Turing, the Internet and a Theory for Architecture

A (Fictional?) Tale in Three Parts

Dirk Trossen

Computer Laboratory
Cambridge University

dirk.trossen@cl.cam.ac.uk

PREFACE

The late noughties have seen an influx of work in different scientific disciplines, all addressing the question of 'design' and 'architecture'. It is a battle between those advocating the theory of 'emergent properties' and others who strive for a 'theory for architecture'. We provide a particular insight into this battle, represented in the form of a story that focuses on the role of a possibly unusual protagonist and his influence on computer science, the Internet, architecture and beyond. We show his relation to one of the great achievements of system engineering, the Internet, and the possible future as it might unfold.

Note from the writer: The tale is placed in a mixture of reality and fiction, while postulating a certain likelihood for this fiction. There is no proof for the assertions made in this tale, leaving the space for a sequel to be told.

1. Part I: Turing and the Foundations

It is the early 1900s. The mathematical world is abuzz with the grand challenges presented by David Hilbert during the International Congress of Mathematicians in 1900. These 23 grand challenges [1] provide a comprehensive collection of open problems that Hilbert sees as crucial to drive forward the axiomatic view on mathematics. As part of this movement, Hilbert poses the *Entscheidungsproblem* (decision problem) in 1928, searching for an answer to the problem whether or not an algorithm could decide upon mathematical statements in a formal expression. In a way, Hilbert asks for an algorithmic foundation to finding solutions to any problem one might be able to (formally) articulate. It becomes clear that a solution to this problem will become crucial to the foundation to scientifically argue about algorithms and design of systems at large.

A young student from Cambridge University, Alan Turing, enters the scene. Following his undergraduate years from 1931 to 1934, Turing turns his interest in 1935 to Hilbert's Entscheidungsproblem. He re-formulates Goedel's earlier results in [2] by introducing a concept of formal and hypothetical machines called *Turing machine (TM)*. He proves that such a concept is capable of performing any conceivable mathematical computation, provided it is representable as an algorithm. Turing's major insight regarding reformulating Goedel's work is to turn any algorithm into data, which in turn can be algorithmically manipulated (by a Turing machine). With this intuitive and easily accessible concept, he is now able to prove that there is no solution to Hilbert's Entscheidungsproblem: he shows that it is not possible to decide algorithmically, whether a given Turing machine would ever halt.

More importantly, however, Turing's work provides another concept that will prove influential at a later stage, the *Universal Turing Machine* (UTM), an incarnation of a Turing Machine that is capable to execute any other Turing Machine. In other words, a UTM is capable of computing anything that is computable; a proposition that is only later recognized for its true importance. It is here that the aspect of "turning an algorithm into data" is utilized. It is the algorithm describing a TM and its input data that altogether becomes "data" for a UTM. This proposition is only later recognized for its true importance.

Turing publishes his work in 1936 [3], around the same time Alonzo Church's work on the *lambda calculus* [4] is published. Hilbert's Entscheidungsproblem is finally solved, some 6 years after the retirement of David Hilbert. Turing goes on to directly work with Church from 1936 to 1938, completing his Ph.D. dissertation at Princeton University. Returning to England in 1938, Turing starts working for the Government Code and Cypher School. With 1939 seeing the outbreak of the Second World War, the British government increases its code-breaking efforts at Bletchley Park with Turing being at the heart of crucial contributions to the success of these efforts[1].

After the war, Turing returns to the emerging field of computer science in his efforts to contribute to the design of the ACE computer. He presents his ideas for a 'stored program computer' in 1946, while von Neumann appears on the scene with his work related to the EDVAC [ref] (a binary computer that advanced the ENIAC decimal computer [ref]). Turing is getting disillusioned by the delay of building the ACE, this delay largely caused by the secrecy of the overall program; a fate that surrounds many of Turing's contributions to science.

Turing joins the Computer Laboratory at Manchester University where he becomes a Reader in 1948 and eventually Deputy Director in 1949. His interests turn towards Artificial Intelligence. He devises the *Turing Test* [5] as a benchmark for artificial intelligence and performs a self-experiment with his colleague Alick Glennie in which

[1] Much has been written about his contributions to the war efforts. Despite these extraordinary contributions, we focus on aspects relevant to this story: architectures and Turing machines.

he simulates a computer executing a chess program – Turing looses against Glennie.

