Nofar Sheffi*

We Accept: Bit-by-bit Constitution

https://doi.org/10.1515/zfrs-2024-1009

Abstract: In "We Accept: The Constitution of Airbnb", published in 2020, I drew on Gunther Teubner's systems theoretical approach, including to digitalisation and societal constitutionalism, to rethink Airbnb as an auto-constitutionalising domain. This dynamic process, the paper proceeded to suggest, instantiates a structural change, what can be described, in systems theoretical terms, as a "co-evolution" of multiple social systems through structural couplings. It renders observable an "emergence" of a general model of functioning – terms of use and engagement – epitomised by a reconceptualisation of the "social contract" as a network of "standard form contracts" or "contracts of adhesion", and of an increasingly dominant mode of subjectivation: the standard formation or auto-constitution of subjects as adherents. In this new contribution, I continue this processing of social systems theory and Teubner's work, this time through a deconstruction and reconstruction of Internet communication. Through this recursive illustration of social systems theory and Teubner's work, the paper develops a social systems theoretical account of "process", "constitution", and "structural change".

Zusammenfassung: In „We Accept: The Constitution of Airbnb", veröffentlicht im Jahr 2020, stützte ich mich auf Gunther Teubners systemtheoretischen Ansatz, unter anderem zu Digitalisierung und gesellschaftlichen Konstitutionalismus, um Airbnb als eine sich selbst konstitutionalisierende Domäne neu zu denken. Dieser dynamische Prozess der Selbst-Konstitutionalisierung, so schlug der Artikel weiter vor, initiiert eine Strukturänderung, die systemtheoretisch als „Koevolution" mehrerer sozialer Systeme durch strukturelle Kopplungen beschrieben werden kann. Er macht die „Emergenz" eines allgemeinen Funktionsmodells – Nutzungs- und Verhaltensbedingungen – beobachtbar, das durch eine Neukonzeptualisierung des „Gesellschaftsvertrags" als ein Netzwerk aus Standardverträgen (*adhesion contracts*) verkörpert wird, und einer zunehmend dominanten Art der Subjektivierung: Standardisierung durch Selbstkonstitution von Subjekten als *adherents*. In diesem neuen Beitrag setze ich meine Auseinandersetzung mit der Theorie sozialer Systeme und Teubners Arbeiten fort, diesmal mit einer Dekonstruktion und Rekonstruktion der Internetkommunikation. Über diese rekursive Illustrierung

***Korrespondenzautor: Nofar Sheffi,** Faculty of Law and Justice, University of New South Wales, Address: Faculty of Law and Justice, UNSW Sydney, NSW 2052, Australia, Tel.: +61 (2) 9385 2227, E-Mail: n.sheffi@unsw.edu.au

der Theorie sozialer Systeme und Teubners Arbeiten entwickelt der Artikel eine systemtheoretische Fassung von „Prozess“, „Konstitutionalisierung“ und „Strukturänderung“.

Keywords: social systems theory; societal constitutionalism; social contract; Terms of Service; Internet communication; digitalisation.

Technically speaking, communication devices do not *communicate* with each other, which may or may not come as a surprise. To begin with, communication devices do not communicate. Systems of communication do (Teubner 2006: 333, building on Luhmann 1995: 176–209; 2012: 57–58). Devices are *used* for communication. They are, precisely, *devices* of communication. What's more, communication is never *with*. Systems communicate but not *with* each other, and that is because "communication *is possible only as a self-referential process*" (Luhmann 1995: 143); it entails mutual translation into a system-specific binary language and emerges once a system reacts to an irritating prompt by initiating a translation process, the aim of which is understanding content (Luhmann 1995: 73–74; 2012: 137).[1] Lastly, as this contribution will show, communication devices are powered and empowered to interpret signs and to output signs, but "transmission" of information from one communication device to another is impossible (Luhmann 2012: 116). The operation of communication devices consists of serving as devices of communication (Luhmann 2012: 59–60). A communication device reacts only to signs that it is configured to interpret as citations of a binary code that the device is constructed to be able to execute, and it reacts only by executing that binary code, that is, by performing an either/or operation that forms part of its configured constitution.

For example, the communication device we call a "digital computer"[2] can be prompted to execute only demands that are issued in "machine language", which is the set of "machine instructions" that a processor (that is, an electronic circuitry)

1 Following Niklas Luhmann, this paper conceptualises communication as a synthesis of three selections: information (a selection from a repertoire of possible utterances), utterance (a selection of a mode of expression, a behaviour, from a repertoire of intentional acts), and understanding (a distinction between utterance and information) (Luhmann 2012: 113; 1995: ch. 4).

2 This paper uses "digital device" and "digital computer" to refer to a wide array of machines that employ the same core hardware technology, for example, personal computers ("PCs"), personal mobile devices ("PMDs") such as tablets and smartphones, servers (computers accessed via a network), supercomputers, and embedded computers ("Internet of Things") (Patterson & Hennessy 2021: 5–7). As argued by Anna Beckers and Gunther Teubner, from a systems theoretical perspective, computer systems serve as devices of communication because interactions between persons and computers produce events that are *understood* as *utterances* that contain certain *information* (2021: 27–30).

is hardwired to be able to convert into electrical signals ("control signals").[3] As this paper will show, because machine language is built into the hardware architecture of a digital device, a device's machine language is distinct and idiosyncratic. Digital devices – even of the same model, that is, devices that are constructed in the same way and appear identical – adhere only to instructions that were issued in a language that digital devices can convert into their distinct machine language. In other words, there is, in effect, no one machine language, a common language that digital devices share; there are as many machine languages as there are digital devices. The messages sent and received by processes that run on digital devices are not and cannot be expressed in machine language or even in a language that mirrors it. To prompt a device to send a message to another computer, that device must be able to interpret the message and the instructions to send it, and, for this to be possible, the message and the instructions need to be translated into a language that mirrors the machine language of the addressor. To get a device to receive a message from another computer, that device must be able to interpret the message and the instructions to receive it, and, for this to be possible, the message and the instructions need to be converted into a language that mirrors the machine language of the addressee.

And it is not only with other digital devices that computers cannot directly interact. Digital computers do not share a language and thus cannot directly interact with any other kind of communication devices, including the semantic artefacts we call "persons" (Teubner 2006: 334; Luhmann 2012, 1995: chs. 6–7).[4] In effect, what appears to be a direct interaction between a device and an operator – an exchange of "inputs" and "outputs" in a shared language – entails a multiplicity of self-referential processes of "conversion", to draw on Piper's (2021: 402–06) linking of the different usages of the term.[5] Persons that operate computers do not express their requests in a language that their device can interpret; both their requests and the data that is provided as input are being automatically translated into the self-ref-

3 See note 61.

4 This contribution follows systems theory in treating both digital devices and persons as communication devices, albeit of a different kind, but leaves aside the question of what distinguishes persons as devices of communication that is explored in Beckers and Teubner (2021: 27–30). On social systems theory's notions of the person as a semantic artefact of communication, see note 63.

5 In information technology, Piper (2021: 403) explains, "conversion" is used to denote the translation of signs/texts into a different medium. Like translation, conversion "is the means of moving between semiotic and material systems, of representing something in a new way rather than simply reproducing it in the same way", and, like the process of moving between religions, implies personal transformation. "Conversion, the act of turning around, also involves the act of leaving behind. It highlights all the losses that occur and uncertainties that are introduced through the practice of transformation."

erential languages of all participating systems and mechanically interpreted by all participating computers. This process – or, more precisely, set of processes – generates an "output" that needs to be rendered in a form that the operator can interpret, a process that itself requires a multiplicity of automatic processes of translation and mechanical processes of interpretation.

If, as this paper will reveal, direct contact between communication devices, digital and human, is not possible, what is "Internet communication", how does it work, and how are devices used for communication? How are we as operators of digital computers able to exchange messages online, visit domains, and navigate from page to page and site to site? What is a Web browser, and how does Web-browsing work? What is the World Wide Web? What are Web sites? What are domains and how are they formed, transformed, and constitutionalised? And how are communication devices, both digital computers and Internet users, configured in the process?

Protocol Followers

Switched on in 1948, the first digital computers were used to perform complex calculations, rather than transmit data to and receive data from (operators of) other devices. They were able to interact – but only with their operators – by performing manipulations on "input data" in response to citations of machine language and by generating "output data" (Abbate 1999: 1). Transfer of data from one digital device to another was possible only by using external storage media, like punch cards or magnetic tapes. To move data from one digital computer to another, in other words, operators had to save the data on an external medium, carry that medium from one computer to another, and save the data in the destination machine's internal storage medium ("non-volatile memory") (Abbate 1999: 1; Augarten 1984: 133–34).

It was only in the late 1960s that the first computer networks were formed. Composed of digital computers that were manufactured by the same vendor, each of these networks operated on behalf of a single entity and used proprietary networking protocols. To enable data transfer, each "dumb" device was plugged into a single, "intelligent" communication controller, which was entrusted with the task of coordinating all network traffic (Hall 2000: 1). Each time a device wanted to send a message to a fellow network member, it had to contact the central authority and request its mediation services. Upon receiving a request from a network member, the central authority would convert the message into a language that mirrored the idiosyncratic language of the addressee and deliver the translation to the addressee. Digital computers that were not connected to a network could not interact with any other devices. Digital devices that were connected to a network were able to interact only with digital computers that were attached to the same communication controller as them, using the central authority's mediation services.

In the early 1970s, efforts were launched to develop a system that would enable any digital device to send data to and receive messages from any other digital device (Hall 2000: 2; Fall & Stevens 2011: 2–3; Galloway 2004: 5). These efforts could have led to the establishment of a single, global network of devices, that is, to the constitution of a universal system for communication between devices, but they did not. In the first place, the architecture of the new communications medium was negotiated not between digital devices, but between (representatives of) the dominant networks operating at the time (and researchers working under their auspices).[6] For an agreement to be reached, its terms must have been acceptable to all networks included in the negotiation.[7] The adoption of a universal language would have required the existing networks to give up their own protocols and dominion, that is, to relinquish their sovereignty and autonomy, to which those existing networks, already established and powerful, had no interest in agreeing (Abbate 1999: 132–33). Moreover, concerned for their privacy and security, many existing networks were pushing for a design that would prevent networks from knowing what was going on in other networks, and hosts from knowing "anything about the nature of or the existence of any networks other than the ones to which the [device] was directly connected" (Cerf 2000: vii). Had all digital devices spoken a single, universal language, it may have been easier to disclose information to devices attached to other networks, but it would have been much more complicated for networks to have, keep, and share secrets.[8] To preserve and protect the dominion of existing networks, the new "universal" network was thus constituted not as a network of communication devices, but as a network of networks: a communication system for communication systems.

The founding networks could have agreed to establish a central authority and to confer on it the task of providing translation services (Abbate 1999: 128). This "design solution", however, was deemed both impractical and incompatible with the political platform for the fulfilment of which the internetwork was intended to serve as a platform.[9] On a practical level, if the networks had agreed to create a centralised translation mechanism, the growth potential of the constituted inter-

6 On the negotiation process, see Abbate (1999: 123–27). The first Internet incorporated the Advanced Research Projects Agency Network ("ARPANET"), the Atlantic Packet Satellite Network ("SATNET"), and a ground mobile Packet Radio Network ("PRNET"). Xerox PARC's 3 MB/s Ethernet joined the internetwork soon after its establishment (Cerf 2000: vii).

