A Practical System towards the Secure, Robust and Pervasive Mobile Workstyle

Zhan Ma, Tao Yue, Xun Cao, Yiling Xu, Xin Li, Yongjin Wang

***Abstract*—We develop an innovative PC2PC (personal computer to pervasive computing) system to enable the secure, robust and pervasive mobile workstyle. PC2PC server compresses the desktop screens of any virtualized system, and delivers the stream through any popular networks to PC2PC client remotely for stream decoding, rendering and end-user interaction (such as keyboard/mouse commands).**

We have implemented the overall system from the scratch, where the emerging screen content coding (SCC) extension of the High-Efficiency Video Coding (HEVC) is implemented to compress and stream the desktop screens in real-time, and three core asset channels (i.e., system, display, inputs, etc) are defined to enable systematic end-to-end communication. Compared with the commercial Red Hat SPICE virtual desktop infrastructure (VDI) scheme, our PC2PC could save the network bandwidth by a factor of 2, 7 and 4 respectively for typical video streaming, web browsing and stationary office applications at same visual quality. Meanwhile, we have also measured the delays in the system and presented the preliminary study on the user experience impact. A simple network estimation is applied to optimize the quality-bandwidth adaptation for both single user and multiuser scenarios to combat the network dynamics.

***Index Terms*—Pervasive computing, virtual desktop infrastructure, high-efficiency video coding, screen content coding**

I. Introduction

Mobile workstyle is an emerging office fashion that allows employees could manage and operate their daily tasks remotely. Such mobile workstyle is supported by the *virtual desktop infrastructure* (VDI) where routine desktop operating systems (OSs) are hosted at centralized server (i.e., private or public cloud) to provide the ubiquitous computing and information storage. It enables people to strike a better work-life balance, able to work from anywhere, unrestricted by location, time or device [1]. Meanwhile, it also provides the effective way for information security where the information processing and storage could be managed inside the server with appropriate protection.

Major industry players in the area of information technology (IT) actively embrace the mobile workstyles and have developed several well-known platforms to enable such service, for instance, VMware Horizon [2], Citrix XenDesktop [3], Microsoft Remote Desktop Protocol (RDP) [4], Red Hat Simple Protocol for Independent Computing Environments (SPICE) [5], etc. Almost all of these popular VDI technologies follows the same procedure, i.e., instantaneous desktop screen is first analyzed to identify different regions, such as text, icon, graphics, images etc, each region will be processed individually using different (compression) methods for network delivery. It requires the complicate protocol to maintain the communications between the client and the remote server. Because of the noticeable involvements from the client for desktop rendering, it is also referred as the "client rendering" solution.

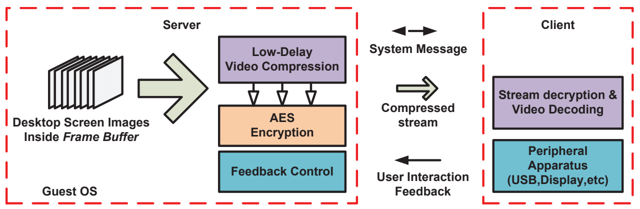

Server
Low-Delay Video Compression
AES Encryption
Feedback Control
Desktop Screen Images Inside *Frame Buffer*
Guest OS
System Message
Compressed stream
User Interaction Feedback
Client
Stream decryption & Video Decoding
Peripheral Apparatus (USB,Display,etc)

Fig. 1: Illustration of our proposed PC2PC architecture and implementation

Due to the advances in communication (such as 4G/LTE, emerging 5G) and mobile computing capability, we envision the massive market size of the mobile workstyle through the underlying VDI technologies. However, almost all existing technical solutions are not suitable very well due to the complicate "client rendering" protocols. In this work, we propose a complete "server rendering" solution where server renders and compresses the instantaneous desktop screen, and client only decodes the compressed screen stream and sends back very few mouse and keyboard commands for user interaction.

Our focus in this work is to enable the interaction between guest OSs and remote clients. Overall PC2PC system is illustrated in Figure 1. A real-time (and simplified) HEVC based SCC based low-delay video compression engine is applied to encode instantaneous desktop screens of virtualized guest OS at server side. This is referred as the **Display Channel** to deliver the compressed streams. To enable the systematic end-to-end communication, we have also defined other two asset channels, such as **System Channel** to convey the messages so as to initiate, maintain/monitor and destroy the communication channel and **Input (feedback) Channel** to collect the user interaction commands (i.e., keyboard, mouse positions, etc from client side. At this moment, TCP is used to manage the stream delivery between the server and the client.

