Dynamic Video Bitrate Adaptation for WebRTC-Based Remote Teaching Applications

Stefano Petrangeli*, Dries Pauwels*, Jeroen van der Hooft*, Jürgen Slowack†, Tim Wauters*, Filip De Turck*
*Ghent University - imec, IDLab, Department of Information Technology
Technologiepark-Zwijnaarde 15, B-9052 Ghent, Belgium, email: stefano.petrangeli@ugent.be
†Barco N.V. – Technology Center, B-8500, Kortrijk, Belgium

***Abstract*—Remote teaching applications are common nowadays. Very often, these applications resemble video-on-demand streaming platforms rather than real virtual classrooms, where a group of students (the receivers) can remotely attend a live lecture held by a lecturer (the sender). To better support this live scenario, Real-Time Communication (RTC) solutions can be used. WebRTC is an open-source project for real-time browser-based conferencing, developed with a peer-to-peer architecture in mind. To use WebRTC, each receiver requires a dedicated encoder at sender-side. Using such approach is expensive in terms of encoders, and does not scale well for a large number of users. To overcome this issue, a WebRTC-compliant framework is proposed, where only a limited number of encoders are used. A centralized node, the conference controller, dynamically forwards the most suitable stream to the receivers, based on their bandwidth conditions. Moreover, the controller dynamically recomputes the encoding bitrates of the sender. This approach allows to closely follow the long-term bandwidth variations of the receivers, even with a limited number of encoders at sender-side. To evaluate the performance of the proposed framework in a realistic environment, a testbed has been implemented using the Chrome browser and the open-source Jitsi-Videobridge. In a scenario with 10 receivers and 3 encoders, and under realistic network conditions, the proposed framework improves the received video bitrate up to 11%, compared to a static solution where the encoding bitrates do not change over time.**

I. Introduction

Video streaming is everywhere and has radically changed the way we spend our free time and interact among each other. Not only is streaming used for pure entertainment, it also plays a fundamental role in making knowledge accessible to everyone on the globe. For example, Coursera and Udacity are among the most famous websites to stream high-quality online courses [1]. Despite that, current remote teaching platforms are actually closer to traditional video-on-demand platforms. In a real *virtual classroom* scenario, the students (or *receivers*) are remotely attending a live lecture given by the lecturer (the *sender*). The receivers are usually geographically distributed and can experience different bandwidth and network conditions. Remote conferencing solutions can be used to implement such a live scenario, which is usually characterized by a high degree of interactivity [2]. Particularly, the Web Real-Time Communication (WebRTC) framework is an open-source project started by Google in 2011 that provides plugin-free real-time communication capabilities to browser-based applications [3]. The WebRTC framework has been developed with a peer-to-peer architecture in mind, where a small group of clients can directly communicate with each other. This approach can suffer from scalability issues when multiple participants are present at the same time, as in the virtual classroom scenario. In such a peer-to-peer architecture, the WebRTC sender would need to encode a separate stream for each of the WebRTC receivers attending the virtual classroom.

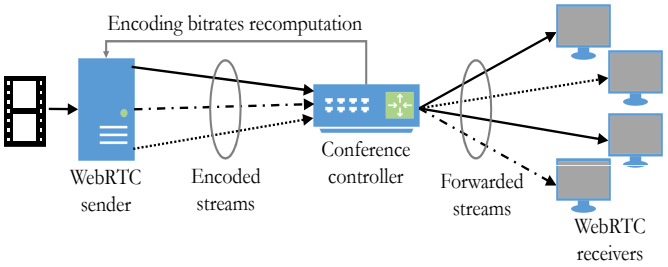


Fig. 1: The conference controller is the terminal endpoint for both the sender and the receivers, and performs the dynamic stream forwarding and dynamic bitrate recomputation tasks.