In trying to understand the wider implications of his early work on UTMs, Turing turns towards biology at the beginning of the 1950s. His interest lies particularly in the mechanisms that underlie the pattern formation within organisms. His contributions to the field of *morphogenesis* are fundamental in the forming of this field but much of his work goes unpublished until being recovered in 1992.

His endeavor to widen the applicability of his seminal work on Turing machines to other areas of design and system architecture is finally disrupted through the events in 1952. Turing is convicted for indecency due to a homosexual relationship. As a result of his conviction, Turing looses his security clearance and is barred from governmental programs. On June 8th, 1954, Alan Turing commits suicide.

Until now, Turing's work certainly had its impact on the beginning of computer science, establishing him as a father figure for what is to become the architecture of modern computers. But there are other children, most notably the modern Internet, which will spring from the foundations he set out; it is this wider fatherhood that will be addressed in the following parts of our tale.

2. Part II: Engineering the Foundations

It is the 1960s. System engineering and computer science have seen a dramatic increase in attention, sparked by the cold war, the race in the space and other large-scale efforts. Computers slowly become a common tool for development, albeit still as a scarce resource in dedicated computing centers. The 'stored program' computer becomes the standard architecture for computer hardware and software. The EDVAC work, as well as the resulting *von Neumann architecture,* have become widely accepted as the foundation to computing architectures – computing has become mainstream in science and system engineering. This continues during the late 60s with the development of networked computers. Foundations are laid to core parts of what is to become the Internet. Turing's influential work abounds in many of the developments albeit its foundations have entered the 'mainstream of system engineering' and are therefore mainly attributed to von Neumann.

Layering emerges as a central design principle for the computing and networking architectures that are being built. It yields powerful concepts such as operation systems (with individual applications utilizing the exposed virtualized resources of the machine where they are executed) and layered network protocol stacks. System engineers utilize the power of layering in order to progress the design of networked architectures, expanding individual computing resources towards large-scale networks.

Fast forward to the late 60s. The *Internet* is the new kid on the block. Born in efforts funded by US agencies, the Internet is touted as the *child of many fathers*. The Internet's fatherhood is commonly shared among at least three fathers: Cerf, Kahn and Berners-Lee[2] due to their core contributions in areas such as IP protocol design, the foundational design of an L2 (Ethernet) layer, and later the addition of the 'linked data' paradigm. Throughout all this, the foundational design of computers is so much engrained into every computer engineer's brain that Turing's influence and contribution is largely overlooked, only upheld in its importance within formal computer science and mathematics.

After the 80s, contributions to understanding the principles of design are making their way into the debates between computer and networking engineers. Layering as a principle is central in these discussions. In an attempt to rationalize the Internet's design decision to limit in-network functionality, Saltzer et al first argue in [6] that functionality should be executed "end-to-end"[3]. Work such as Day's [8] returns to the aspect of layering as a principle with the intention to introduce a foundational view on layering and its usage in large-scale systems, aided by developments in object-oriented computing as well as interface specification in requirements engineering. Through this, layering finds its entry into mainstream system design by eventually standardizing the layers that are widely used in the various communication systems of today[4]. Again, a foundational part for the design of large-scale systems, *layering*, becomes almost naturally engrained in every system engineer's brain.

The rising importance of the Internet and its increasing inclusion into every day's life and the social fabric as a whole attracts the attention of those who want to understand the Internet and its foundation of design from a larger societal perspective. Clark et al. [9] argue that design is more than a technological question: it is the power to shape societal tussles at large, enabling or preventing them, but ultimately defining the *good* or *bad* nature of any design in accommodating conflicts that arise within the artifact created. As Clark et al. put it so pointedly "*We, as technical designers, should not try to deny the reality of the tussle, but instead recognize our power to shape it.*" [9] *Modularization of functions* becomes key to this power, as outlined in [9], confining conflicts within well defined 'tussle spaces' that characterize the conflicts over the control exerted on these functions. While seminal in its

[2] The fatherhood of the Internet is not without contention and mainly depends on the importance that various communities place on the contribution of each of the prospective fathers.

[3] The "end" in this argument is mainly associated with the "end" of a transport connection and still remains the common viewpoint in many debates, e.g., on network neutrality. Clark et al., however, introduce the view in [7] that a 'trusted endpoint' might well be the center of an end-to-end communication.

