The Liquid Media System – a Multi-Device Streaming Media Orchestration Framework

Jayanth Mysore, Venu Vasudevan, Jay Almaula, Anwar Haneef
{jayanth, venuv, almaula, anwar}@labs.mot.com

About us

The authors are members of the Mobile Platforms and Services Laboratory of Motorola Labs. We have been examining the problem of enabling multi-device streaming media delivery in multi-device computing environments for over a year. Our efforts have resulted in the development of a system to enable multi-device content delivery in mobile computing environments, which we call the Liquid Computing System.

The focus of the Liquid Computing System is to enable content access, delivery and manipulation in multi-device mobile computing environments. While we have given some thought to human-computer interaction in such environments, it has not been the focus of our study so far. Our emphasis has been on the design of the distributed system to enable multi-device streaming media content delivery.

Assumptions about our networked future

The project makes the following assumption about a reasonably networked future in the 2-3 year time frame:

1. Users will at least one mobile device that is wirelessly connected to the Internet which they carry with them
2. Users will own multiple devices that are always connected to the Internet - these could include their PCs at home or office, for example
3. Shareable computing resources will be available in physical locations with sufficient traffic – for example, conference rooms, coffee shops, theme parks etc.
4. Shared resources will be owned and operated by entities different from the users actually using them – users will rent and release these resources on the go

So what's lacking today?

Computing and communication models currently in use, we believe are insufficient to exploit such a future for the following broad reasons:

1. **Devices ordinarily lack local peripheral vision** – devices do not naturally detect other resources not part of the same physical device. Technologies such as BlueTooth address this problem to an extent as long as all devices use BlueTooth. However we believe assuming homogeneity in the wireless protocol used universally is not strong.
2. **Devices lack network peripheral vision** – an application on a given device is written with the implicit assumption that the only resources available to it are those on the device or on remote machines whose locations are known at the time of developing the software. However, a resource may be detected at run-

time in the physical or virtual proximity of the user, which could improve the utility of the application to the user. A device in virtual proximity is one that a user has access to, that is connected to the network, but far from his physical location.

3. **Devices may have multiple network interfaces** – applications are written to detect a given interface and use that interface for the lifetime of a session. However, it may be more optimal to change the actual interface used during the course of a session or use multiple interfaces to achieve better quality.

A number of projects in the area of pervasive computing [Oxygen], [Gaia], [Aura], [EasyLiving] consider significantly more sophisticated environments and propose more comprehensive solutions. However, we have made a deliberate attempt to look at a more focused application (communication) and a much more shorter time frame. This has forced us to stay firmly rooted in industry trends, existing standards and available technologies.

The Liquid Computing Architecture

In order to overcome the above limitations, we have developed an architecture called the Liquid Computing Architecture, which

1. Decouples a user's communication session management from resource ownership and management.
2. Enables intelligent splitting of session content (such as multi-media content) based on a user's preferences for stream quality, cohesion (the extent to which a user cares about component parts of a session playing out on the same device) and cost.
3. Enables automatic device and network discovery
4. Enables users to rent shared resources and them to their pool of resources on-the-fly
5. Enables dynamic re-orchestration of content to devices as the resource pool of a user changes
6. Enables users to move content to devices in their resource pool irrespective of the device's location – for example, a user can move an audio stream from his MP3 player to his hard-disk on the PC at home.

In a nutshell, the liquid computing system consists of the **liquid computing server** - an application layer "bridge" between multi-device and single device computing environments, which uses an orchestration algorithm for selecting the best set of resources for rendering content belonging to sessions that a user is participating in, a **device manager** per computing neighborhood - a "resource vending machine" for locating and selecting resources in a shared computing space and a **liquid computing user process** – a process that identifies a user uniquely to a server and enables a user to leverage the richness of the liquid computing environment.