In light of the above, we propose a WebRTC-compliant framework to support the delivery of real-time communication streams in a remote teaching scenario (see Figure 1). In this framework, the WebRTC sender only needs to encode a limited number of streams, much smaller than the number of receivers, at different bitrates. This approach allows to overcome the aforementioned limitation, where each receiver would need to be associated to an independent, dedicated encoder. In our framework instead, multiple receivers are assigned to the same encoder at sender-side. A centralized node, called the *conference controller*, is aware of the bandwidth conditions of the WebRTC receivers and dynamically forwards the stream at the best bitrate in order to follow the bandwidth variations of the receivers. Besides this dynamic stream forwarding, the conference controller has another fundamental task. Instead of keeping the encoding bitrates of the sender fixed to predefined static values, the conference controller can dynamically and periodically recompute them based on the changing bandwidth conditions of the receivers. This approach allows to better follow the bandwidth conditions of the receivers, even though only a limited number of encoders is actually used. In the WebRTC domain, the conference controller functionalities can be easily carried out by a Selective Forwarding Unit (SFU), whose task is to receive all the streams and decide which

stream should be sent to which participant [4].

The contributions of this paper are two-fold. First, we present in detail the proposed framework and the conference controller functionalities. Particularly, we model the bitrate recomputation problem as an Integer Linear Programming (ILP) formulation, which is periodically solved by the centralized node. Second, we present a real-world testbed, implemented using the WebRTC framework and Jitsi-Videobridge [5], to evaluate the performance of the proposed framework. Preliminary results illustrate the benefits of the developed solution.

The remainder of this paper is structured as follows. Section II presents related work on remote conferencing solutions for WebRTC. Section III describes the functionalities of the conference controller and the ILP formulation for the dynamic encoding bitrates recomputation, while Section IV details the implementation of the real-world testbed. Preliminary results are shown in section V. Section VI concludes the paper and discusses future work.

II. Related Work

Xu et al. perform a measurement study on real-world conferencing systems [6]. The authors report that a complete peer-to-peer architecture is never used as it does not scale to a large number of users. A Multipoint Conferencing Unit (MCU) can be used in WebRTC to improve scalability. The MCU receives all the streams, decodes and composes them in a single common stream that is sent back to the participants. Grand et al. divide the participants into regional clusters, each associated to an MCU [7]. All the MCUs are interconnected in a mesh network. This hybrid architecture allows to support a large number of users. Ma et al. use an MCU to transcode the sender stream and adjust it to the viewing conditions of the receiver. The authors consider the viewing distance and the pixel density of the receiver's screen to transcode the stream to an optimal bitrate, in order to save bandwidth. Nevertheless, MCU operations are extremely computationally intensive, due to the decoding-mixing-encoding processes to be carried out. To reduce this issue, MCU functionalities can be dynamically migrated among conference participants to meet certain bandwidth, latency and CPU constraints [8]. Alternatively, MCU low-level functionalities can be virtualized and deployed on-the-fly [9]. Unlike an MCU, an SFU does not require decoding/encoding operations. Its main task is to receive all the streams and selectively forward one or more streams to each participant. In this case, the amount of forwarded streams should be selected to avoid wasting bandwidth. Grozev et al. develop a speaker identification algorithm to be deployed on an SFU, to identify the last *N* dominant speakers of the conference [10]. Only these *N* streams are forwarded to the conference participants. The authors also propose to use simulcast in combination with an SFU [11]. Each participant can send up to three streams, encoded at different bitrates. The SFU forwards the highest quality to participants involved in the conversation, and the lowest quality to the remaining ones. In a measurement study, Xhagjika et al. find that the load pattern on an operational system of SFUs is periodic and can be easily predicted [12]. The prediction can be used to allocate the streams to the right SFU and avoid overloading the system. The software-defined networking principle can be used to optimize a system of distributed SFUs [13], by dynamically creating a multicast tree for the optimal delivery of the streams.

Unlike previous works, the proposed conference controller, implemented using the SFU principle, forwards the most suitable stream based on the bandwidth conditions of the receivers. Moreover, the dynamic computation allows the encoding bitrates to be always representative of the network conditions of the receivers, and therefore maximize the received bitrate.

III. WebRTC Conference Controller

The conference controller is the main component of the proposed WebRTC framework for remote teaching applications. The goal of the controller is to maximize the video bitrate sent to the receivers, given the constraint on the available bandwidth. Two functionalities are implemented by the controller, namely dynamic stream forwarding and dynamic encoding bitrates computation.

