Challenges of Remote Access to Emulation of Original Environments

Functional Longterm Archiving Group
Albert-Ludwigs University Freiburg
Hermann-Herder Str. 10
79104 Freiburg i. B., Germany

ABSTRACT

Offering emulation services in the cloud (remotely) generates several issues as the user does not interact directly with an emulator but moderated through a network connection. Additionally to the challenges of proper translation of user input from the client side a network link introduces the problem of limited bandwidth and delayed responsiveness to user interaction. As the requirements for remote digital environment access share similarities with thin client computing and desktop virtualization, there are a couple of implementations available which can be considered and evaluated for remote emulation. While standard desktop remote access is solved, especially multimedia streams and fast interactive applications still pose a challenge. Different to a couple of remote desktop protocol implementations which have a certain knowledge on the screen content to be transferred, this information is mostly missing for abstract emulator framebuffer output. Due to technical limitations like bandwidth, processing and network delay the possible solutions depend on user expectations.

1. INTRODUCTION

The shift of the usually non-trivial task of emulation of obsolete software environments from the end user to specialized providers through Emulation-as-a-Service (EaaS, [4, 6]) helps to simplify digital preservation and access strategies. End users interact with emulators remotely through standardized (web-)clients on their various devices. Beside offering relevant advantages, EaaS makes emulation a networked service introducing new challenges like remote rendering, stream synchronization and real-time requirements. A set of objectives, like fidelity, performance or authenticity may be required depending on the actual purpose and user expectations. Various original environments and complex artefacts have different needs regarding expedient and/or authentic performance.

Remote Display Systems have been evolved over a longer period and are popular in many forms like VNC [5], RDP, THINC [1], Citrix, PCoIP or X11.[1] They provide solutions for thin client computing, remote administration, desktop virtualization or mobile users and they were implemented at all levels of the display software stack, like virtual hardware, device drivers and window system extensions (Fig. 1). Thus, there is a large variance in semantic properties of the protocols. Lots of work has gone into optimizing remote displays, predominantly to decrease bandwidth requirements

[1]See the various project home pages.

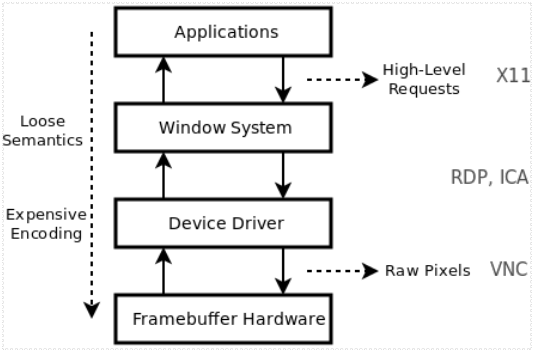

Figure 1: Display Pipeline of GUI systems

[3]. Emulators can also be a source for remote displays and audio like desktops or server management consoles. The generated streams need to be properly grabbed, transcoded if required and transported to the user's client.

While rather static remote screen access is accomplished for a while, dynamic content like multimedia streams or highly interactive software with fast screen refreshes require more complex solutions. Such solutions usually significantly exceed computational power or hardware requirements compared to the desktop case. They might be "solved" partially by rising network capacity and processing power. Future requirements are rising screen resolutions (like the upcoming 4K display standard) and 3D rendering. Even with ever improving networks, bandwidth and processing delays remain a limiting factor requiring compromises. User expectations on the service could be used as input, when different solutions exist to chose from.

For future implementations of both emulators and remote emulation services it needs to be discussed what should be achieved to make services like EaaS widely adoptable and acceptable both to institutions and users. The problem should be analyzed and classified, if it is a *conceptual*, *computational*, *bandwidth* or just *implementation* challenge. Conceptual problems may produce some interesting research questions, while the other challenges remain on a technical level.

2. STREAM DE-MULTIPLEXING, DELAYS AND REMOTE EMULATION

The performance of various artefacts, especially ones with multimedia components, has several implications as the object is not directly rendered within the end user's environ-

ment but send through additional layers (Fig. 2). Already in standard setups where the object is rendered in it's native environment, the synchronization is not predictable due to different buffers in the sound and video pipelines.[2] Different to direct rendering where the artefact is (uncompressed and) de-multiplexed (1) and where streams are directly sent to the devices like screen and loudspeaker, it is moderated through the emulator executing the original environment. The emulator itself runs as a normal application within a hosting environment. This step may already de-synchronize the original performance as different buffers can be in place for screen and audio output. It can be worsened in scenario A where in-

