Collaborative Music Creation and Performance with Soundcool Online

Roger B. Dannenberg
Carnegie Mellon University
Pittsburgh, PA
rbd@cs.cmu.edu

Huan Zhang
Carnegie Mellon University
Pittsburgh, PA
huanz@andrew.cmu.edu

Amit Kumar Meena
Indian Institute of Technology
Indore, India
amitmeena094@gmail.com

Ankitkumar Joshi
University of Pittsburgh
Pittsburgh, PA
ahjoshi@pitt.edu

Amey Kiran Patel
Indian Institute of Technology
Indore, India
ameykpatel@gmail.com

Jorge Sastre
Universitat Politècnica de València
Valencia (Spain)
jsastrem@upv.es

ABSTRACT

Soundcool Online is a Web Audio re-implementation of the original Max/MSP implementation of Soundcool, a system for collaborative music and audiovisual creation. Soundcool has many educational applications, and because Linux has been adopted in many school systems, we turned to Web Audio to enable Soundcool to run on Linux as well as many other platforms. An additional advantage of Soundcool Online is the elimination of a large download, allowing beginners to try the system more easily. Another advantage is the support for sharing provided by a centralized server, where projects can be stored and accessed by others. A cloud-based server also facilitates collaboration at a distance where multiple users can control the same project. In this scenario, local sound synthesis provides high-quality sound without the large bandwidth requirements of shared audio streams. Experience with Web Audio and latency measurements are reported.

1. INTRODUCTION

Soundcool [12] is a free system for collaborative audiovisual creation. It consists of a set of modules such as audio and video players, effects and mixers, video switchers, and virtual instruments that run in Mac or PC computers as part of a Max/MSP stand-alone application. These modules can be controlled with the Soundcool OSC app for iOS or Android smart phones and tablets through local Wi-Fi or at a distance over the Internet. In 2013, version 1.0 was released, dealing only with audio, and was tested in a high school music class in Spain. Since then, Soundcool has been adopted by educators in several European Erasmus+ projects and in the Americas. In 2018, version 3.0 was released adding video, and now it is a powerful tool for Project Based Learning in STEAM (Science, Technology, Engineering, Arts and Mathematics) collaborative projects.

One of the challenges of using technology in education is the (lack of) availability of computing resources. One of the original motivations to use mobile devices as controllers in Soundcool is the simple fact that many students already have mobile phones they can use. The original implementation assumed the availability of a Windows or macOS personal computer to act as "server" and user interface, while mobile devices act as controllers.

In some schools, there is a push for open-source software to reduce licensing costs, and there are even special versions of Linux designed for school use in Spain. The Soundcool project was asked to run on this Linux platform, but that was not possible using the original Max/MSP implementation. Therefore, we considered other implementation options and decided to use Web Audio.

With Web Audio, we have a wider range of operating systems, a fairly device- and system-independent execution environment (web browsers and HTML5), and thus greater and simpler portability.

This paper reports on our experience with Web Audio and HTML5 to implement a flexible, modular sound synthesis system for collaborative use in education. Related work is discussed in the next section. Then, we present our motivation and design goals (Section 3). In Section 4, we describe some of the challenges of our Web Audio implementation. In Section 5, we describe how we have designed Soundcool Online to support collaboration. In Section 6, we describe some work to extend the built-in signal processing options of the Web Audio standard and give some performance measurements to compare built-in signal processing to signal processing in ScriptProcessorNodes. Finally, Sections 7 and 8 describe future work and present our conclusions.

2. RELATED WORK

There are many music audio projects with an educational objective, such as EarSketch [5] and BlockyTalky [13]. In the Web Audio sphere, iMuSciCA [7] offers web-based musical activities to support STEM subject learning, and Scott Fradkin's Snap Music (www.fradkin.com/snapmusic-0.3/tone-snap.html) is a music programming system for children. WebPd (https://puredata.info/downloads/webpd) is

Web Audio Conference WAC-2021, July 5–7, 2021, Barcelona, Spain.

an experimental release of Pure Data [8], a visual programming tool for multimedia, that allows a subset of Pure Data audio patches to run in the browser.

A number of virtual synthesizers exist for Web Audio as well, including Synth Kitchen [10] and Zupiter (z.musictools.live/), and more can be found at Web Audio Modules (www.webaudiomodules.org/wamsynths/).