7 On the process through which the terms of the eventual agreement were "accepted", see note 13.

8 On the constitutive role of secrecy, see Simmel (1906).

9 My play on the multiple meanings of "platforms" draws on Gillespie (2010: 349–50).

network would have been limited. The more that networks would have joined the internetwork, the more complex the task of the central translation service would have become, not only rendering the arrangement unworkable but also thwarting the effort to constitute a universal system of communication (Abbate 1999: 128). At the time, centralised systems of communication were also seen as both more prone to internal failures and more vulnerable to attacks from the outside, as malfunction or destruction of a central node destroys communication between nodes, causing the whole system to break down (Baran 1962). The internet was designed as a distributed network also to *serve* a political platform and *as* a political platform. Distributed networks were considered not only more robust than centralised and decentralised networks, but also more sound. At the height of the Cold War era, distributed networks were seen as less vulnerable (than centralised and decentralised networks) to calculated attacks emanating from anti-liberal, anti-democratic forces (Galloway 2004: 4–6, 29–30). Rather than subjecting a collection of dumb devices to "a central politburo", distributed networks were claimed to enable self-deterministic, autonomous agents to transfer data privately, freely, and directly, "without having to talk to a central host first" (Hall 2000: 4–6, 86). They were said to facilitate the formation of an open association of equals antithetical to the "closed society" of the Soviet Union (Edwards 1996: 7–15). Paradoxically, perhaps, distributed networks were also seen as platforms for political change. Bound up not only with the imaginary and rhetoric of the American Declaration of Independence[10] and of the American expansion westward (Yen 2002: 1207–1263),[11] but also with the utopic ideals of the California-based counterculture movement of the 1960s, radically distributed internetworking claimed to represent and enable an alternative mode of governance, to form an open, universal, and global system of communication that would decentralise governance and dismantle barriers to entry and to transfer, to create an unbounded world where anyone can belong and share (Hall 2000: 4–6, 86). Still today, the young are encouraged to head West – to the United States of America, to Silicon Valley, Redmond, and San Francisco – in pursuit of gold, opportunities, and "unicorns". Still today, in the U. S. context, the East is portrayed as the domain of law and order, the seat of incumbents, and the West as a wild, unruly frontier, the territory of boundary-pushing pioneers and boundary-transgressing outlaws. Still today, conflicts between state jurisdictions and technology companies are depicted as clashes between old and new, conventional and disruptive, centralising and decentralising, incumbent and revolutionary, conservative and progressive, or inhospitable and hospitable forces.

10 The most emblematic expression of the 1990s vision of the Internet is John Perry Barlow's e-mail, known as *The Declaration of the Independence of Cyberspace*, modelled after the American Declaration of Independence. The e-mail was published as Barlow (2001).

11 An illustrative example is Carveth and Metz (1996: 72–90).

Instead of establishing a distributed or centralised network of devices or a centralised internetwork, the founding networks developed an auxiliary language of exchange, a "lingua franca for computers" (Galloway 2004: 42), which would be used to transfer data from any device to any device, including between devices that are attached to the same network (RFC 1122: s. 1.1.2). The grammar of the adopted "Esperanto for machines" (Wu 2010: 196) is said to represent *general agreement*,[12] expressed through *voluntary adherence* to suites of technical standards called "protocols",[13] which are *codified* in a series of technical memoranda called "Requests for Comments" ("RFCs").[14] The term "Internet" (with a capital "I") is used to refer specifically to the internetwork constituted through conformity to the terms of use; the collection of protocols relating to Internet communication is referred to as the "Internet Protocol suite" or the "TCP/IP suite"; and communication devices that, so to speak, follow protocol are referred to as "Internet hosts".[15]

But "protocol" – the term devised by Alexander Galloway to refer to this "principle of organization" and "management system" (2004: 3, 7–8) – does not really offer itself to adherents as an auxiliary language or as a language at all, and that is because it purports to necessitate no translation or interpretation (Galloway 2004: 164–67, 2006: 325–29). In other words, rather than as a language to which and from

12 For example, RFC 1122, s. 1 and RFC 1123, s. 1. Presented as an expression of consensus, as many have noted, the first Internet protocols were devised by a small number of academics and government employees, working mainly in the United States, for or under the auspices of the U. S. Department of Defense and other military authorities. They were developed by "a small entrenched group of techno-elite peers", consisting largely of electrical engineers and computer specialists from a small number of countries. "Many of them are university professors. Most all of them either work in industry or have some connection to it" (Galloway 2004: 122–23). See also Abbate (1999).

13 The Internet was rendered operational in 1977, interconnecting the ARPANET, PRNET, and SATNET. As Abbate (1999: 131–38) suggests, however, those who were not actively involved in internetworking experiments had no immediate motivation to implement the Internet protocols in place of their existing protocols. For example, ARPANET did not initially require implementation of the suite and, indeed, many of the organisations connected to it chose to continue using their existing protocols, which met their needs. Implementing TCP/IP was costly and time-consuming; "to make matters worse, the specification kept changing as the Internet team adopted new ideas and as experimental use revealed shortcomings in the design". ARPANET users all switched to using the Internet protocols only following the 1981 issuance of an ultimatum by the U. S. Defense Communications Agency, which had assumed control over ARPANET in 1975.

14 For an exposition of the standardisation process, see Galloway (2004: 131–38) and Hall (2000: 399–406).

15 RFC 1122, s. 1.1.1: "A host computer, or simply 'host', is the ultimate consumer of communication services. A host generally executes application programs on behalf of user(s), employing network and/or Internet communication services in support of this function. ... An Internet communication system consists of interconnected packet networks supporting communication among host computers using the Internet protocols."

which information must be translated, protocol is seen as a formal framework that is indifferent to the semantic meaning of transferred information and makes possible the transmission (within "wrappers" that prevent access to messages) of *any* content from *any* source to *any* destination (Galloway 2004: 52).

Like Airbnb (Sheffi 2020: 489–97), that is to suggest, protocol claims to facilitate direct transfer between users, to enable travel (albeit of information rather than persons) from any address to any address, to platform the constitution of a global community to which everyone can belong, to help "create a world where anyone can belong anywhere".[16] The use of a "common protocol" for internet communication, as observed by Janet Abbate, "would create a particular type of experience for Internet users" (1999: 128). Had networks continued to use different protocols, the boundaries between networks would have been "emphasised", disclosing the being of the Internet as a system for communication between autonomous networks, rather than communication devices (ibid).[17] Had a sense of seamlessness not been fabricated, computer operators would have realised that a device's connection to the Internet is, in the words of RFC 1122, "only conceptual".

Like Airbnb's terms of service and use (Sheffi 2020), what protocol works to "create" is not a boundaryless "world where anyone can belong anywhere", but a sense of belonging to a global community of hosts. Like Airbnb, what protocol aims to manufacture is not a radically distributed network of hosts, but an appearance of global boundarylessness. Like Airbnb (Sheffi 2020), what protocol encodes and enacts is not unconditional hospitality, but hospitality subject to terms and conditions, the constitution of a domain which can be visited and within which conditional hospitality can be offered as a service. Like Airbnb (Sheffi 2020), what protocol standardly ultimately forms are adherents that accept and perform terms of service and use, that follow protocol.

Akin to a building code, protocol construes the Internet as a multilayer architecture and specifies rules and technical standards that must be followed for implementations (in the form of software) to be deemed compliant (Fall and Stevens 2011: 8; for example, RFC 793: s. 1.3). While alternative conceptualisations of the Internet architecture exist,[18] protocol itself distinguishes four nested layers that represent

16 The reference here is to Airbnb's mission of "creating a world where anyone can belong anywhere", discussed in Sheffi (2020: 489).

17 As admitted in an oral history interview by Vinton Cerf, one of the key players: "We wanted to have a common protocol and a common address base so that you couldn't tell, to first order, that you were actually talking through all these different kinds of nets. That was the principal target of the INTERNET protocols" (1990: 23–24).

18 Notably, the Open Systems Interconnection ("OSI") model, defined by the International Organization for Standardization, distinguishes seven abstraction layers: the physical layer, the data-link

different aspects of Internet communication: the “link layer”, the “Internet layer”, the “transport layer”, and the “application layer” (RFC 1122: s. 1.1.3). The *link layer* is the bottom-most layer. Concerned with the hardware that physically connects the sending and receiving endpoints (for example, wires, jacks, and telephone wiring), link-layer protocols define methods for moving information across phone lines, fibre-optic cables, and so forth. Encapsulated within the link layer is the *Internet layer*. Internet-layer protocols are responsible for providing reliable and securable connection service. As will be explained below, they institute an addressing scheme, procedures for the fragmentation and reassembly of long messages, and routing algorithms. The core Internet-layer protocol is called the Internet Protocol (“IP”). Nested within the Internet layer is the *transport layer*. Designed to ensure the reliable transmission of data across the network, transport-layer protocols provide senders and addressees with error-correction and flow-control services. The Transmission Control Protocol (“TCP”) and the User Datagram Protocol (“UDP”) are the core transport-layer protocols. The top layer is the *application layer*, and it is encapsulated within the transport layer. Application-layer protocols deal with the details of specific applications that may be used in Internet communication, and provide applications with access to the data being passed in accordance with the transport-layer protocols.[19] Most protocols use a client-server model: a “client” sends a request to a “server” running on another system; the server processes the request and services it, oftentimes by sending data to the client.[20] For example, the Domain Name System (“DNS”), introduced below, is used by “DNS clients” to request IP addresses from “DNS servers”. The Hypertext Transfer Protocol (“HTTP”), also presented below, is used by “Web clients” to ask for instructions on how to render “Web pages” from “Web servers”.

The transfer process instituted by the IP and the TCP proceeds in three stages: connection establishment, data transmission, and connection termination. Before any data can be transmitted, a connection between a sender and a receiver must be set up. The procedure involves an exchange of three messages, a process called a “three-way handshake”. The server sends the client a request for synchronisation (a message called a “SYN”). The client acknowledges the server’s request (with a message called an “ACK”), initiating, in parallel, its own SYN request. Connection is established once the server acknowledges receipt of the client’s request for synchro-

layer, the network layer, the transport layer, the session layer, the presentation layer, and the application layer. On the OSI model, see Hall (2000: 7–11).

19 For a more detailed presentation of the protocological architecture, see Galloway (2004: 40–41), Hall (2000: ch. 1), and Fall and Stevens (2011: 8–16).

20 UDP servers, for example, do not return data to UDP clients, but rather monitor network activity (Hall 2000: 26). On the history of service and the server metaphor, see Krajewski (2018).

nisation (with its own ACK message) (RFC 793: s. 3.3). Once a reliable connection is set up, the data transfer phase commences. If the message to be sent (a “datagram”) is determined to be too long for transmission, it is “fragmented” into several shorter datagrams, which are then treated as distinct entities.[21] Each datagram is “encapsulated” inside a technically defined wrapper called a “packet”, which functions as an “envelope”, preventing access to the transferred data (referred to as the “payload”) (RFC 793: s. 1.1). The delivery service is provided solely on the basis of the information indicated in a packet’s “header”, including the IP addresses of both the addressee and the sender. Once the encapsulation process is completed, a different route is selected for each packet (a process called “routing”). The various packets are sent on their way separately, with specialised computers called “routers” passing the packets from one to another until the destination is reached (a process called “hopping”). The routers sending a packet to its next hop are indifferent to the content of the packeted datagram. They merely receive the packet and pass it off to the specified “next-hop router”. Once all the packets arrive at the “destination address”, the fragments are reconstructed into the original datagram, so that the original message is re-created.[22] The connection between the two endpoints is terminated through a pair of “two-way handshakes”: each endpoint transmits a message called a “FIN”, which the other endpoint acknowledges with an ACK message. Once both sides send a FIN message and receive an ACK message, the connection ends (RFC 793: s. 3.5).[23]

Internet Hosts

One of the primary tasks of the IP is to institute and maintain an addressing mechanism.[24] In accordance with the scheme, each Internet host (each protocol-following digital computer used for Internet communication) is assigned a unique identifier

21 “The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise)” (RFC 791, s. 1.4).

22 For an outline of the process, see RFC 791, s. 2.3. The process is based on a technology called “packet-switching”, which was developed by Paul Baran at the Rand Corporation and first used, in 1969, by the Advanced Research Projects Agency (“ARPA”) at the U. S. Department of Defense. On packet-switching, see Abbate (1999: ch. 1), Baran (1964), and Galloway (2004: 4–5).

23 For a more detailed exposition of the transfer process, see Fall and Stevens (2011: 12–13), Galloway (2004: ch. 1), and Hall (2000: ch. 1).

24 In addition to addressing, the IP is responsible for determining the path a packet must take, based on the receiving host’s IP address, for assembling packets into datagrams, and for fragmenting and reconstructing datagrams.

called an “Internet Protocol address” (“IP address”), which renders the device identifiable and locatable and, thus, *addressable*.[25] Only devices that have an address can address messages to others and be the addressees of messages, that is, serve as Internet hosts.

