Network sonification

https://github.com/mrkva/Sonodump

Jonáš Gruska
Sonology Royal Conservatory, Den Haag

June 18, 2011

1 Introduction

The project of network sonification is my contribution to the mass of sonification attitudes and works. My code is still a work in progress, therefore not all technical issues are solved, nevertheless I will try to describe the main principles to give you a clearer idea.

2 Inspiration

When I started to work on this project, I was of course inspired by the whole branch of sonficiation. Pieces like "Spherics" by Alvin Lucier or work of Michael Takezo Chinen had a strong influence on my perception of data streams as sound material. Generally, the concept of sonification may seem to many contemporary composers as overcome and boring, yet I have decided to give it my own try.

3 Early beginnings

Idea of sonificating of wireless networks came to me 2 years ago. I was internet-less, and looking for an (illegal) ways of acquiring access to some of password protected networks in my neighborhood. On this mission I came to program called KisMac, Mac OS X implementation of *nix Kismet. This program basically allows user to sniff for networks, their MAC address and most importantly - data packets. After certain amount of data packets is stolen, it allows to "crack out" packets and get the password of the network.

KisMac has also other function except malicious ones, which interested my musical part. It contains a simple audio interface, which "sonifies" the events on networks. Like when good packet is received, when new network is found and so on.

At that point I was without any C knowledge and had no idea how to get those events and use them for my needs. But, it was definitely inspiring.

Idea of live stream of events, so close to my everyday realm, surrounding – yet invisible.

My friend advised me to try the already mentioned Kismet. Kismet is a powerful terminal application allowing similar things as Kismac. It has one more feature - while sniffing, it creates special server, which then sends out the information about networks. I was dreaming about connecting my programming language (as Pure Data) to this server and processing those events, but didn't found any good documentation.

4 Experiments with tcpdump

Tcpdump is an is a common packet analyzer that runs under the command line. It allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network to which the computer is attached. It is used mainly by network administrators, security consultants and hackers.

My first try was to get some "sniffed" data to the file and try to play it as audio file. For this I simply used `tcpdump -i en1 -w test`, where I set it up to listen on `en1` device – wireless adapter and to write down everything to file `test`.

For conversion of this raw information, I have used Audacity. Audacity has built-in feature consisting of importing of the raw data. In the dialog, user is asked to setup the predicted (or desired - depends on the point of view) sampling frequency, encoding (PCM, float, etc.) and byte order (ordering of the subcomponents of the data item).

Using this feature I have imported my `test` file and set up Audacity to read it as WAV file with sampling frequency of 24000 Hz. For the rest I have left default values. I have attached the resulting file to this document as `tcpdump_example.m4a`.

5 Libpcap - library for packet capture

As the title reveals, libpcap is indeed a library for TCP/IP packet capture designed by the same group of people currently developing tcpdump. Designed for use with C/C++, but available as port for other languages (e.g. Python – pylibcap).

When I found out about this library, it was clearly the way for my project. It allowed me to implement tcpdump functionality to my own audio program and process the data very directly.

6 Programming

The programming part consisted of two parts, and both of them are still in progress. First part was to get the program to sniff desired data and second was, obviously, to transform – sonify the data for my musical purposes.

First part was rather easy and well documented on the various security and hacker websites, because the core of the program acts like a basic sniffer, which is the essential network-security tool.

In short, this part opens the the stream from the network device using `pcap_open_live()` and consequently loops trough it using `pcap_loop`. In the end, one can receive the stream in `packet[]` array.

Second part, the sonification is very primitive so far. I just basically get the packet data and play them as audio data at desired sample rate. This part needs to be extended, but I didn't managed to get more advanced yet.

Third part (which I didn't mentioned, but is also important) is the console GUI, which I take care of by using ncurses library. I have also added couple of fool-proof checks which limit user for doing non-safe or non-sense operations.

7 Conclusion

As you can see, the project is still in progress. I am planning to continue to work on it in future and hopefully perform as well. So, please excuse my errors and do not try the software with high amplification, it still behaves quite wildly.