[4] The ISO/OSI model of seven layers is one of these attempts, largely the basis for many other standardization bodies. The less layered "Internet model" leaves out session and presentation layers in favor of a single application layer on top of transport.

importance, the work provides insight largely through examples and therefore contributes little towards an underlying scientific approach to decompose the functions; a criticism often brought against the work.

Together with the increasing socio-economic interest in the Internet, a growing branch of science appears on the stage. It interprets the Internet as a system with *emergent properties*, with popular examples such as the scale-free nature of the Internet's topology (with Barabasi [10] being a prominent advocate[5]). Such emergent properties are seen as key to understanding the Internet and possibly designing alternatives or evolutions of it. One such property is the *hourglass nature* of the Internet's protocol stack. The work by Akhshabi and Dovrolis in [12] studies this aspect from the perspective of a characteristic emergent property of any architecture. In other words, they argue that any layered architecture would inevitably have an hourglass shape. However, their work provides little insight regarding the overall performance of any such (hourglass-shaped) architecture. In other words, an hourglass property can still be the property of a 'bad' architecture that is wasteful in its overall utilization of (key) resources. Such viewpoint, however, is characteristic for a movement that interprets design as *evolution with minimal tuning*. Here, certain emergent properties are seen as the result from careful perturbations that are imposed on a system by various technological and societal stakeholders; i.e., the system is seen as being too large to be 'designed' overall. It has led to a trendy direction of *new sciences of complex networks* (NSCN), which sees architecture overall as an emergent property that can hardly be designed upfront.

The start of this millennium sees several efforts that aim for an underlying theory regarding the design of large-scale systems, such as but not limited to the Internet. One area is that of microbiology. The work of Gerhart and Kirschner [13], for example, connects the structure and behavior of biological systems to key principles of design, most notably layering. Through their observations, they argue that layering provides a framework to "deconstrain" highly constrained biological sub-systems so that each sub-system can optimally operate within its constraints, while being integrated into a larger system. Expanding on Gerhart and Kirschener, Doyle et al. [14] connect the behavior of the Internet with that of other control systems. In this connection, the authors characterize the nature of the Internet as that of *robust-yet-fragile systems*; robust in its operations within the intended parameters but often catastrophically fragile when unintended perturbations occur. It is the beginning of connecting well-known principles for design, such as layering, with advances in control and optimization theory. Isolated successes showcase the potential that lies in forward-engineering novel solutions for key parts of the Internet, such as TCP, based on solid control theory [15] and such improved understanding of design principles that are utilized in the system engineering community.

Chiang et al. observe in [16] that there exists an extreme heterogeneity in the space of content and services as well as in the set of resources at the infrastructural level. They argue that layered architectures and the protocols defined within each layer introduce homogeneity through strictly defined interfaces, constraining interactions through a well-defined protocol[6]. These individual protocols can be optimized against well-defined constraints with the interfaces enabling well-defined feedback across layers. This strictness in layered structure and its well-defined interactions ensures the robustness against perturbations for which the individual protocols are designed.

From an architectural perspective, Chiang et al. argue that this structural approach allows for decomposing the architecture along boundaries within which each sub-problem can be optimized against well-defined constraints. These constraints can (be) change(d) over the lifetime of the system along as the structural components adhere to the well-defined feedback through the exposed interfaces. With that, the authors position the recognized layering principle as an approach to decompose an optimization problem into well-defined (and solvable) sub-problems. Through such early work in optimization and control theory, Chiang et al. provide a possible mathematical foundation for designing as well as engineering robust solutions within each layer, while allowing for the composition towards a larger architecture. Chiang et al. refer to this framework for optimization decomposition as *deconstraining constraints*[7] and we will encounter this term later again.

Interfaces define a process of information exposure (and hiding) and therefore an information flow across layers. This flow enables the creation of economic markets through defined *information asymmetries* [17][8]. While the well-defined constraints within individual layers provide the robustness for which the individual protocols are designed, the process of layering itself modularizes the individual protocols along functional spaces. This combination introduces flexibility in the presence of optimized resource utilization at each layer, with feedback

[5] Work in [11] has shown since that power law relations do not necessarily apply to engineered systems like the Internet, as reflected by its technological topology of routers and links.

[6] It is the heterogeneity above and below the internetworking layer together with the homogenization through layering that leads to the observed hourglass nature in the Internet; an observation utilized in the evaluation approach in [12].