In accordance with Internet Protocol version 4 (“IPv4”), an IP address is a string of 32 binary (base-two) digits (“bits”), for example, “11101000000000010000 000001010111”. This 32-bit address consists of two sub-addresses: an identifier of the network to which the specific host is connected and an identifier of the particular host to that network. Another string of 32 bits, called an “IP-mask”, helps networks determine which portion of the 32-bit IP address signifies the network and which portion relates to the host. For example, an IP-mask of “1111111111111111 1111110000000000” would indicate that the first 22 bits of “11101000000000010000 000001010111” signify the network portion of the address, and that the last 10 bits identify the host on that network. The 32-bit binary IP-mask would also disclose the maximal scale of the network to which the addressee is attached, that is, that the relevant network was allotted 2^{10} (1024) IP addresses, of which the initial 22 bits are “1110100000000001000000”.

While digital computers use the binary system to, so to speak, address, persons use a “dotted-decimal” notation (for example, 232.1.0.87), which is a translation of the “machine-readable” 32-bit sequence into the decimal (base-ten) number system. The “human-friendly” notation consists of four decimal numbers from 0 to 255, separated by periods. Each decimal number represents one of the four strings of 8 bits (“bytes” or “octets”) of which a 32-bit address is composed. For example, “232.1.0.87” is the dotted-decimal notation of “11101000000000010000000001010111”: “232” is the decimal expression of “11101000”, “1” is the decimal value of “00000001”, “0” represents “00000000”, and “87” is the decimal expression of “01010111”. A mask, annotated as a “slash prefix” (a “/” character followed by an integer in the range of 13 to 27), trails the dotted-decimal notation, helping to identify the network portion of the 32-bit address. For example, on 22 July 2020, Airbnb Inc. was allocated the “IP block” (or “IP range”) “103.107.42.0/24”, that is, the 256 (2^8) 32-bit IP addresses that begin with “011001110110101100101010” (AS137437 n.d.).

The theoretical maximum of 4,294,967,296 (2^{32}) available IP addresses – from “00000000000000000000000000000000” (“0.0.0.0”) to “1111111111111111111111111111 1111” (“255.255.255.255”) – is referred to as the “address space”.

The seeming semantic meaninglessness of both the “machine-readable” sequence and the “human-friendly” notation serves to generate a sense of the

25 In the words of Luhmann, “self-reference on the level of basal processes is possible only if at least two processing units that operate with information are present and if they can relate to each other and thereby to themselves” (1995: 138).

address space as a radically distributed network of nodes.[26] Nonetheless, as the above implies, IP addresses have never been randomly allocated or assigned,[27] and addressing authority has never been radically distributed. If address space control had indeed been radically distributed, every Internet host would have been able to decide how it wants to be addressed, which has never been the case (Mueller 2009: ch. 2). Distributive techniques of address interpretation constantly work to classify and stratify the address space, and the addressing scheme has been "reformed" multiple times, since its introduction, to enable the centralisation, aggregation, and agglomeration of authority.

While alternative methods of addressing could have been devised, the addressing scheme is structured so that only a predetermined number of IP addresses is available for assignment, constituting the address space as a "finite resource" that "can be used up if it is not conserved properly" (Mueller 2009: 19). Indeed, many of the reforms introduced through the years have been framed as urgent responses to the management and scaling problems arising out of the growth of the Internet, as well as the "[e]ventual exhaustion" (RFC 1519: s. 1) of the address space – "a scarce shared resource that must be managed for the good of the community" (RFC 1518: s. 1).[28]

When the addressing scheme was first deployed, to provide a brief overview of the major "reforms", every assigned IP address was interpreted as consisting of two sections: an 8-bit "network number" and a 24-bit "host number".[29] This method of allocation and assignment assumed an equality of scale and need. The address space was effectively partitioned into 256 (2^8) blocks, allowing a maximum of 16777216 (2^{24}) host addresses for each network. Networks may have varied in scale, but their ultimate growth was subject to a common limit, implying an equal growth potential. Allocation procedures were based entirely on the first-come, first-served principle. With the "realisation" that networks vary in scale and needs, a new method of partitioning was devised to allow for the allocation of address blocks of different sizes,

26 "The Internet does not have a hierarchical topology, rather the interpretation of addresses is hierarchical. In this two-level model, each host sees its network as a single entity; that is, the network may be treated as a 'black box' to which a set of hosts is connected" (RFC 950, s. 1).

27 "[T]he terms 'allocate' and 'assign' have specific meaning in the Internet address registry system; 'allocate' refers to the delegation of a block of address space to an organization that is expected to perform further sub-delegations, and 'assign' is used for sites that directly use (i.e., number individual hosts) the block of addresses received" (RFC 4632, s. 3.1).

28 "The IP address space is a scarce shared resource that must be managed for the good of the community. The managers of this resource are acting as its custodians. They have a responsibility to the community to manage it for the common good" (RFC 1518, s. 1).

29 For a more detailed exposition of the Internet addressing architecture and its history, see Fall and Stevens (2011: ch. 2), Hall (2000: 407–16), and Mueller (2009: 32–38).

the implementation of administrative rationing rules, and the imposition of administrative fees (Mueller 2009, pp. 36–37). The new scheme partitioned the address space into five classes named A, B, C, D, and E, representing five different trade-offs between the number of available network numbers of a given size and the number of hosts that can be assigned to the given network (Fall & Stevens 2011: 36). Alas, with the advent of the Internet, the so-called "classful approach" to Internet addressing was also deemed inadequate. To allow for additional flexibility in allocation, a third level was soon introduced into the hierarchy. Local networks were allowed to further partition the host portion of their base address allocation into a "subnet number" and a host number, and to delegate control and responsibility over these "subnets" to smaller networks (for example, universities or companies).[30] Despite these efforts (but unsurprisingly), the scarcity problem persisted. By 1994, over half of all class B addresses had been allocated. With the exponential growth of the Internet, it was predicted that all class B addresses would be consumed by 1995 and that the entire address space would be exhausted by the early 2000s. In 1993, the "classful" structure of IP addresses was eliminated. A new "semantics", called "Classless Inter-Domain Routing" ("CIDR"), was implemented, introducing the concept of an "IP-mask" (also known as a "subnet mask" or "CIDR mask"), which, as suggested above, makes it possible to group together numerically contiguous addresses and to allocate address blocks in a wider variety of sizes.[31]

None of these, so to speak, "reforms" increased the maximum number of IP addresses available for allotment, which remained at 4,294,967,296 (2^{32}), but, in 1998, "Internet Protocol version 6" or "IPv6" (RFC 2460) declared the creation of a new 128-bit address space, a *terra nullius* to which all Internet hosts are eventually expected to migrate. Like IPv4, IPv6 makes available only a finite number of available addresses (340,282,366,920,938,463,463,374,607,431,768,211,456 or 2^{128}), expressed as a series of four hexadecimal digits separated by colons, with each hexadecimal digit (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, or f) expressing a group of 4 bits. For example, "5f05:2000:80ad:5800:58:800:2023:1d71" is the hexadecimal expression of 010111110 000010100100000000000000100000001010110101011000000000000000000000001011000 000010000000000000100000001000110001110101110001 (Fall & Stevens 2011: 32). The new "empty" space has been gradually settled since the mid-2000s. On 17 January 2018, 18.41 % of computers used to request services from Google identified them-

30 Subnetting was introduced in August 1985 and is outlined in RFC 950. On subnetting, see Fall and Stevens (2011: 36–42) and Hall (2000: 408–10).

31 CIDR had been introduced in RFC 1518 and RFC 1519 and was clarified and modified in RFC 4632 (which obsoleted RFC 1519). Alongside the new semantics, CIDR instituted "route aggregation", which is a method for reducing routing complexity, simplifying routing decisions, and shortening packet-delivery times (Fall & Stevens 2011: 48–50; Hall 2000: 40–41; Mueller 2009: 37–38).

selves using a 128-bit address, whereas, on 4 April 2024, 41.45 % of computers used to request services from Google had already "settled" the new address space (Google IPV6 n.d.).[32]

As can be seen, control over chunks of the address space is conferred, not claimed. Until 1998, authority over assignment functions (referred to as "Internet Assigned Numbers Authority" or "IANA") was assumed, claimed, and asserted by the U. S. Department of Defense and "delegated" under contracts to Jon Postel at the Information Sciences Institute of the University of Southern California.[33] In 1998, the U. S. government published a nonbinding statement of policy (known as the "White Paper"), announcing its intention to relinquish authority over IANA functions by transferring them to a "globally representative" not-for-profit corporation, incorporated and headquartered in the United States (United States Department of Commerce 1998).[34] The same year, addressing authority was "delegated" under a U. S. Department of Commerce contract to the Internet Corporation for Assigned Names and Numbers ("ICANN"), a not-for-profit corporation incorporated in the State of California. While ICANN's contract with the Department of Commerce expired in 2006, the "transition" process was not concluded until 2016 and is, today, framed as entailing the transfer not of ownership or control but of "stewardship". Today, IANA authority is said to derive from ICANN's status as representative of the "community of stakeholders that has flawlessly coordinated the Internet's domain name and addressing systems since their inception [and] will continue to do so".[35]

What's more, IP addresses are distributed in a hierarchical system. IANA (today, a department of ICANN) allocates large chunks of addresses and delegates IANA authority not directly to organisations but to a limited number of top-level Internet Service Providers ("ISPs") and to five Regional Internet Registries ("RIRs"): APNIC (Asia/Pacific), ARIN (Canada, the United States, and many Caribbean and North Atlantic islands), RIPE NCC (Europe, the Middle East, and Central Asia), LACNIC

32 Hosts that are assigned a 32-bit address cannot interact directly with hosts that use a 128-bit address and vice versa. Exchange of messages between inhabitants of different address spaces is made possible and mediated using special protocol translators and other mediation techniques (ICANN 2011, p. 10).

33 Postel's operation was funded by the U. S. Defense Advanced Research Projects Agency ("DARPA") (Mueller 2009: 98).

34 See also Goldsmith and Wu (2006: ch. 3) and Mueller (2009: ch. 8).

35 "The transition isn't the U. S. Government handing over the Internet to any one country, company or group. The truth is that nobody, including the U. S. Government, has a 'control of the Internet' to hand over. The community of stakeholders that has flawlessly coordinated the Internet's domain name and addressing systems since their inception will continue to do so" (The IANA Stewardship Transition: What You Need to Know n.d.).

(Latin America and portions of the Caribbean), and AFRINIC (Africa and portions of the Indian Ocean). In turn, the RIRs allocate smaller blocks of addresses to regional ISPs. Top-level ISPs and regional ISPs assign smaller blocks of IP addresses to mid-level carriers that provide lower-speed connection services directly to end-users (Hall 2000: 414).

Significantly, under the current scheme, dominion over chunks of the address space (that is, networks) is delegated to and held by ISPs, instead of organisations. ISPs maintain control of the chunks of the address space that were allocated to them. In other words, ISPs do not assign the addresses that form the address blocks over which they hold IANA authority and do not delegate IANA authority (Mueller 2009: 37–38; Hall 2000: 4–5). Rather than a right to an address, Internet *Service Providers* grant a right of use,[36] framing Internet access as a *service* that is provided pursuant to a bilateral agreement and the addressed (Internet hosts as well as their operators) as *clients*.

Clients and Servers

"IP addresses are enough to identify a host, but they are not very convenient for humans to remember or manipulate (especially the long addresses used with IPv6)" (Fall and Stevens 2011: 19). Imagine needing to memorise or look up and type in 32-bit IP addresses, especially given the seeming meaninglessness and randomness of their assignment. Imagine having to remember the 128-bit IP addresses of the Web sites you visit most frequently (Mueller 2009: 39–40). Another difficulty with IP addresses stems from their instability. Users may change ISPs; databases that are used to provide Web sites may be moved from one server to another. Large-scale changes may also take place. Not only has the Internet addressing scheme been "reformed", multiple times, since its introduction, but also the "topology" of networks and subnetworks tends to change relatively frequently as they grow, shrink, or are being reconfigured. Any "change in the identifier of a computer connected to a network could wreak havoc on connectivity if thousands of internal and external users have stored the old address in their routers or browsers or email address books" (Mueller 2009: 40). This means that, for the Internet to be usable, a change of address needs to "be implemented seamlessly without disrupting users, without even requiring their participation" (Mueller 2009: 40).

