Music-Defined Networking

Mary Hogan*
Princeton University
mh43@cs.princeton.edu

Flavio Esposito
Saint Louis University
flavio.esposito@slu.edu

ABSTRACT

For several years researchers have used the term "network orchestration" as a metaphor. In this paper, we make the metaphor reality; we describe a novel approach to network orchestration that leverages sounds to augment or replace various network management operations. We test our Music-Defined Networking approach with both a real and a virtual network testbed, on several mechanisms and applications: from datacenter server fan failure detection to authentication, from load balancing to explicit congestion notification and detection of heavy hitter flows. Our approach can be used with and without a Software-Defined Network controller. Despite its limitations, we believe that sound-based network management has potential to be further explored as an effective and inexpensive out-of-band orchestration technique.

1 INTRODUCTION

Delivering management traffic is essential to operate and orchestrate network services. In existing datacenters, for example, thousands of servers, storage units or switches run a vast plethora of operations and management tasks: from simple device booting, restart or configuration, to complex and computationally expensive anomaly detection, intrusion detection systems, monitoring and diagnostics.

To tame such complexity, many Software Defined Network (SDN) solutions, as well as creative network and traffic engineering designs, have been proposed; see *e.g.*, [7, 17, 23]. Despite those advances, management traffic is still carried in-band with data plane traffic both inside and out of datacenters. It is common for operators to isolate management traffic with VLANs or other forms of slicing. However, even with such logical separation, sharing the infrastructure for data and management traffic is risky [10, 41], since data plane or hardware failures could cut off network management traffic as well, aborting important management tasks such as diagnostics, intrusion detection systems, congestion notification or recovery signals. Previously, researchers have shown how an out-of-band management network could be desirable to reduce the growing complexity of datacenter operations [10, 14, 16, 18, 40, 41]. Some of these approaches have been criticized, however, as deploying them may result in significant costs or changes to the datacenter infrastructure. For example, they may require construction of a parallel network, which must support a prohibitively large port count to reach all data and control devices [10]; other obstructions include the installation of reflective surfaces [14, 16, 18, 40] or a raise of the ceiling level [18, 40]. In addition to these concerns, WiFi management networks are limited by coordination between transmitters and interference [41]. Other solutions that reduce out of band costs by using the existing infrastructure require installing expensive equipment per server rack [41].

In the past, requirements for out-of-band management networks have focused, at best, on three main design criteria: (*i*) *reliable*, *i.e.*, the management network should survive (datacenter) failures and faults, so that recovery and diagnostics can always be performed; (*ii*) *scalable*, so that all devices in a datacenter can be reached, and (*iii*) *deployable*, that is, a practical out-of-band network should be compatible with existing equipment [10]. Aside from these wise requirements, we believe a management network should also be (*iv*) *simple*, *i.e.* able to run without major interventions to the existing (datacenter) infrastructure, such as inserting complex logic to a switch, and (*v*) *inexpensive*. To our knowledge, none of these five design principles have been simultaneously considered for an out-of-band management network.

To this aim, in this paper we propose Music-Defined Networking, a paradigm where several network mechanisms can be programmed in response to specific sound sequences, *i.e.*, music, coming from real or virtual devices. We explore both active applications, in which we program network devices to emit a certain sound, and passive applications, where we monitor sounds produced by devices to identify when they (may) have failed. Using low-cost speakers, microphones and Raspberry Pis, we augment with sound capabilities datacenter components, such as real and virtual switches, hosts, SDN controllers. With a local testbed and with sound measurements in a real datacenter, we demonstrate how Music-Defined Networking can provide a low-cost out-of-band management network. Such a network is able to replace existing management operations, hence limiting management traffic, or to provide additional information, *e.g.*, detecting hardware failures.

The rest of the paper is organized as follows. Related work is discussed in the next section. In Section 3 we discuss the details of both our real and virtual network testbeds that we

*Work done while at Saint Louis University.

HotNets-XVII, Redmond, WA, USA

DOI: 10.1145/3286062.3286085

use to demonstrate several active music-defined operations in which a device emits sounds to in response to an event. In particular, in Section 4 we show how sounds, if played in the right sequence, can be used as an (additional) out-of-band authentication mechanism, or to implement any finite state machine for network state processing. In Section 5 we show how sounds can be an effective measurement tool by implementing two use cases, (*i*) a heavy hitter detection and (*ii*) a port scan detection. In Section 6 we show how we can perform music-defined traffic engineering by mapping specific sounds to different queue sizes in different switches. In Section 7 we show an example of passive sound-based network management in which we listen to a hardware component to detect its (imminent) failure. Sounds coming from a server cooling fan can be encoded and decoded using the Fast-Fourier Transform (FFT) [32], and their absence due to a hardware failure can be easily detected despite the datacenter background noise from other fans. We then conclude our work in Section 8 discussing limitations and a few Music-Defined Networking research directions.