Soundcool Online mainly differs from this other work in its emphasis on collaborative performance, versatile high-level sound modules, and a large body of supporting materials and experience in primary and secondary music education.

3. MOTIVATION AND DESIGN

Soundcool [12] is a modular sound synthesis system implemented in software. A feature of Soundcool is that modules are high-function units akin to plug-in effects or Eurorack hardware modules. For example, the Soundcool SignalGen module implements multiple waveforms, FM and AM modulation, a user interface with amplitude and frequency sliders, and access via Open Sound Control. With such complete modules, users can focus on making music and have a minimal amount of implementation and configuration to realize interesting music performance systems.

As mentioned earlier, an important motivation of Soundcool Online is to allow Linux-based systems in schools to run Soundcool. Moreover, browser-based software can run with little or no modification on many different systems.

The second attraction of Web Audio for Soundcool is the possibility of greater sharing. One of our inspirations is Scratch, a browser-based visual programming environment for young people that is immensely popular. [9] Scratch users can share their creations with peers, see, try, and modify examples online, and display galleries of their creations. Inspired by this approach, Soundcool Online allows users to name and save projects and make them public or private.

Recently, we have been experimenting with network performances using Soundcool [11] (see https://youtu.be/kRp4SMfpL0Y?t=5534 for an example). Since the controllers for Soundcool objects are mobile devices communicating over Open Sound Control, it is simple to give access to multiple performers over the Internet and to share the Soundcool screen and audio with all the performers using a third-party communication system such as Zoom or Microsoft Teams. Web Audio offers the possibility of setting up communications through a Soundcool server, which is simpler than dealing with IP addresses and port numbers needed to configure Open Sound Control.

3.1 Project and Modules

In Soundcool Online, the Project panel provides a list of available modules, which users can connect and control. Users are able to instantiate modules by clicking on the icon from the panel (shown in Figure 1). In the Project panel interface, modules are stacked in columns with even margins. The arrangement of the modules is flexible, and users are free to drag and reorder modules among columns. Rather than using lines or "wires," modules are interconnected by clicking on an output ("Out"), then clicking on an input ("In"). The output and input boxes indicate the connection status, and hovering over a connection box highlights all the connected modules to make inspection easy. This approach was created for the original Soundcool system to simplify the implementation, but we found it works well even for novice users. As a large project usually involves a lot of interconnected modules, explicit wires significantly increase the visual complexity on the interface and confuse the user.

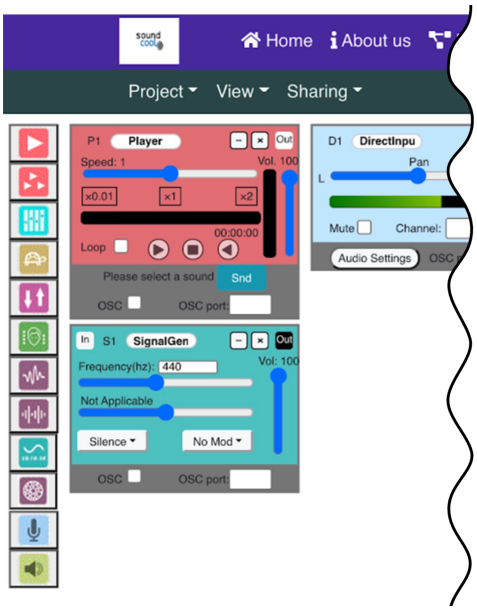

Figure 1: A sample project with Player, DirectInput, and SignalGen modules.

Underlying each module, we employ the Web Audio API to implement audio processing. In Figure 2, we show the Web Audio nodes composing a Soundcool delay module, where users are allowed to control delay time and feedback. Most modules have input (button In generally at the top left) and output (button Out generally at the top right). More implementation and analysis of modules will be illustrated in Sections 4 and 6.

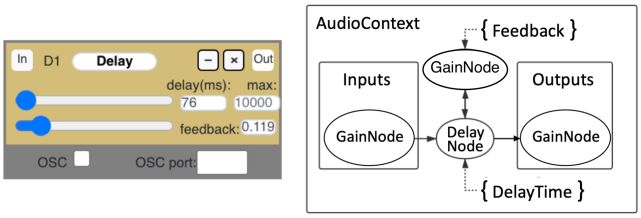

Figure 2: Soundcool Online's Delay module (left) and its Web Audio implementation (right).