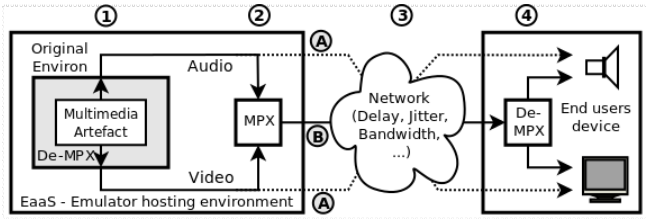

Figure 2: AV Stream De-/Multiplexing Challenges in Remote Emulation

dependent streams are sent over an (uncontrollable) network channel to the end user's device (3). The transport layer protocol might add additional buffers.Various AV protocols optimize on quality, reliability or low-latency and thus add differently to end-to-end delay. System buffers in (2,4) can influence the end-to-end delay and clock skew significantly. In the other approach B the previously separated AV streams are re-multiplexed within the hosting environment and sent as a single stream. The second de-multiplexing takes place in a better controllable environment of the the end user's device (4). The stream is however prone to network delays and limitations. Emulators have to provide APIs for local and/or remote AV output.

3. POSSIBLE SOLUTIONS

Different implementations of streaming protocols are already known from Thin Client Computing and Desktop Virtualization. They attach to different layers in the Device Pipeline (Fig. 1). On the top layer high-level requests of the windowing system can be seen, which are system dependent. The well-known network transparent X11 protocol works below the application layer, where proxies like NoMachine NX[3] intercept. Unfortunately, the X-Window System does not provide any audio functionality. With the rising abstraction from concrete windowing systems semantics are lost when traversing the pipeline. The drawing becomes more generic in the form of raw pixels. This allows for much more generic protocols independent of the windowing system, but increases the encoding costs.

The latter limitation is overcome by RDP, a proprietary protocol designed by Microsoft as well as Citrix' ICA and Sun Microsystems Appliance Link Protocol. These implementations attach to the layer between window system and device driver.

A bandwidth optimized, platform-independent protocol for remote desktop access is Virtual Network Computing (VNC) which attaches between display driver and framebuffer in the display pipeline. It uses the Remote Frame Buffer (RFB) protocol. To achieve the significant compression VNC needs to run over small bandwidth connections, it tiles the desktop into regions, and transfers only the differences between two consecutive images. Those are usually pretty small in standard desktop interaction. Plus, the protocol normally runs within the operating system whose's desktop is accessed over the net. Thus it can make heavy use of the knowledge if e.g. a window is moved, resized or is partially hidden to efficiently redraw the remote end. Optimizations like special codecs to update parts of the screen or well understood content are discussed and algorithmically improved in [3] and [2]. A significant limitation for multimedia artefacts is the missing audio channel in VNC.

A different trend, completely agnostic of the actual underlying operating systems and specific screen content, are various hardware solutions directly attached to the device output.[4] Further options in that domain like Apple Airplay,[5] and similar framebuffer mirror devices[6] which mirror device output by using specialized hardware to encode the screen updates (and audio streams) into a wireless H.264 stream. Different to the remote framebuffer solutions the screen content is treated as a movie and transmitted at a fixed update rate. Only slightly different are the comparably recent Gaikai and OnLive proprietary streaming services primarily focusing on cloud-based gaming. They require a previously installed Java, Flash or client application. The encoding of the actual AV stream takes place within specialized 3D graphic adaptors capable of handling high resolution and frames per second. Another method are virtual, remote PCIexpress adaptors.[7]

Another way of remote display access are screencasting solutions,[8] which produce video streams from standard desktop environments e.g. to show gameplay. They are usually not meant for interaction. Thus, latency of e.g. encoding the screen and audio output play a minor role.[9]

3.1 Latency vs. Bandwidth

Large amount of work in optimizing bandwidth usage of Remote Displays compression, encoding, caching and protocol design [3]. Effective usage of bandwidth is important, but so is interactivity. If the round-trip-time for an event triggered by a user is too long, it becomes difficult to link that event to the original trigger. This can be observed, for example when moving the cursor over complex menus folding out. Every user interaction implies waiting for server

[2]This is usually not an issue as the deviation is small.

[3]See e.g. `http://www.nomachine.com`

[4]These range from analog screen capture appliances such as NCast, `http://ncast.com`, which digitize VGA streams or the Wireless home digital interface, `http://www.whdi.org`, offering a proprietary point-to-point wireless transfer expecting the network capable of supporting 1080p uncompressed video.

[5]See `http://www.apple.com/ipad/features/airplay`

[6]Like Sony wireless screen mirroring or Intel WiDi, urlhttp://www.intel.com/content/www/us/en/architecture-and-technology/intel-wireless-display.html

[7]Compage, e.g. `http://www.amd.com/us/products/workstation/graphics/firepro-remote-graphics/rg220/Pages/rg220.aspx`

[8]See the references in `http://en.wikipedia.org/wiki/Comparison_of_screencasting_software`

[9]Like compared to delays in life broadcasting in digital TV for sports events.

side updates. Thus, encoding and network latency are critical components.

With regard to complex content the problem is that most protocols focus on office applications over LAN links and reduce data transfer on low bandwidth links. Therefore there is poor support for display intensive interactive applications (games, ...) and poor support for video and audio. The same is true for higher latency WAN environments as is to be expected for remote access to emulated environments.