2 RELATED WORK

Acoustic data transmission. The idea of using sound waves to transfer information has been floated before [19, 20, 27, 31, 33, 34]. With the aim of transferring data messages, audio-based networking has been presented as an alternative form of communication. In [27], for example, audio networking was utilized for both short- and long-range data transfer. Acoustic waves have been used for underwater communication [33, 34]. The same principles used underwater have also been adapted to the air medium [15, 15, 19, 25, 31]. However, given the low throughput capability (it can take up to six seconds to send a 20 bytes packet over a single hop [19]), effective acoustic transmission has been limited to replacement of magnetic induction in *Near Field Communication* [31]. Differently from all these *sound* approaches, our focus in this paper is instead on the control and management plane, not on the data plane.

Acoustic insecurity. As sounds are often overlooked when securing systems, researchers have developed several forms of acoustic attacks. For example, audio signals can facilitate covert communication with ultrasound frequencies [19], or they can be leveraged to exploit security vulnerabilities, both by listening to sounds produced by devices and by generating sounds to gain control over devices [15, 25]. These attacks detect the different acoustic emanations generated by processors based on the operations being performed. By eavesdropping on these sounds, researchers have been able to manipulate emanations to transmit data [25], to attack cryptographic schemas [15], and to reconstruct user information [3, 4]. In addition to gaining information from a system, audio signals can also be used to trigger unexpected and unwanted behavior in mobile devices [20] and in popular speech recognition systems such as Apple Siri and Amazon Alexa [39].

We also consider sound signals to actively or passively control devices for security applications, for example, to open

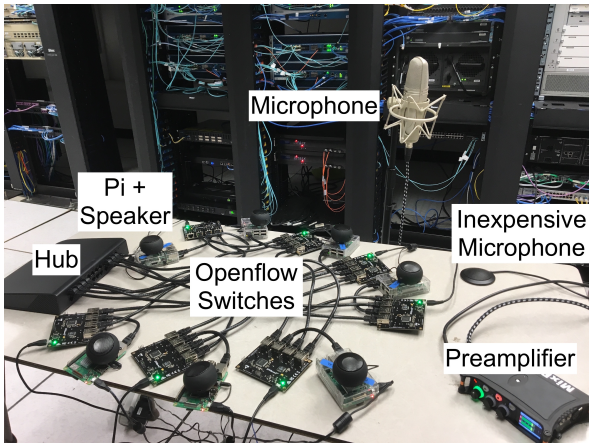
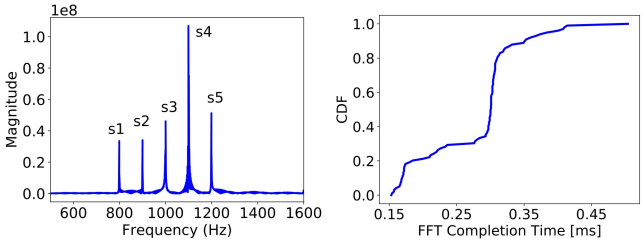


Figure 1: Music-Defined Networking testbed: We extend the firmware of the Zodiac FX switches so they could send our Music Protocol messages, which trigger signals to be played by a connected host. An application listening for sounds then interprets the sound sequence (music) and launches the appropriate action, *e.g.*, send an OpenFlow Flow-MOD message (§ 6) or open a previously closed port on a given switch (§ 4).



(*a*) FFT of audio from 5 switches (*b*) CDF of FFT processing time

Figure 2: We use the Fast Fourier Transform to process multiple sounds captured by the listening device and to identify the frequencies played by a switch.

a previously closed switch port (§ 4) or to detect a denial of service or a (naive) port scanning attack (§ 5). Differently from these papers though, this work is the first attempt, to our knowledge, to use sound signals, with single-tone alerts or in a well-defined sequence (music), to trigger or execute a wide range of network management operations, such as active queue management and load balancing (§ 6), access control (§ 4) or hardware failure detection (§ 7).

3 TESTBED

To establish the practicality of our approach, we built two testbeds, one using real switches and one using virtual switches.

Since switches today do not have built-in speakers, we connected a Raspberry Pi to each switch in order to generate sounds. In particular, we connected 7 Zodiac FX switches (whose cost is currently under 80 USD) to 7 Raspberry Pis. We attach the Pi to a port on the FX switch; each Pi is connected to an inexpensive speaker (Figure 1).

