Adaptive Rate Control over Mobile Data Networks with Heuristic Rate Compensations

[1]Ke Liu, [1]Zhuang Wang, [2]Jack Y. B. Lee, [1]Mingyu Chen, [1]Lixin Zhang
[1] *State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences*
[2]*Department of Information Engineering, The Chinese University of Hong Kong*

***Abstract* — Mobile data networks exhibit highly variable data rates and stochastic non-congestion-related packet loss. These challenges result in key performance bottlenecks in current Transmission Control Protocol (TCP) implementations: bandwidth inefficiency and large end-to-end delay. This work addresses these challenges by first developing a Sliding Interval based Rate Adaptation (SIRA) that tracks bandwidths with a fixed time interval and applies them to its transmission rate periodically. Extensive experiments confirmed that SIRA achieves 96.3% bandwidth utilization and reduces the average queueing delay by a factor of 1.37, compared to TCP CUBIC, the preferred variant for Internet servers. However, the resultant end-to-end delay is still much larger for interactive applications, thus we complement SIRA with two heuristic rate compensation algorithms (SIRA-H) given that the bandwidth does not vary significantly in long time scales. Specifically, SIRA-H first reduces the transmission rate of SIRA if the estimated RTT is above a prefigured threshold. Meanwhile, it computes the amount of unsent data that would be transmitted if SIRA were used, and compensates the rate reduction with those unsent data as if their ACKs were received, when the queue is detected to be empty. We evaluated SIRA-H through a combination of trace-driven emulations and real-world experiments, and showed that it reduces the 95th percentile queueing delay by a factor of over 3.9, while maintains a similar throughput compared to the original SIRA. In comparison to state of the art protocols such as Sprout and Verus, SIRA-H also reduces the 95th percentile queueing delay by a factor of over 0.8.**

I. INTRODUCTION

Mobile data networks such as 3G and LTE demonstrate unique properties that are fundamentally different from those of wired networks, such as fluctuating bandwidths over short time scales and stochastic non-congestion-related packet losses. These difference results in severe performance bottlenecks [1-3] in TCP and its related variants [4-6]. To address this problem, mobile operators prioritize the TCP bandwidth efficiency by employing a *separate* large buffer [7-9] for every user at the radio link to absorb the bandwidth variations and packet losses, thus TCP could accumulate sufficient packets inflight hence resulting high bandwidth efficiency.

Having large backlog of queueing packets does improve the bandwidth efficiency, but it hurts the Quality of Service (QoS) of applications that requires low end-to-end delay to support interactivity, such as video conferencing (e.g., Skype, Google Hangout, etc.). Additionally, users are expected to perform multitasking more often, e.g., a user is on video call and downloading a file using TCP in the background at the same time. The delay-sensitive video call is very likely to experience large queueing delays resulted from the file download.

This work addresses this challenge of achieving both high throughput and low delay by first developing a Sliding Interval based Rate Adaptation (SIRA) tuned for mobile data networks. Specifically, SIRA periodically tracks the available bandwidth with a fixed time interval and update the sending rate accordingly. Theoretically, SIRA can be shown to guarantee the optimal bandwidth efficiency with a bounded link queue length. Extensive experiments confirmed that SIRA achieves 96.3% bandwidth utilization, while reduces the average queueing delay by a factor of 1.37, compared to TCP CUBIC, the preferred variant for Internet servers. However, the resultant end-to-end delay (i.e., 203.7 ms) is still much larger than the conventional standard for interactivity, i.e., every packet has 95% probability of clearing the queue within 100ms in [7].

The SIRA is designed under the assumptions that the radio link bandwidth over an *arbitrary* interval is bounded from above by the maximum bandwidth and from below by the minimum bandwidth, and varies in *arbitrary* manner from one interval to the next, thus prepares a sufficient buffer at the radio link so that it won't underflow even if under the worst case scenario – the bandwidth varies from the minimum to the maximum from the current interval to the next, which does occur if intervals are in short time scales (e.g., milliseconds) [7, 10], thus are non-trivial to predict using current predictors [10,11]. However, the bandwidths in longer time scales (e.g., seconds) do not vary substantially thus exhibit significant correlations [11]. We also find that the bandwidth does not change over 1Mbps in the time scale of 1s for over 94.5% of time of one day after investigating the bandwidth traces.

Inspired by this observation we relax the original assumptions by assuming that the bandwidths in long time scales do not vary dramatically, thus complement SIRA with two heuristic rate compensation algorithms (SIRA-H) that first reduces the transmission rate of SIRA when the estimated RTT is above a preconfigured threshold. In the meantime, it records the amount of unsent data that would be sent if SIRA were used according to the SIRA's model, then checks whether the queue underflow occurs due to the previous rate reductions with the estimated queueing length, and compensates the rate with those unsent data as if their ACKs were received if the queue underflow occurs.