The reminder of this paper is organized as follows: Section II details the PC2PC architecture and its implementation. Comparative study is performed in Section III to demonstrate the performance of proposed system and finally we will conclude this work in Section IV.

II. Personal Computer to Pervasive Computing (PC2PC): Architecture and Implementation

A. PC2PC Architecture

We build up this baseline PC2PC framework from the scratch with three core asset channels to fulfill the inevitable functionalities. System channel is used to deliver the message for communication initialization, maintenance/monitoring and destroy; SCC is used for screen compression in display channel while input channel offers the capability to manage the end user interaction. Other channels could be extended in future to support other functionalities, such as USB, network sharing folder, printer, etc.

Note that with the proposed PC2PC implementation, we provide the complete "server rendering" solution in comparison to the current client rendering methods. Client could be an ultra-light-weight device with just video stream decoding and user interaction capabilities. Of course, client could be hosted at any popular Android or iOS powered device to leverage the ecosystem.

B. SCC Encoder

As aforementioned, computer screen is another typical video frame. Rather than the traditional camera captured natural video, it is a mixed resolution of text, icon, glyphs, graphics, image, video, etc. It plays at a fixed frame rate, saying 60 frame per second aligned with the display refresh rate. To analyze and understand the impact of applying video compression for VDI solution, we implement the emerging SCC into our proposed PC2PC framework to provide the ubiquitous computing environment. To provide the real-time application, encoder is developed with the significant tool simplification shown in Table I.

C. A Simple Network Bandwidth Estimation

We often face the network dynamics because of the congestion, channel fading, etc. The key issue to maintain the quality of the service is adapting the sending rate based on the real-time network status probing.

Google Congestion Control (GCC) algorithm offers great performance and is adopted in WebRTC framework for real-time communication. GCC employs two controllers: a sender-side controller, which computes the target sending bitrate A_s, and a receiver-side controller, which computes the rate A_r that is returned to the sender. Sender will adjust the sending bitrate to the $\min(A_s, A_r)$. The sender-side controller estimates the sending rate according to the packet loss rate reported periodically by the client. The receiver-side controller is a delay-based congestion control algorithm based on the packet arrival time estimation.

In our baseline system, we provide a simplified version of the GCC where only the client side bandwidth estimation is utilized. Such network status is conveyed in system asset channel to adapt the server sending rate. Here, we can simply refer the sending rate as the screen compression bitrate.

As a greedy strategy, GCC will consume any residual bandwidth. So the "best" quality of all users can not be guaranteed. Allocating each user equal bandwidth is the easiest but obviously not the best solution because different services have different requirement for network resources. Some efforts have been made to provide a good solution of bandwidth allocation for competitive users, such as the game theoretic framework [6] and dynamic bandwidth allocation [7].

For VDI service, sever is aware of the screen content characteristics of different users. Hence, server can manage network bandwidth according to the individual application service. A simple solution is to allocate the network resources to get the best video quality (such as Peak Signal-to-Noise Ratio PSNR) balance among all the users. Intuitively, more bandwidth should be given to the user who is watching videos (e.g., high frame rate and high bit rate case) than the user who is just editing a slides (e.g., lower frame rate and lower bit rate case).

III. Experimental Study and Discussion

Two major tests are carried out to demonstrate the performance of our proposed PC2PC system. One performs the actual field tests for practical virtual desktop applications using our PC2PC platform as well as the default SPICE system; the other one discusses the quality-bandwidth adaptation for our PC2PC system.