[7] In the remainder, we use the term 'deconstraining constraints' in reference to the decomposition framework that Chiang et al. postulate. In part III, we connect this framework to the design principles of layering and modularizing.

[8] The constraints within each layer potentially include societal concerns as much as technological ones. Hence, the information flow should be interpreted as being possibly influenced by policy decisions such as open access obligations.

mechanisms across layers ensuring robustness of the overall architecture. Although the advantages of modularity are well recognized in the systems community, even prominent examples in the Internet do not strictly follow this principle, for example TCP includes IP-level information in its headers therefore tying TCP very strongly to IP. While this does not preclude robustness per se, it introduces the potential for fragilities outside the constraints that were assumed during its design.

Within the context of designing large-scale communication systems, available resources are that of *computation* over and *storage* of information as well as the *transfer* of information through communication links. The constraints that limit this optimization potential are technological and economic ones. Two trends exemplify the dramatic shift in constraints since the beginning of the Internet. The first one is that of available storage and computing resources. At the beginning of the Internet, these resources were scarce. Hence, transmitting bits represented an alternative to purchasing more local computing and storage resources; remote access was key. Computing and storage have become significantly cheaper, while also becoming almost ubiquitously available, in particular in the developed world. This has influenced the attitude of many users. Services that utilize distributed resources are on a constant rise. Bittorrent-like large-scale dissemination of files has found widespread adoption in the Internet. This adoption recognizes that the WHAT of a communication is likely to exist in many more places than the WHO that originally disseminated the information.

The second trend is that of wireless communication. While in its infancy at the time of designing today's Internet, it has become a dominant form of accessing communication resources worldwide. Wireless communication differs significantly from wireline communication in that a notion of a priori endpoint location hardly exists. While this has led to many solutions for mobile communication, an encompassing architectural framework that properly integrates wireless resources as yet another optimization dimension into the communication system, is still missing.

As observed in [9], system design largely decomposes the larger problem a priori. This inevitably leads to inefficiencies and fragility in the long run due to the emergence of perturbations that diverge from the original expectations of the designer. The quest to find an answer to sustainable evolvability of any large-scale system, however, has become a battle between the camps studying the Internet as a system of complex networks (with few emergent properties that can be minimally tuned over time) and those who aim at finding a unifying set of concepts and theories that allow for engineering robust and evolvable solutions in the future (with [18][19] only being recent examples of this movement).

In all this, the impact of Turing's work, including his early insights into the relation between Turing machines and morphogenesis, all seem long forgotten.

3. Part III: Return to Theory

We are still in 2012, Turing's centenary. The battle to develop a greater understanding on architecture and design turns towards the role of his theory. While optimization and control theory have proven helpful in backward-engineering the Internet as we know it, doubts are cast that it alone will provide the set of tools required for forward-engineering a sustainable (future) Internet or any large-scale system for that matter. It is here that Turing finally re-appears on stage. This re-appearance is driven by connecting the Internet to Turing's insights, placing Turing at the heart of a theory for design of distributed systems. *It therefore points to Turing when it comes to the true fatherhood of the Internet specifically and the understanding of distributed systems in general!*

But how do we possibly move from Turing's understanding of what has become the foundations of modern computers to a theory for architecture that allow for building distributed systems of any kind? Let us start by connecting the development of store-and-program computers and the Internet to Turing's major insight, namely the universal execution of any algorithm [3]. For this, we interpret services in the Internet as Turing Machines with their input being the service-specific parameters and the output being the result(s) of the service implementation. We can then formulate the Internet and its enabled content and services space in a 'Turing-esque' manner:

> ***Corollary 1[9]:*** *The Internet (as a distributed system) constitutes a UTM, implementing the specific TMs of its enabled service and content space.*

Intuitively, this extension to Turing's main theorem is not surprising, given the UTM nature of the individual computational resources (i.e., computers) that contribute to the overall service execution. Furthermore, our understanding of the Internet as an environment that can execute any service over any topology and link layer technology intuitively confirms Corollary 1, too. We have to understand, however, that this corollary does not interpret a particular 'layer' of the architecture as the universal TM[10]. What constitutes the UTM is the Internet as a whole, i.e., as a system as well as a way of assembling the software components that realize its various design and implementation choices.