We modified the firmware of the Zodiac FX switches [1], so that when we want the switch to play a sound, a *Music Protocol* (MP) message is sent to the Pi. The MP payload

[1] https://northboundnetworks.com/products/zodiac-fx

contains the frequency at which we want to play the sound, its duration and intensity (volume). To support MP message marshaling on the Zodiac FX switches, we had to disable OpenFlow on the switch Ethernet port connected to the Pi.

Two major limitations of the Zodiac FX switches forced us to implement some of our use cases on a virtual network testbed using Mininet [24]: (*i*) the RAM is limited to 120KB and (*ii*) multi-packet queues are not supported (only a single packet can be sent at once). The limited memory also forced us to use the *raw API* of the Lightweight IP stack [2] to send messages.

Sound length, duration and intensity can be treated as a policy to allow programmability of the music-defined mechanism to be deployed. Ranges for these policies are dictated by the hardware capabilities of speakers and microphones.

In our Music-Defined Networking testbed, we empirically found that a distance of approximately 20 Hz between frequencies is needed to accurately differentiate them. Each switch in our testbed was assigned a unique set of frequencies, so that we can identify sounds played by different switches at the same time (Figure 2a). Although we only tested one application at a time, it is possible to support multiple MDN applications simultaneously, as long as each task uses a different set of frequencies and the listening application knows the frequency mappings. The minimum intensity to detect a sound clearly depends on the distance from speaker to microphone, but in our experiments we played sounds of at least 30dB. Normal conversation is on the order of 50 dB [30].

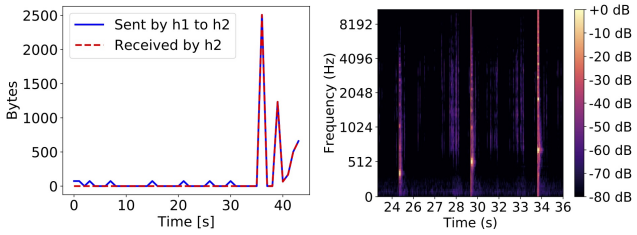
We also found that, although sound duration varied between devices, the shortest possible length generated in our testbed was approximately 30ms. The length of the sound dictates how long it takes the Music-Defined controller (or the listening application) to process the sound; smaller sounds means we can listen for shorter durations, and smaller samples take less time to process. Figure 2b shows the distribution of FFT processing time for audio samples of approximately 50ms; as we can see, approximately 90% of our samples were processed in 0.35ms or less.

We tested our applications with and without background noise. In both cases, we could accurately distinguish the sounds from switches. The level of noise may, however, grow significantly based on other applications, as well as on full-duplex sound communications (that we did not implement). Scaling an MDN application to even a medium size datacenter may result in environments that are even more uncomfortable for operators, who must already wear noise canceling headphones. We believe that accurately tuning sound parameters to manage sound interference, mitigate operator discomfort and support multiple MDN applications is an interesting research direction.

4 STATE PROCESSING

By design, Software-Defined Networking focuses on a centralized controller governing stateless switches [29]. Other

[2]https://github.com/dreamcat4/lwip.



(*a*) Bytes sent/recvd (*b*) Mel-scaled spectrogram

Figure 3: Port knocking: the controller receives 3 sounds in a correct sequence, each corresponding to a port number (as a form of authentication), and allows TCP traffic on a specific previously closed port.

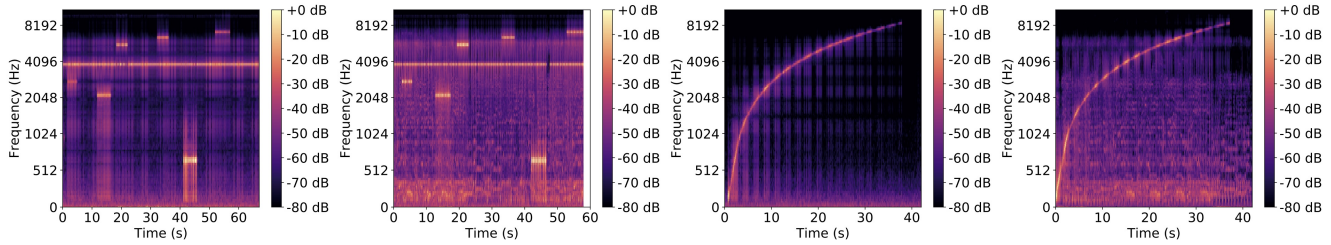
work, *e.g.*, [9] proposed inserting data plane state machines on switches, so they could run some of the functionalities achieved today through middleboxes. Data plane states are probably impossible to maintain (today) with sounds, since playing a sound every packet received seems unfeasible with current hardware and datacenter traffic rates. Management plane states instead have larger timescales, so sounds can be processed inside routers, switches or other middleboxes, or within the application process running the logically centralized SDN controller. It simply depends on where we attach speakers and microphones.

