Microtemporality: At The Time When Loading-in-progress

Winnie Soon
School of Communication and Culture, Aarhus University
wsoon@cc.au.dk

Abstract

Loading images and webpages, waiting for social media feeds and streaming videos and multimedia contents have become a mundane activity in contemporary culture. In many situations nowadays, users encounter a distinctive spinning icon during the loading, waiting and streaming of data content. A graphically animated logo called throbber tells users something is loading-in-progress, but nothing more. This article investigates the process of data buffering that takes place behind a running throbber. Through artistic practice, an experimental project calls *The Spinning Wheel of Life* explores the temporal and computational complexity of buffering. The article draws upon Wolfgang Ernst's concept of "microtemporality," in which microscopic temporality is expressed through operational micro events. [1] Microtemporality relates to the nature of signals and communications, mathematics, digital computation and dynamic network within these deep internal and operational structures. [2] Through the lens of microtemporality, this article offers a new understanding of a throbber icon beyond its symbolic form and human sensory reception. It opens up the cultural and computational logics that are constantly rendering the pervasive and networked conditions of *now*.

Introduction

Loading images and webpages, waiting for social media feeds and streaming videos and multimedia contents have become a mundane activity in contemporary culture. In particular, this includes network connected devices from fixed desktop computers to portable tablets and smart watches, all involving data transmission across multiple sites—referred to as data streams. In many situations nowadays, users encounter a distinctive spinning icon during the loading, waiting and streaming of data content. A graphically animated logo called throbber tells users something is loading-in-progress, but nothing more. Unlike a progress bar, a throbber is perceived as repeatedly spinning under constant speed. In contrast to a progress bar which is more linear in form, a throbber does not indicate any completed or finished status and progress. Commonly, a progress bar explains such computer operations, for example, transferring and copying specific files and directories, and illustrating installation procedures. Arguably, when a software application connects to a technological network, such as a home or mobile Internet network, things get more complex.

This article investigates the process of data buffering that takes place behind a running throbber. In particular, it examines the temporal complexity of data streams, in which data processing and code inter-actions are operated in real-time. The notion of inter-actions mainly draws references from the notion of "interaction" from Computer Science and the notion of "intra-actions" from Philosophy. [3][4][5] The term code inter-actions highlights the operational process of things happen within, and across, machines through different technical substrates, and hence produce agency.

This article is informed by artistic practice, including close reading of a throbber and its operational logics of data buffering, as well as making and coding of a throbber. These approaches, following the tradition of artistic research, allow the artist/researcher to think in, through and with art. [7] Such mode of inquiry questions the invisibility of computational culture. By focusing, using and producing a throbber, it suggests a different engagement and possibility of seeing this cultural icon and its related background activities in a different way. This article also draws upon the concept of "microtemporality" in the work of Wolfgang Ernst, in which microscopic temporality is expressed through operational micro events. [1] Microtemporality relates to the nature of signals and communications, mathematics, digital computation and dynamic network within deep internal and operational structures. [2] Ernst's microtemporality is also linked to his notion of discontinuity which is grounded on Michel Foucault's *The Archaeology of Knowledge*. [8] According to Foucault, his concept of discontinuity offers an alternative perspective to understand knowledge beyond its stable form of narration and representation. [9] Both Foucault and Ernst use the term discontinuity as a means to examine the gaps, silence and ruptures of things that go beyond signs or representational discourses. Although a throbber icon becomes a standard way of implementation in contemporary software culture, the micro events that happen behind a throbber indeed react differently. This complexity of time conception is explored and exemplified in an experimental and artistic project calls *The Spinning Wheel of Life*.

Rethinking the notion of time beyond users' perception, this article and the artwork open up the cultural and computational logics that are constantly rendering the pervasive and networked conditions of *now*.