We implemented SIRA and SIRA-H in a custom transparent proxy called *mobile accelerator* [12] to be located inside the mobile network so that all TCP traffic to/from mobile users passes through the device, which provides a *plug-and-play* platform to perform on-the-fly protocol optimizations. Because mobile operators commonly deploy

transparent proxies in their networks [13, 14], our approach is practical and can be readily deployed into current mobile data networks. We evaluated SIRA-H through a combination of trace-driven emulations and real-world experiments, and demonstrated that it reduces the 95th percentile queueing delay by a factor of 3.9 while maintains a comparable throughput compared to original SIRA. In comparison to the state of the art protocols such as Sprout [7] and Verus [10], SIRA-H also reduces the 95th percentile queueing delay by a factor of over 0.8.

The remainder of this paper is organized as follows. In Section II, we review the background and related works. Section III presents SIRA algorithms. Section IV introduces SIRA-H algorithms. Section V evaluates SIRA-H via trace-driven emulated experiments and real experiments over a production of mobile data networks and Section VI concludes this work.

II. Related Works

Prior to ours various approaches have been proposed to address the challenge that achieves both high bandwidth efficiency and low delay. Extend from comprehensive reviews [3, 7, 8, 10, 15] of existing works we classify them into *three* categories: active queue management (AQM), proxy-based and end-to-end based solutions (E2E). AQM such as RED [16] and CoDel [17] drops or marks queueing packets with congestion indicators such as the queue length and delay, and endpoints react to this signals before queueing lengths grow significantly, but they exploit accesses to or integration with routers, RNC, or Node-B, which are proprietary in mobile networks, thus AQM might not be deployed readily. E2Es generally employ a delay-based rate control algorithm such as TCP Vegas [6], Compound TCP [18], FAST TCP [19], Google Congestion Control (GCC) [20] and LEDBAT [21], that adjust the transmission rate with the delay information. On the other hand, some recent E2Es address the challenge through network learning: such as Sprout [7] Verus [10]. We evaluate them through emulated and real experiments for comparative study in Section V. All E2E or any protocol optimizations can be deployed and implemented in an intermediate network device that maintains compatibility with the TCP sender, the TCP receiver, or both, thus become proxy-based solutions such as [9, 12]. Compared to the other two approaches proxy-based solutions do *not* require any modification to the client/server implementations and the lower layers, and second distinguish users from both wired and mobile networks and apply appropriate protocol optimizations accordingly hence can be readily deployed in today's mobile data networks.

III. Sliding Interval Rate Adaptation

Our study was motivated by the fact that TCP is designed for any network hence tracks little underlying network characteristics, e.g., bandwidth variations, thus an intuitive approach to address the challenge that achieves both high throughput and low delay is to track bandwidths continuously and adapt them accordingly, thus we first invent a Sliding Interval based Rate Adaptation (SIRA) that periodically tracks the available bandwidth with a fixed time interval, and validate our intuition through trace-driven emulated experiments.

A. SIRA Algorithms

SIRA operates in two phases: the *startup phase* and the *adaptive phase*. Beginning with the startup phase, which lasts from the time that the first TCP data segment is transmitted to the time that the first ACK is returned from the receiver, SIRA sets the transmission rate to be a preconfigured rate X bps that is no lower than the possible maximum link capacity C_{max}, i.e., $X \geq C_{max}$, where C_{max} can be obtained from the maximum bandwidth advertised by mobile operators, e.g., 7.2Mbps for 3G/HSPA, 21Mbps for 3G/HSPA+ or a long-term bandwidth measurement at different locations. This phase serves two purposes. First, before any ACK packet returns, there is no information on the link bandwidth. The high initial transmission rate X *guarantees* the accuracy of the subsequent bandwidth estimations. Second, the high initial rate builds up packets at the radio link to provide the subsequent adaptive phase with sufficient buffered data to react to *any* bandwidth fluctuations.

After receiving the first ACK, SIRA enters the *adaptive phase*, which continuously tracks the bandwidth with a fixed time interval. Specifically, it divides time into small fixed intervals of Δ seconds in duration. SIRA begins transmission at rate of X and marks the time when the first ACK from the receiver is received as the beginning of the first interval and then performs bandwidth estimation at the end of each interval using the ACK arrival data of the past $M\Delta$ seconds – where M controls the *duration* of the sliding time interval.