As a state processing example, we implemented a *port knocking* finite state machine, similar to the one presented in OpenState [9]. In our implementation, however, states are stored in a Music-Defined Network (MDN) controller attached to the Zodiac FX switches, not in the switches themselves. In particular, the controller keeps track of what sounds it has heard thus far from the switch; each sound is then mapped to the destination port number received by the switch. In the controller, we know what frequencies are associated with each port for a switch, so we know which frequencies to listen for. Once we hear the frequencies in the correct sequence, we allow traffic to be forwarded by adding a flow table entry at the switch. The match that specifies the port opening event depends not only on the sound (and so on packet header information), but also on the current state of the finite state machine; an incoming packet with port x is associated to a forwarding action when the port is open, but to a drop action when the system is in any other state.

In Figure 3*a* we show how the sender is attempting to send packets on the port to be opened for about 34 seconds (blue continuous line). After the third sound has been interpreted in the correct combination by the state machine, (see Figure 3b) the port is opened and all traffic sent by host 1 is received by host 2 (red dashed line in Figure 3a).

5 MUSIC-DEFINED TELEMETRY

Operators continuously monitor traffic to track events ranging from performance limitations to attacks. This monitoring requires continuous, real-time measurement and analysis — a process commonly referred to as network telemetry [37]. The introduction of SDN has permitted the deployment of new



(*a*) Heavy hitter detection (*b*) Noisy heavy hitter detection (*c*) Port scanning detection (*d*) Noisy port scanning detection

Figure 4: Music-Defined Telemetry: (a-b) Heavy hitter detection. (a) switches are programmed to play a sound based on the hash of the flow. (b) Same as (a) but while playing Sia's Cheap Thrills, a popular song, as random background noise. (c-d) Port scanning detection: real switches are programmed to play a sound based on the destination port number. The MDN controller is programmed to listen to a set of available port numbers. (d) Same as (c) but with Sia's Cheap Thrills, a popular song, as random background noise.

centralized network solutions that have improved many network operations. The majority of these solutions rely on the assumption that a centralized controller collects and merges measurements from different monitoring points to obtain a network-wide view. This task is challenging when the same packet may pass through several monitoring points, as packets can be double-counted [5]. Existing measurement solutions either assume that each packet is measured at a single monitoring point or that the routing of each packet is known by the controller. Another active measurement approach is to mark the sampled packet so that other measurement points do not process it, as they are aware that the packet was already considered.

In this section we demonstrate with two use cases that our Music-Defined Networking approach can be an effective strategy for several network monitoring tasks and even for detecting traffic anomalies, such as misconfigurations and (some naive) attacks. By assigning each network component to a different set of sound frequencies, we can accurately perform, with a fairly fine-grained granularity, measurements that are (*i*) passive, *i.e.*, do not require traffic modification in any way, (*ii*) routing oblivious, (*iii*) have very flexible placement of the measurement point (the set of microphones and speakers), and (*iv*) are network topology oblivious.

We focus on demonstrating two Music-Defined Telemetry use cases: one for monitoring purposes (heavy hitter detection) and one for security (port scanning attack detection). Both are shown in Figure 4. By heavy hitter detection we mean the identification of a flow that consumes more than a fraction of the link capacity during a given time interval.

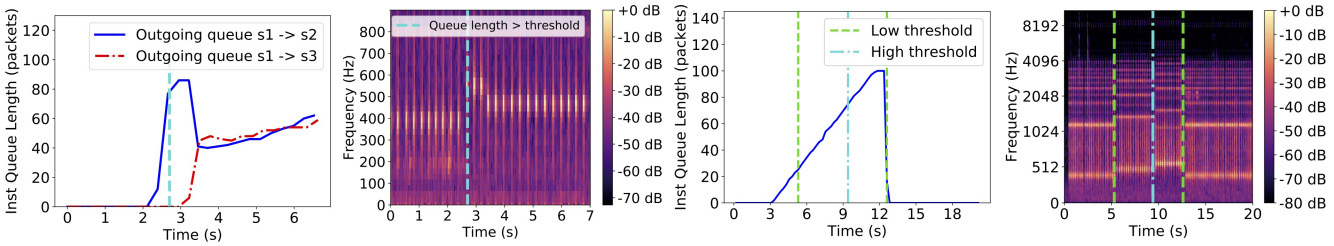
Heavy-hitter detection. To detect a heavy hitter flow, we hash a flow tuple defined by source port, destination port, source IP, destination IP and protocol type [22] and map it to a given frequency. We then demonstrate how our Music-Defined Network controller, or any application process capable of listening and processing sounds, can recognize when a sound with a similar frequency is played more than a threshold in a given time interval (Figure 4a-b), with (b) and without (a) random background noise.