A cultural reading of a throbber

With its distinct characteristic of spinning design that indicates background processing, the throbber icon acts as an interface between computational processes and visual communication. One of the earliest uses of the throbber can be found in the menu bar of a Mosaic web browser in the early 1990s, developed by National Center for Supercomputing Applications (NCSA) and the browser interface was designed by scientist Colleen Bushell [10][11]. The browser throbber contains a letter 'S' and a globe that could spin when loading a web page. This kind of a spinning throbber, with the company browser's graphical logo, has also been seen in subsequent software browsers. While the throbber spins, it visually indicates actions are in progress. These actions, from a user point of view, could be interpreted as the loading of web data or connecting to a website by a software browser. From a technical perspective, it involves Internet data transmission and a browser that renders the inter-actions of code. The spinning behavior stops when a webpage is finished loading within a browser. A web browser, according to Tali Garsiel and Paul Irish, is software able to render and display requested content, make network calls and requests, and store data locally. [12] In this respect, the spinning throbber icon represents complex inter-actions of code under network conditions. A throbber with its spinning characteristic, therefore, can be said to be rooted in, and specific to, Internet culture.

More recently, the throbber icon is no longer only attached to software browsers, it also appears on different web and mobile applications, and social media platforms in particular. The contemporary throbber transforms into a spinning wheel[1] that consists of lines or circles that are arranged in radial and circular form, moving in a clockwise direction. Each individual element of a wheel[2] sequentially fades in and out repeatedly to create a sense of animated motion (see Fig 1).

Fig 1. *Throbber sequences in the form of circles and lines*, 2015, designmodo, web, the image is retrieved from http://designmodo.com/css3-jquery-loading-animations.

These spinning wheels appear after a user has triggered an action, such as swiping a screen with feeds to requesting updated information. They also appear after a user has confirmed an online payment, or is waiting for a transaction to complete. Most commonly, it is seen when a user cannot watch a video clip loading smoothly over an Internet connection. As a result, an animated throbber appears as a spinning wheel on a black colour background, occupying the whole video screen while the video is buffering.

A throbber represents the speed of network traffic that also seems tied to our emotional states and perception of time. Emotionally, it can be annoying and frustrating while one encounters buffering because it involves interruption. [14][15] Things do not flow smoothly, and users become impatient in waiting for an unknown time or watching for something yet to come. As James Charlton puts it: "It is a gaze that goes beyond the screen to an event not yet here." [16] To Charlton, the loading time of the throbber is wasted and unproductive as it is often associated with the perception of the slowness of a network.

Perhaps, there is a desire in which things would flow continuously, as something like broadcast television. The notion of flow was theorized by a media scholar Raymond Williams in 1974, in which the programming of content implies continuity, stemming from the experience of viewing and reception. [17] The interruption, what Williams calls "natural break," of the advertisement on television, is a planned flow as part of the television production. [18] Most importantly, the notion of flow is an expected sequence, such as the number of breaks and the corresponding duration, to engage with audiences. Therefore, television exhibits a relatively stable temporality. However, in a networked medium, the interruption, such as buffering, cannot be planned as with television insofar it is subjected to its material conditions at any moment of time.

The material nature of the network exhibits something that is unpredictable, unstable and discontinuous, which is beyond seemingly 'natural' breaks, and beyond visible and apparent interruptions. In the following section, I will elaborate on what I refer to as 'discontinuous microtemporality' as a way to rethink the notion of flow and stream in networked environments.