A. Dynamic Tests in Real Life

We perform the experiments by collecting the actual network bandwidth in real life applications for commercial Red Hat SPICE system and our proposed PC2PC platform. Towards this goal, we build two small private cloud shown in Figure 2. These two clouds are hosted at N-city and S-city with 200 miles distance, respectively. Each one uses 10 high performance servers. Each server is equipped with two 20-core Xeon CPUs, 128 Gigabytes RAM and 2 Tera-bytes hard

TABLE I: Basic tools in our real-time SCC encoder

Tools	Description
Input Source	8-bit YUV 420
Prediction Structure	IPPP
Coding Unit	64×64 to 8×8
Prediction Unit	2N×2N, N×2N, 2N×N
Transforms	32×32 to 4×4 (no recursive decision)
Filtering	deblocking, sample adaptive offset
Modes	spatial intra, inter, intraBC, palette
Motion Vector	integer MV resolution (no interpolation)
Parallelism	Multiple slices

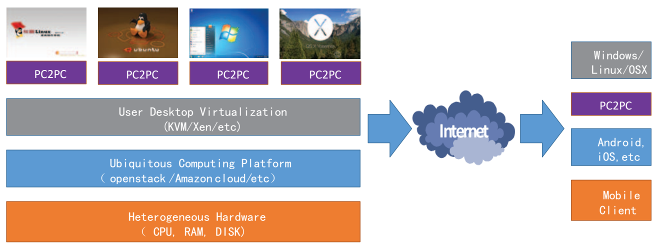

PC2PC
PC2PC
PC2PC
PC2PC
User Desktop Virtualization (KVM/Xen/etc)
Ubiquitous Computing Platform (openstack /Amazon cloud/etc)
Heterogeneous Hardware (CPU, RAM, DISK)
Internet
Windows/ Linux/OSX
PC2PC
Android, iOS, etc
Mobile Client
Fig. 2: Illustration of the PC2PC based private cloud service

TABLE II: Statistics of network bandwidth consumed by the default SPICE and PC2PC

scenario (KB/s)		ave.	std.	max.	min.
office	SPICE	178	228	1188	0
	PC2PC	44	80	671	3.3
web	SPICE	1401	1743	9750	0
	PC2PC	182	181	717	2.5
video	SPICE	1324	880	7580	0
	PC2PC	770	313	1740	2.8

disks. Servers are interconnected using the 10G Ethernet. For instance, an engineer would prefer a Linux OS with 8-core CPU, 64G RAM and 2T disk space for computing intensive environment, while a secretary would just require a Windows 7 setup with a 1-core CPU, 2G RAM, and several hundreds gigabytes disk space for routine office tasks (i.e., word editing, slides presentation, etc). For every user, our PC2PC server algorithm is utilized to compress its instantaneous desktop screen and stream it to the client.

At client side, we choose the Firefly RK3288 [8] development board to implement our PC2PC client algorithm. At this moment, all functions are realized in pure software fashion on its Quad-core ARM platform. Since the stream is compliant with the emerging SCC standard and there is not any available ASIC chip in the market yet, it is preferable to use the optimized software decoder.. Basically, PC2PC client decodes the received stream and renders it on the display. Meanwhile, it also captures the user interaction commands, such as the keyboard and mouse messages, and sends back to the server. Since Firefly RK3288 offers the similar computing power as the recent tablets and SmartPhones, we can migrate our implementation on those platform as well.

1) PC2PC versus Red Hat SPICE: In this section, we mainly compare the (downlink) display channel data bandwidth given that we are using complete different methods for screen display delivery in Red Hat SPICE and our proposed PC2PC. System and input channels are implemented similar to the SPICE counterparts (i.e., main and inputs channels respectively) in our PC2PC system. We would expect the similar network behavior for both methods. Experiments are carried to ensure the visual lossless quality for both SPICE and our PC2PC methods.

We have collected the data for three typical scenarios when users are using their Microsoft Windows 7 powered environments, i.e.,

- "office": represents the scenario that user performs the typical daily assignments, such as the word editing, slides show, email drafting, etc.
- "web": is for the typical web surfing where users usually scrolls the web pages quickly to track and locate the interested topic.
- "video": is for the on-line video streaming service (such as YouTube) where users also fast scrolls the web browser to locate the interested video (at the beginning).

It is often very stationary with few keyboard and mouse hits and very slow page scrolling for "office" application, while "web" and "video" applications would have an intensive interaction in the very beginning to fast locate the specific content.