For *Soundcool Online*, we have (so far) implemented the following modules: SignalGen, Player, SamplePlayer, Mixer, Delay, Transposer, Pan, Oscilloscope, Spectroscope, GranSynth, DirectInput (microphone input), and Speaker (audio output).

Projects can be saved on the server in an SQL database using a JSON representation of modules, connections and project meta information. Users can also download and upload projects to and from local files using the same JSON format.

3.2 New Features

Besides saving and sharing projects, the interface offers a dashboard for quick access to a user's personal projects and to projects shared by others.

Although Soundcool projects often incorporate audio samples and loops, we are concerned about storing audio content, which could create a considerable demand for storage and bandwidth. Although we currently store samples on our server, we consider it a "feature" that users can obtain sounds from other online sources such as Freesound [4]. Figure 3 illustrates our server page where users click "Add Online Sound" to add a sound to their library. Any Soundcool module that uses a sound, e.g. *Player*, has a button to select a sound from the user's library.

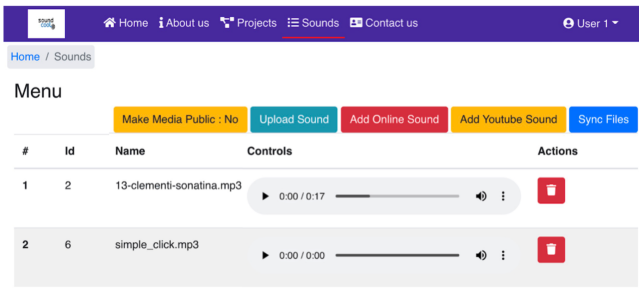

Figure 3: User sound inventory page.

4. IMPLEMENTATION

Our implementation consists of three major parts:

- Audio modules are the basic signal processing modules seen by Soundcool users. These are of course implemented using the Web Audio API.
- Soundcool module interfaces and the overall graphical user interface are implemented in Javascript. The interface components are designed using the popular framework React, and the application state is managed using the Redux framework.
- The backend consists of storage (we support both SQLite and MySQL databases), project serving and saving, and an Open Sound Control interface, all implemented with Node.js and React.

Web Audio poses some interesting challenges for the implementation of Soundcool Online, and we discuss some implementation details and limitations that we have encountered.

4.1 Limitations of AudioNodes

Like many computer music systems, Web Audio is based on connecting simple predefined signal-processing modules (unit generators or "AudioNodes" in Web Audio terminology) to implement signal creation and processing algorithms. This is a convenient approach, but unlike most programming languages where almost any algorithm can be implemented, unit generator libraries are never "complete." I.e., there are interesting, reasonable signal processing algorithms that cannot be reasonably implemented using existing unit generators. This is a well-known and long-standing problem of computer music languages. [1]

One "escape" provided in Web Audio is the ScriptProcessorNode that allows signal processing to be performed directly using JavaScript at the possible cost of speed and latency. We used a ScriptProcessorNode to implement a granular synthesis module for Soundcool, and we give further details on this in Section 6. AudioWorklet [3] is replacing ScriptProcessorNode, and we expect to adopt AudioWorklets in the future.

4.2 Limited Bandwidth in Schools

Another problem we face is the availability of Internet connectivity and bandwidth in primary and secondary schools. Often, networks are limited or even restricted. This is especially a problem for Soundcool projects that make use of audio files, resulting in project sizes of tens of megabytes. The upload and download times can be prohibitive.

For situations where bandwidth is limited, we provide a locally executable version of Soundcool Online. This executable is easily created using the `pkg` package, which generates a Node.js application in the form of an executable binary (e.g. a Windows `.exe` file or macOS application). This makes installation very simple because the user merely downloads and runs the application to start a local server. Once the server is running, users can connect either on the same machine or from some other machine on a local-area network, for example a laptop connected in a classroom by Wi-Fi, or even over the Internet.