The controller acts as an endpoint for the sender and for the receivers (see Section IV for more details). This aspect entails that the controller can intercept the WebRTC Receiver Estimated Maximum Bitrate (REMB) messages, which are used in WebRTC to estimate the bandwidth of the remote receivers [14]. Using this bandwidth estimation, the conference controller forwards the stream to each receiver at the highest sustainable bitrate. This dynamic forwarding is performed at a very fine-grained timescale, in the order of 250 to 500 ms, to accommodate the bandwidth variations of the receivers and guarantee a continuous playback. This period mainly depends on the timing of REMB messages reported by the receivers, which are used to update the bandwidth estimation.

Together with this short-term control loop, a second long-term optimization is performed on a longer timescale, to recompute the set of encoding bitrates at sender-side. By allowing the encoding bitrates to dynamically vary, it is possible to follow the long-term bandwidth variations of the receivers and, therefore, maximize the delivered video quality. A shorter period is desirable, but is more computationally expensive when the number of receivers is high. We formulate the dynamic bitrate computation problem as an ILP formulation. The virtual classroom is composed of R receivers and one sender, where l_{max} encoders are available (with $l_{max} \ll R$). Each receiver $r \in \{1, \ldots, R\}$ is associated with a bandwidth measure b_r. Different options are possible on how to compute b_r for a given time window of multiple seconds, as for example the average or the minimum estimated bandwidth over the period. In this work, b_r is simply equal to the latest bandwidth estimation available to the conference controller. We also assume a set of L encoding levels are available, each associated to a nominal bitrate B_l. The goal is to select the l_{max} encoding levels, among the possible L levels, which are the closest to the bandwidth measure b_r of the receivers. The optimization problem, which is executed periodically by the controller, is given in the following:

$$
\begin{aligned}
\min_{\alpha} \quad & \sum_{r=1}^{R}\sum_{l=1}^{L} \alpha_{r,l}(b_r - B_l)^2 & & \\
\text{s.t.} \quad & \alpha_{r,l} \in \{0,1\} & & \forall r,l \\
& \beta_l \in \{0,1\} & & \forall l \\
& \beta_l \geq \alpha_{r,l} & & \forall l \\
& \sum_{l=1}^{L} \alpha_{r,l} = 1 & & \forall r \\
& \sum_{l=1}^{L} \beta_l \leq l_{max} & & \\
& \sum_{l=1}^{L} \alpha_{r,l} B_l \leq b_r & & \forall r
\end{aligned}
\tag{1}
$$

The solution of the problem is characterized by two sets of boolean decision variables, namely $\alpha_{r,l}$ and β_l. $\alpha_{r,l}$ is equal to 1 when client r is associated to encoding level l, and 0 otherwise. Similarly, β_l is equal to 1 when encoding level l is selected, and 0 otherwise. The optimization problem tries to find the l_{max} encoding levels whose bitrates allow to minimize the quadratic difference with the receiver bandwidth estimations. The first three constraints set up a consistent relation between the decision variables α and β. The fourth constraint indicates that each receiver can only be associated with one specific encoding level. The last two constraints are representative of the analyzed problem. First, only l_{max} encoding levels can be selected (constraint 5), as l_{max} indicates the maximum number of encoders available at sender side. Second, the encoding bitrate associated to receiver r must be lower than the bandwidth measure for r (constraint 6), in order to guarantee a continuous playout.

It is worth noting that the two tasks carried out by the conference controller are performed at different timescales. The bitrate computation presented in Equation 1 is executed on a tens of seconds timescale, and is used to adjust the encoding bitrates to take into account long-term variations of receivers' network conditions. On the contrary, the stream forwarding is executed on a millisecond timescale, to closely follow the changing bandwidth conditions of the receivers.