Heavy hitter detection may be a far more complex task than detecting sound frequencies, and could require complex algorithms [35], sampling or sketching techniques [26, 38], that may operate in the data plane [35] or in the control plane [26, 38].

We remark that scalability is also a concern: there may be thousands of active flows per minute on an ISP backbone link or a datacenter top-of-rack switch [6]. Despite the number of distinct feasible frequency-flow mappings that we can perform with today's microphones, even including ultrasounds we would probably not be able to detect every single flow. To this end, we do not claim that Music-Defined Telemetry is a scalable replacement for all these complex solutions, but we believe that it could be suitable for smaller sized networks, or to offload some of the measurement tasks within a datacenter, reducing their input size or increasing their processing speed. Moreover, Music-Defined Networking could be a viable solution for other telemetry applications that do not need single-level flow scalability. With our inexpensive testbed hardware alone, we could distinguish up to 1000 distinct frequencies played simultaneously only considering the human-hearable frequency range. An average size Internet Service Provider controls several Autonomous Systems; counting the distinct number of source addresses who send traffic to a set of destinations, merely using sounds, seems conceivable if we merely map AS source-destination identifiers to frequencies.

Port Scanning. As another telemetry example, this time with focus on security, we implemented a port scanning attack detection (Figure 4c-d.) In particular, we generated a port scanning attack on a host, forcing traffic to pass through the same real switch. When hit by a packet, the switch plays a sound whose frequency is based on the destination port number. As we can see in Figure 4c, the port scan can be identified by a clear logarithmic line on the Mel-scaled spectrogram. The log is merely given by the Mel-scale on the y-axis. Even in this case we repeated the experiment adding random noise (Figure 4d).

Similar to the discussion above for heavy hitters, we admit that detecting a port scanning attack may be a far more complex operation. Its detection may require network-wide knowledge [38] and more in-depth analysis [8]; although we believe that it is possible to detect naive port scan attacks, sound analysis alone is surely insufficient. Other applications, however, may be more feasible and may alert network



(*a*) Queue length (*b*) Mel-scaled spectrogram (*c*) Queue length (*d*) Mel-scaled spectrogram

Figure 5: (a-b) Load balancing application: when the MDN controller hears a sound associated with an overloaded queue, it installs a new FlowMOD rule to split traffic across two ports. (a) shows the queue length evolution, (b) shows when we play the sound that tells us the queue is congested (marked with the vertical blue line). (c-d) Queue size monitoring application: ¡25 pkts in queue play 500Hz, 25¡pkts¡75 play 600Hz, ¿75 pkts play 700Hz. Frequency values in the spectrogram are normalized by the mel-scale.

operators to events that call for further investigation. For example, detection of Distributed Denial of Service (DDoS) via *k-superspreaders*: a *k*-superspreader is a host that contacts more than *k* unique destinations during a time interval. A DDoS victim is a host that is contacted by more than *k* unique sources. By mapping destination addresses to frequencies, we can presumably detect k-superspreaders and hence a DDoS. We leave that as an open problem.

6 TRAFFIC ENGINEERING

Traffic engineering solutions can be classified according to two main design dimensions: one that focuses on the choice of forwarding paths [11, 12], and another in which sending rates are dynamically adjusted to balance incoming traffic flows [21, 28]. Some recent solutions even use a combination of the two [23]. In this section we show how sound can be used as an effective signal to trigger any traffic engineering approach, whether for traffic steering or to respond to congestion events.

Load balancing. In particular, as a proof of concept, we implemented a music-defined load balancing application on a virtual network testbed. We attach to an MDN controller four switches connected in a rhomboid topology, with the two hosts attached to two opposite vertices of the rhombus. The source host continuously sends traffic with a progressively increasing rate to the destination, initially using a single path. Every 300ms, each switch is programmed to send a sound whose frequency depends on the number of packets currently in the switch's queue. Switches whose queues have less than 25 packets play a sound at the lowest frequency; a higher sound is played if the queue has a number of packets between a low threshold (which in our implementation was set at 25 packets) and a high threshold (set at 75 packets) and the highest sound is played when the switch is (getting) congested *i.e.* the queue has more than 75 packets (Figure 5b).

When the MDN controller application hears a sound associated with an overloaded switch (in our experiment, at time 3.7s), it sends an OpenFlow flow-MOD message so that the source traffic gets split across two ports, balancing the traffic load across the two different available routes (Figure 5a).