TABLE III: Server processing delay measurement

slice#	stationary (ms)	motion intensive (ms)
1	5.1	8.8
2	3.6	5.5
4	2.9	4.4
8	2.4	3.6
16	2.7	3.7

Figure 4 has shown the down-link traces of three difference application scenarios for both SPICE and PC2PC systems. Note that display data channel in default SPICE platform require tremendous amount of bandwidth for visual lossless user experience. Particularly the instantaneous bandwidth jumps to about 9.8 MB/s when the screen content has changed significantly due to the intensive user interaction, such as fast content scrolling in "web" application around 125 second shown in Figure 4(b). As shown in Table II, on average, our PC2PC system only requires 25%, 13% and 59% network bandwidth of default SPICE protocol for "office", "web" and "video" use cases with same picture quality, respectively. We also give the maximum bandwidth requirement for both systems, where our PC2PC system uses 56%, 8% and 23% bandwidth of default SPICE for "office", "web" and "video".

SCC video encoder is deployed to compress the instantaneous desktop screen where the first picture is placed using the intra frame (I-frame) followed by the all forward predictive frames (P-frame). I-frame typically requires more than 5x bits compared with the P-frame. Thus, there is a bit rate jump for the first frame. If the following pictures share the similar content, bit rate consumption will be reduced massively because of the temporal prediction (see 0-10 seconds of Figure 4(b)(c)); But, if the screens changes intensively (i.e., window switch, page scrolling) with completely content, bit rate consumption will be jumped noticeably (see 0-10 and 150-275 seconds of Figure 4(a) where we switch the widow very often!), given that intra prediction is preferred for such scenario, rather more efficient temporal inter prediction. Even we fast scroll the pages for "web" and "video" applications, since there are still stationary regions (i.e., typical while black block, similar text block in web pages), temporal prediction still provides excellent coding efficiency with much less bandwidth requirement compared with the SPICE system (cp. Table II).

2) Delay Impact: There are various delays in the overall system. To well understand this problem, we plot the process-

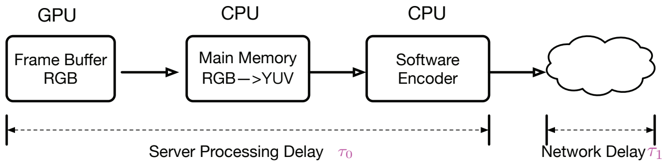
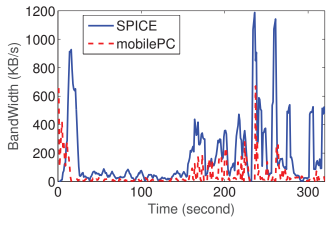
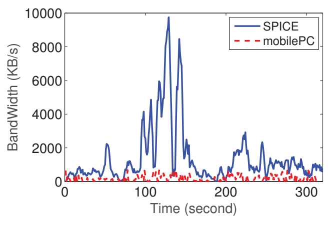


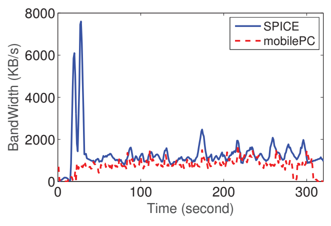
Fig. 3: Potential delays in PC2PC system pipleline.



(a)



(b)



(c)

Fig. 4: Down-link bandwidth comparison for SPICE and PC2PC system (a) office (b) web (c) video

ing pipeline in Figure 3. Delays are categorized to two parts: one is the server processing delay τ_0 and the other one is the network transmission delay τ_1.

Since we only implement the slice threading in our baseline SCC encoder., we evaluate the overall server processing delay using different numbers of slices for 1080p screen video. Both stationary (office and web surfing) and motion intensive (full screen action movie or gaming) screen scenarios are considered with collected data shown in Table III. When we increase the slice number, we see the server processing delay is decreased noticeably, for instance, 8 slices per frame requires almost 50% processing time compared with the 1 slice per frame. Note that the there isn't clear benefits when increasing the slice number from 8 to 16. Hence, we constrain our SCC encoder using 8 slices per frame configuration.

Network delay is simulated in an intranet setup to avoid unexpected traffic over the Internet. Intranet is connected using the 1000Mbps wired connection. Normally, its delay is less than 1 ms. We use the clumsy [9] to control the intranet delay scale. We have invited 10 gaming fans to play the League of Legends (i.e., one of the most profitable games in China) with different network delay setup, and ask them to provide the subjective feedback about the delay configuration, as listed in Table IV.