The liquid computing server acts like an operating system of a virtual computer whose peripherals change over time as users add and delete devices based on their location and preferences. It hides the underlying complexity and heterogeneity in device types,

network resources. The device manager exposes devices as resource collectives and exposes an intuitive interface for renting a resource. The orchestration algorithm used by the liquid computing server maps components of a stream to resources in a user's resource pool optimizing the mapping to satisfy a user's preference for stream cohesion, rendering quality and rendering cost. Users express these preferences using utility functions that are stored as part of a user's profile in the liquid computing server.

Implementation status

We have implemented the liquid computing system in our laboratory test bed. In our set up, the Liquid Computing User Process runs as a J2ME midlet on a cellphone (the Nextel i85c phone). Other personal devices carried by a user include iPAQs and laptops connected to the network using 802.11b networks. We use laptops and desktops as shared devices and a PC using Pentium III 800MHz, running Linux as our Liquid Computing Server. The entire implementation is in Java (J2SE for all devices except the phone where we use J2ME).

We have used the system to demonstrate how streaming media bundles can be split and rendered across multiple devices, and have these sessions migrate with the user as a user's location changes. We have enabled users to move sessions from a rendering device to a remote device where the content can be stored, using a file system like interface to enable session manipulation (copy, move, redirect, pipe etc.)

Most of our experiments have been qualitative and observational. The key insights we drew from our experiences have been captured as challenges for further research in the next section.

Challenges

1. **Avoiding user surprise**: The flip side of automatic device selection by the orchestration algorithm is what we term "user surprise" – when a user enters a space, he has little clue on which device is going to be used for rendering the content. While the algorithm makes every attempt to satisfy the user's preferences, we find it important to give users an indication of where the action is going to occur, and provide users an opportunity to change the orchestration before content actually gets rendered. This is especially true in social environments, as users may not want their private conversation with a family member to play out on public speakers in a theme park, for instance. The short term solution to this problem is to replace the request-response model with a request-notify-response model, where in the notify phase, the system notifies the user of the selected target device and modality and provides an opportunity for the user to modify the system's selection before the response is streamed to the selected devices. However, this approach has the drawback of breaking the continuity of sessions. A better solution would be to learn user preferences over time and use this to drive the device selection process.

2. **Multi-device input**: In many instances, it may be very natural for users to use multiple devices to input related information. For example, one may find it natural to take a picture on a camera, but annotate it using his phone (which he is used to "talking into").

While combination devices, such as cell-phones with built-in cameras, are a potential solution, they solve a narrow sub-set of the larger problem.

3. **Avoiding event spamming**: Users in highly connected, always on environments may be in touch with multiple information conduits – e-mail, telephone, web, video conferences etc. In addition, users may be maintaining multiple to-do lists. While location detection, multi-device computing and always on connection may be effectively leveraged to expose events on a user's information conduits and to notify users of opportunities to accomplish tasks on their to-do lists, it is important that the technology does not morph into a nuisance. Specifically, user preferences need to be studied and profiled to tune the signal-to-noise ratio in these cases.

Conclusion

In this brief paper, we have presented our approach to enabling multi-device content delivery using the Liquid Computing System. While our solution enables automatic content orchestration based on user's preferences, we believe there are still significant challenges that need to be addressed before the technology is ready for the mass market. However, there may be specific vertical applications where the technology may be ready for adoption, and such opportunities may provide key insights into user behavior, which hold the key to developing practical solutions to the hard problems.

References

[Oxygen] http://oxygen.lcs.mit.edu/index.html

[Gaia] "Gaia: A Middleware Infrastructure to Enable Active Spaces" Manuel Román, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt, IEEE Pervasive Computing, pp. 74-83, Oct-Dec 2002.

[EasyLiving] "EasyLiving: Technologies for Intelligent Environments". Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S. Handheld and Ubiquitous Computing, September 2000.

[Aura] "Project Aura: Toward Distraction-Free Pervasive Computing" Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P. IEEE Pervasive Computing, April-June 2002