The local executable version uses SQLite for storage, and SQLite in turn uses an ordinary local file. This eliminates the need to install and configure a database server. Thus, a classroom with only Wi-Fi and local computing can run Soundcool Online with fast local storage, the ability to support multiple client machines with Soundcool running in browsers, and multiple mobile devices controlling Soundcool modules over Open Sound Control, with practically zero configuration.

4.3 Implementation Tools

Another challenge of the Web Audio approach is the large number of tools that increase the complexity of the application, as opposed to the monolithic implementation of Soundcool in Max/MSP.

As mentioned previously, Soundcool Online is implemented in Javascript under the Node.js environment. In the front-end graphical user interface design, the React framework utilizes `jsx` syntax, a combination of Javascript and HTML, to create more intuitive React components. In the development, we utilized npm, a package manager for Node JavaScript, to keep track of necessary libraries and hundreds of dependencies. In the locally executable version, these dependencies are compiled and linked with the `pkg` library to create a self-contained executable with no external dependencies on the Node.js environment.

For Soundcool Online, we serve the project with Amazon AWS ElasticBeanstalk, a service for deploying and scaling web applications. Under this service, we run the application on an EC2 server, and keep the SQLite as database manager for objects in S3 storage. In order to achieve collaboration and sharing, we also implemented login services in the back end and store user accounts in databases.

5. COLLABORATION

5.1 Shared control over OSC

With Open Sound Control, we can set up communication between mobile devices and the server. Latency is critical, and we performed the following latency test:

- Create a project, and connect a mobile phone Soundcool App to a Player module in the project.
- Load a clear click sound into the Player module.
- Tap the play button on the phone to start the sound from the Player module.
- From an audio recording, measure the time between audibly tapping the control and the audio output of the click sound, giving an end-to-end estimation of system latency using OSC control.

Our assumption is that the overall latency measured above can be roughly broken down into four parts, as follows:

- **Touch screen latency**: Latency includes the response time of the mobile device, from tapping the screen until an OSC UDP packet is transmitted. We do not have network measurements, but GameBench reports around 90 ms response times for graphics response on both iPhones and Galaxy devices, as measured with a high-speed camera (blog.gamebench.net/touch-latency-benchmarks-iphone-xs-max-galaxy-note-10). These measurements include game graphics but not message transmission over Wi-Fi, so it is only an approximation for our application.
- **Network latency**: We used the command `ping` to measure the time it takes for network transmission to the Soundcool server, which is currently located in Ohio, US. We take the average round-trip time, and we report measurements from both China (over 11,000 km) and Pittsburgh (about 200 km). The "Local" location refers to the Soundcool server running on the same laptop as the browser, and the Network time there is an estimate of OSC over Wi-Fi plus inter-process communication times.
- **Browser and Audio latency**: The time that Web Audio needs to respond to an event with audio output. This is mostly due to audio buffers, but includes front-end scheduling and processing time. To estimate this component, we simply measure the time it takes for a GUI button click to start the player, without going through OSC control. This is measured by recording the mouse click and sound output with a microphone and analyzing the result. Of course, this uses mouse event processing that is not strictly comparable to reacting to an incoming network message, but the major component, audio latency, is the same.
- **Server response latency**: In principle, we could subtract estimates for other time components from the total to estimate server processing time, but we suspect the server time is on the order of a few ms, which cannot be resolved with these measurements.

Location	Touch Screen	Network	Web Audio	**Total Measured**
China	90	260	52	**410**
Pittsburgh	90	33	45	**190**
Local	90	6	45	**150**

Table 1: Latency estimates (ms).

Although latencies in Table 1 seem high, it should be noted that human response time for critical listening and adjusting faders is on the order of hundreds of milliseconds [6]. While it might not be possible to play Soundcool modules like conventional instruments, this delay is reasonable for collaborative performances that involve cuing ambient sounds—but not in rhythm—and adjusting filters, reverberation, gain, and pitch. In fact, we have been performing with Soundcool over the Internet using Zoom to share screens and audio where the delays are as high as 500 ms (see for example https://youtu.be/kRp4SMfpL0Y?t=5532).

As an alternative to using a cloud server, we can run a local Soundcool Online server at home or in the classroom. However, even with no network delay, we can estimate Touch Screen delay plus Web Audio delay to be around 140 ms. (And this is confirmed with an actual measurement of 150 ms.) We can conclude that, compared to Soundcool implemented in Max/MSP, the Web Audio version adds about 50% to the latency, and putting the server in the cloud adds another 20%.

