SonNet: A Code Interface for Sonifying Computer Network Data

KatieAnna E Wolf
Princeton University
Department of Computer Science
kewolf@princeton.edu

Rebecca Fiebrink
Princeton University
Department of Computer Science (also Music)
fiebrink@princeton.edu

ABSTRACT

As any computer user employs the Internet to accomplish everyday activities, a flow of data packets moves across the network, forming their own patterns in response to his or her actions. Artists and sound designers who are interested in accessing that data to make music must currently possess low-level knowledge of Internet protocols and spend significant effort working with low-level networking code. We have created SonNet, a new software tool that lowers these practical barriers to experimenting and composing with network data. SonNet executes packet-sniffing and network connection state analysis automatically, and it includes an easy-to-use ChucK object that can be instantiated, customized, and queried from a user's own code. In this paper, we present the design and implementation of the SonNet system, and we discuss a pilot evaluation of the system with computer music composers. We also discuss compositional applications of SonNet and illustrate the use of the system in an example composition.

Keywords

Sonification, network data, compositional tools

1. INTRODUCTION

Networked digital communications, including communication over the Internet, have a rich potential for use in the real-time control of sound and music. Data flowing over the Internet as a person browses the Web constitutes a complex, multi-dimensional information signal. Some aspects of this signal are under the user's control and might be used to intentionally influence sound in an expressive manner; the signal will also vary according to complex and unpredictable external processes, leaving open the possibility for serendipity and novelty. The sound installation *Ping* is one example: it generated sound based on the Round Trip Time (RTT) of "call and response" messages initiated by the "ping" network utility [4, 5]. Because each host produced RTTs with varying latency and jitter, viewers at the exhibit were able to influence the sound by selecting the remote hosts with which the system communicated. Other musical works exploring the use of networks as instruments can be found at the CROSSFADE[1] online exhibit.

[1]http://crossfade.walkerart.org/index.html

Network music and multimedia systems can also forgo direct user interaction and control over the network data, and instead rely on latent properties of the network to drive music and sound. An example of this approach is *Leech* [10], which visually and sonically renders BitTorrent traffic in an attempt to give audience members insight to music piracy as it occurs in the real world. Network sonification has similarly been used by technologists who may not have an artistic agenda, but who wish to use sound as a convenient means for network monitoring [2, 8] and supporting situational awareness [1, 9]. The *TresnaNet* application enables a user or performer to manipulate the parameters related to how the network data is used to generate sound [15].

To build systems like those above, the creator must write code from scratch, or modify existing extraction code, to gain access to the low-level networking data. Then additional code needs to be written to parse and analyze this data to obtain the information used to drive sound and music. To accomplish this requires a significant amount of coding effort as well as an intimate understanding of networking protocols. SonNet lowers the barrier for composers and developers to create these types of systems by automatically extracting low-level packet data and higher-level network information and exposing it to the user's sound-generating code via a simple code interface. In this regard, SonNet is similar to other music composition toolkits that seek to make computational techniques more useful to people without a preexisting deep technical background in an area, and to toolkits that seek to make the process of composition more efficient or satisfying by automating or abstracting processes that are not central to music creation. For instance, the *Unsupervised Play* toolkit for Max/MSP provides several machine learning algorithms and utilities, allowing computer musicians of any background to easily experiment with machine learning, and to do so without having to implement or port any learning algorithms themselves [14]. MoMu [3] and UrMus [6] are toolkits for musical interaction design for mobile phones. SMELT [7] provides lightweight tools in the ChucK programming language [16] for creating digital musical instruments that are controlled using built-in laptop inputs. Like SonNet, each of these toolkits has the effect of making a technique or technology usable by a wider group of users, and of making system implementation and experimentation easier for all users.

In the following sections, we present the design and implementation of the SonNet code interface, an evaluation and discussion of our current implementation, and an overview of some compositional applications of SonNet.

2. DESIGN

We designed our system to make computer network data easily accessible for composers. In this section we discuss the network data, the design details of our system to make that data accessible, and descriptions of the data our system makes available, divided into three abstraction levels.