Let c_i denote the bandwidth estimate of the i^{th} interval, r_j denote the arrival time of the j^{th} ACK at SIRA, and assume every ACK acknowledges the same length of data S. Then

$$c_i = \frac{\sum_{\forall j} \{S \mid (i-1)\Delta \leq r_j < i\Delta\}}{\Delta} \quad (1)$$

is the average data rate over the i^{th} interval.

Two cases are considered to determine the transmission rate for the i^{th} interval, denoted by R_i, where $i \geq 1$.

Case 1: $i \leq M$

This case is the initial condition case where the actual sliding window is shorter than M intervals, thus

$$R_i = \frac{X(M-(i-1))\Delta + \sum_{j=0}^{i-1} c_j \Delta}{M\Delta} \quad (2)$$

where we define $c_0 \equiv 0$ to simplify the notation. This step is equivalent to assuming a bandwidth of X for interval $i=1$ as well as those *imaginary* intervals before that.

Case 2: $i > M$

This case represents all subsequent cases with a full sliding interval of $M\Delta$ seconds. The transmission rate is set according to the average bandwidth of the sliding interval:

$$R_i = \frac{1}{M\Delta} \sum_{j=i-M}^{i-1} c_j \Delta \quad (3)$$

Table 1. Throughput-delay performances comparison for SIRA and TCP

Protocol	Queueing delay (ms)		Throughput
	Average	95th percentile	(Mbps)
CUBIC	483.1	895.7	6.77
Vegas	127.5	211.4	5.16
Reno	520.2	981.6	6.30
SIRA	203.7	363.3	6.67

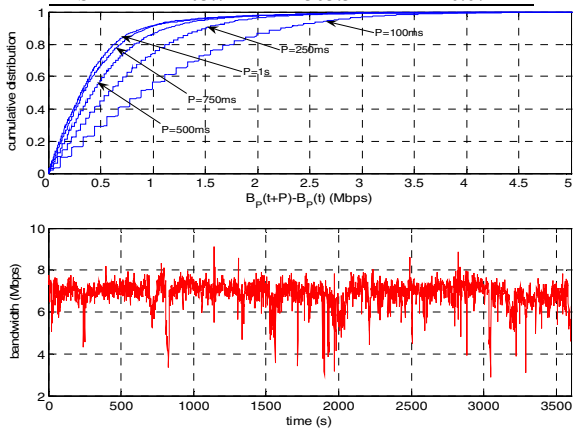

Fig. 1. (a) The cumulative distribution of $\lambda_P(t)$s that are less than 1Mbps with different P (up); (b) the bandwidth variations of a segment example (down).

SIRA as defined by (2) and (3) *guarantees* full utilization of the link bandwidth, the intuition is that with a sufficiently high initial transmission rate X, the radio link buffer will queue up sufficient data awaiting transmission such that subsequent bandwidth fluctuations, no matter how rapid, even under the worst case – changing from the minimum bandwidth denoted by C_{min}, where $C_{min}\geq 0$, in one interval to C_{max} in the subsequent intervals, will never deplete the queue with SIRA. On the other hand, the queueing length decreases with the increase in bandwidth, thus by average, SIRA results a smaller queueing length than TCP that always fills up the buffer regardless the bandwidth.

B. Performance Validation

In this section we validate our intuition by evaluating SIRA through emulated experiments over the testbeds depicted in Fig. 2 (a) (c.f., Section V.A), and compare its throughput-delay results to the ones of conventional TCP variants including TCP CUBIC, TCP Reno, TCP Vegas. The details of the experimental setup can be found in Section V.A. Table 1 summarizes the throughput-delay results for SIRA and the TCP variants. SIRA reduces the average queuing delay and 95th percentile queueing delay by at least a factor of 1.37, and 1.47 respectively, while achieves the comparable throughput performance, compared to TCP CUBIC and TCP Reno. TCP Vegas with its default parameters cannot efficiently utilize the bandwidth thus results in a lower queueing delay, which is resulted from the inappropriate parameter settings for our networks thus needs to be tuned them again to adapt a different network.

IV. Heuristic Rate Compensations

Although SIRA reduces the average queueing delay by a factor of at least 1.37 compared to conventional TCP variants, it still far away from the conventional standard for interactivity, i.e., every packet has 95% probability of clearing the queue within 100ms [7].