Discontinuity in streams

The network structure of today's communication channels and their streams are often understood as providing a direct connection between users and services or between two communication partners, even though there cannot be any direct links on digital networks. The metaphor of the flow conceals the fact

[1] The use of lines that indicates the progress activity of a computer can be found in the early operating system of Unix. [13]

[2] Coincidently, the visual design of a throbber is similar to the design of early wristwatches (with crystal guards) that were made for soldiers in World War I. Both include the concept of a wheel in the form of circles or lines of the petal shape. See: http://www.oobject.com/category/earliest-wrist-watches/

> that, technically, what is taking place is quite the opposite. There is no stream in digital networks. [19]

In a general sense, data is generated under different circumstances in the network environment, traversing at various speeds and spaces across platforms and continents. In the context of data buffering within computational networks, temporality refers to the processing and the unfolding of data over time that generates differences and rhythms. Although there are numerous scholarly works discuss temporality in relation to the subjectivity of time, [20] less attention is paid to the material aspect and the nondiscursive realm of temporality. This comes close to what Ernst describes as 'microtemporality,' focusing on the detailed processes of computation. Instead of examining obsolete objects,[3] as demonstrated in much media archaeology, the emphasis is more associated with the nature of signals and communications, mathematics and computation within its deep internal and operational processes. [21]

In today's networked communication, data is regularly perceived as a stream, indicating its characteristics of vast volume, the speed of update and delivery. However, Florian Sprenger reminds us that the notion of flow and stream are metaphors. The temporality of the perceived flow/stream involves imperceivable discrete-time system, [22] the transmission of data packets, [23] temporal storage and transfer, [23] and micro-decisions by numerous protocols. [24] A stream produces differences and rhythms due to things—calculations, mathematics and logics—that are executing and running in real time. The implication is that these operative processes cannot be seen as planned sequence of flow, and that there is a temporal dimension to such operative logics that reconfigure our understanding of temporality in computational processes.

As opposed to continuous-time signals in analogue systems, the digital adopts the model of a discrete-time system with independent variables in signal processing. This model means that each discrete state is countable and measurable with a distinct value, and can be represented by a sequence of numbers. The signals are discrete in time (see Fig 2), alluding to the value between two discrete-time instances is not defined. Therefore, the 'flow' of data that we experience through a screen is discrete in its nature. Within digital signal processing, the data stream is discontinuous (discrete) regarding its time signal.

[3] For instances, Friedrich Kittler's analysis of the gramophone and typewriter in 1999, or Wolfgang Ernst's analog radio and phonograph in 2013.

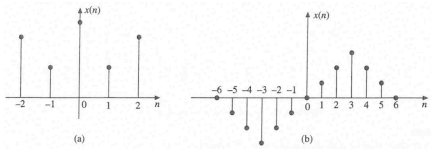

Fig 2. *Discrete time signals*, 2015, A. Anand, Kumar, Print, Copyright 2015 by PHI Learning Private Limited.

In the early design of modern communication networks, the concept of 'packet switching' was fundamental to understand how data was organized and flowed. A data stream was chopped into smaller blocks as 'packets' that were then sent a communication channel in and through different routes, rates, and sequences, known as packet switching. According to Paul Baran, one of the inventors of the packet switched computer network, real-time connections between a sender (transmitting end) and a user (receiving end) are an illusion. Instead, the fast enough data rate gives a *sense* of real-time connection between a sender and receiver. [23] Fundamentally, the routing of a data packet transmits through different nodes and routers. Although a selected path is based on "adaptive learning of past traffic," there are real-time decisions that have to be made to locate the shortest path due to the dynamic of network conditions [23]. In other words, data travels "via highly circuitous paths that could not be determined in advance". [23] Data packets are inscribed with a sequence of numbers and the function of checksum that made them possible to reformulate a correct sequence that makes sense of the perceived content. This assembling process involves the use of a temporary buffer. Baran explains as follow:

> On the transmitting end, the functions include chopping the data stream into packets, adding housekeeping information and end-to-end error control information to the out-going packets. On the receiving end, each multiplexing station uses terminating buffers temporarily assigned to each end addressee to unscramble the order of the arrived packets, and buffer them so that they come out as an error-free stream, only slightly but not noticeably delayed. [23]

What is interesting here is the barely noticeable delay time that gives the perception and illusion of a stream. In this journey of data packet transmission, Sprenger argues that there are numerous "micro-decisions" that are made through network protocols. [24] For instances, the decision of locating "the most efficient path to the destination," the speed of data processing and "the priority of incoming packets." [25] All these decisions are made to control how data is distributed. Even though the time may not seem significant, still there is time lost along the journey. [26] This journey involves interruption at different nodes, transmitting and receiving ends in which data disassembling and assembling occur.