IV. Testbed implementation

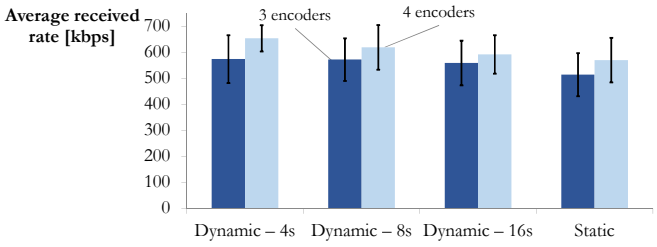
In order to evaluate the performance of the proposed solution in a realistic environment, we implemented our framework on the imec iLab.t Virtual Wall emulation platform [15]. To implement the receivers and the sender, we use the Google Chrome browser. Nothing was changed of the Google's original WebRTC stack, which makes our solution completely WebRTC-compliant. From an implementation perspective, the sender is decoupled into l_{max} WebRTC sub-senders, each encoding the stream at different bitrates. To implement the conference controller, we used the Jistsi software, a set of open-source projects to easily build and deploy secure video-conferencing solutions [16]. Particularly, the Jitsi-Videobridge, a WebRTC SFU, has been used as the main component. Its default functionality is to relay all the streams generated by the conference to all the participants. Jitsi-Meet, a JavaScript application running on top of the browser WebRTC stack, is used at the sub-senders and receivers to interface with the Jitsi-Videobridge [17]. We modified the code so that the receivers do not generate any video stream. In the remaining of this section, we will explain the modifications we made on the Jitsi-Videobridge to implement the stream forwarding and bitrate recomputation functionalities. To implement these functionalities, the conference controller has to estimate the available bandwidth of the receivers first. This estimation is performed by default by the Jitsi-Videobridge, which forwards all WebRTC traffic among the conference participants, implemented using the RTP/RTCP protocol suite. In WebRTC, bandwidth estimation is performed using RTCP REMB messages, a feedback used by a receiver to notify its media stream sender over the same RTP session of the estimated available bandwidth on the path to the receiving side. The Jitsi-Videobridge intercepts these RTCP REMB messages and is therefore aware of the available bandwidth at the receivers.

A. Stream Forwarding Selection

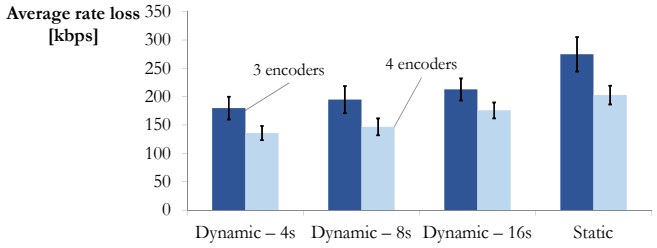
By default, the Jitsi-Videobridge forwards multiple streams to a participant. The stream of the participant who is currently speaking, the so-called *dominant speaker*, is automatically detected by the software and is always included in these streams. We override this logic so that a different *dominant speaker* can be manually set per receiver. Moreover, we limited the amount of streams that can be sent to a specific receiver to only one, selected as previously explained. Using this mechanism, the Jitsi-Videobridge dynamically assigns a sub-sender per receiver, so that the encoding bitrate is lower than the receiver's estimated bandwidth.

B. Dynamic encoding bitrates recomputation

As the long-term network conditions of the receivers can change over time, it is required to periodically recompute the set of encoding bitrates of the sub-senders, as described in Section III. Once these values are computed, they have to be enforced on the WebRTC sub-senders. However, there is no standardized way to set the encoding bitrate of a WebRTC client. To perform this task, we use the RTCP REMB messages, which contain the receiver's estimated available bandwidth. In WebRTC, the congestion control mechanism of a sender considers this estimation as the maximum bitrate that can be sent to a receiver. Consequently, the sender's encoder uses this value as its current target bitrate. Once the new bitrates are computed, the Jitsi-Videobridge modifies the REMB feedback messages for the sub-senders by setting the newly computed bitrate instead of the bandwidth estimation of the receivers. This way, the sub-senders are forced to modify their encoding bitrates. To implement this mechanism, we changed the way RTCP messages are generated in lib-jitsi [18], the underlying Java media library used by Jitsi-Videobridge. Instead of setting the maximum bitrates in the REMB messages for the sub-senders to the latest estimated remote bandwidth, we set them to the bitrates generated by the ILP presented in Section III.