To further reduce the resultant end-to-end delay without degrading the bandwidth efficiency, we first find out that the bandwidth does vary in an arbitrary manner in short time scales, e.g., changing from C_{min} to C_{max} in milliseconds, but it does not vary significantly in long time scales thus manifest significant correlations, which agrees with [11] that shows that the throughput prediction in mobile networks is more accurate in a prediction horizon of seconds. Motivated by that observation we complement SIRA with two heuristic rate compensation algorithms (SIRA-H) that reduces the transmission rate if the estimated RTT is above a prefigured threshold instead of continuing to queue up sufficient data as SIRA. In the meantime, SIRA-H records the unsent data size that should be queued up to guarantee full bandwidth utilization if SIRA were used, and defers those data transmissions to the later intervals at which the queue is nearly empty.

A. Bandwidth Variations

To quantitatively demonstrate the bandwidth variations we first divide the bandwidth traces into fixed-length segments, i.e., 1 day. For each segment we measure the average bandwidth every fixed interval from time 0 to the end of the segment. Let P denote the periodicity; $B_P(t)$ denote the measured bandwidth over interval $[t–P, t]$, and a set of $B_P(t)$s can be derived for each segment. Then we compute the *absolute difference* between the bandwidths of any two adjacent intervals to characterize the bandwidth variations in the time scale of P, denoted by $\lambda_P(t)$, at time t, i.e., $\lambda_P(t)=B_P(t+P)–B_P(t)$, from which a set of $\lambda_P(t)s$ can be derived, thus the cumulative distribution of $\lambda_P(t)$s for a segment can be plotted to characterize the bandwidth variations for that segment. As shown in Fig. 1 (a), we plot the cumulative distributions of $\lambda_P(t)$s that are less than 1Mbps with different P from 100ms to 1s. We find that over 94.5% of $\lambda_P(t)$s that are less than 1Mbps when P=1s, i.e., the time scale is 1s, and over 56.3% of $\lambda_P(t)$s that are less than 1Mbps when P=100ms, which strongly suggests that the bandwidth does not vary in a long time scale quantitatively, e.g. 1s. Fig. 1(b) also qualitatively shows that the bandwidth variations of a segment do not vary significantly in long time scales.

B. SIRA-H Algorithms

SIRA-H operates in the adaptive phase of SIRA's rate control and further divides the adaptive phase into two states: the *delay adaptation* state and *rate compensation* state.

Delay Adaption State: SIRA-H enters the delay adaptation state if the estimated RTT exceeds a preconfigured threshold. Specifically, SIRA-H estimates the RTT from each received ACK by first subtracting the current time, i.e., the received time of that ACK, from the sent time of the TCP data packet that ACK acknowledges, which is then weighted by an Exponential Weighted Moving Average (EWMA). Let s_k denote the sent time of TCP packet k; r_k denote the reception time of the ACK that acknowledges the TCP packet k; d_k denote the RTT of TCP packet k, thus we have

$$d_k=\alpha d_{k-1}+(1-\alpha)(r_k-s_k) \quad (4)$$

where the smooth factor α=0.875.

Similar to SIRA, SIRA-H also adapts to the latest estimated RTT at the end of every Δ. Let d_i denote the latest RTT estimation at the end of the i^{th} Δ, where $i\geq 1$, i.e., $d_i=d_k$, given that $(i-1)\Delta<r_k\leq i\Delta$ and $r_{k+1}>i\Delta$; t_k denote the transmission time of packet k; thus D, the propagation delay, can be approximated by the minimum RTT estimated so far, i.e.,

$$\min\{d_k, k\geq 1\}=D+\min\{t_k, k\geq 1\}\approx D. \quad (5)$$

If $d_i>T+D$, where T denotes a preconfigured target queueing delay, SIRA-H reduces the transmission rate below the estimated bandwidth to prevent the queueing delay from increasing further instead of continuing to apply the transmission rate to the estimated bandwidth as SIRA. Let $R_{i,H}$ denote the transmission rate of SIRA-H for the i^{th} Δ, thus

$$R_{i,H}=\frac{T+D}{d_i}R_i \quad (6)$$

where R_i is defined in (2) and (3) thus $R_{i,H}$ also has two cases by substituting (2) and (3) into (6) respectively. Meanwhile, SIRA-H computes the unsent data size over the i^{th} Δ at the end of the i^{th} Δ for all i during the delay adaptation state, which would be sent and queued up if the SIRA's rate control were used. Specifically, SIRA-H first measures the actual amount of data that transmitted over the i^{th}Δ, denoted by σ_i. Because we cannot measure the total amount of data that would be transmitted during the i^{th} Δ if SIRA were used, we compute it according to the SIRA's rate control, i.e. $R_i\Delta$. Let u_i denote the unsent data size that would be transmitted if SIRA were used over the i^{th} Δ, thus we have

$$u_i=R_i\Delta-\sigma_i \quad (7)$$

where u_i also has two cases according to R_i. SIRA-H keeps track of u_i every Δ and adds them in a set, denoted by $\vec{u}$, implemented by a *circular buffer*. Similarly, let sn_i denote the sequence number of the last sent packet during the i^{th} Δ, $sn_i\geq 1$, we also add sn_i into a set denoted by $\vec{S}$ at end of the i^{th} Δ. If there is no packet transmitted during the i^{th} Δ, sn_i becomes the sequence number of the previous transmitted packet plus one that will be transmitted during later Δs. Both sets will be used in the rate compensation state at which $d_i\leq T+D$.