36 Because IP addresses remain under the control of ISPs, to give an example of the implications, IP addresses are not portable across ISPs. If a client wants to switch ISPs, the client cannot take their address with them, which makes the process of changing ISPs costly and labour intensive (Mueller 2009: 22–23, 38).

To help computer operators rather than computers, the DNS allows for the use of mnemonic and stable alphanumeric names that serve as pointers for the retrieval of “resource records” (“RRs”) from databases that are used to provide services via a Web site (RFC 1034: s. 3.6). Rather than entering IP addresses, users can refer to Internet hosts on which such databases are stored using a semantically meaningful “domain name”, which is 1 to 63 octets (series of 8 bits) in length (RFC 1034: ss. 3.1, 3.5). Domain names, in other words, are not assigned to “sites”. Nor are they, technically speaking, “human-friendly” notations of “machine readable” IP addresses. Rather, each domain name is associated with a database that resides on an addressed and hence addressable Internet host. Processes (rather than Internet hosts or software applications) that provide services by transmitting RRs to other devices in response to queries are called “servers”, while processes that request RRs from servers on behalf of Internet users are referred to as “clients” (Fall and Stevens 2011, pp. 20–21). For example, “www.airbnb.com” is, technically speaking, not the name of a “Web site”. It is the name of a database, a set of resource records that is stored on an addressed Internet host, on which server processes run. When you press the “enter” key after typing “www.airbnb.com” into your Web browser, you trigger a client process that runs on your digital device. Acting on your behalf, the client requests data from a server that runs on a device that can be found in the IP address associated with the domain name (“www.airbnb.com”), rather than being transported to some other site.

Computer operators do not type domain names into the address bar of their Web browsers. What they usually enter is a type of “Uniform Resource Identifier” (“URI”) called a “Uniform Resource Locator” (“URL” or “Web address”). A URL is a hierarchical sequence that consists of three elements: a protocol name (“HTTP”), a domain name (“www.airbnb.com”), and a file path, separated by “/” (for example, “/help/article/971/how-do-i-know-if-an-email-or-website-is-really-from-airbnb”). For example, the URL “http://www.airbnb.com/help/article/971/how-do-i-know-if-an-email-or-website-is-really-from-airbnb” indicates the name of the resource record stored in the database associated with the domain name (“how-do-i-know-if-an-email-or-website-is-really-from-airbnb”), and the path (“/help/article/971/”) that the server process running on the computer hosting that database would have to follow to locate this resource.

Internet users, in other words, are not granted access to databases stored on Internet hosts on which server processes run. Rather, as will be explained in more detail below, upon receiving a navigation request, a client process that runs on the device operated by the user establishes a connection with a server process that runs on the device hosting the relevant database, requesting guidance on how to render the requested Web content (for example, a particular Web page). The requested rendering directions are transmitted to the client process by the server process.

The client follows the instructions, generating a "hypertext": a representation of the Web page to which the Internet user "navigates".[37]

The set of all domain names is referred to as the "domain name space" and is structured hierarchically as a "tree".[38] The domain name of a node represents its location in the name hierarchy (RFC 1034: s. 2.1). From the perspective of Internet hosts, it consists of a sequence of labels, where each label is a length octet (a byte or 8-bit string) followed by an octet string. However, from the perspective of Internet users, it is the list of the labels on the path from the node to the root of the tree, separated by periods ("."). The "labels can contain only alphabetical characters (A–Z), numeric characters (0–9), and the minus sign (–), and are printed or read left to right, from the most specific (lowest, farthest from the root) to the least specific (highest, closest to the root)" (RFC 1034: s. 3.1). The unnamed starting node, the root of the tree, does not have a pronounceable name that can be used to refer to it, but it does have a name, which is represented as a null (that is, zero length) string: " ". Because all domain names end at the root, a length byte of zero terminates the domain name of all other nodes. In other words, all domain names include the name of the root at the end. The root is the "parent" of multiple "top-level domains" ("TLDs") or "generic top-level domains" ("gTLDs"), including "com", "net", "org", "edu", and "tv", but also over 250 country-code top-level domains ("ccTLDs"), including "us", "eu", "de", and "kr".[39] A null label at the end of each top-level domain signifies that it is a "child" of " ", that is, of the unnamed root.[40] The root's children are themselves "parents". For example, the "subdomain" that bears the name "airbnb.com" is a child of "com", and "org" is the parent of "icann.org". The name "airbnb.com" indicates that the parent of the subdomain is "com", and that "com" is the child of " ". Many subdomains are "parents", including "www.airbnb.com". For example, "es.airbnb.com" is the child of "airbnb.com", while "airbnb.com.co"

37 A "hypertext" is a text that contains one or more "hyperlinks". "Hyperlinks are links to other [Web] resources that are generally exposed to the user by the user agent so that the user can cause the user agent to navigate to those resources, e.g., to visit them in a browser or download them" (HTML Standard: s. 4.6.1).

38 Prior to the introduction of the DNS in 1984, the domain name space was construed as a flat structure in the form of a two-column table mapping. All domain names were listed in a single text file called "hosts.txt" containing a simple two-column table: one column contained all allocated domain names and the other contained the IP addresses associated with each of these names. Internet users (mostly computer scientists) would download the file to their computers periodically, consulting it whenever needed (Fall & Stevens 2011: 516; Galloway 2004: 47–48; Mueller 2009: 40–41). See also notes 33 and 41 and accompanying text.

39 A "parent" is the "domain in which the Child is registered" (RFC 7344: s. 1.1).

40 A "child" is the "entity on record that has the delegation of the domain from the Parent" (RFC 7344: s. 1.1).

is the child of "com.co". The domain name "www.airbnb.com.pe" indicates that the parent of the subdomain is "com.pe", that "com.pe" is the child of "pe", and that "pe" is the child of " ".

The DNS is maintained by a "distributed and hierarchical" database system (RFC 1032). A key feature of this database is that it is not stored in a single location or on a single computer. Nor is the database managed by a central authority. Rather, different parts of the domain name database are stored on different Internet hosts and are maintained by different "organisations".[41] To elaborate, the tree structure of the domain name space allows for the partitioning of the database used to maintain the domain naming system into more manageable subsets, as well as for the delegation of DNS responsibility down the levels of a hierarchy. A subset of the database over which responsibility was delegated is called a "zone"; the holder of "authority" over and responsibility for a zone is called an "organisation"; and the process whereby authority is devolved so that a child can become a parent and thereby an authority is called "delegation" (RFC 1034: s. 2.4). Each organisation is required to maintain its own database of "authoritative" data, which includes data about the top node and the echelons directly below it, data that describes delegated subzones (that is, cuts around the bottom of the zone), and the contact information of the devices on which the DNS server programs employed by their children run.[42] "Because all of the data is expressed in the form of RRs", in other words, "a zone can be completely described in terms of a set of RRs" (RFC 1034: s. 4.2.1), that is, as "the complete database for a particular 'pruned' subtree of the domain space" (RFC 1035: s. 2.1).

When a process is instructed to request a service from another process, it must know the 32- or 128-bit IP address of that other device, not the name associated with the database that contains the indicated RR. To assist processes, the DNS treats the name space database as a directory that comprises name-to-address mappings and defines a query/response protocol for translating a domain name into an IP address, which also uses the client-server model. The process of looking up the IP address that corresponds to a particular domain name using the DNS is known as "resolution". In accordance with protocol, authorities are required to employ a "name server" to which other authorities and Internet hosts working for Internet users

41 As mentioned in note 38, before the introduction of the DNS, the name space database was contained in a simple text file, which was managed by a single authority. The distribution of DNS authority was framed as a response to the scaling problems arising out of the growth and commercialisation of the Internet. On the history of the naming scheme, see Fall and Stevens (2011: 516), Galloway (2004: 47–48), and Mueller (2009: 40–41).

42 A node is only responsible for providing data about itself and its children. Nodes are not expected or required to be able to provide information about, for example, their parents, "grandchildren", or "great-grandchildren".

can message in order "to provoke a response" (RFC 1034: s. 3.7).[43] Devices acting for Internet users employ client processes called "resolvers" to retrieve information associated with a particular domain name. Because the domain name database is distributed among multiple devices, the resolver must first locate the authoritative name server. The resolver starts by contacting the one name resolver known to it and submitting a request for resolution. If this name server is employed by the relevant authority, it provides the resolver with the IP address of the Internet host on which the data associated with the name resides. If the name server does not have the requested authoritative data, it refers the resolver to another name server that is "closer" to the desired information. In case of a referral, the resolver contacts the name server to which it was referred. If this turns out to be the authoritative name server, the IP address that is associated with the relevant domain name is sent to the resolver. If the name server is not authorised to provide the requested information, it refers the resolver to another name server. The process continues until the resolver locates the authoritative name server and retrieves the 32- or 128-bit IP address of the Internet host on which the data associated with the domain name specified by the user is stored (RFC 1034: ss. 3.7, 4; RFC 1035: s. 6).[44]

Just as end-users become "Members" of the "Airbnb Community" and, thereby, bearers of rights and obligations upon adherence to terms of service and use (Sheffi 2020: 520), legal persons constituted as service providers become recognised as owners of domains and employers of DNS servers upon acceptance of terms of service and use. To become an authority, a legal person has the DNS server or servers that it employs named and registers those domain names. To become an employer of a DNS name server, a legal person performing as a service provider must not only adhere to the terms set by the relevant parent domains, but also follow protocol, that is, accept and perform protocol's terms of service and use.[45]

While the address space is conceptualised as the set of all potential IP addresses, the domain name space is conceived of as the collection of all registered domain names, which means that each act of naming, unlike each act of addressing, serves to extend the name space. Accordingly, the domain name space has been conceptualised not as an exhaustible resource that must be preserved, but as an unbounded,

43 "Name servers are the repositories of information that make up the domain database. The database is divided up into sections called zones, which are distributed among the name servers. While name servers can have several optional functions and sources of data, the essential task of a name server is to answer queries using data in its zones" (RFC 1034: s. 4.1).

44 For an overview of the process, see Fall and Stevens (2011: 518–65).

45 "When some organization wants to control its own domain, the first step is to identify the proper parent zone, and get the parent zone's owners to agree to the delegation of control" (RFC 1034: s. 4.2.2).

constantly expanding space. The process of registering new domains has been construed as a race to register rights to "new" semantically meaningful names, unused areas outside the boundaries of the domain name space that can become part of the name space. Like the "extra spaces" that Airbnb purportedly enables us to add into the market, domain names are conceived of as potential resources, objects of property that can be utilised to provide a profitable service.[46] At the same time as it claims to create a world where anyone can belong, the DNS encourages us to become servers whose business is hospitality and for whom hospitality is a business: to acquire domains and offer hospitality, for personal gain, subject to conditions and limitations, terms of service and use.

Internet Explorers

To press the enter key ("↵") after typing "http://www.airbnb.com" into the address bar of a Web browser or to follow a "hyperlink"[47] to the Airbnb "homepage" is to be teleported to Airbnb, a recognisable site with a distinct and distinctive identity. Almost immediately, a hospitable Web page opens itself up, welcoming the browser to a domain named Airbnb.