As such, "[t]he stream never flows uninterruptedly." [25] This constant interruption constitutes the notion of microtemporality that includes decision-making processes, controls and regulations that are programmed at the level of protocols and are inscribed in the stream.

Such processes do not only occur at the network level but also at the memory and storage level which are highly relevant to code inter-actions that involve both hardware and software. [27] Data is processed at a receiver's end (as input data in the buffer) and is stored temporary and locally until the data is further processed by the software application (as output data). The term buffering describes the process of input and output of the buffer – the activities of writing and storing, reading and processing that are happening at the same time but not acting on the same bit and piece of data. Buffer refers to the processing of all kinds of data with different "data transfer rates and/or data processing rates between sender and receiver." [27] In other words, the processing of data consists not only of the transferring part but rather as Ernst remind us, through "a coupling of storage and transfer in realtime." He continues, "[w]hile we see one part of the video on screen, the next part is already loaded in the background." [22]

Theoretically, Just in Time (JIT) delivery is used in streaming media, allowing for the playback of partially received data temporarily stored in the client's buffer. In this sense, both the playback of buffer data and receiving the remaining data can be made simultaneously (and, in addition to the case of video and audio, this is also commonly experienced in loading any relatively large size file such as a PDF or an image within a browser). Therefore, streaming "is achieved by buffering the transmitted data before the actual display." [28] Ideally, according to Meinel and Sack, "buffer empties itself at one end just as quickly as it fills up at the other end". If there is transmission delay that is within a threshold time t, it is regarded as unnoticeable in playback. [28] However, if the delay of the individual segment exceeds the threshold time t, a throbber will display. A throbber is seen when loading a big chunk of data, which is commonly seen on video sites such as Youku or YouTube, mostly due to the instability or low bandwidth of a network that causes the delay of data segment arrival (exceeds the threshold time t).

Indeed, buffering is highly related to time, inasmuch as the primary purpose of a buffer "is to reduce time dependencies of the data and to decouple input/output from the program execution." [27] As a result, data can be consumed (as input) and processed (as output) at a different rate. Data, in the case of streaming, is actively and repetitively being stored (write) and removed (read) in the buffer with different rhythms (see Fig 3), oscillating between the invisible and visible, and this is how we can only immanently experience the microtemporality of buffering.

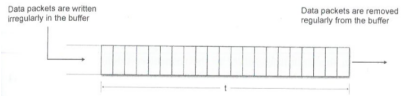

Fig 3. *Principle organization of a playback buffer*, 2013, Christoph Meinel & Harald Sack, Print, Copyright 2013 by Springer.

From the operative logic of streaming, we know there are calculable processes, data transmissions, reading and writing of the buffer at different rates. The operative logic is automated, and is built into the infrastructure of software as code. What has been written in the buffer will be automatically read and processed. However, technology does not guarantee that all the data is written in the buffer.

Dropped frames (frames of video that are dropped during playout) are a relatively common experience in real-time communications and video streaming. Sometimes the issue of dropped frames is seamless because it does not create significant quality degradation. Such visible and invisible dropped frames are caused by packet loss, the absence of certain parts of data during data transmission across nodes and routers through the journey. Time lost, as mentioned above, includes micro-decisions making as well as interruptions and delays. Packets are required to queue up and wait for the transfer while the network is congested. Under streaming conditions, data is continuously transmitted from a sender to a receiver across multiple sites. However, the amount of buffer space is limited at each site which means the newly arriving packet has no space to the stored while the stored packet is still queuing for its next routing. In this situation, "packet loss will occur-either the arriving packet or one of the already-queued packets will be dropped." [30]