The interpretation of the Internet as a UTM suggests a generality with respect to the services and applications that can be executed. However, many attempts exist to limit this generality through specific mechanisms such as deep packet inspection, firewalls, and others. These attempts are

---

[9] A popular version of this corollary was postulated by John Gage in 1984 with his phrase "the network is the computer" [20]. Although meant as a marketing slogan, it captures much of our understanding of distributed systems such as the Internet.

[10] One is easily tempted to see the current IP layer as such UTM. Such view would miss the real power behind this corollary.

often driven by various economic interests (such as service differentiation) or by regulatory interventions into the generality of the Internet (such as censoring content according to some regional policy). In the following, we leave out this societal dimension. However, when thinking of policies being expressed through utility functions, they can be brought back into consideration.

Applying Turing's work to distributed tasks, such as Internet routing, is not novel (as [21], among others, outlines). But it is this formulation of the corollary and the *deconstraining constraints* decomposition framework of Chiang, Doyle, Gerhart and Kirschner [16][13] that leads us to formulating the relation between the *layering principle* and Turing machines in the following manner:

> ***Hypothesis 1:*** *Layering deconstrains individually constrained TMs within each layer.*

According to Chiang et al., layering (and the strictness it introduces through its well-defined interfaces) decomposes the overall architecture along the individual constraints that are introduced through the protocol at each layer, i.e., layering allows for *deconstraining the individual constraints introduced at each layer*. Following Corollary 1, we see the protocol for each layer being realized by a Touring machine, with Corollary 1 extending the execution of the protocol at each contributing node to the overall distributed system.

Decomposing the architecture along boundaries that are set out by well-defined constraints of individual sub-problems that occur within the architecture is not limited to the principle of layering. In the following corollary, we argue that Hypothesis 1 allows for linking the decomposition outlined by Chiang et al. to the *modularity principle* that is well known and accepted in the systems community (discussed, among others, in [9])[11].

> ***Corollary 2:*** *Deconstraining constraints allows for modularizing the overall distributed system in order to localize computation.*

Through such modularization, resources are virtualized towards functionality outside of this modularization. Well-known 'virtualization' boundaries in today's computing systems exist, for instance, between application and the OS. Usually, finer-grained virtualization is provided throughout the system. For instance, memory could be virtualized through a page file abstraction. The work in [22] provides a memory virtualization that is closer to Goedel's work in that it introduces associative memory in which data and its association algorithm is encoded as data again[12], providing powerful tools for accessing this memory. We return to this work in at a later stage.

Having interpreted the (constrained) protocol within each individual layer as executing individual TMs, we now link the execution of these TMs to the decomposition framework that is defined by Chiang et al. [16]:

> ***Corollary 3:*** *Deconstraining constraints provides a framework for efficient and optimal TM execution.*

Here, efficiency and optimality are defined through utility functions, which in turn are defined through the modularity that is determined by the particular Turing machine[13]. Defining such utility is part of the overall protocol design process albeit its definition very rarely happens in explicit, i.e., mathematical, form. Examples for possible utility functions span from traffic elasticity over resource allocation efficiency and fairness to user satisfication; they therefore span from the engineering over the economic to the behavioral (end user) domain. The particular selection of the most appropriate utility function is driven by the particular optimization problem but also by the interests of the parties involved in its solution.

We now connect the deconstraining aspect to the global operation of the Internet. For this, we define *recursively deconstraining* as the recursive application of Hypothesis 1 and Corollary 2, i.e., recursively decomposing through modularization and layering (see [23] for an intuitive application of this recursive decomposition). With that in mind, we formulate:

> ***Hypothesis 2:*** *Recursively deconstraining introduces flexibility with the ability to optimize the global operation of the underlying UTM (that is the Internet).*

This hypothesis reconciles the need for *flexibility*, as expressed in Corollary 2, and for *optimality*, as expressed in Corollary 3, within the decomposition (through layering and modularization) that occurs in a divide-and-conquer approach for designing systems. The hypothesis states that such reconciliation 'naturally' leads to a recursive form of decomposition (in the form of layering and modularizing) while providing the ability to optimize the overall performance of the system.

This breaks with notions of strictly defined layers, such as defined in the OSI model, in favor of a flexible layering where the arrangement of layers can be changed due to adjustments in utility functions[14]. Flexible layering, however, is not new to the Internet. The notion of *overlaying* introduces flexible layering into the

[11] Modularity and layering within this decomposition framework leads us to the principle of subsumption.

