc. 18th Int'l Teletraffic Congress (ITC-18)*, 2003.

[19] M. Fiedler, L. Isaksson, S. Chevul, J. Karlsson, and P. Lindberg, "Measurement and Analysis of Application-Perceived Throughput via Mobile Links," *Proc. Performance Modelling Evaluation Heterogeneous Networks (HET-NETs)*, 2005.

[20] D.V. Lindley, "The Theory of Queues with a Single Server," *Proc. Cambridge Philosophical Soc.*, vol. 48, pp. 277-289, 1952.

[21] G. Hasslinger et al, "Variability of Broadband ADSL Traffic: Measurement and Time Slotted Modelling," *Proc. EURO-NGI Conf. Next Generation Internet (NGI)*, 2006.

[22] S. Aalto, "Output from an A-M-S Type Fluid Queue," *Proc. Int'l Teletraffic Congress (ITC)*, pp. 421-443, 1994.

[23] M. Fiedler and K. Tutschku, "Application of the Stochastic Fluid Flow Model for Bottleneck Identification and Classification," *Proc. SCS Conf. Design, Analysis and Simulation of Distributed Systems (DASD)*, pp. 35-42, 2003.

[24] J.D.C. Little, "A Proof for the Queuing Formula: $L = \lambda W$," *Operations Research*, vol. 9, no. 3 pp. 383-387, 1961.

[25] M.N. Ismail and A.M Zin, "Comparing the Accuracy of End-to-End Network Performance Measurement Testbed and Simulation Model for Data Transfers in Heterogeneous Environment," *Proc. Second Asia Int'l Conf. Modeling and Simulation (ICMS)*, pp. 124-131, 2008.

[26] C. Sarr and I. Guerin, "Estimating Average End-to-End Delays in IEEE 802.11 Multihop Wireless Networks," technical report, CITI Insa Lyon / INRIA Grenoble Rhône-Alpes - ARES, 2007.

[27] S. Borst, "Flow-Level Performance and User Mobility in Wireless Data Networks," *Philosophical Trans. Royal Soc.: Math., Physical and Eng.*, vol. 366, no. 1872 pp. 2047-2058, 2008.

[28] M.M. Barry, A. Campbell, and A. Veres, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks," *Proc. IEEE INFOCOM*, 2001.

[29] H. Graja et al, "A Statistical Estimation of Average IP Packet Delay in Cellular Data Networks," *Proc. IEEE Wireless Comm. Networking Conf. (WCNC)*, 2005.

[30] M. Garetto and D. Towsley, "Modeling, Simulation and Measurements of Queuing Delay under Long-Tail Internet Traffic," *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 31, pp. 47-57, 2003.

[31] G. Aniba and S. Aissa, "A General Traffic and Queueing Delay Model for 3G Wireless Packet Networks," *Proc. IEEE Int'l Conf. Telecomm. and Networking (ICT '04)*, pp. 125-137, 2004.

[32] B. Ngamwongwattana and R. Thompson, "Measuring One-Way Delay of VoIP Packets without Clock Synchronization," *Proc. IEEE Instrumentation Measurement Technology Conf. (I2MTC)*, 2009.

[33] M. Cola et al., "Covert Channel for One-Way Delay Measurements," *Proc. Int'l Conf. Computer Comm. Networks*, pp. 1-6, 2009.

[34] J. Kogel, "One-Way Delay Measurement Based on Flow Data: Quantification and Compensation of Errors by Exporter Profiling," *Proc. Int'l Conf. Information Networking*, pp. 25-30, 2011.

[35] K. Wac, M. Fiedler, R. Bults, and H. Hermens, "Estimations of Additional Delays for Mobile Application Data from Comparative Output-Input Throughput Analysis," *Proc. IEEE Network Operations Management Symp. (NOMS)*, pp. 171-178, 2010.

[36] N. Dokovsky, A. van Halteren, and I. Widya, "BANip: Enabling Remote Healthcare Monitoring with Body Area Networks," *Proc. Int'l Workshop Scientific Eng. Distributed Java Applications (FIJI '03)*, N. Guelfi, E. Astesiano, and G. Reggio, eds., pp. 62-72, 2004.

[37] "The Jini Technology Surrogate Architecture Overview," technical report, SUN Microsystems, 2001.

[38] R. Bults, K. Wac, A. van Halteren, D. Konstantas, and V. Nicola., "Goodput Analysis of 3G Wireless Networks Supporting M-Health Services," *Proc. IEEE Eighth Int'l Conf. Telecomm.*, 2005.

[39] P. Arlos and M. Fiedler, "Influence of the Packet Size on the One-Way Delay on the Down-Link in 3G Networks," *Proc. IEEE Int'l Symp. Wireless Pervasive Computing (ISWPC '10)*, pp. 573-578, 2010.

**Markus Fiedler** received the PhD degree in electrical engineering/ICT from Universität des Saarlandes, Saarbrücken, Germany, in 1998. Since then, he has been with Blekinge Institute of Technology (BTH), Karlskrona, Sweden. As a professor within the School of Computing and the head of the Communications and Computer Systems Research Lab at BTH, he performs and supervises research on quality of experience, seamless communications, network virtualization; service chains, and networks of the future. He leads and participates in several national and European projects. He serves on the steering board of the European Network of Excellence Euro-NF and coordinates its research activities. Furthermore, he is a cochair of the Future Internet Cluster of the EC. He is a member of the IEEE.

**Katarzyna Wac** received the BSc and MSc degrees in computer science, the MSc degree in telematics, and the PhD degree in information systems. Since 2010, she has led the Quality of Life Group at the Institute of Services Science of University of Geneva (Switzerland). From 2003-2004, she was a researcher at University of Twente, the Netherlands, and from 2004-2009, at the University of Geneva. From 2009-2010, she was on leave from Carnegie Mellon University. She researches pervasive mobile computing and communication in mHealth. She is a member of the IEEE.

**Richard Bults** received the BSc degree in technical computer science and the MSc (cum laude) degree in telematics. He is the cofounder and CTO of MobiHealth B.V., a company specialized in the development of (mobile) telemonitoring services for the healthcare market. He has more than 20 years of experience in performance analysis of (mobile) data communication networks. He researches methods and techniques for auto-adaptation of mobile health applications behavior based on performance measurements of data communication networks.

**Patrik Arlos** received the MSc degree in electrical engineering from Hogskolan i Karlskrona/Ronneby and the PhD degree in telecommunications systems from the Blekinge Institute of Technology in 2005. He is an assistant professor at the School of Computing, Blekinge Institute of Technology. His research interests lie in network measurements, network performance, quality of service, and experience. He is a member of the IEEE.