A domain Name, however, is neither a "place" nor a "space", to play on Michel de Certeau's distinction (1984: 117–18).[48] As ICANN explains, just as "the purchase of a piece of land does not automatically result in a house being built on it", registration of a right to a domain name grants the licensed holder only the right to use the domain name during the restoration period (ICANN 2012: 7). A domain name is akin to an inaccessible empty lot, which means that visitors cannot be welcomed and serviced unless infrastructure and mechanisms are established. To offer hospitality, to continue the analogy, it is also not enough to build a structure such as a house. To again build on de Certeau, a house may be a place, but it is not a hospitable space. As Jacques Derrida observes, "in order to constitute the space of a habitable house and a home, you also need an opening, a door and windows, you have to give up a passage to the outside world [*l'etranger*]. There is no house or interior without a door or windows" (Derrida & Dufourmantelle 2000: 61). Hospitality, in other words, requires an interface, a surface that encloses a domain but also opens

46 On the political economy of the DNS, see Mueller (2009).

47 For a definition of "hyperlink", see note 37.

48 A "place" is an instantaneous configuration of positions, the abstract, theoretical order "in accord with which elements are distributed in relationships of coexistence"; whereas a "space" is a practised, experienced place (de Certeau 1984: 117–18).

this domain up to an outside, a portal that serves as an opening and that creates openings, opportunities for interactions with comers. This interface transforms the place into a domain that belongs but also is visitable (for example, a house into a habitable space, a home). It serves as a threshold, allowing for hospitality to be offered subject to threshold terms and conditions (Derrida & Dufourmantelle 2000. 55).

The place that needs to be "built" and operated for hospitality to be made possible is called a "Web site", and the user interface ("UI") that transforms this abstract, empty place into a practised space that welcomes and furnishes services to visitors is provided by a software application called a "Web browser".

Technically speaking, however, it is not *into* a domain or a site that visitors are welcomed. Airbnb users, for example, never really "access" the inhospitable domain or the hospitable site. As Derrida suggests, the very interface that renders a site hospitable by serving as a portal, that opens an "inside" to an "outside", forms a protective surface or barrier that can be used to control and prevent access: a gate, a door, or a window (Derrida & Dufourmantelle 2000: 55–73). Just like the door that the law created especially for Josef K., the Web page generated each time a computer operator uses their Web browser is a portal, one that is opened especially for that operator, at that very moment. Just like the door that the law created especially for Josef K., the portal created by the user interface is a gateway, one that the user will never be able to cross. Just like the door that the law created especially for Josef K., the interface generated each time a Web page is displayed to a computer operator is merely the effect of communication and an opening for mediated self-referential communications.[49]

A web browser, in other words, is precisely an interface. From the perspective of end-users, it is an interface that enables computer operators to visit Web sites and to request and receive services from their providers: Please allow me to view the site's homepage. Please search for listings that meet these specified criteria. Please display this listing. Please send a booking request to this host. Please accept the booking request made by this guest. Please make a payment to this host. Please send this message to this user. Please post this review to this listing. Technically speaking, however, a Web browser is an interface that enables coded processes called "clients" to request and receive from processes called "servers" guidance as to how to display a specified "Web resource" or "Web content" to the device operator. Requests for service, in other words, are not submitted by computer operators themselves. Nor are requests for service sent by Web browsers. Services are requested by multiple prompted processes that run on a particular digital device. Similarly, services are not provided by legal persons – the corporate entities that

49 For a systems theoretical reading of Kafka's fable, see Teubner (2013b: 405–22).

act as the service providers – or by their organs. Services are provided by processes that run on special digital devices employed for this task. When a computer operator takes any action using a Web browser, processes that run on the operated communication device format a request for services, locate the relevant server process, establish connection with it, and submit the query to it. Upon receiving a request for service, processes that run on the digital device working for the Web site provider send a set of rendering instructions to the processes running on the visitor's digital device. Processes that run on the device operated by the visitor follow the provided directions, make adjustments with the aim of adapting the representation to the features of the particular device, and generate the requested Web content (for example, a new Web page) by displaying it to the computer operator.[50]

To request and receive service, a process must adhere to the application-layer HTTP.[51] Like most other application-layer protocols, HTTP is a request/response protocol. Processes that initiate HTTP connections are called "clients", and processes that accept HTTP connections in order to service requests from clients are called "servers" (RFC 2616: s. 1.3; RFC 7540: s. 2.2). Whenever a device operator takes any action using the interface provided by their Web browser, a process acting as a DNS client discovers the IP address of the Internet host on which the database associated with the specified domain name resides. The same or another prompted process formats an "HTTP request message" in accordance with the HTTP protocol and sends the message down to the lower-level protocols. A TCP connection is established with a process that runs on the addressee. The data is fragmented, encapsulated, sent down to the link-layer protocols, and routed across the network at the direction of the addressed Internet host on which the server process runs. Once all the packages arrive at the destination, the message is reassembled and transmitted to the HTTP server. The HTTP request message is processed by the server and an "HTTP response message" is formatted. The server then sends the message down to the lower-level protocols. The data is fragmented, encapsulated, sent down to the link-layer protocols, and routed across the network at the direction of the Internet host on which the client process runs. Once all the packages are received, they are sent to the higher-level protocols. The response message is reassembled and transferred to the HTTP client. As will be explained below, the client follows

50 User agents are "encouraged to offer settings that override this default to improve the experience for the user, e.g. changing the color contrast, using different focus styles, or otherwise making the experience more accessible and usable to the user" (HTML Standard: s. 2.1.8).

51 The addition of "http://" before a URL instructs the client process to follow the HTTP protocol. The protocol is set out in multiple RFCs, notably RFC 2616 ("HTTP/1.1") and RFC 7540 ("HTTP/2"). Hypertext Transfer Protocol Version 3 ("HTTP/3"), which uses QUIC rather than TCP for the underlying transport protocol, was published as a Proposed Standard in RFC 9114.

the directions provided by the server, rendering the Web content that its operator requested to "access".[52]

HTTP response messages are composed in the markup language Hypertext Markup Language ("HTML") and also contain embedded directions written in style sheet languages such as Cascading Style Sheets ("CSS") and scripting languages such as JavaScript ("JS"), as well as links to external style sheets and scripts (HTML Standard: s. 1.3). They contain no tables, no graphics, and no pictures. Using text only, they provide instructions on how to present Web resources or Web content for end-users.[53] HTML is a mark-up language that is used to describe the structure of a Web page, giving "authors the means to: [p]ublish online documents with headings, text, tables, lists, photos, etc."; "[r]etrieve online information via hypertext links, at the click of a button"; "[d]esign forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc."; and "[i]nclude spread-sheets, video clips, sound clips, and other applications directly in their documents" (HTML 4.01 Specification: s. 2.2).[54] HTML further allows for the incorporation of links to "style sheets" and "scripts" or for the embedding of style sheets and scripts directly into HTML documents. The style sheet language CSS provides styling features, which can be used to specify the design and presentation of a Web page, for example, to define fonts, set text and background colours, and add decorative features (W3C 2021). The scripting language JS enables the creation of dynamically updating content, the animation of images, and the control of multimedia (HTML 4.01 Specification: s. 18).

Like Airbnb, the World Wide Web claims to enable and promote universal accesses: "to facilitate use of the Web by all people, regardless of their language, script, writing system, and cultural conventions" (Dürst et al. 2005).[55] Like Airbnb, however, the World Wide Web supports radical diversity by setting and executing terms of

52 For an outline of the process, see RFC 2616 and RFC 7540.

53 In accordance with the HTML specification, "[u]ser agents are not required to present HTML documents in any particular way" (HTML Standard, s. 15). The specification rather provides "a set of suggestions for rendering HTML documents that, if followed, are likely to lead to a user experience that closely resembles the experience intended by the documents' authors" (HTML Standard: s. 15).

54 On the relationship between HTML 4.01 Specification and HTML Standard, see HTML Standard, s. 13. HTML allows Web designers to encapsulate certain content by "tags" and instruct the Web browser how to display the encapsulated content to the user. For example, the tag "<p>" designates the beginning of a paragraph, and the tag "</p>" marks its end. Each heading is enclosed by a "<h>" tag and a "</h>" tag. The tag "<i>" instructs the software application to render all text from that point onwards in italics, while the tag "</i>" lets it knows when to stop italicising.

55 HTML was devised to serve as "a universally understood language, a kind of publishing mother tongue that all computers may potentially understand" (HTML 4.01 Specification: s. 2.2).

use that constantly work to standardise and obscure difference. Web resources are rendered by protocol-adherent processes that "implement" instructions expressed in a single, universal language, HTML, with the goal of standardisation and organisation and the aim of generating "a user experience that closely resembles the experience intended by [HTML] … documents' authors" (HTML Standard, s. 15).[56] While clients are "encouraged" to offer settings that override "suggested default rendering" in order to "improve the experience for the user" or make it "more accessible and usable to the user" (HTML Standard: s. 2.1.8), both implementation of the default suggestions and deviations from the suggested default rendering work to obscure rather than highlight divisions and the situatedness of user experiences, that is, to fabricate a "fantasy of global seamlessness" – to draw on Patricia Williams (1997: 1–16) – rather than to create a world where anyone can belong.

Every Web resource to which clients can request access is identifiable using a URI. A URL, for example, is a type of URI that is used to identify Web pages. As discussed above, a URI typically consists of three elements: the naming scheme of the mechanism used to access the resource (for example, "http://"), the domain name of the database containing the resource (for example, "www.airbnb.com"), and the name of the resource itself, given as a path (HTML 4.01 Specification: s. 2.1.1); the "www." part of the domain name tells your Web browser that you are looking for the Web interface for that domain name. The collection of identifiable Web resources that can be retrieved by client processes is known as the "World Wide Web" ("WWW" or the "Web") (HTML 4.01 Specification: s. 2.1).

Internet users, however, do not view the Web as a collection of Web resources. By allowing for the use of design language systems,[57] HTML enables the visual linking of Web pages and the formation of "Web sites" as distinct, stable, consistent, and coherent spatial identities,[58] generating a sense of the World Wide Web as an assemblage of radically diverse sites that can be visited.[59] Each rendering of a Web page both forms and transforms the user experience and user's experience of the

56 "[P]rotocols like HTML were specifically designed to allow for radical deviation in screen resolution, browser type, and so on. And HTML (along with protocol as a whole) acts as a strict standardizing mechanism that homogenizes these deviations under the umbrella of a unilateral standard" (Galloway 2004: 141).

57 On the concept of "design language" and the use of design language systems by Airbnb, see, for example, Saarinen (n.d.) and Hughes and Han (n.d.).

58 As highlighted by Butler, identities are not established at once, once and for all, but are continuously formed through iterative processes of materialisation that generate an impression of stability, fixity, and distinctiveness (2011 [1993]: xviii, xix).

59 Sites are commonly identified by a domain name or by a portion common to a set of domain names. For example, the site name "Airbnb" is used to refer to the collection of Web resources stored in the variety of databases associated with some or all of the over six hundred domain

Web site of which the page forms part, as well as the user experience and user's experience of the World Wide Web. Each generation of a Web page installs and orients views and perspectives, forming and promoting a platform.[60]

Much like pictorial representations drawn in linear perspective (Panofsky 1991: 27–28), Web browsers present themselves as windows, portals through which Internet users can explore a pre-existing, external world that opens up to the user. Offering themselves as books, they invite users to pick any page as their starting point, to jump from any page to any page, to browse through pages that are generated especially for them, upon request. Like the Members of the Airbnb Community, Internet users are installed with a sense of mastery, unrestrained choice, and entitlement, incited to embark on self-directed journeys across a wide web of welcoming domains: to choose any site as their point of embarkment, visit any site they wish, and navigate from any site to any site, at any point, instantaneously. Like the Members of the Airbnb Community, they are encouraged to travel, explore, and discover, but also to make places their own. They are iteratively configured as Internet explorers.

Communication Devices

The Internet, as has been revealed, is not, technically speaking, a network or a community of Internet users or "netizens". Nor is the Internet a network or community of Internet hosts. The Internet is also not an association of networks. Nor is the Internet a collection of protocols. The Internet is not a space composed of potential IP addresses. Nor is the Internet the set of all registered domain names. The Internet is further not composed of deposits of resources, databases associated with domain names, that can be used to provide a service and generate surplus value. Nor is the Internet a collection of Web resources such as Web pages that can be retrieved or a collection of Web sites that can be visited. Rather, like any other system of communication, the Internet is iteratively formed through coded processes that are initiated and carried out by devices: both digital computers and their operators.

names registered in the name of Airbnb, Inc., including "Airbnb.co.uk", "airbnb.com.ee", "airbnbblog.com", and "airbnbhelp.com" (Domain Names Owned by AirBnB Inc 2020).