Packet loss does not only limit to streaming applications but also other kinds of software applications in contemporary software culture. For real-time conversational applications and media streaming platforms, such as Skype and YouTube, the delay time for each packet is crucial because the transmission demands have to be perpetual as conversations and live concerts are unceasing. On the one hand, the absence of data is central as packet loss is related to the degradation of quality, and it could immediately impact the visual or audio quality with jitters or glitches in a live environment. On the other hand, if data arrives with a significant delay, the application design at the receiver's end is then required to determine if such data will still make sense in playback, in particular where conversation and data are continuously played-back as a stream. In deciding whether the data should be played-back or ignored, acceptable latency becomes a decision that is inscribed in the software and platform design. A serious data loss may even result in the automatic termination of a connection. More precisely, data packets are not only transmitted at different rates (speed), but also with the

potential to be dropped at any time as absent data. In addition, absent data might not cause noticeable effects in digital communication as it is subjected to the amount of loss and the level of acceptable latency that are designed into software applications.

Therefore, not all data is treated equally and has the right to arrive at the destination and able to take a perceptible form. The automated micro-decisions and computational processes, again, reconfigure the temporality of networks by discarding absent data. The notion of microtemporality explicates the invisibility of computational culture by shifting our attention from what is visible on a screen to invisible micro events that are running in the background.

The Spinning Wheel of Life

Such reflection of invisibility is made apparent in the artistic project *The Spinning Wheel of Life*, emphasizing the microtemporal dimension of code inter-actions that are manifested in the throbber. The title of the project is borrowed from a 'wait cursor' in the Macintosh Operating System X designed by Apple. The wait cursor is colloquially known as "The Spinning Wheel of Death," referring to the malfunction or failure of a running program or a system that leads to screen freezes. The name takes on negative connotations, as the problems are usually difficult to diagnose. My version of the spinning wheel is designed to reveal the microtemporal complexity of data transmission and storage, and takes buffering to be a cultural activity that is nonhuman-oriented. The project does not involve human manipulation directly, but rather it processes in real-time through code inter-actions. The visual outcome is subjected to the technical conditions of its operation at a given moment in time.

The Spinning Wheel of Life consists of a throbber that animates in different rates. Each ellipse within the throbber represents a new data packet arrival. The time for each fading ellipse is adjusted to an optimal level in which a balance of the visual composition is achieved. Since packets arrive at multiple time and space, and sometimes a huge amount of packets arrive at almost the same time (the time is down to milliseconds), the visual throbber yields an unusual and uneven spinning wheel—from having just a few ellipses to a full throbber with all the ellipses displayed brightly. Each ellipse fades in and out with different tempo, subject to the network conditions in real time. The project makes apparent the underlying notion of discontinuous microtemporality.

Figures 4-9 below have documented the animated movements of *The Spinning Wheel of Life*; it reacts to the network packets that are generated from running a YouTube playlist in real-time.

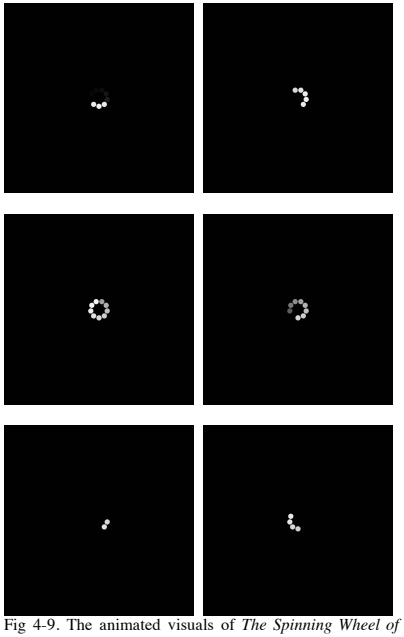

Fig 4-9. The animated visuals of *The Spinning Wheel of Life* (2016)

There are different rates, tempo, pulses, pauses and rhythms at multiple scales - from the operations of the CPU to network routers, from the transmissions of senders to receivers, from the writing to the reading of buffers, and from continuous streams to discontinuous packets. The artwork presents an ongoing operative processing, addressing the microtemporality of its inter-actions that underpin the networked logic of contemporary software culture.