3.2 Input Translation

With the evolution of computer systems the concepts of input devices and layouts of keyboards have changed. First of all, users and the intermediate systems need to be aware of two different problem set: basic keyboard layouts defined by the original hardware manufacturers (1) and localizations of these keyboards (2). A key pressed on the client side needs to be properly transported and if required translated to trigger the appropriate action within the original environment. A pressed key is usually presented by a hardware scan-code.

Special keys and modifiers need to be present to properly interact with the remote system.[10] In some cases the input needs translation, if e.g. a PC desktop is accessed from a tablet device. Touch gestures are to be translated to mouse movements and on-screen keyboards are to be offered to use the keyboard available on the original platform. The keyboard even might need translation between different layouts regarding languages and function keys. The same is true for totally new sources of input like GPS signals, position sensors, electronic compass or gyroscope. The problem has some conceptional components if present remote access protocols lack significant features, but is otherwise an implementation issue.

The various implementations of emulators depend on information on keys pressed on the concrete keyboard. Usually the technical implementation of the actual keyboard is different from the original hardware and needs to be translated. Here, already the frameworks like windowing systems, or SDL play a major role as they filter and act upon the input. Keyboard events usually generate scan-codes (hardware representation of a concrete key on a concrete keyboard) which is translated to symbol-codes and represents e.g. the letter "a". Unfortunately the mapping does not need to be unique, as e.g. a hardware scan-code of "y" maps to "y" or "z" (just thinking of German and English keyboard layouts). It might get even nastier thinking of special keys like "|" which are directly accessible on the English layout but need a modifier on the German version (problem 2).

In SDL both codes are transferred (but only the latter is present in VNC streams, which is a problem because of the mapping issue outlined before). The effect is, that locally emulators like QEMU or DOSBOX work fine, but accessed over VNC they o nly see the whole keyboard if configured to symbol-codes (and a matching keyboard layout).[11] Many emulators try to "optimize" the input received e.g. via SDL. SheepShaver is an example of an emulator, which has built-in code for proper keyboard translation.

[10] E.g. special modifier keys were present on many older platform keyboards but not on today's systems. Thus, certain scan-codes may not be present on today's keyboards, which might require additional virtual replacements.

[11] Parts of the keyboard get ignored using scan-codes.

4. DISCUSSION

To provide convenient remote emulation services[12] several of questions and challenges can be discussed. Some of them address a more general audience, others might just serve as input for the emulator community. From practical experience of accessing different emulators remotely we acquired a number of technical and research questions:

- What kind of service is acceptable for users and which (rendering) quality, responsiveness should be achieved?
- VNC is the prevailing remote access method but lacks audio capabilities and is limited regarding multimedia and 3D. Does it make sense to define a next generation standard remote access API for future emulators in an ever changing technical landscape?
- Can RDP (proprietary), offering at least built-in sound support, take over as the next generation protocol?
- Would a dedicated remote access client (because of more easy to control input and output features, be better suited for synchronization and better control of the rendering environment) and thus be preferable over a more general web access?
- What kind of remote desktop protocols are useable for generic remote access to emulation?
- What are the bandwidth and delay requirements for future (multimedia, research) data, virtual research environments?
- How fast should the system output react to user input?
- How can the user input be properly translated, for example, in order to adapt to different keyboard layouts used by different platforms or required for different localizations?

5. REFERENCES

[1] R. A. Baratto, L. N. Kim, and J. Nieh. Thinc: a virtual display architecture for thin-client computing. In *ACM SIGOPS Operating Systems Review*, volume 39, pages 277–290, 2005.

[2] R. A. Baratto, L. N. Kim, and J. Nieh. Thinc: a virtual display architecture for thin-client computing. *SIGOPS Oper. Syst. Rev.*, 39(5):277–290, Oct. 2005.

[3] S. Chandra, J. T. Biehl, J. Boreczky, S. Carter, and L. A. Rowe. Understanding screen contents for building a high performance, real time screen sharing system. In *Proceedings of the 20th ACM international conference on Multimedia*, MM '12, pages 389–398, New York, NY, USA, 2012. ACM.

[4] K. Rechert, I. Valizada, D. von Suchodoletz, and J. Latocha. bwFLA – a functional approach to digital preservation. *PIK – Praxis der Informationsverarbeitung und Kommunikation*, 35(4):259–267, 2012.

[5] T. Richardson. The rfb protocol. Online, *http://www.realvnc.com/docs/rfbproto.pdf*, 2009.

[12] Some of the issues remain like input or mapping of keyboard translations, remain if emulation runs locally.

[6] D. von Suchodoletz, K. Rechert, and I. Valizada. Towards emulation-as-a-service – cloud services for versatile digital object access. In *Proceedings of the 8th International Digital Curation Conference, iDCC08, Amsterdam*, 2013.