2.1 Network Data

Our tool supports the sonification of data communicated over the Internet using the User Datagram Protocol (UDP) and Transport Control Protocol (TCP). As applications communicate with each other on the Internet, packets are sent over different protocols. For TCP and UDP each packet contains both a message "payload" (i.e., the data being sent from one application to another over the network) as well as other data fields, including information on where the packet came from and where it is going. Each TCP packet also contains fields with information about the connection on which the packet is sent, including information about the ordering of the packet on a connection, and data that signals changes in the connection as it is established, torn down, or reset.

2.2 Design Details

The raw packet-level data described above is typically obtained with a packet sniffer. As the data flows across a network, a packet sniffer captures each packet and decodes it to display the values of each of its fields. Without SonNet, a user desiring to use packet-level information in music performance or composition typically has to implement a packet sniffer or build on a third-party tool to access the packet information, and write code to track and analyze packet-level data over time to determine higher-level information about the connections on which packets are sent. Additionally, users have to implement the mechanisms for using this data to control sound. Although it is possible to build on a third-party packet-sniffing application or library such as Wireshark[2] or Carnivore[3], users still must write code to modify these tools to send the network information to a music environment in real-time (e.g., by sending OSC messages), and users still must have a low-level understanding of network protocols in order to navigate these tools and correctly compute network state information from the packet-level data.

SonNet simplifies the process of creating compositions and sonifications from network data. With SonNet, users no longer need to write code to access the low-level networking data or write code to track network state information from packet data; they must only write the system for generating sound and music. The SonNet application alerts an event handler in the user's code each time a new packet is sent or received on the network; SonNet also tracks packet-level data and higher-level network properties in real-time, and the user's application can query SonNet for this information at any time. For further information on the implementation of SonNet, see Section 3.

2.3 Levels of Abstraction

SonNet tracks network data at several levels of complexity and abstraction, from packet-level information to network connection state information. Figure 1 shows SonNet's approach to organizing all available data about network properties into three levels of abstraction, where subsequently higher-level properties are computed from lower-level properties, beginning with the raw packet information in the first level. Table 1 lists the SonNet variables used to represent packet and network properties, along with their descriptions and corresponding abstraction levels. In the following sections, we define each level of abstraction and describe the information that SonNet makes available at each level.

[2]http://www.wireshark.org/
[3]http://r-s-g.org/carnivore/

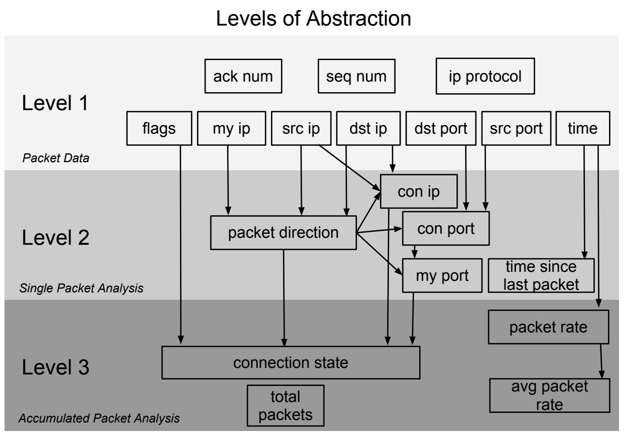

Figure 1: This figure displays the properties in each of the three levels of abstraction; the arrows show how fields at the lower levels are used to determine those at higher levels.

Variable	Network Information	Level
`src_ip`	source IP address	1
`dst_ip`	destination IP address	1
`my_ip`	IP address of user's device	1
`src_port`	source port in TCP header	1
`dst_port`	destination port in TCP header	1
`flags`	TCP control flags	1
`time`	time stamp on packet	1
`ip_protocol`	type of packet	1
`seq_num`	sequence number	1
`ack_num`	acknowledgment number	1
`direction`	direction of the packet	2
`con_ip`	IP address of connecting device	2
`con_port`	port number of connection	2
`my_port`	port number of user's device	2
`time_since`	time since last packet	2
`total_packets`	total count of packets	3
`state`	current state of the connection	3
`packet_rate`	number of packets per period	3
`avg_packet_rate`	running average of packets	3

Table 1: Variables available in the code interface, their relation to network packets, and their abstraction level.