B. Quality-Bandwidth Adaptation

1) Quality-Bandwidth Optimization for Single User Case : Previously, we have developed an analytical model [10] to express the relationship between video stream bit rate and its perceptual quality, i.e.,

$$Q(R) = \frac{(1 - e^{-\kappa(\frac{R}{R_{\max}})^{0.55}})}{(1 - e^{-\kappa})}, \qquad (1)$$

with κ as the video content dependent model parameter, ranging from 2 to 6 for typical samples (i.e., from stationary to motion intensive). $R_{\max}$ is the maximum bandwidth allocated to a single user.

TABLE IV: Network delay τ_1 and overall delay $\tau = \tau_0 + \tau_1$ (ms) impact on user gaming experience

network	overall	user feedback
<20	<25	excellent experience without noticing delay
20 - 50	25 - 55	3 out of 10 feel some certain delay
50 - 100	55 - 105	7 out of 10 feel the obvious delay
>100	>105	Too bad. Gaming is over!

In 4G test case, we mark the 200 KB/s as the highest sustainable bandwidth. Server encoding is adapted according to the periodic channel probe from the client. As shown in Figure 5, the connection starts from the very low bit rate, and the quality is improved as long as the bit rate is increasing. Once it reaches the ceiling, it will fluctuate because of the congestion-induced packet drop. User's visual system typically tolerates a certain degree of packet drop (such as frame drop with slight frame rate variation) without noticeable visual degradation. Hence, we can observe the quality fluctuation is much smaller than the rate variation as captured in the simulation.

2) Quality-Bandwidth Optimization for Multiuser Case: Although GCC can provide good network utilization for a single user, it has obvious drawbacks for typical multi users applications. We propose a two-step scheme to facilitate the quality-bandwidth adaptation for multiuser scenario. We first allocate the appropriate (maximum) bandwidth for each individual user. Here we would have a upper bound of the total network bandwidth shared by all users. As discussed in the previous sections, we could have two schemes to do the bandwidth allocation shown in Figure 6, i.e., one is the **equal allocation** and the other is the **application driven allocation**.

For application driven allocation, our intuition is shifting more resource for bandwidth hungry applications so as to improve the overall quality for all users. Ideally, we would like to find out the optimal combination of different users (with

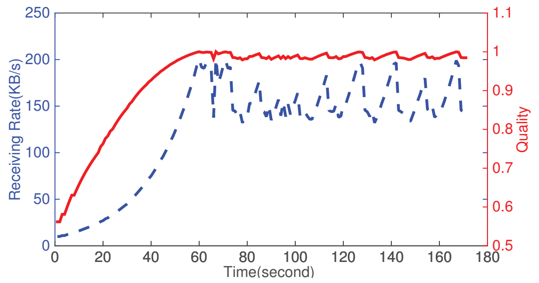


Fig. 5: Quality-rate adaptation using GCC: quality improves as the rate increases; quality fluctuates as the rate varies due to the packet drop but with smaller variation.

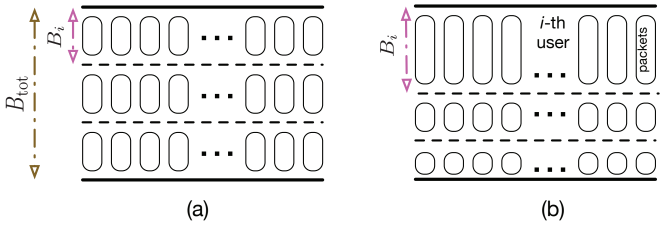


Fig. 6: Multiuser quality-bandwidth optimization: (a) equal allocation; (b) application driven allocation. $B_{\texttt{tot}}$ is the total bandwidth for all users and B_i is the bandwidth allocation for i-th user. Both $B_{\texttt{tot}}$ and B_i are adaptive because of the network fluctuation.

different applications) that yields the maximum overall quality $Q_{\texttt{tot}}$. For different applications, its quality is dependent on its content characteristics and bit rate [10]. Together with Eq. (1), we could formulate the optimization problem as

$$\max \quad Q_{\texttt{tot}} = \sum\nolimits_i Q_i(R_i), \text{subject to} \sum\nolimits_i R_i \le B_{\texttt{tot}}, \quad (2)$$

where $B_{\texttt{tot}}$ is the total bandwidth presumedly allocated for all users. However, due to the network dynamics, it is also a variable over the time. For i-th user, its bandwidth is B_i. For simplicity, we refer it the same as the sending rate R_i.