60 The iterability of service provision allows for the introduction of changes to design, layout, presentation, and text, affecting user experience. Most changes to the user experience of a site are subtle (discernible or unnoticeable), and are introduced in stages as "tweaks", but changes can be part of a noticeable redesign and revealed as part of a brand campaign. For example, in 2014, alongside a brand campaign called "Belong Anywhere", the "entire Airbnb experience" was redesigned to "better reflect" what Airbnb "really" is and is "really" about: a "shared vision of belonging" (Chesky 2014).

As RFC 793 explains, “all communication is viewed as inter-process communication” (RFC 793: s. 2.1). “[F]rom the communication network’s point of view”, Internet hosts “are the sources and destinations of packets” but are not the “active agents that produce and consume messages” (RFC 793: s. 2.1). Messages are produced and consumed by “the active elements in host computers”, which are *processes*, with the “fairly common definition of a process” being “a program in execution” (RFC 793: s. 2.1).

The protocol-adherent programs (that is, implementations, codes, software, applications, and, colloquially, algorithms) that coders (that is, programmers) write are not, however, the instructions that target devices execute. Contrary to common notions of programs, with Lawrence Lessig’s canonical work being a paradigmatic illustrative example (1999: 236–37, 2006: 341–43), the programs that coders write are, technically speaking, neither “self-executing” nor “executable”, as highlighted by Wendy Chun (2011: 22–29). Expressed in “high-level” programming languages, as will be explained below, these software applications make absolutely no sense to the digital devices on which they are intended to run. As target machines cannot make sense of these programs, instructions that are issued in the programming languages in which coders write cannot prompt digital devices to act. Digital computers respond only to the pulses or frequencies of electricity that drive the different components inside a “processor”, treating these as signals that require response.

How then are devices interpellated to adhere? If the programs devised by coders today are incomprehensible to the digital computer and thus cannot trigger action, what prompts execution? How is execution enabled and of what does the apparatus consist?

Loosely speaking, the operation of a digital device involves the triggering of a serial issuance of electrical signals (“control signals”) that drive a processor to perform a set of mechanical either/or operations.[61] A processor might have more or less resources at its disposable. A processor might have a stronger or weaker – a bigger or smaller, faster or slower – memory. A processor might be able to rely on

61 In the past, an integrated circuit (“chip”) contained only one processor, and the terms “Central Processing Unit” (“CPU”), “processor”, “microprocessor”, and “core” were used interchangeably to refer specifically to that hardware component (Patterson & Hennessy 2021: 19). Today, however, chips not only consist of multiple processors that can execute instructions in parallel (Patterson & Hennessy 2021: 8, 43), but also are accompanied by at least one Graphics Processing Unit (“GPU”), which is a type of accelerator (in more simple terms, processor) that assists CPUs by performing more specialised tasks (Patterson & Hennessy 2021: 548–49). To avoid implying that execution is performed exclusively by one processor, distinguishing between processor types, and delving into the implications of parallel hardware, this contribution will use “processor” instead of technical terminology and indefinite articles when speaking generally about the operations of processing units.

bigger or smaller storage. A processor might be able to perform slower or faster. Performance by a processor might consume more or less resources. Regardless, a processor can be prompted to perform only those roles that it is configured to be able to perform in response to control signals.

Like social systems, digital communication systems use binary coding (Luhmann 2012: 132–35), meaning that they process by recursively executing a constitutive binary distinction. In broad strokes, "0" and "1" denote neither numerical values nor quantities. "0" and "1" also do not refer to states such as "open" and "close" or "on" and "off". Rather, as will be explained in this section, computers are constructed to interpret "0" and "1" as instructions relating to the state of a mechanical device called a "logic gate". A logic gate implements basic logic functions such as AND and OR by preventing or allowing the flow of electrical current (Petzold 2023: 65). Each processor contains not one logic gate but millions of them, and the state of each of these logic gates affects voltage levels. "0" triggers an electrical signal to block the flow of electrical current through the relevant circuit; "1" triggers a control signal to allow electrical current to flow through that circuit. Any change to the state of any gate affects the flow of current through the circuit and thus voltage levels. That is basically it, what the computer can do: lowering or increasing voltage by controlling the state of a logic gate, that is, by doing either/or (Petzold 2023: ch. 8; Patterson & Hennessy 2021: app. B).

Technically speaking, however, processors are configured to follow not commands that are expressed using abstract symbols like "0" and "1", but a limited number of "machine instructions" that are kept in the computer as sets of high and low electronic signals (Patterson & Hennessy 2021: 86). The set of machine instructions that a digital device is constructed to be able to respond to is called "machine language". As hinted in the opening of this paper, because machine instructions are rendered in hardware (rather than software), machine language is considered device-specific and is commonly referred to as "idiosyncratic" (Abelson & Sussman, with Sussman 1996: 768; Petzold 2023: 319). It is the very configuration of the machine, its constitution, its form. Computers of the same model may appear identical, that is, they may be standardly formed, but they do not share a language. Each computer converts software instructions into electrical signals and responds to electrical signals using the unique machine language that it is hardwired to be able to use, that is, using the configured language that forms part of its distinct constitution.

As processors are constructed to use machine language and can use machine language only, what they respond to is electrical signals that are mechanically prompted, rather than abstract commands that are expressed using abstract language. To bridge this "gap" between hardware (configured machine instructions) and software (the abstract language used to refer to these machine instructions), processors are constructed to be able to, so to speak, "interpret" unique sequences of "0" and "1" as citations of machine instructions, a process that is conducted by a component of the processor called the "Control Unit" ("CU") (Patterson & Hennessy 2021: 273–84; 269). The set of software instructions that a computer is configured to be able to interpret can be called "binary machine language" because it is designed to mirror machine language.[62] Each element of this binary language, that is, each of the unique sequences of bits of which the language consists, corresponds to a particular element of machine language.

As computers are configured to respond only to citations of machine language, the building blocks or atoms of *any code* must be citational representations of machine instructions (Patterson & Hennessy 2021: 79). Put differently, as will now be explained, programming may be seen both as a process of breaking down a complex task into subtasks that correspond to machine instructions and as a process of building abstractions with procedures and data. The former represents a top-down view of programming, whereas the latter expresses a bottom-up view of it. To return to the point, all computer programs are, in effect, configurations of citations of binary machine instructions, which are themselves citations of machine instructions. Just like persons (to be distinguished from psychic systems),[63] digital devices are conferred interpretive authority, the power to make sense of demands that are expressed in abstract language, to translate citations into a language that makes sense, to give meaning to citations of embodied binary codes. Just like persons, digital devices carry out the execution process by making judgement, by determining what is expected of them by way of performance.[64] Just like persons, they are interpellated to take charge of the execution process, to assume "respon-

62 While the two terms are often conflated in the technical literature, in this contribution, I distinguish between "machine language" and "binary machine language", using the former to refer to the set of electrical operations that the machine is hardwired to be able to perform and the latter to denote the binary language that computer designers and scientists use to represent these operations.

63 Social systems theory distinguishes between "persons" and "individuals" (Teubner 2006: 333–36; Luhmann 2012: 57; 1995: chs. 6–7): persons are conceived of as semantic constructs that serve as communication devices. What appears to be a use of one social language by a person to convey meaning, in effect, entails simultaneous self-referential processes of meaning-generation internal to the various social language systems in effect being used. Individuals (mind-bodies) are conceptualised as psychic systems (rather than social systems) that belong to the environment of social system. Because language is necessarily social, psychic systems cannot themselves communicate, but this does not mean that psychic systems do not affect the reproduction and transformation of social systems. Psychic systems make a difference, but only indirectly, by "irritating" communications of social systems.

64 On citationality as a formative and transformative communicative practice, see Butler (1993: 71–72, 171–72), building on Derrida (1988: 18), and also Butler (1997: 48–52).

sibility for the constraints of power", and to become "the principle of [their] own subjection" (Foucault 1995: 202–03).

While processors perform electrical operations and respond only to instructions that are expressed as control signals, programmers do not think in terms of hardware. Rather, programmers think in computational terms, which is precisely why we came to call digital devices "computers". What enables the employment of digital devices as computers is that each of the elements of a binary machine language is also assigned a computational meaning. In other words, while processors interpret binary machine language instructions as signs that correspond to machine instructions as to the control of voltage levels, programmers understand these very binary machine language instructions as signs that refer to basic computational operations. These include data handling operations such as read, write, move, load, and store data; arithmetic operations such as addition and subtraction; logical operations such as NOT, AND, OR and Exclusive-OR (XOR); and control-flow operations (decision-making during the running time of a program) such as branch, jump, and loop (Patterson & Hennessy 2021: ch. 2).

It is not only that programmers do not think in terms of the electrical operations with which machine instructions are associated. Programmers today also do not think in terms of those basic computational operations that the elements of binary machine languages simultaneously represent. Software applications are today rarely written in binary language and are virtually never written in a binary machine language, that is, in a binary language that mirrors the hardware of a particular device. The reasons cited by computer scientists are diverse. Programming in binary language is considered a tedious, time-consuming, and error-prone process (Petzold 2023: 425; Patterson & Hennessy 2021: 14). It also requires familiarity with the underlying structure and particular features of the device on which the program is expected to run (Bryant & O'Hallaron 2016: 200; Petzold 2023: 425–28). Furthermore, with the growth in the number of digital devices, the development of general-purpose computers, and the commercialisation of the computer, it has become inefficient and impractical to write programs that are hardware dependent. The number of computer models is ever-growing and new "generations" of processors are constantly introduced (Petzold 2023: 428). Requiring coders to constantly learn new binary languages and to keep up with changes to existing languages would have had significant implications for digital technology. What's more, computing today strives to devise not software applications that individual devices can execute, but programs of, so to speak, "general applications" or programs that can run on populations or classes of devices (Abelson & Sussman, with Sussman 1996: 489; Petzold 2023: 428–30). Computing also claims to offer computational solutions to almost all problems that humans as individuals or populations or that organisations and institutions face, including problems that are deemed "big" and "complex" and are said to require computational "thinking" at a high level of abstraction (Abelson & Sussman, with Sussman 1996: xxii–xxiii, 294–95).[65]

Virtually all computer programs devised today are written in hardware-independent languages that are "erected on a machine-language substrate, [and] hide concerns about the representation of data as collections of bits and the representation of programs as sequences of primitive instructions" (Abelson & Sussman, with Sussman 1996: 489).[66] These programming languages make it possible to express commands in general language by describing "primitive expressions" that represent the simplest entities with which the particular language is concerned, that is, "primitive procedures" and "primitive data", and providing methods for combining and abstracting procedures and data, that is, for thinking at an even higher level of abstraction (Abelson & Sussman, with Sussman 1996: 6). While certain primitive expressions in higher-level languages may correspond to binary machine language instructions, they are often defined by abstracting binary machine language (operations and data). To give a simple and simplified example, binary machine languages generally assign the meaning of "addition" to a particular machine instruction, but do not include a primitive instruction that expresses that idea of "exponentiation" (Patterson & Hennessy 2021: ch. 3). When programming in binary machine language, coders would need to use addition to define a more abstract function that computes exponentials. Higher-level programming languages, however, can include a primitive procedure called "exponentiate", which would be conceived of as corresponding to a compound procedure in binary machine language that performs exponentiation by means of repeated addition. Coders would be able to define even more abstract procedure using this primitive operation, which would save them the need to use repeated addition (Abelson & Sussman, with Sussman 1996: 17–18, 57–62).[67]

65 While it is admitted that certain problems cannot be described in computational terms, the theory of computation also concedes that three problems that can be described computationally nonetheless cannot be solved through computation (Sipser 2012: 3). Demonstrated over a decade before the first digital computer was turned on, the insolvability of those problems reveals not only the limits of computational thinking (Sipser 2012: 195), but also the paradox of self-reference, the very foundational paradox observed by social systems theory (Teubner 2019: 319–20, citing Luhmann 2012: 219). In 1931, Kurt Gödel showed that the completeness and consistency of formal systems cannot be proved (Gödel 1931), and, in 1936, both Alonzo Church (1936) and Alan Turing (1937) demonstrated the insolvability of the decidability problem ("Entscheidungsproblem").

66 The term "primitive" is used to denote the most basic units of a programming language.

67 Remember, exponentials can also be computed using successive squaring or multiplication, and both squaring and multiplication are themselves compound procedures that represent repeated addition.