Switch Congestion Monitoring. As another use case application, we show how MDN can be used to detect thresholds on queue size. This in turn can be used to drive in-network flow or congestion control decisions, without waiting for source reactions, without having to modify the transport protocol, as in DataCenter TCP (DCTCP) [1], and without using the less efficient Explicit Congestion Notification (ECN) mechanism of TCP. DCTCP has been shown to have greater performance but fairness and convergence drawbacks [2].

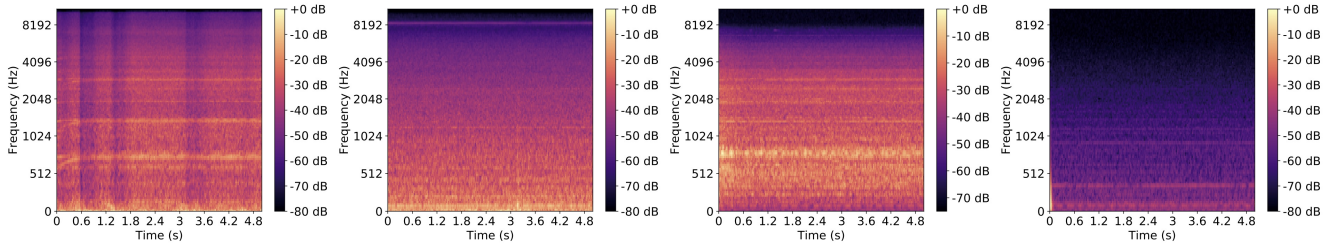
In Figure 5c-d, we show a simple use case in which we have a virtual switch notify the MDN controller with a sound at progressively higher frequencies, depending on how many packets are in its queue.

In our experiments, we send data traffic through the switch and we measure the instantaneous queue length using the traffic control Linux utility `tc` every 300ms. We then play a different sound based on the number of packets in the switch queue; the MDN controller is programmed to listen for those specific frequencies, so if it hears a frequency it recognizes, it knows the range for the number of packets in the queue (and can then make a congestion decision based on that). After all traffic has been sent to the destination, the queue size gets again lower than 25 packets and the controller is notified with another sound at a lower frequency (500 Hz).

7 SERVER FAN FAILURE DETECTION

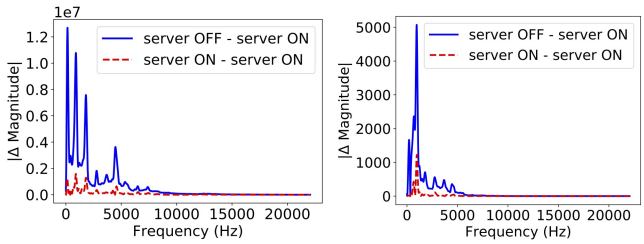
Detection of malfunctioning hardware is a key element of network management. In this section, we show how passively listening to sounds can be an effective out-of-channel mechanism that may avoid severe hardware failures. In datacenters today, these failures are a major financial risk, and the efficiency of countermeasures is far from ideal [13, 36]. To summarize a recent study by Wang et al. [36] that looked at 290,000 hardware failure reports collected over the past four years, automatic hardware failure detection and handling have potential to be very accurate, significantly reducing human labor, "*but we need to improve these systems, especially in "bad spots" where the failure rate is higher.*"

Fire alarms are a widely adopted disaster countermeasure in today's datacenters. Aside from reacting to problems when it is already too late, fire alarms can cause side effects [13]. Personal communication with experienced IT operators exposed us also to recent stories of ineffective emergency responses in which servers that were powered off due to an emergency



(*a*) Datacenter: server ON (*b*) Datacenter: server OFF (*b*) Office: server ON (*c*) Office: server OFF

Figure 6: Sound waves of a single server are detectable despite the datacenter noise: mel-scaled noise spectrograms of a server with (a-c) and without a functioning fan (b-d) in a datacenter (a-b), and in an office environment (c-d).



(*a*) Datacenter (*b*) Office

Figure 7: Fan noise is processed to detect a failure: after a difference of amplitude is detected we launch an out-of-band signal alert.

were immediately restored by Uninterruptible Power Supply (UPS) units.

In this section we show an inexpensive countermeasure for datacenter failure detection that operates by monitoring the sound of server fans and detecting when one has failed. To this aim, we set up several sound-listening scenarios with different types of microphones (from very cheap to fairly expensive), and we test their ability to detect fan failure with different background noise levels.[3] The open question we explore is: *Can we detect the failure of a single server despite the typical datacenter noise?* [4] After capturing the noisy signal, we were able to answer positively to this question with a closely placed microphone (Figures 6 and 7).