Referring to the typical scenarios aforementioned, i.e., "office", "web" and "video", we have demonstrated the application driven bandwidth allocation as shown in Figure 7. Here, we assume the maximum network bandwidth at 8 Mbps for these three typical users. Total bandwidth varies due to the network dynamics, such as the wireless channel fading, congestions, etc. Upon instant total bandwidth, we could allocate different user with optimal resource so as to achieve the maximum overall quality. As we can see, motion intensive content (e.g., video) demands more network bandwidth while stationary content (i.e., office) could be managed with less bandwidth shown in Figure 7(a) with better overall quality compared with the most straightforward equal allocation strategy depicted in Figure 7(b). But if the network is too bad or too good, there isn't difference between two allocation mechanisms. This also fits our intuition.

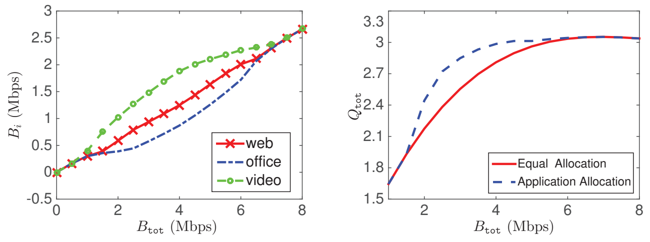


Fig. 7: Application driven bandwidth allocation (a) individual bandwidth for each user (b) overall quality vs. total bandwidth

IV. Concluding Remarks

We present an innovative PC2PC framework as the underlying VDI technologies to support the emerging mobile workstyle. PC2PC sever compresses the desktop screens using HEVC based SCC encoder for each user, and delivers the stream through any popular networks to PC2PC client for stream decoding, rendering and end-user interaction. Network bandwidth is estimated at client side and carried to the server side using system message for quality-bandwidth adaptation.

Extensive studies evident the efficiency of our proposed system, by massively reducing the network bandwidth for the same visual quality in comparison to the default SPICE environment. For instance, SPICE system requires 4x bandwidth for typical stationary office applications, such as word editing, email drafting, slides preparation, etc; it needs more than 7x bandwidth for web applications, such as intensive user interaction like fast page scrolling; Even for on-line video playback (i.e., YouTube), we can save about 50% bandwidth.

Referring to the popular Google congest control algorithm, we have implemented a very simple client side network bandwidth estimation scheme based the packet arrival timing measurement. Such estimation network bandwidth is encapsulated as the system message for the server to adapt the SCC encoder bitrate. Single user and multiuser scenarios are both considered when optimizing the quality-bandwidth trade-off.

Acknowledgment

Dr. Z. Ma is partially supported by National Science Foundation for Young Scholar of Jiangsu Province, China (Grant # BK20140610) and the National Science Foundation of China (Grant # 61571215).

References

[1] Citrix Mobile Workstyles Survey, "Workplace of the future: a global market research report," 2014, Available online at https://www.citrix.com/content/dam/citrix.

[2] VMware Horizon, *http://www.vmware.com/products/horizon-view*.

[3] Citrix XenDesktop, *https://www.citrix.com/products/xendesktop*.

[4] Microsoft RDP, *https://en.wikipedia.org/wiki/Remote_Desktop_Protocol*.

[5] SPICE, *http://www.spice-space.org/*.

[6] Haïkel Yaïche, Ravi R Mazumdar, and Catherine Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 5, pp. 667–678, 2000.

[7] Chadi M Assi, Yinghua Ye, Sudhir Dixit, and Mohamed A Ali, "Dynamic bandwidth allocation for quality-of-service over ethernet pons," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 9, pp. 1467–1477, 2003.

[8] Firefly RK3288, *http://en.t-firefly.com/en/*.

[9] clumsy, *http://jagt.github.io/clumsy/index.html*.

[10] H. Hu, Z. Ma, and Y. Wang, "Optimization of spatial, temporal and amplitude resolution for rate-constrained video coding and scalable video adaptation," in *Proc. of IEEE ICIP*, Oct. 2012.