While all programming languages provide methods for abstraction in the way illustrated above, programming languages are commonly distinguished according to their own level of abstraction. Binary machine languages are considered the "lowest level" programming languages because their elements mirror those of machine language. All other programming languages are considered of "higher level" than binary machine languages but may be labelled as "higher" or "lower" level depending on the context. Today, most programming is done in programming languages that are generally considered to be "high level" rather than in binary machine languages or in languages such as assembly languages that are generally considered "low level" (Petzold 2023: 428).

As explained above, those "high-level programs" are neither self-executing nor automatically executable. To be executable, a high-level program must be translated into the binary machine language of the particular device on which it needs to run. This task is not performed by the same programmers that wrote the high-level program. In fact, the task is not performed by programmers at all. Rather than manually, translation is generally done automatically, using a special type of computer programs known collectively as "translators". Three types of translators are used to translate from higher-level into lower-level languages: "compilers", "assemblers", and "interpreters".[68] A compiler accepts as input entire programs (referred to as "source codes" or "source programs"), written in a certain high-level language (referred to as the "source language"), and translates these programs completely into a lower-level language such as assembly language or binary machine language. The outputs generated by compilers are called "object codes". Specifically, translations into a model-specific binary language of instructions that correspond to machine instructions are called "executables", even though, as will be explained below, these translations are, technically speaking, *not* executable. Once a program is compiled, the generated translation can be stored on the device for future execution, meaning that the object code can be executed repeatedly, without further translation (Abelson & Sussman, with Sussman 1996: 770–882). An assembler, which can be considered a type of compiler, translates into binary machine language from assembly language: a low-level programming language of which elements are mnemonics that correspond to binary machine instructions such as MOV, ADD, JMP, and HLT (Petzold 2023: 425–28; Patterson & Hennessy 2021: 130–32; Abelson & Sussman, with Sussman 1996: 704–08). Finally, an interpreter not only translates source programs written in a particular source language, but also arranges for their execution

68 Other types of translators are used to translate between high-level languages, between low-level languages, from machine code to assembly language, from machine language into one or another higher-level language, etc. On the advantages and disadvantages of compilation versus interpretation, see Abelson and Sussman, with Sussman (1996: 769, 827–29) and Petzold (2023: ch. 27).

by the processor. It is written in the binary machine language of the device on which it runs, and its execution involves going over the inputted program line-by-line, alternately converting lines into a binary language that mirrors machine language without generating an object code, and passing the commands to the processor one by one (Abelson & Sussman, with Sussman 1996: 768–69; Bryant & O'Hallaron 2016: 42–43).

Like all other translations (Teubner 2000, drawing upon Derrida 2007), the outputs generated by compilers, assemblers, and interpreters cannot be considered "proper" translations (Chun 2011: 21–25). In technical terms, different programming languages allow for different operations, which means that compilation is not a straightforward process. Each statement in a high-level language needs to be broken down into multiple instructions in the lower-level language. In many cases, this can be achieved in a variety of ways, but, because translation is a mechanical process, these are not considered, as the execution process mechanically follows the program instructions. The higher the level the source language is, the more general are the statements that need to be translated and the more numerous are the choices that must be made in the process of translation. What's more, compilers include instructions on how to adapt programs to the specific characteristics of the executing device, which means that translation in the form of compilation involves the making of adjustments (Petzold 2023: 430). Many compilers also incorporate optimisation techniques, which are devised to allow for the optimal utilisation of a machine's specific capacities and to minimise resource consumption (Abelson & Sussman, with Sussman 1996: 827).

Proper or not, it is also not even these automatically generated translations that digital devices execute. The object codes generated by compilers and assemblers, including executables, are not executable, and that is because they are expressed using languages that correspond to machine language, rather than being in machine language. Machine language is rendered in hardware, forming part of a device's constitution. The device is configured to process using it. And because of this, it is only the device of which constitution machine language forms part that can use it. The final stage of any execution process, that is to explain, *requires* an assumption of responsibility by a target device, constitutional conversion, and a transformation of a communication device into the principle of its own subjection.[69] Like all other communication devices, digital computers are powered and empowered to process using their own "brain", "brawn", and "memory", to borrow the terminology employed by Patterson & Hennessy (2021: 19). Like all other communication devices,

69 As explained by Petzold, operation codes are known as "machine codes" precisely "because they are used directly by the machine – the circuitry that constitutes the central processing unit" (2023: 319).

digital computers are prompted to process information by exercising interpretive power. At the final stage of the communicative process, the device is interpellated to determine what is required of it by way of performance. It must translate the binary machine instructions into machine instructions, that is, into a series of electrical signals (Patterson & Hennessy 2021: 90–91; Petzold 2023: 364). These electrical signals are then transmitted to the relevant mechanical components that drive the operation of the device, triggering performance: the carrying out of either/or operations, the execution of binary codes, the making of a difference.[70]

Constitutional Processors

In a previous article, "We Accept: The Constitution of Airbnb", I investigated the transformation of Airbnb as a platform for a more general reflection on the constitution of "domains" and subjects, as well for observing a structural change: an evolution of our social systems', so to speak, "terms and conditions" (Sheffi 2020).

That article proceeded in four moves: it began by proposing a novel conceptualisation of digital platforms as "domains", linking the use of the term in information technology and in legal, social, and political theory. Prompted by Gunther Teubner's "irritations" (including in 2004; 2012: 55-56, 106-107; 2013a; 2017), the article continued by rethinking the transformation of one such domain, Airbnb, in terms of auto-constitutionalisation. This process, the article suggested, also renders observable a "structural change", what appears as an emergence of *a* "generalizable model of functioning", to draw on Michelle Foucault (1995: 205), and, in systems theoretical terms, can be described as the "co-evolution" of our social discourses/systems through "structural couplings" (Teubner 1993: ch. 4, 2002; Luhmann 2012: ch. 3). These emergent terms of use and engagement are legitimated through iterative citation of a reformulation of the social contract idea: the social contract is no longer conceived of as a multilateral agreement between community members on the constitution of a sovereign on which powers of defined scope are conferred. Rather, the social contract is framed as a network of bilateral contracts between a service provider and individual clients, Terms of Service agreements that are drafted by the service providers and are not open for negotiation, what is known doctrinally as "standard form contracts" or "contracts of adhesion". The article concluded by tracing the links between the auto-constitution of social systems, domains, and the subject, describing the rise to dominance of a new mode of subjectivation.

70 As defined by Luhmann, "Information is a difference that changes the state of a system, thus generating another difference" (2012: 113).

Through the operation of digital devices and the use of the Internet, to unpack the last of those four moves, subjects are iteratively constituted as “clients”; the organisation that assumes and exercises dominion assumes the role of a “service provider”; and its agents are employed as “servers”. The relationships between clients and the service provider are encoded as bilateral, and so are the relationships between them and other clients. Because the contracts being cited are understood as bilateral, clients, servers, and service providers are iteratively framed as “third parties” (but not “third party beneficiaries”) – clients as third parties to the bilateral relationships between corporate entities and other clients and to the bilateral relationships between other clients, servers as third parties to agreements between a service provider and a client and to agreements between the service provider and other servers, and service providers as third parties to the bilateral relationships between clients. Through iterative acceptance and performance of these encoded terms of service and use, subjects are standardly formed as uninterested yet self-interested, uninformed, uninquisitive, and unquestioning adherents.

Three questions that the previous article left open are how domains are constituted and constitutionalised, how subjectivities as discursive artefacts are standardly formed, and how structural change takes effect and becomes observable, and it is these three questions that have concerned the current contribution. In answering the three questions, this article has drawn on the insight that both social systems and digital technology recursively generate content by executing constitutive system-specific binary codes, which has also been highlighted by Luhmann (2012: 119) as well as by Vismann and Krajewski (2008: 91). It has built up an account of digital processes as a method for illustrating systems theory and for elaborating a general systems theoretical account of “process”, “constitution”, and “structural change”.

To avoid transposing institutional and statist notions of process, as systems theory urges us to do, the contribution has “broken” the Internet, to turn the phrase (Break the Internet n.d.), countering common notions of what the Internet is, before proceeding to present a description of how Internet communication works. To avoid treating the Internet as a black box and focusing on the meaning and presentation of the “content” that digital communication makes available (texts, audio files, video files, images, etc.), as systems theory further urges us to do, the contribution has deconstructed and reconstructed Internet communication, illustrating the internal processes through which social systems as communication systems “constitute themselves with the aid of the distinction between *medium and form*” (Luhmann 2012: 113–20).[71]

71 Also calling for a shift in focus from content (media as experienced by end-users) to the process of transfer, Vismann and Krajewski demonstrate that media are not “mere tools” for the transmission of content, but rather “the conditions of possibility for communication” (2008: 101–102).

As has been shown, the Internet, the World Wide Web, and Web sites such as Airbnb are, to again borrow from Foucault, "transactional realities" (*réalités de transaction*): domains of reference born "from the interplay of relations of power and everything which constantly eludes them, at the interface … of governors and governed" (2008: 297). These domains are not places in which persons interact. They are also not collections of communication devices or communities of which persons are members. Rather, these domains are iteratively constituted and constitutionalised through prompted communicative processes of binary-code execution that make use of communication devices. Each operation of a digital computer through use of an interface does not entail *a* transmission of information from one Internet user to another in a language that is shared, or *a* transfer of data from one digital device to another in a common language (Luhmann 1995: 139, 2012: 116). Each operation of the device via an interface prompts multiple, automatic, and mechanical processes that run on multiple devices (digital computers and persons), generating communication.

The constitutional processes through which domains are iteratively formed, transformed, and auto-constitutionalised, the very processes through which subjects are standardly formed and transformed, the very processes through which discourses evolve and structural change comes to be observable, take place iteratively, *bit by bit*, through serial executions of system-specific binary codes that form part of our constitution as configured communication devices.

Bibliography

Standards

Internet Engineering Task Force [IETF] (1981) *Internet Protocol, STD 5, RFC 791*. Status: Internet Standard. Available from: https://www.rfc-editor.org/info/rfc791 [Accessed 16 July 2023].

IETF (1981) *Transmission Control Protocol, RFC 793*. Status: Internet Standard. Obsoleted. Available from: https://www.rfc-editor.org/info/rfc793 [Accessed 16 July 2023].

IETF (1985) *Internet Standard Subnetting Procedure, STD 5, RFC 950*. Status: Internet Standard. Available from: https://www.rfc-editor.org/info/rfc950 [Accessed 16 July 2023].

IETF (1987) *Domain Administrators Guide, RFC 1032*. Status: Unknown. Available from: https://www.rfc-editor.org/info/rfc1032 [Accessed 16 July 2023].

IETF (1987) *Domain Names – Concepts and Facilities, STD 13, RFC 1034*. Status: Internet Standard. Available from: https://www.rfc-editor.org/info/rfc1034 [Accessed 16 July 2023].

IETF (1987) *Domain Names – Implementation and Specification, STD 13, RFC 1035*. Status: Internet Standard. Available from: https://www.rfc-editor.org/info/rfc1035 [Accessed 16 July 2023].

IETF (1989) *Requirements for Internet Hosts – Communication Layers, STD 3, RFC 1122*. Status: Internet Standard. Available from: https://www.rfc-editor.org/info/rfc1122 [Accessed 16 July 2023].