To identify failures, we find the total amplitude of each frequency in recorded sounds with a server fan both on and off; we obtain such amplitudes by computing the FFT of each given sound sample. We then use these amplitudes to classify the state (health) of the fan. The difference in amplitude for certain frequencies is considerably larger when comparing two audio signals of the fan on and off (blue continuous line in Figure 7) than when comparing two samples of a functioning fan (red dashed line in Figure 7). This is reflected in the spectrograms reported in Figure 6: when the fan is operating, the specific frequencies it generates have noticeably greater amplitudes than when the fan is off. Two interesting questions that remain open are: *(1) How many distinct server anomalies can we recognize and (2) what is the optimal microphone-server distance to be able to correctly distinguish multiple fan signals?*

[3]Note that some servers in modern datacenters may be equipped with temperature (or other) sensors. Our datacenter today does not have those. Moreover, some of these sensors emit sounds when they detect a problem, but they often require a human to hear it; *e.g.*, while conducting our experiments, we heard a misconfigured server beeping for weeks.

[4]Datacenter noise may exceed 85 dBA [30].

8 CONCLUSION

In this paper, we have shown that Music-Defined Networking can be used to orchestrate network management functions with a simple and inexpensive out-of-band channel whose implementation requires minimal infrastructure changes. We analyzed frequencies produced by datacenter server cooling fans as an efficient disaster countermeasure.

We also leveraged sound to transmit control plane information. In particular, we implemented several applications in which a listening device performs management operations after learning the state of a switch from audio signals, without relying on traditional control messages.

Aside from the limitations that we have already discussed throughout the paper, our Music-Defined Networking approach poses several challenges for practical implementations that warrant further exploration. First, we limit our evaluation to close-range applications, as we transmit sound signals between devices over a single hop. Practical systems are limited to devices that are placed close enough to each other to transmit sounds without significant signal degradation. Sound waves can, and have been, however, relayed, although with very low throughput and for data plane transfers [19].A more efficient multi-hop sound transmission would allow greater flexibility in device placement. We leave this as an open question.

Furthermore, as discussed in Section 5, with our equipment, we found that we could feasibly use approximately 1000 unique frequencies simultaneously. An interesting research direction is to coordinate an array of microphones listening to different groups of switches, as well as to allow cooperation of classical and ultrasound speakers and microphones. Including frequencies outside the spectrum of human hearing would allow for an increase in the number of discernible sounds and for more complex and scalable network management operations.

9 ACKNOWLEDGEMENT

We thank A. Johnson and D. Thomas for their support, and the anonymous reviewers and our shepherd for their valuable feedback. Work supported by NSF CNS-1647084 and CNS-1836906.

REFERENCES

[1] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data Center TCP (DCTCP). In *Proc. of ACM SIGCOMM 2010*.

[2] M. Alizadeh, A. Javanmard, and B. Prabhakar. Analysis of DCTCP: Stability, Convergence, and Fairness. In *Proc. of the ACM SIGMETRICS*, pages 73–84. ACM, 2011.

[3] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 3–11, May 2004.

[4] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder. Acoustic side-channel attacks on printers. In *Proc. of the 19th USENIX Conference on Security*, pages 20–20, 2010.

[5] R. B. Basat, G. Einziger, S. L. Feibish, J. Moraney, and D. Raz. Network-wide routing-oblivious heavy hitters. In *Proc. of the 2018 Symposium on Architectures for Networking and Communications Systems*, ANCS '18, pages 66–73, 2018.

[6] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proc. of the 10th ACM SIGCOMM Conf. on Internet Measurement*, IMC '10, pages 267–280, 2010.

[7] T. Benson, A. Anand, A. Akella, and M. Zhang. MicroTE: Fine Grained Traffic Engineering for Data Centers. In *Proc. of the 7th COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 8:1–8:12, 2011.

[8] M. H. Bhuyan, D. Bhattacharyya, and J. Kalita. Surveying port scans and their detection methodologies. *Comput. J.*, 54(10):1565–1581, Oct. 2011.

[9] G. Bianchi, M. Bonola, A. Capone, and C. Cascone. Openstate: programming platform-independent stateful openflow applications inside the switch. *ACM SIGCOMM Computer Communication Review*, 44(2):44–51, 2014.

[10] L. Chen, J. Xia, B. Yi, and K. Chen. PowerMan: An Out-of-Band Management Network for Datacenters Using Power Line Communication. In *NSDI 18*, pages 561–578, Renton, WA, 2018.

[11] M. Chiesa, G. Kindler, and M. Schapira. Traffic engineering with equal-cost-multipath: An algorithmic perspective. *IEEE/ACM Trans. Netw.*, 25(2):779–792, Apr. 2017.