Level 1

Level 1 contains the information carried by an individual packet. A unique IP address is assigned to each device that participates in a computer network using IP for communication [11]. The Level 1 data contains the source and destination IP addresses of the current packet, the source and destination port numbers associated with that packet, and the IP address of the user's device (`my_ip`). The type of IP protocol is also given as either TCP or UDP. For TCP packets, TCP control flags are used to control the state of a connection. The flag data is a Level 1 abstraction, however by analyzing these flags over multiple packets on the same connection, the state of a connection (a Level 3 abstraction discussed later) can be determined. Finally, the sequence number keeps track of how much data has been sent over the connection, and the acknowledgement number is used to inform the sending device that the data was received successfully. For more details about packet-level data, we direct the reader to the Request for Comments (RFC) pages for IP (RFC 791) [12] and TCP (RFC 793) [13].

Level 2

Level 2 contains information computed by analyzing the single-packet information in Level 1. Because packet-level source and destination IP addresses do not explicitly convey

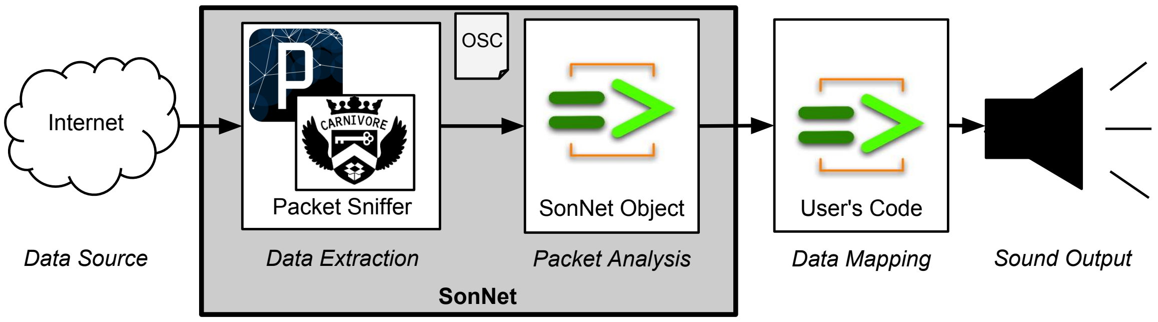

Figure 2: Packet information is taken from the Internet via the Carnivore Processing Library. A Processing sketch sends the packet information using the OSC protocol to the code interface written in ChucK. The program listens for OSC messages, stores and analyzes the packet data, and then sonifies it using the composer-defined mappings.

the packet's direction in relation to the user's device (incoming or outgoing), SonNet compares the IP address of the user's device to the source and destination IP addresses in the packet and stores the direction in the `packet_direction` variable. The same analysis determines the IP address and port of the connecting device (`conn_ip` and `conn_port`), and the port on the user's device (`my_port`). Level 2 also computes the time since the last packet.

Level 3

Level 3 aggregates information over multiple packets. SonNet uses its prior packet history and the `flags` of the current packet to compute the `state` of each new packet's connection (being set up, established, being torn down). Level 3 also updates a running count of the total number of packets and tracks the rate at which packets are flowing on the network. Two rates are updated when a packet arrives: `packet_rate` is the rate over a user-defined period (the default period is one second), and `avg_packet_rate` is a running average of the number of packets per period over a window of time (the default window size is five seconds).

3. IMPLEMENTATION

SonNet is an open-source and cross-platform application implemented using the Carnivore Processing library[3] and the audio programming language ChucK [16]. An overview of the SonNet system is depicted in Figure 2. Carnivore is based on Jpcap, a Java native interface for the libpcap standard sniffing library. Each time a packet is sent or received, the sniffer sends the data to the ChucK component of SonNet using the Open Sound Control (OSC) protocol [17]. The ChucK component populates the Level 1 variables with the packet data, and analyzes those variables to update variables at the higher abstraction levels.