Time is an important element in contemporary software culture as it governs how a signal is processed, how data is transmitted and how micro-decisions are made. As a result, a stream is constantly being interrupted since the start of data transmission, but not at the time one encounters a throbber animating on a screen. The complex process of buffering, as I argue, exhibits a very particular temporality that sheds light on an understanding of software culture and how it processes time.

Conclusion

The notion of discontinuous microtemporality highlights the temporal dimension of the stream as part of our contemporary condition. As Peter Osborne remarks, contemporaneity "is primarily a global or a planetary fiction," which suggests a stream is highly capitalized under these globalized processes that disseminate into

every part of the world. [31] *The Spinning Wheel of Life* calls for critical attention to these networked but also mediated processes, not only at a planetary scale but at the microtemporal level of operations. In this contemporary software culture, things are immanently networked, and data are constantly generated, the notion of discontinuous microtemporality highlights not only the differences and rhythms of code inter-actions, but also the absent and silent of data that is beyond human sensory reception. The complex process of data buffering, as I have argued, exhibits a very particular kind of discontinuous microtemporality. This sheds light on not only the computational logic behind a throbber, but also the real-time dynamics of computational networks in a general sense, and hence the rendering of the pervasive and networked conditionings of *now*.

In other words, the artwork is a reflection of the perpetual, cultural and social changing conditions. On the one hand, the existence of a throbber is a by-product of a commercial application that informs users to wait for an unknown period of time. On the other hand, through the development of various services, such as live streaming, big data analysis, social media platforms, data predictions and transactional applications, it offers a critical space to understand how a throbber is being made operative. A throbber is a cultural phenomenon that appears in almost every application that requires a live computational environment. A throbber is not only a technical or visual object but also entangled with other cultural and micro processes that render the unknowable more knowable.

Acknowledgements

This paper was supported by The Center for Participatory IT. I would like to thank my supervisors for providing useful feedback and criticism. This paper was developed through the seminars and conferences on execution in 2015. I am thankful to the team of Critical Software Thing, as well as the participants of the execution conference for their comments on an earlier version of this paper. Finally, I would like to thanks for 4 anonymous reviewers for their feedback.