Rate Compensation State: SIRA-H breaks the SIRA's rate control by reducing the transmission rate below the estimated bandwidth, thus guarantees the accuracy of the subsequent bandwidth estimations with ACK returning rates. To address it, we invent the first heuristic rate compensation algorithm (H1) that checks whether the queue underflow occurs during the i^{th} Δ in the delay adaptation state, resulted from $R_{i,H}$ in (6), and then compensates it with the unsent data, i.e., u_is, as if they were absorbed by the radio link thus their ACKs were received, by simply adding them into the sliding time interval as normal ACKs, which could make the estimated bandwidth still valid.

To detect queue underflow during the i^{th} Δ, we adopt the SoD algorithm [22] to estimate the *queueing length* denoted by l_i ahead of packet sn_i when it arrives at the queue, which can achieve near-zero estimation accuracy [22]. After receiving the ACK for packet sn_i during the m^{th} Δ, where $m>i$, we have

$$l_i=\min\left\{x\left|\sum_{k=sn_i-x-1}^{sn_i-1}t_k>\left(d_{sn_i}-\min\{d_k,k\geq 1\}-t_{sn_i}\right),\forall x\geq 0\right.\right\} \quad (8)$$

If $l_i\geq L_q$, where L_q denotes the preconfigured threshold for l_i, and L_q=6 in our prototype, the queue underflow does not occur during the i^{th}Δ, thus the current transmission rate is only determined by the ACK returning rate from SIRA's rate control, i.e., $R_{m,H}=R_m$. If $l_i<L_q$, the queue underflow occurs during the i^{th} Δ, thus $R_{m,H}$ might be underestimated by the invalid ACK returning rate from SIRA, hence SIRA-H compensates R_m with the unsent data of the first L_u number of Δs, where $L_u\geq 1$, as if those unsent data were absorbed by the radio link thus their ACKs were received during the m^{th} Δ as normal ACKs, by simply adding the first L_u number of u_is in set $\vec{u}$ into the sliding time interval, i.e., $M\Delta$, thus we have

$$R_{m,H}=R_m+\sum_{j=1}^{\min(L_u,|\vec{u}|)}u_j/M\Delta \quad (9)$$

where u_j is the first j^{th} element in $\vec{u}$ and deleted from that set after (9), $|\vec{u}|$ is the current size of $\vec{u}$ and L_u=1 in our prototype, thus only the unsent data of one interval is used for rate compensations. Note that sn_i is deleted from $\vec{S}$ after receiving its ACK regardless the value of l_i, thus SIRA-H applies H1 until all sn_i are deleted from $\vec{S}$, or it enters the delay adaptation state, i.e., $d_i>T+D$, at which the residual unsent data of $\vec{u}$ during the previous delay adaptation state is added to a set denoted by $\vec{U}$.

SIRA-H also can stay at the rate compensation state when all sn_i are popped, which suggests that the compensated rate still cannot utilize the bandwidth efficiently, because H1 so far does not transmit all the unsent data in $\vec{u}$ that should be queued up in SIRA to absorb large bandwidth variations in long time scales, which is rare but does occur.

To address it, SIRA-H invents the second heuristic rate compensation algorithm (H2) that compensates the transmission rate with the residual unsent data. Specifically, let U_n denote the residual unsent data of $\vec{u}$ during the n^{th} delay adaptation state in $\vec{U}$ implemented by a FIFO queue, thus the oldest U_n is popped from $\vec{U}$ if it is used or the size limit of $\vec{U}$ exceeds a preconfigured limit denoted by L_U, where L_U=1 in our prototype, i.e., the residual data of the previous delay adaptation state is used in H2. SIRA-H first checks the queue length every RTT, if the number of Δs with zero queue length estimation exceeds α, e.g., 2/3, of the total number of Δs over that RTT, it spreads the unsent data size in $\vec{U}$ equally over the sliding time interval, as we cannot know which Δ the ACKs arrived in, thus at the i^{th} Δ, c_j in (1), where $\min(0, i-M)\leq j\leq i-1$, the estimated bandwidth during the j^{th} Δ in the current sliding time interval becomes

$$c_j=\frac{\sum_{\forall k}\{S\mid(j-1)\Delta\leq r_k<j\Delta\}+\sum_{U_n\in\vec{U}}U_n/M}{\Delta} \quad (10)$$

The transmission rates in (2) and (3) are compensated by substituting (10) into (2) and (3) respectively.