Users can easily write music or sonification code in ChucK by instantiating and running a SonNet object in their own code, listening for the SonNet ChucK event that is broadcast each time a packet is sent or received, and using member variables of the SonNet object to access any of the network properties in Figure 1. The instantiation and configuration of the SonNet object requires the user to write only five lines of ChucK code, so the practical barrier to accessing network data is greatly reduced. Users working in other environments can easily interface with ChucK via OSC.

The user may easily customize a few properties of SonNet. By default, SonNet only tracks TCP packets and only processes packets sent via port 80 on the connecting device (where all HTTP traffic is directed). However, the user can opt to include UDP using SonNet's `include_udp` flag, and to include traffic on all ports using the `include_ports` flag. Additionally, the period and window sizes used for tracking packet rate in Level 3 can be modified.

The SonNet code and a demonstration video are found on the SonNet website: `http://sonnet.cs.princeton.edu`.

4. EVALUATION AND DISCUSSION

For an initial evaluation of the SonNet interface, we ran a pilot test with four computer music composers and students. Their level of exposure to computer networks varied from none to having taken an advanced graduate level course. Their level of ChucK programming experience also ranged from beginner to expert. Our main objective was to determine whether composers could use SonNet effectively to create a musical piece. We sat down individually with each participant, described the SonNet code interface, watched as they experimented with the system, and then asked the users about their interaction with SonNet.

While the composers had different backgrounds in computer networking, they all found the network data interesting, and they felt the SonNet interface made this data easily accessible. The two more advanced ChucK users were excited to explore the sonification possibilities, and in a short time, they were able to come up with unique mappings from the network data to sound. One particular user, who will be referred to as the expert user, spent an additional three hours working with the system to develop short sonification sketches, which we discuss in Section 5. The two participants with limited ChucK knowledge found it more difficult to develop the sonification, and they requested additional example code to illustrate sound creation in ChucK.

All users expressed interest in having additional infrastructure to assist in understanding how the network properties could be used. The expert user, who had a minimal amount of computer network experience, wrote his own ChucK code to print out the values of SonNet variables in order to understand their behavior as he browsed specific sites. In order to help streamline the understanding of the variable fields, he proposed developing a visual interface to display the state of all variables of a packet, which he could monitor to understand how network properties changed as he browed different websites. Additionally, several composers suggested adding new variables to SonNet, including the amount of data contained in the packet, the packet header size, and the time-to-live and header checksum fields.

Based on users' feedback, our next steps will be to add additional data fields to SonNet, to add simple visualization of network information within the Processing interface, and to enable richer OSC mechanisms for query and notifications for improved integration with arbitrary sonification platforms (e.g., Max/MSP). This includes implementing the Level 2 and 3 packet analyses within the Processing sketch. We also plan to employ the next iteration of the system in an undergraduate course in order to evaluate the usefulness of SonNet in teaching students about computer networking.

5. COMPOSITIONAL APPLICATIONS

In this section we discuss SonNet-based compositions by the expert pilot study user and this paper's first author, each of which takes a different approach to the musical use of network data. Demonstrations of these compositions appear in a video on the SonNet website.

The expert user created three musical sketches that can be used to sonify any website, giving a performer the ability to browse the web and explore the sonification changes from one site to the next. Here we describe the user's third sketch, *Varying Rhythmic*, which is built upon concepts the composer developed in his first two sketches, and which utilizes a musical technique called *isorhythm* (arranging fixed pattern of pitches with a repeating rhythmic pattern). For each packet, the 5 TCP flags and the 4 numbers the source and destination IP addresses create a 5-note rhythm and a 4-chord sequence. Stepping through this isorhythmic sequence, the beat rests if the flag is 0 and plays a chord if it is 1. The two pitches of the chord depend on one IP address number field from each of the source and destination IP addresses. As packets are sent and received, repetitions in the IP address and flags variables create melodies that repeat as a performer communicates with the same web domain. Other packet information influences sound properties such as the decay time of sound envelopes, the pitch ranges of the IP address sonification, the rate the notes are being played, and the number of times the rhythm repeats.

This sketch demonstrates how the expert user was able to use his compositional skills to develop interesting sonifications of the network data. The expert user intends to continue developing these sketches into full compositions or performance pieces, but this is not the only way he plans to use SonNet for composition. His sketches generate interesting chord sequences, which he plans to use as inspiration in his acoustic compositions.