IETF (1989) *Requirements for Internet Hosts – Application and Support, STD 3, RFC 1123*. Status: Internet Standard. Available from: https://www.rfc-editor.org/info/rfc1123 [Accessed 16 July 2023].
IETF (1993) *An Architecture for IP Address Allocation with CIDR, RFC 1518*. Status: Historic. Available from: https://www.rfc-editor.org/info/rfc1518 [Accessed 16 July 2023].
IETF (1993) *Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy, RFC 1519*. Status: Proposed Standard. Obsoleted. Available from: https://www.rfc-editor.org/info/rfc1519 [Accessed 16 July 2023].
IETF (1998) *Internet Protocol, Version 6 (IPv6) Specification, RFC 2460*. Status: Draft Standard. Obsoleted. Available from: https://www.rfc-editor.org/info/rfc2460 [Accessed 16 July 2023].
IETF (1999) *Hypertext Transfer Protocol – HTTP/1.1, RFC 2616*. Status: Draft Standard. Obsoleted. Available from: https://www.rfc-editor.org/info/rfc2616 [Accessed 16 July 2023].
IETF (2006) *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan, BCP 122, RFC 4632*. Status: Best Current Practice. Available from: https://www.rfc-editor.org/info/rfc4632 [Accessed 16 July 2023].
IETF (2014) *Automating DNSSEC Delegation Trust Maintenance, RFC 7344*. Status: Proposed Standard. Available from: hhttps://www.rfc-editor.org/info/rfc7344 [Accessed 16 July 2023].
IETF (2015) *Hypertext Transfer Protocol Version 2 (HTTP/2), RFC 7540*. Status: Proposed Standard. Obsoleted. Available from: https://www.rfc-editor.org/info/rfc7540 [Accessed 16 July 2023].
IETF (2017) *Internet Protocol, Version 6 (IPv6) Specification, STD 86, RFC 8200*. Status: Internet Standard. Available from: https://www.rfc-editor.org/info/rfc8200 [Accessed 16 July 2023].
IETF (2022) *HTTP/3, RFC 9114*. Status: Proposed Standard. Available from: https://www.rfc-editor.org/info/rfc9114 [Accessed 16 July 2023].
World Wide Web Consortium [W3C] (1999) *HTML 4.01 Specification*. Superseded Recommendation. Available from: https://www.w3.org/TR/html401/ [Accessed 16 July 2023].
Web Hypertext Application Technology Working Group [WHATWG] (n.d.), *HTML Standard*. Available from: https://html.spec.whatwg.org/multipage/ [Accessed 16 July 2023].

Literature

AS137437 (n.d.) Available from: https://ipinfo.io/AS137437 [Accessed 16 July 2023].
Abbate, Janet (1999) *Inventing the Internet*. Cambridge, Mass: MIT Press.
Abelson, Hal & Sussman, Gerald Jay with Sussman, Julie (1996) *Structure and Interpretation of Computer Programs.* 2nd ed., Cambridge, Mass: MIT Press.
Augarten, Stan (1984) *Bit by Bit: An Illustrated History of Computers.* New York: Ticknor & Fields.
Baran, Paul (1962) *On Distributed Communications: I. Introduction to Distributed Communications Networks*. Santa Monica, CA: RAND Corporation.
Barlow, John Perry (2001) A Declaration of the Independence of Cyberspace, pp. 27–30 in Peter Ludlow (ed.), *Crypto Anarchy, Cyberstates, and Pirate Utopias.* Cambridge, Mass: MIT Press.
Beckers, Anna & Teubner, Gunther (2021) *Three Liability Regimes for Artificial Intelligence: Algorithmic Actants, Hybrids, Crowds.* Oxford: Hart Publishing.
ICANN (2011) *Beginner's Guide to Internet Protocol (IP) Addresses*. Available from: https://www.icann.org/resources/files/ip-addresses-beginners-guide-2011-03-04-en [Accessed 6 September 2023].
ICANN (2012) *Beginner's Guide to Domain Names*. No longer available online.
Break the Internet (n.d.) Available from: https://www.merriam-webster.com/wordplay/break-the-internet [Accessed 6 September 2023].

Bryant, Randal E. & O'Hallaron, David R. (2016) *Computer Systems: A Programmer's Perspective*. 3rd ed., Boston: Pearson.
Butler, Judith (2011 [1993]) *Bodies That Matter: On the Discursive Limits of "Sex"*. London: Routledge.
Butler, Judith (1997) *Excitable Speech: A Politics of the Performative.* New York: Routledge.
Carveth, Rod & Metz, J. (1996) Frederick Jackson Turner and the Democratization of the Electronic Frontier. *The American Sociologist* 27(1): 72–90.
Cerf, Vinton G. (1990) *Oral history interview with Vinton G. Cerf*. Charles Babbage Institute. Available from: https://hdl.handle.net/11299/107214 [Accessed 16 July 2023].
Cerf, Vinton G. (2000) Forward, pp. vii–ix in Eric A. Hall, *Internet Core Protocols: The Definitive Guide*. Sebastopol, CA: O'Reilly.
Chesky, Brian (2014) Belong Anywhere. Available from: http://blog.atairbnb.com/belong-anywhere/ [Accessed 27 November 2020].
de Certeau, Michel (1984) *The Practice of Everyday Life*. Berkley: University of California Press.
Chun, Wendy Hui Kyong (2011) *Programmed Visions: Software and Memory*. Cambridge, Mass: MIT Press.
Church, Alonzo (1936) An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics* 58(2): 345–363.
Derrida, Jacques (1988) Signature Event Context, pp. 1–23 in Jacques Derrida, *Limited Inc*. Evanston, IL: Northwestern University Press.
Derrida, Jacques & Dufourmantelle, Anne (2000) *Of Hospitality.* Stanford: Stanford University Press.
Derrida, Jacques (2007) Des Tours de Babel, pp. 191–225 in *Psyche: Inventions of the Other, Volume I*. Stanford: Stanford University Press.
Domain Names Owned by AirBnB Inc (2020). Available from: https://robbiesblog.com/domain-names-owned-by-airbnb-inc/7744 [Accessed 16 July 2023].
Dürst, Martin J., Yergeau, François, Ishida, Richard, Wolf, Misha & Texin, Tex (2005) *Character Model for the World Wide Web 1.0: Fundamentals*. Available from: https://www.w3.org/TR/charmod/ [Accessed 16 July 2023].
Edwards, Paul N. (1996) *The Closed World: Computers and the Politics of Discourse in Cold War America*. Cambridge, Mass: MIT Press.
Fall, Kevin R. & Stevens, W. Richard (2011) *TCP/IP Illustrated, Volume 1: The Protocols*. 2nd ed., Upper Saddle River, NJ: Addison-Wesley.
Foucault, Michel (1995) Discipline and Punish: The Birth of the Prison. 2nd Vintage Books ed., New York: Vintage Books.
Foucault, Michel (2008) *The Birth of Biopolitics: Lectures at the College de France, 1978–1979*. Basingstoke: Palgrave Macmillan.
Galloway, Alexander R. (2004) *Protocol: How Control Exists after Decentralization*. Cambridge, Mass: MIT Press.
Galloway, Alexander (2006) Language Wants to Be Overlooked: Software and Ideology. *Journal of Visual Culture* 5(3): 315–331.
Gillespie, Tarleton (2010) The Politics of 'Platforms'. *New Media & Society* 12(3): 347–364.
Gödel, Kurt (1931) Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik Physik* 38: 173–198.
Goldsmith, Jack L. & Wu, Tim (2006) *Who Controls the Internet? Illusions of a Borderless World*. New York: Oxford University Press.
Google IPV6 (n.d.). Available from: https://www.google.com/intl/en/ipv6/statistics.html [Accessed 4 April 2024].
Hall, Eric A. (2000) *Internet Core Protocols: The Definitive Guide*. Sebastopol, CA: O'Reilly.

Hughes, Hayley & Han, Yujin (n.d.) Systems Thinking, Unlocked: Building an Inclusive Design Language System. Available from: https://airbnb.design/systems-thinking-unlocked/ [Accessed 16 July 2023].

Krajewski, Markus (2018) *The Server: A Media History from the Present to the Baroque*. New Haven: Yale University Press.

Lessig, Lawrence (1999) *Code and Other Laws of Cyberspace*. New York: Basic Books.

Lessig, Lawrence (2006) *Code: Version 2.0*. New York: Basic Books.

Luhmann, Niklas (1995) *Social Systems*. Stanford: Stanford University Press.

Luhmann, Niklas (2012) *Theory of Society, Volume 1*. Stanford: Stanford University Press.

Mueller, Milton L. (2009) *Ruling the Root: Internet Governance and the Taming of Cyberspace*. Cambridge, Mass: MIT Press.

Panofsky, Erwin (1991) *Perspective as Symbolic Form*. New York: Zone Books.

Patterson, David A. & Hennessy, John L. (2021) *Computer Organization and Design: The Hardware/ Software Interface. MIPS Edition*. 6th ed., Kidlington: Morgan Kaufmann.

Petzold, Charles (2023) *Code: The Hidden Language of Computer Hardware and Software*. 2nd ed., Redmond, WA: Pearson Education.

Piper, Andrew (2021) Digitization, pp. 402–406 in Ann Blair et al (eds.), *Information: A Historical Companion*. Princeton: Princeton University Press.

Saarinen, Karri (n.d.) Building a Visual Language Behind the Scenes of Our New Design System. Available from: https://airbnb.design/building-a-visual-language/ [Accessed 16 July 2023].

Sheffi, Nofar (2020) We Accept: The Constitution of Airbnb. *Transnational Legal Theory* 11(4): 484–520.

Simmel, Georg (1906) The Sociology of Secrecy and of Secret Societies. *American Journal of Sociology* 11(4): 441–498.

Sipser, Michael (2012) *Introduction to the Theory of Computation*. 3rd ed., Boston, MA: Cengage Learning.

Teubner, Gunther (1993) *Law as an Autopoietic System*. Oxford: Blackwell.

Teubner, Gunther (2000) Contracting Worlds: The Many Autonomies of Private Law. *Social & Legal Studies* 9(3): 399–417.

Teubner, Gunther (2002) Idiosyncratic Production Regimes: Co-evolution of Economic and Legal Institutions in the Varieties of Capitalism, pp. 161–182 in John Ziman (ed.), *The Evolution of Cultural Entities: Proceedings of the British Academy*. Oxford: Oxford University Press.

Teubner, Gunther (2004) Societal Constitutionalism: Alternatives to State-centred Constitutional Theory? 3–28 in C. Joerges, I.-J. Sand & G. Teubner (eds.), *Transnational Governance and Constitutionalism*. Oxford: Hart.

Teubner, Gunther (2006) The Anonymous Matrix: Human Rights Violations by 'Private' Transnational Actors. *Modern Law Review* 69(3): 327–346.

Teubner, Gunther (2012) *Constitutional Fragments: Societal Constitutionalism and Globalization*. Oxford: Oxford University Press.

Teubner, Gunther (2013a) The Project of Constitutional Sociology: Irritating Nation State Constitutionalism. *Transnational Legal Theory* 4(1): 44–58.

Teubner, Gunther (2013b) The Law before Its Law: Franz Kafka on the (Im)possibility of Law's Self-Reflection. *German Law Journal* 14(2): 405–422.

Teubner, Gunther (2017) Horizontal Effects of Constitutional Rights in the Internet: A Legal Case on the Digital Constitution. *Italian Law Journal* 3(1): 193–206.

Teubner, Gunther (2019) Exogenous Self-Binding: How Social Subsystems Externalise Their Foundational Paradoxes in the Process of Constitutionalisation, pp. 317–338 in Gunther

Teubner, *Critical Theory and Legal Autopoiesis: The Case for Societal Constitutionalism*. Manchester: Manchester University Press.
The IANA Stewardship Transition: What You Need to Know (n.d.). Available from: https://www.icann.org/iana-transition-fact-sheet [Accessed 16 July 2023].
Turing, Alan M. (1937) On Computable Numbers, with an Application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society* s2-42(1): 230–265.
United States Department of Commerce (1998) *Management of Internet Names and Addresses* (980212036-8146-02). Available from: https://www.icann.org/resources/unthemed-pages/white-paper-2012-02-25-en [Accessed 6 September 2023].
Vismann, Cornelia & Krajewski, Markus (2008) Computer Juridisms. *Grey Room* 29(12): 90–109.
W3C (2021) CSS Snapshot 2021. Available from: https://www.w3.org/TR/css-2021/ [Accessed 16 July 2023].
Williams, Patricia J. (1997) *Seeing a Colour-blind Future: The Paradox of Race*. New York: Farrar, Strauss & Giroux.
Wu, Tim (2010) *The Master Switch: The Rise and Fall of Information Empires*. New York: Atlantic Books.
Yen, Alfred, C. (2002) Western Frontier or Feudal Society? Metaphors and Perceptions of Cyberspace. *Berkeley Technology Law Journal* 17(4): 1207–1263.