Table 2. Parameters and their default values used in SIRA and SIRA-H

Symbol	Description	Value
M	The duration of the sliding time interval	250
Δ	Update interval	2ms
X	The initial transmission rate	12Mbps
T	The target queueing delay	10ms
L_q	The threshold for queueing lengths	6
L_u	The number of the intervals for rate compensations	1
L_U	The size limit of $\vec{U}$	1

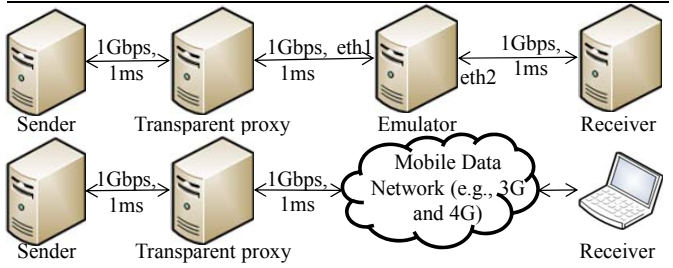

Fig. 2. (a) Emulated network testbed setup (up); (b) Real network testbed setup (down).

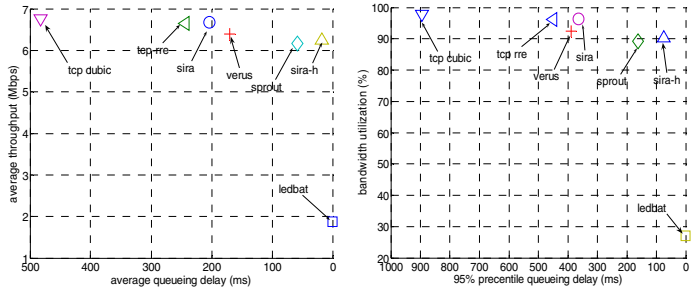

Fig. 3. (a) The average queuing delay and average throughput results (left); (b) the 95th percentile queueing delay and bandwidth utilization results (right) of every rate control over an emulated network.

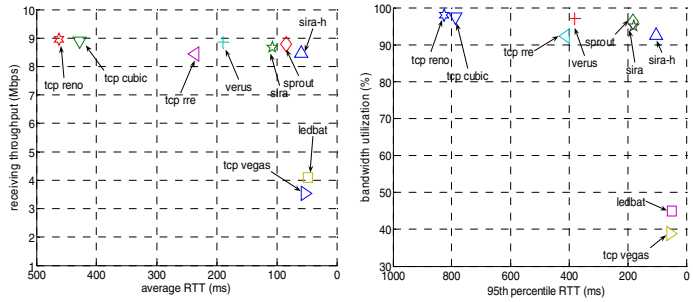

Fig. 4. (a) The average RTT and average receiving throughput results (left); (b) the 95th percentile RTT and bandwidth utilization results (right) of every rate control over a real 4G network.

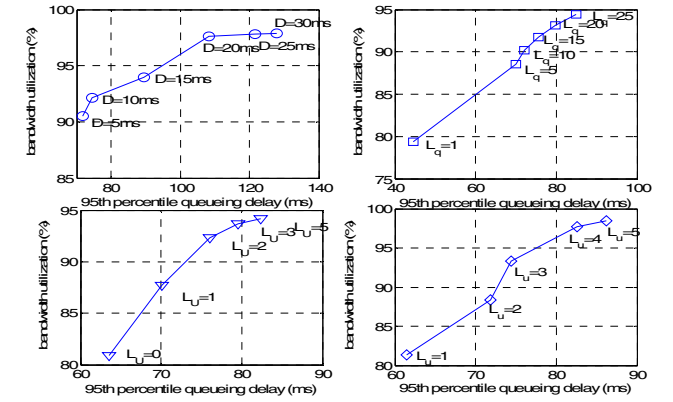

Fig. 5. (a) The sensitivity of D (up left); (b) the sensitivity of L_q (up right); (c) the sensitivity of L_U (down left); (d) the sensitivity of L_u (down right) on throughput-delay tradeoff evaluated through emulated experiments.