The first author of this paper explored a different approach to web browsing sonification. A website "installation" designed by the author is populated with hyperlinks to specially-chosen other sites. SonNet distinguishes among traffic between the user's machine and any of those sites, and each site's data is mapped to sound in a unique way. This website installation was inspired by, and revolves around, Sergei Prokofiev's *Peter and the Wolf*, Op. 67. When a user visits the installation with the SonNet *Peter and the Wolf* application running in the background, clicking on images of the various characters in the story brings the user to different sites, and the data from each site renders the musical theme associated with that character. The synthesis of each theme is modified according to properties of the network connection between the user's computer and the character's website. For instance, a bowed-string physical modeling synthesis algorithm represents Peter, and the position of the "bow" is driven by the state of the connection. The SonNet website contains links to the *Peter and the Wolf* website and to download the sonification code.

6. CONCLUSIONS

While current creative applications using network data require the composer to have an intimate knowledge of network information, and to write code to tap into that information, SonNet allows users direct access to the data via a code interface. Through an initial pilot study, users found the code interface easy to understand and the data easily accessible. One expert user even spent additional time with the system developing sonification sketches, which he is interested in developing and utilizing in the future for both digital and acoustic compositions. Our future work includes adding more data fields to SonNet, creating a visualization of the network data, moving packet analysis into Processing, and enabling richer OSC mechanisms to improve integration with other sonification platforms. These changes will make network data even more easily accessible to a wider range of users, allowing them to explore an area of musical creation that may have otherwise been inaccessible.

7. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] M. Ballora, N. A. Giacobe, and D. L. Hall. Songs of cyberspace: An update on sonification of network traffic to support situational awareness. In *SPIE Proc. on Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, 2011.

[2] M. Barra, T. Cillo, A. D. Santis, U. F. Petrillo, A. Negro, V. Scarano, T. Matlock, and P. P. Maglio. Personal webmelody: Customized sonification of web servers. In *Proc. ICAD*, 2001.

[3] N. J. Bryan, J. Herrera, J. Oh, and G. Wang. MoMu: A mobile music toolkit. In *Proc. NIME*, 2010.

[4] C. Chafe and R. Leistikow. Levels of temporal resolution in sonification of network performance. In *Proc. ICAD*, 2001.

[5] C. Chafe and G. Niemeyer. Ping. In *010101: Art in Technologial Times, exhibition catalog*, pages 54–55, San Francisco Museum of Modern Art, 2001.

[6] G. Essl. Urmus: An environment for mobile instrument design and performance. In *Proc. ICMC*, 2010.

[7] R. Fiebrink, G. Wang, and P. Cook. Don't forget the laptop: Using native input capabilities for expressive musical control. In *Proc. NIME*, 2007.

[8] R. Giot and Y. Courbe. Intention: Interactive network sonfication. In *Proc. ICAD*, 2012.

[9] F. Kilander and P. Lönnqvist. A whisper in the woods: An ambient soundscape for peripheral awareness of remote processes. In *Proc. ICAD*, 2002.

[10] C. McKinney and A. Renaud. Leech: Bittorrent and music piracy sonification. In *Proc. SMC*, 2011.

[11] L. L. Peterson and B. S. Davie. *Computer Networks, Fourth Edition: A Systems Approach.* Morgan Kaufmann Publishers Inc., 2007.

[12] J. Postel. Internet Protocol. RFC 791 (Standard), Sept. 1981. Updated by RFCs 1349, 2474.

[13] J. Postel. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168, 6093, 6528.

[14] B. D. Smith and G. E. Garnett. Unsupervised play: Machine learning toolkit for Max. In *Proc. NIME*, 2012.

[15] P. Ustarroz. Tresnanet: Musical generation based on network protocols. In *Proc. NIME*, 2011.

[16] G. Wang and P. R. Cook. ChucK: A concurrent, on-the-fly audio programming language. In *Proc. ICMC*, 2003.

[17] M. Wright, A. Freed, and A. Momeni. Open Sound Control: State of the art 2003. In *Proc. NIME*, 2003.