Note that audio latency varies among browsers and operating systems. Also, at least on recent MacBook Pro computers, approximately 10 ms can be saved by using an external audio interface as opposed to built-in audio. (www.cs.cmu.edu/~rbd/blog/latency-blog22sep2020.html). We used built-in hardware on the assumption that this would be typical for Soundcool users.

Further reduction in latency could be obtained in several ways. The "low-hanging fruit" from a technical perspective is Web Audio. Web Audio latency is a limiting factor for many applications and there is much room for improvement. Fortunately, the problems here are internal to browsers, so no special support is required from hardware, kernels, or device drivers, which already support low-latency audio on most platforms. There is interest in low-latency networking [14], and perhaps as audio communication moves increasingly to the Web, we will see improvements reaching consumers. Finally, the advantages of lower-latency touch screens have been recognized (https://www.theguardian.com/technology/askjack/2019/jul/25/which-is-the-best-tablet-for-an-artist) and we hope faster touch sensing will become commonplace.

5.2 Mirrored Soundcool Projects

In return for some additional latency, the cloud-based server solution offers an interesting opportunity for collaborative performance over the web. Instead of sharing a screen and low-quality audio over a conferencing system such as Zoom, we can run "clones" of a Soundcool project locally for every performer and for audiences too. Local copies of the project can compute and deliver high-quality audio. Collaborative control is achieved by sending copies of all control changes to every instance of the Soundcool project. Due to

variations in network timing, not every instance will compute exactly the same sound, but given the overall response time, small timing variations are not critical.

To enable this type of distributed performance control and synthesis, our server supports named *performances* which are projects with a set of reserved OSC port numbers. Players that want to collaborate open copies of the performance by name and then share all control changes. A reservation system ensures that other projects can use other port numbers for OSC control without interference. For example, we can have one student in China operating player module 1 and one student in Spain operating player module 2, and they hear approximately the same audio from their local browsers.

During performances, editing should cease and only shared control changes should be made. In principle, collaborative project editing is also possible, but we have not implemented this yet.

6. EXTENSIONS VS. BUILT-IN DSP

One of the limitations of Web Audio is the fixed set of DSP functions that are provided. Without plug-ins or dynamically loaded libraries, the extension mechanisms are limited. In the original Soundcool application, we often use multiple VST plug-ins, and Soundcool offers a VST Host module which makes this very easy.

For the Web Audio implementation, a Web Audio plug-in standard would be extremely helpful in providing a user-oriented extension mechanism for Soundcool Online. Buffa et al. propose a Web Audio plugin standard and describe prograss. [2].

Our current implementation uses only standard Web Audio Nodes for DSP with one exception: Our GranSynth granular synthesizer module uses Web Audio's ScriptProcessorNode (SPN) to enable filling the delay buffer and scheduling grains in the future to be implemented in JavaScript. We used SPN because the newer Audio Worklet implementation was not available on all browsers.

We found that we needed SPN buffer sizes of 2048 samples to avoid most dropouts, but even larger buffers do not eliminate all problems. Depending on the browser used, this adds a delay of either one or two buffer lengths (about 46 or 93 ms at 44100 Hz sampling rates for 2048 sample buffers). This latency is tolerable for this particular effect, but Audio Worklets combined with disciplined programming with WebAssembly should provide more reliable scheduling, less variability in execution times, and lower latency.

7. FUTURE WORK

So far, we have created a basic set of modules to support simple Soundcool projects. The original Soundcool application has many more modules such as Sequencer, Filter, Keyboard and Router as well as supporting synthesizers through the VST Host module. We expect to extend Soundcool Online with similar capabilities.

Soundcool also supports video processing. Live camera input, video playback, and even full-screen rendering have become quite practical in modern browsers, and we plan to add video modules to Soundcool Online.

We have already mentioned the possibility of performances where Soundcool projects are replicated in the browser of each performer to provide high-quality audio. We use existing Soundcool Apps (available at the Apple and Play Stores) on mobile devices as controllers, but since controls pass through the Soundcool Online web server, it makes more sense to re-implement the Soundcool App in the mobile device *browser* and make the connection through Web Sockets or other web technology rather than Open Sound Control. This would eliminate the need to download and install an App and to configure Open Sound Control addresses and ports. It would also enhance security, preventing someone from guessing OSC port numbers and injecting uninvited control changes to a performance.