SIRA-H keeps compensating the rate using (10) until it transmits $\sum_{U_n \in \vec{U}} U_n$ more data than the case without H2. Similar to (7), the additional amount of data transmitted with (10) during the i^{th} Δ, denoted by a_i, can be derived as

$$a_i = \sigma_i - R_i \Delta \tag{11}$$

Thus SIRA-H stops compensating with (10) until the summation of a_i exceeds the data size of $\vec{U}$.

V. Performance Evaluations

We implemented SIRA and SIRA-H in a user-space transparent proxy called *mobile accelerator* [12] and compared their performances to existing approaches including the recent ones such as Sprout and Verus, by using an emulated and real network testbeds depicted in Fig. 2.

A. Network Testbed Setup

Fig. 2 (a) depicts the setup of the emulated network testbed, with the emulator node that is used to emulate the behavior of the mobile base station. To introduce the network variability of mobile data network into the emulated experiments, we built a custom bandwidth shaper with libpcap [23], which runs in the emulator node, to replay the bandwidth traces that are collected in 3 locations over a production 3G/HSPA+ network. The bandwidth traces last 285.4 hours, exhibiting bandwidth variations with an overall max/mean/min of 9.98, 7.95 and 2.85 Mbps. The sender and receiver in Fig. 2 (a) run Ubuntu Linux 12.04 LTS with kernel version 3.5, and the proxy and bandwidth shaper run Centos Linux 6 with kernel version 2.6. The proxy can be configured to switch on to use one of rate controls such as SIRA, SIRA-H and TCP-RRE, and off to forward packets as bridge, thus the rest of rate control algorithms running in the sender and receiver are applied. Fig. 2 (b) depicts the real network testbed over a production of a 4G network, with the receiver replaced by the HUAWEI E5 mobile data receiving terminal plugged into a notebook that runs Ubuntu Linux 12.04 LTS. The rest of the network nodes (without emulator) are the same ones in Fig. 2 (a).

Besides conventional TCP variants we also reproduce the following state of the art approaches for comparisons: Sprout [7], Verus [10], TCP-RRE [14] and LEDBAT [21].

B. Trace-driven Emulated Experiments

We ran trace-driven emulation to evaluate the delay-throughput performance of every rate control with the emulated network setup in Fig. 2 (a). The parameters for both SIRA and SIRA-H's implementations are set to the default values in Table 2. Because every rate control approach has its own configurations, we separately ran each approach over the same bandwidth traces for the same number of experiments that were running one after another, spaced by 5 s, and each experiment lasts for 20 s. The average bandwidth of that trace data is 6.92Mbps.

We plot the combination of the average throughput and queueing delay and the combination of the bandwidth utilization and 95th percentile queueing delay in Fig. 3 (a) and (b) respectively for every approach, which are measured and

collected at the bandwidth shaper in Fig. 2 (a). One point per approach on each figure is averaged from a total of over 4020 data points. As shown in Fig. 3, TCP CUBIC achieves the highest throughput, i.e., 6.77/6.92=97.8%, but has the largest delay in both average and 95th percentile one. SIRA-H achieves the lowest delay in both average one, i.e., 17.8ms, and 95th percentile one, i.e., 90.2ms, reducing the average queueing delay and 95th percentile queueing delay by a factor of 10.4 and 3.9 respectively, compared to the original SIRA, and a factor of 2.3 and 0.8 compared to the next-best performer Sprout that achieves 58.7ms in the average queueing delay and 161.8ms in the 95th percentile queueing delay, while the throughput of SIRA-H is only degraded by only 6% compared to SIRA (6.24Mbps vs. 6.67Mbps), and is comparable to the ones of Sprout and Verus.

C. Real Experiments over a Production of 4G Network

We repeated the experiments (c.f., Section V.B) over a production of a 4G network with a max/mean/min bandwidth of 4.36, 9.13 and 10.1Mbps using a real network testbed setup in Fig. 2 (b). Different from emulated experiments that can measure the relevant metrics at the bandwidth shaper, we measure the average receiving throughput at the receiver and measure the delay from a custom application transmitting data at a low data rate running concurrently with every rate control, thus the RTT experienced by that application can be used to characterize the delay performance of every rate control. Similarly, we present the combination of the average RTT and receiving throughputs and the combination of the 95th percentile RTT and bandwidth utilization respectively in Fig. 4 (a) and (b) for every approach, with each point on both figures is averaged from a total of over 200 data points.