References

1. Jussi Parikka, "Archival Media Theory: An Introduction to Wolfgang Ernst's Media Archaeology," in *Digital Memory and the Archive*. (Minneapolis: University of Minnesota Press, 2013), 19.
2. Jussi Parikka, "Operative Media Archaeology: Wolfgang Ernst's Materialist Media Diagrammatics," *Theory, Culture & Society* 28 (2011): 52-74.
3. Peter Wegen, "Why interaction is more powerful than algorithms," *Communications of the ACM 40* (1997): 80-91.
4. Peter Bentley, "The meaning of code," in *Code: The language of our time*, eds. Gerfried Stocker and Christine Schöpf (Linz: Hatje Cantz, 2003), 33-36.
5. Karen Barad, *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning*. (Durham: Duke University Press, 2007), 140.
6. Jussi Parikka, "Archival Media Theory: An Introduction to Wolfgang Ernst's Media Archaeology" *in Digital Memory and the Archive*, 8-9.
7. Henk Borgdorff, "The Production of Knowledge in Artistic Research," in *The Research Companion to Research in the Arts*, eds Michael Biggs & Henrik Karlsson (Oxon, UK: Routledge), 44.
8. Wolfgang Ernst, "Dis/continuities: Does the Archive Become Metaphorical in Multi-Media Space?" in *New Media, Old Media: a history and theory reader*, ed. Wendy Hui Kyong Chun (New York, London: Routledge, 2006).
9. Michel Foucault, *The Archaeology of Knowledge and the Discourse on Language* (New York: Pantheon Books, 1972), 3.
10. Michael C, Albers, "Auditory cues for browsing, surfing, and navigating," *Proceedings of the 3rd International Conference on Auditory Display (ICAD 1996), Palo Alto, California,* (1996), accessed December 10, 2015, https://smartech.gatech.edu/handle/1853/50793
11. Kevin Roebuck, *Virtual Desktops: High-impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. (Dayboro: Emereo Publishing, 2011).
12. Tali Garsiel & Paul Irish, "How Browsers Work: Behind the scenes of modern web browsers (2011)", How Browsers Work website, accessed December 15, 2015, http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/
13. Kevin Roebuck, *Virtual Desktops: High-impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*, 349
14. Rick Broida, "Stop Frustrating Pauses in YouTube Videos", PCWorld website, accessed December 15, 2015, http://www.pcworld.com/article/201089/Stop_Frustrating_Pauses_in_YouTube_Videos.html
15. Brian Stelter, "Debate Web Stream Does Not Flow Smoothly for All (2011)," New York Times website, accessed December 15, 2015. http://thecaucus.blogs.nytimes.com/2011/10/11/debate-web-stream-does-not-flow-smoothly-for-all/?_r=1
16. James Charlton, "Post Screen Not Displayed," in *Post-Screen: Device, Medium and Concept,* eds. Helena. Ferreira & Ana. Vicente (Lisbon: CIEBA-FBAUL, 2014), 171.
17. Raymond Williams, *Television: Technology and Cultural Form*, (London: Fontana, Collins, 1974), 80-84.
18. Raymond Williams, *Television: Technology and Cultural Form*, 90.
19. Florian Sprenger, *The Politics of Micro-Decisions: Edward Snowden, Net Neutrality, and the Architectures of the Internet,* (Lüneburg: meson press, 2015), 88-89.
20. David M Berry, "The Social Epistemologies of Software," *Social Epistemology* 26 (3-4), (2012): 379-398.
21. Wolfgang Ernst, "Experimenting with Media Temporality: Pythagoras, Hertz, Turing," in *Digital Memory and the Archive*, ed. Jussi Parikki (Minneapolis: University of Minnesota Press, 2013), 186-189.
22. Wolfgang Ernst, "Dis/continuities: Does the Archive Become Metaphorical in Multi-Media Space?" in New Media, Old Media: a history and theory reader, ed. Wendy Hui Kyong Chun, 108.
23. Paul Baran, "The beginnings of packet switching: Some underlying concepts," *IEEE Communications Magazine* 40(7), (2002): 42-48.

24. Florian Sprenger, *The Politics of Micro-Decisions: Edward Snowden, Net Neutrality, and the Architectures of the Internet,* 73-102.
25. Florian Sprenger, *The Politics of Micro-Decisions: Edward Snowden, Net Neutrality, and the Architectures of the Internet,* 19.
26. Florian Sprenger, *The Politics of Micro-Decisions: Edward Snowden, Net Neutrality, and the Architectures of the Internet,* 75.
27. Philip A Laplante, *Dictionary of Computer Science, Engineering and Technology,* (CRC Press, 2000), 55.
28. Christoph Meinel & Harald Sack, *Internetworking: Technological Foundations and Applications*, (Berlin: Springer, 2013), 783.
29. Christoph Meinel & Harald Sack, *Internetworking: Technological Foundations and Applications*, 780.
30. Kurose James F & Keith W. Ross, *Computer Networking: A Top-Down Approach*. (Pearson Education, 2013).
31. Peter Osborne, *Anywhere or Not at All: Philosophy of Contemporary Art*. (London: Verso, 2013), 26.