Finally, we would like to support shared editing of projects during collaborative rehearsals and performances. At present, changing a performance patch requires everyone to stop and reload a new performance version.

8. CONCLUSIONS

Soundcool Online takes advantage of Web Audio and other Web technologies to implement a simple and powerful modular synthesizer. We have used Soundcool extensively in its original form as a laptop/desktop application. We believe Soundcool Online has the potential to reach a much larger group of users, especially young people for whom Soundcool was intended. The shared cloud environment of Soundcool Online is ideally suited to the collaborative nature of Soundcool, and we look forward to developing our current experimental installation of Soundcool Online into a very capable system for education and collaborative music performance.

9. ACKNOWLEDGMENTS

We would like to thank Manuel Sáez and Saúl Moncho for their early prototyping and testing contributions to the project and their work on the Soundcool Team. We also wish to acknowledge many contributions by supporters and members of the Soundcool project (soundcool.org). Soundcool has been partially supported by Generalitat Valenciana grants AICO/2020/151 and GJIDI/2018/A/169.

10. REFERENCES

[1] E. Brandt. *Temporal Type Constructors for Computer Music Programming*. PhD thesis, Carnegie Mellon University Computer Science Department, 2002.

[2] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin, and S. Letz. Towards an open web audio plugin standard. In *Companion Proceedings of the The Web Conference 2018*, pages 759–766, 2018.

[3] H. Choi. Audioworklet: the future of web audio. In *Proceedings of the 2018 International Computer Music Conference, ICMC 2018, Daegu, South Korea, August 5-10, 2018*. Michigan Publishing, 2018.

[4] F. Font, G. Roma, and X. Serra. Freesound technical demo. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, page 411–412, New York, NY, USA, 2013. Association for Computing Machinery.

[5] J. B. Freeman, B. Magerko, D. Edwards, R. Moore, T. McKlin, and A. Xambo. Earsketch: A steam approach to broadening participation in computer science principles. In *2015 Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*, pages 1–2, 2015.

[6] A. Jain, R. Bansal, A. Kumar, and K. Singh. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *Int J Appl Basic Med Res*, 5(2):124–127, May-Aug 2015.

[7] K. Kritsis, M. Bouillon, D. Martín-Albo, C. Acosta, R. Piéchaud, and V. Katsouros. imuscica: A web platform for science education through music activities. In A. Xambó, S. R. Martín, and G. Roma, editors, *Web Audio Conference WAC-2019*, WAC '19, Trondheim, Norway, 2019. NTNU.

[8] M. Puckette. Pure data: another integrated computer music environment. In *in Proceedings, International Computer Music Conference*, pages 37–41, 1996.

[9] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: Programming for all. *Commun. ACM*, 52(11):60–67, Nov. 2009.

[10] S. Rudnick. Synth kitchen. In A. Xambó, S. R. Martín, and G. Roma, editors, *Proceedings of the International Web Audio Conference*, WAC '19, page 143, Trondheim, Norway, December 2019. NTNU.

[11] J. Sastre, N. Lloret, S. Scarani, R. B. Dannenberg, and J. Jara. Collaborative creation with soundcool for socially distanced education. In *Korean Electro-Acoustic Music Society's 2020 Annual Conference Proceedings*, pages 47–51, 2020. http://computermusic.asia/2020/Proceedings.pdf.

[12] S. Scarani, A. Muñoz, J. Serquera, J. Sastre, and R. B. Dannenberg. Software for interactive and collaborative creation in the classroom and beyond: An overview of the soundcool software. *Computer Music Journal*, 43(4):12–24, Winter 2019.

[13] R. Shapiro, A. Kelly, M. Ahrens, B. Johnson, H. Politi, and R. Fiebrink. Tangible distributed computer music for youth. *Computer Music Journal*, 41(2):52–68, 2017.

[14] A. Singla, a. Chandrasekaran, P. B. Godfrey, and B. Maggs. The internet at the speed of light. In *HotNets '14*, 2014.