(a) Average received rate



(b) Average rate loss

Fig. 2: Dynamically recomputing the encoding bitrates allows to both increase the received rate and reduce the error compared to a static association. Results are reported for the static and dynamic associations with several optimization periods.

V. Performance Evaluation

We evaluate the performance of the proposed framework using the testbed presented in Section IV. The virtual classroom is composed of ten receivers and one sender (Figure 1). Depending on the experiments, the sender is decoupled into three or four sub-senders, each encoding the video at different bitrates (i.e., l_{max} can be equal to three or four). No real video capture is carried out. Instead, we use a predefined video as input for the encoders[1]. Two parameters have to be defined for the conference controller. First, the period of the optimization executed in Equation 1, which has been fixed to 4, 8 and 16 seconds. A shorter optimization period allows to better follow the network conditions of the receivers, but it is more computationally expensive. Second, the number of possible encoding levels L to be used in the optimization problem, with the corresponding bitrates. Nineteen different levels are used, with bitrates equal to $250 + 125l$ kbps, with $l = 0, 1, \ldots, 18$. Consequently, the lowest and highest encoding bitrates are 250 and 2500 kbps, respectively. To evaluate the approach under realistic network conditions, we apply a different bandwidth pattern on each link connecting the conference controller to the receivers, collected on a real 3G network [19]. Each experiment configuration, in terms of optimization period and number of encoders, has been repeated five times.

We compare the performance of the proposed framework to a solution where the encoding bitrates are statically fixed and are not dynamically recomputed during the experiment. The static bitrates are equidistantly assigned to $250 + \frac{2500-250}{l_{max}-1} l$ kbps, with $l = 0, 1, \ldots, l_{max} - 1$. For each receiver, we keep track of two metrics. First, the average video rate received during the experiment. Second, the average *rate loss*, computed as the difference between the available bandwidth at the receiver and the actual received rate. The rate loss represents the gap between the rate a receiver is able to sustain (i.e., the available bandwidth) and the rate actually received. Figure 2 reports the results of this investigation, averaged over the entire group of receivers with the corresponding standard deviations.

In all cases, increasing the number of encoders allows to increase the received rate and reduce the error, up to 10% and 35% in the static case. When more encoders are used, it is possible to follow the bandwidth variability of the receivers in a more fine-grained way. This behavior is clear when the encoding bitrates are dynamically recomputed. In the case with an optimization period of four seconds and three encoders, the received rate is increased by 11%, while the rate loss is reduced by 35%. Similar conclusions can be drawn when using four encoders instead. The dynamic recomputation allows to adjust the encoding bitrates to the network conditions of the receivers. This aspect entails that the encoding bitrates are usually closer to the actual bandwidth conditions of the receivers, which improves the analyzed metrics. As expected, a longer optimization period degrades the performance of the system. When compared to a 4 seconds optimization period, the average rate loss increases by 5% and 12% when the period is set to 8 and 16 seconds, respectively. The relevance of the encoding bitrates tends to decrease over time, as the long-term bandwidth conditions of the receivers might change. A shorter period can reduce this side effect, but it is more computationally expensive, especially when the number of receivers is large. Even in the 16 seconds case though, a dynamic approach is able to outperform a static one. Interestingly, using a dynamic approach with three encoders leads to a similar received rate and rate loss compared to a static approach with four encoders. The dynamic framework can reach the same performance of a static association while reducing the number of encoders, and is therefore more efficient.

VI. Conclusions

In this paper, a framework has been proposed for the efficient delivery of WebRTC streams in the context of remote teaching applications. At sender-side, only a few encoders are used, which allows to scale the proposed approach to a large number of receivers and reduce the encoding costs. A conference controller, implemented using the Jitsi-Videobridge software, periodically recomputes the set of encoding bitrates to better follow the network conditions of the receivers. Besides this long-term optimization, the controller also dynamically forwards the most suitable stream to each receiver, to accommodate fast bandwidth variations and ensure a continuous playout. Compared to a static, fixed association of the encoding bitrates, the proposed dynamic bitrate recomputation results in 11% higher video rate at the receivers, when three encoders are used. Moreover, the dynamic recomputation is more efficient than a static approach, as less encoders are needed to obtain similar performance. Future work will focus on large-scale evaluations of the proposed framework to test the scalability of the ILP formulation and on machine learning algorithms to perform the bitrate recomputation task.