[12] M. Chiesa, G. Rétvári, and M. Schapira. Lying your way to better traffic engineering. In *Proc. of CoNEXT*, pages 391–398, 2016.

[13] C. Cimpanu. Gas based fire suppression system shuts down NASDAQS scandinavian datacenter https://goo.gl/VjqqHZ, 2018.

[14] Y. Cui, S. Xiao, X. Wang, Z. Yang, C. Zhu, X. Li, L. Yang, and N. Ge. Diamond: Nesting the data center network with wireless rings in 3d space. In *13th USENIX NSDI*, pages 657–669, Santa Clara, CA, 2016.

[15] D. Genkin, A. Shamir, and E. Tromer. Acoustic cryptanalysis. *J. Cryptol.*, 30(2):392–443, Apr. 2017.

[16] Ghobadi, Monia et al. ProjecToR: Agile Reconfigurable Data Center Interconnect. In *Proc. of SIGCOMM 2016*, pages 216–229.

[17] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian. Drill: Micro load balancing for low-latency data center networks. In *Proceedings of SIGCOMM 2017*. ACM.

[18] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *Proc. of SIGCOMM 2014*.

[19] M. Hanspach and M. Goetz. On covert acoustical mesh networks in air. *CoRR*, abs/1406.1213, 2014.

[20] R. Hasan, N. Saxena, T. Haleviz, S. Zawoad, and D. Rinehart. Sensing-enabled channels for hard-to-detect command and control of mobile devices. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, pages 469–480, New York, NY, USA, 2013. ACM.

[21] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proc. of SIGCOMM 2014*, pages 187–198, 2014.

[22] Internet Assigned Numbers Authority (IANA). Assigned internet protocol numbers https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml, 2018.

[23] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé. Semi-oblivious traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 157–170, Renton, WA, 2018.

[24] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA, 2010. ACM.

[25] M. LeMay and J. Tan. Acoustic surveillance of physically unmodified pcs. In *Security and Management*, 2006.

[26] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proc. of the 2016 ACM SIGCOMM Conference*.

[27] A. Madhavapeddy, R. Sharp, D. Scott, and A. Tse. Audio networking: the forgotten wireless technology. *IEEE Pervasive Computing*, 4(3):55–60, July 2005.

[28] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proc. of ACM SIGCOMM 2017*, SIGCOMM '17, pages 197–210, New York, NY, USA, 2017. ACM.

[29] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM CCR*, 38(2):69–74, Mar. 2008.

[30] D. Miljkovic. Noise within a data center. In *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1145–1150, May 2016.

[31] R. Nandakumar, K. K. Chintalapudi, V. Padmanabhan, and R. Venkatesan. Dhwani: Secure Peer-to-peer Acoustic NFC. In *Proc. of the ACM SIGCOMM 2013*, pages 63–74, 2013.

[32] K. R. Rao, D. N. Kim, and J.-J. Hwang. *Fast Fourier Transform - Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[33] H. Riksfjord, O. T. Haug, and J. M. Hovem. Underwater acoustic networks - survey on communication challenges with transmission simulations. In *2009 Third International Conference on Sensor Technologies and Applications*, pages 300–305, June 2009.

[34] M. Sharif-Yazd, M. Khosravi, and M. K. Moghimi. A Survey on Underwater Acoustic Sensor Networks: Perspectives on Protocol Design for Signaling, MAC and Routing. 05:12–23, 03 2017.

[35] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford. Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*, SOSR '17, pages 164–176, New York, NY, USA, 2017. ACM.

[36] G. Wang, L. Zhang, and W. Xu. What can we learn from four years of data center hardware failures? In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 25–36, June 2017.

[37] Q. Wu, J. Strassner, A. Farrel, and L. Zhang. Network telemetry and big data analysis (expired). Internet-Draft draft-wu-t2trg-network-telemetry-00, IETF Secretariat, March 2016. http://www.ietf.org/internet-drafts/draft-wu-t2trg-network-telemetry-00.txt.

[38] M. Yu, L. Jose, and R. Miao. Software defined traffic measurement with opensketch. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 29–42, Berkeley, CA, USA, 2013. USENIX Association.

[39] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 103–117, 2017.

[40] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng. Mirror mirror on the ceiling: Flexible wireless links for data centers. *SIGCOMM CCR*, 42(4):443–454, Aug. 2012.

[41] Y. Zhu, X. Zhou, Z. Zhang, L. Zhou, A. Vahdat, B. Y. Zhao, and H. Zheng. Cutting the cord: A robust wireless facilities network for data centers. In *Proc. of MobiCom '14*, MobiCom '14, pages 581–592, 2014.