Similar to Fig. 3, SIRA-H achieves the lowest average RTT, i.e., 59.2ms, and 95th percentile RTT, i.e., 103.3ms, which reduces the average RTT and 95th percentile RTT by a factor of 6.2 and 6.6 respectively, compared to TCP CUBIC, and a factor of and 0.4 and 0.8 respectively, compared to sprout, and achieves a comparable bandwidth utilization, 8.45/9.13=92.6%.

D. Parameters Validations

The throughput-delay tradeoffs of SIRA-H is govern by four parameters: threshold for queueing length estimations L_q, the target queueing delay D, the size limit of $\bar{U}$, L_U and the number of intervals for rate compensations L_u. To explore the sensitivity of each parameter that affects SIRA-H's delay-throughput tradeoff we vary one of the four parameters and evaluate SIRA-H over 2 hour bandwidth traces separately with the other parameters set to the default values.

Fig. 5 (a), (b), (c) and (d) plot the results of the bandwidth utilization and 95th percentile queueing delay of SIRA-H affected by D, L_q, L_U and L_u, respectively, all of which are averaged from a total of 50 data points. As expected, increasing one of the parameters allows SIRA-H to achieve a greater throughput by assuming the bandwidth might vary more significantly, but at cost of more delay.

VI. Conclusions

This work presented SIRA-H, a set of heuristic rate compensation algorithms that achieve low delay and high throughput over mobile data networks.

Acknowledgement

The research was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61502459, 61331008, and 61221062, the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA06010401.

References

[1] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling Bufferbloat in 3G/4G Networks," in *ACM IMC*, 2012.

[2] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of LTE: effect of network protocol and application behavior on performance," in *ACM SIGCOMM*, 2013.

[3] C. Chirichella, D. Rossi, C. Testa, T. Friedman and A. Pescape, "Passive Bufferbloat Measurement Exploiting Transport Layer Information," in *IEEE GLOBECOM*, 2013.

[4] S. Ha, I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *ACM SIGOPS Operating System Review*, 2008.

[5] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control," *RFC 2581*, 1999.

[6] L. Brakmo and L. Peterson, "TCP Vegas: End-to-End Congestion Avoidance on a Global Internet," *IEEE JSAC*, 1995.

[7] K. Winstein, A. Sivaraman and H. Balakrishnan, "Stochastic Forecasts Achieve High Throughput and Low Delay over Celluar Networks," in *USENIX NSDI*, 2013.

[8] F. Ren and C. Lin, "Modeling and Improving TCP Performance over Cellular Link with Variable Bandwidth," in *IEEE TMC*, 2011.

[9] R. Chakravorty, S. Katti, I. Pratt and J. Crowcroft, "Using TCP Flow-Aggregation to Enhance Data Experience of Cellular Wireless Users," in *IEEE JSAC*, 2005.

[10] Y. Zaki, T. Potsch, J. Chen, L. Subramanian and G. Gorg, "Adaptive Congestion Control for Unpredictable Cellular Networks," in *ACM SIGCOMM*, 2015.

[11] Y. Liu, Jack Y. B. Lee, "An Empirical Study of Throughput Prediction in Mobile Data Networks," in *IEEE GLOBECOM*, 2015.

[12] K. Liu and Jack Y. B. Lee, "On Improving TCP Performance over Mobile Data Networks," to appear in *IEEE TMC*, 2015.

[13] Y. Xu, W. K. Leong, B. Leong and A. Razeen, "Dynamic Regulation of Mobile 3G/HSPA Uplink Buffer with Receiver-Side Flow Control," in *IEEE ICNP*, 2012.

[14] W. K. Leong, Y. Xu, B. Leong and Z. Wang, "Mitigating Egregious ACK Delays in Cellular Data Networks by Eliminating TCP ACK Clocking," in *IEEE ICNP*, 2013.

[15] H. Balakrishnan, V. Padmanabhan, S. Seshan and R. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM TON*, 1997.

[16] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," in *IEEE/ACM TON*, 1993.

[17] K. Nichols and V. Jacobson, "Controlling queue delay," in ACM Queue, 2012.

[18] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A Compound TCP Approach for High-Speed and Long Distance Networks," in *IEEE INFOCOM*, 2006.

[19] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," in *IEEE/ACM TON*, 2006.

[20] L. De Cicco, G. Carlucci and S. Mascolo, "Experimental Investigation of the Google Congestion Control for Real-time Flows," in FhMN, 2013.

[21] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)," *RFC 6817*, 2012.

[22] Stanley C. F. Chan, K. Liu, K. M. Chan and Jack Y. B. Lee, "On Queue Length and Link Buffer Size Estimation in 3G/4G Mobile Data Networks," in *IEEE TMC*, 2013.

[23] Libpcap, Available: http://www.tcpdump.org/#latest-release.