[1] https://media.xiph.org/video/derf/y4m/factory_1080p30.y4m

ACKNOWLEDGEMENT

Jeroen van der Hooft is funded by grant of the Agency for Innovation by Science and Technology in Flanders (VLAIO). This research was performed partially within the imec PROFLOW project (150223).

REFERENCES

[1] Coursera, "Online courses from top universities." [Online]. Available: https://www.coursera.org

[2] J. Jang-Jaccard, S. Nepal, B. Celler, and B. Yan, "Webrtc-based video conferencing service for telehealth," *Computing*, vol. 98, no. 1, pp. 169–193, Jan 2016. [Online]. Available: https://doi.org/10.1007/s00607-014-0429-2

[3] S. Loreto and S. P. Romano, "How far are we from webrtc-1.0? an update on standards and a look at what's next," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 200–207, 2017.

[4] B. Grozev, L. Marinov, V. Singh, and E. Ivov, "Last n: relevance-based selectivity for forwarding video in multimedia conferences," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2015, pp. 19–24.

[5] Jitsi Videobridge, "Jitsi videobridge, is webrtc compatible video router or sfu." [Online]. Available: https://github.com/jitsi/jitsi-videobridge

[6] Y. Xu, C. Yu, J. Li, and Y. Liu, "Video telephony for end-consumers: Measurement study of google+, ichat, and skype," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 826–839, June 2014.

[7] J. C. Granda, P. Nuño, F. J. Suárez, and D. F. García, "Overlay network based on webrtc for interactive multimedia communications," in *2015 International Conference on Computer, Information and Telecommunication Systems (CITS)*, July 2015, pp. 1–5.

[8] M. A. Hossain and J. I. Khan, "Distributed dynamic mcu for video conferencing in peer-to-peer network," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, Dec 2016, pp. 1–8.

[9] P. Rodríguez, Álvaro Alonso, J. Salvachúa, and J. Cerviño, "Materialising a new architecture for a distributed mcu in the cloud," *Computer Standards and Interfaces*, vol. 44, no. Supplement C, pp. 234 – 242, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920548915001014

[10] B. Grozev, L. Marinov, V. Singh, and E. Ivov, "Last n: Relevance-based selectivity for forwarding video in multimedia conferences," in *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '15. New York, NY, USA: ACM, 2015, pp. 19–24. [Online]. Available: http://doi.acm.org/10.1145/2736084.2736094

[11] B. Grozev, G. Politis, E. Ivov, T. Noel, and V. Singh, "Experimental evaluation of simulcast for webrtc," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 52–59, 2017.

[12] V. Xhagjika, . D. Escoda, L. Navarro, and V. Vlassov, "Load and video performance patterns of a cloud based webrtc architecture," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2017, pp. 739–744.

[13] E.-z. Yang, L.-k. Zhang, Z. Yao, and J. Yang, "A video conferencing system based on sdn-enabled svc multicast," *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 7, pp. 672–681, Jul 2016. [Online]. Available: https://doi.org/10.1631/FITEE.1601087

[14] H. Alvestrand, "Rtcp message for receiver estimated maximum bitrate," *Internet-Draft draft-alvestrand-rmcat-remb-03 (work in progress)*, 2013.

[15] imec, "Virtual wall emulation platform." [Online]. Available: http://doc.ilabt.iminds.be/ilabt-documentation/virtualwallfacility.html

[16] Jitsi, "Jitsi, an open-source projects to build and deploy secure videoconferencing solutions." [Online]. Available: https://jitsi.org/

[17] Jitsi Meet, "Jitsi meet - secure, simple and scalable video conferences." [Online]. Available: https://github.com/jitsi/jitsi-meet

[18] Libjitsi, "Advanced java media library for secure real-time audio/video communications." [Online]. Available: https://github.com/jitsi/libjitsi

[19] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 8, no. 3, pp. 24:1–24:19, Aug. 2012.