[12] Linear association is commonly used in computing architectures, while simple forms of associative memory exist. The authors in [22] suggest a broad replacement of the simple linear form with more complex associations that are generally connected to the 'purpose' of the information being stored.

[13] In practical terms, the modularity of the Turing machine is defined by the particular requirements of the desired service and content delivery being implemented by the Turing machine. The utility function captures the boundaries for 'optimally' implementing these requirements in mathematical form.

[14] The occurrence of *cross-layer violations* in today's Internet in favor of optimizing operations across existing layers is an expression of the engineering community of this property.

architecture. In most cases, however, overlays are oblivious to the overall performance to the system. The same goes for tunneling techniques, utilized, e.g., for partial deployment of IP multicast or newer IP versions such as IPv6. What we can observe from these techniques is that their desire for indirection is often born from the need to circumvent optimization goals introduced by the layers the indirection itself depends on, therefore often leading to a non-optimal operation of the overall system.

Key to the reconciliation in Hypothesis 2 is *choice*. The role of choice in systems has been observed at an architectural level in work such as [9], while economic arguments in [17] study the role of choice in establishing (viable) markets. While recursive deconstraining provides the flexibility that we seemingly want and need, it is choice that provides the ability to optimize the global operation of the system through 'picking and choosing' the right sub-problem realizations for implementing the overall system. We return to this aspect when outlining the concepts for designing an Internet UTM.

For now, we turn our attention to a key concept in Turing's work from 1936, namely that of *symbols*. In his approach to the Entscheidungsproblem, symbols take the role of encoding *data* as well as *algorithms* in order to intuitively embed Goedel's work in a model for what has become *computing* as we know it. This emphasis on information can be connected to recent work (such as [24][25][26]) that creates a paradigm shift at the internetworking level based on purely information-centric foundations[15]. A key observation in these ideas is the importance of *information* within a world of experiencing the Internet that moves from the WHO is communicating (which is the basis for the current internetworking layer) to the WHAT is communicated. Common across all these proposals is the identification of information (either at object or packet level) with a service model that is centered on disseminating the information throughout the network.

With this connection to recent work in the networking domain and the enabling nature of *choice* in the quest for optimality, we can now relate the engineering of distributed systems back to Turing's insight that formulates algorithms as data, which can itself be manipulated[16]. For this, let us define the following concepts:

Definition 1: *We define* ***Information*** *as encoded symbols in the Turing sense, i.e., data as well as algorithms being represented as data. We further define* ***Mediation*** *as a function for matching demand for and supply of information under given utility functions.*

[15] The authors in [27] argue that the current IP-based Internet can already been seen as an information-centric system, so we include the current Internet in our viewpoint.

[16] See [22] for an example of such algorithmic data interpretation in the OS space, while [28] provides examples of algorithmic encoding of flows in information-centric networking systems.

Corollary 4: *The following concepts implement an Internet UTM: (i)* ***Information*** *as per Definition 1, (ii)* ***Mediation*** *as per Definition 1, (iii)* ***Deconstraining constraints*** *as a decomposition framework per Hypothesis 1 and Corollary 2 and 3, and (iv)* ***Recursive deconstraining*** *as per Hypothesis 2.*

It is striking that this corollary extends our initial Corollary 1 by laying out the set of concepts that defines the assembly of a variety of design and implementation choices as a UTM. The TMs in this case are the software components that implement these specific choices. The selection of these choices is driven by the flexibility/ optimality tradeoff that is captured in Hypothesis 2.

We recognize that for many in the community, this connection of principles for large-scale system design to Turing's foundational theory appears far-fetched and might cause incomprehension, disbelief as well as rejection. However, ongoing engineering efforts to re-design the internetworking layer, such as the work in [29], postulate similar design concepts as outlined in Corollary 4. The similarities are largely (i) the focus on information-centric identification, (ii) the brokering of resources through dedicated demand/supply matching, and (iii) the construction of constrained communication relationships at runtime. Dedicated solutions along these lines are embedded into a recursive design framework, outlined for instance in [29][23], that is similar to Corollary 4.

In addition, we recognize the substantial work on formal methods since Turing's work, removing many of the restrictions (such as sequentiality) of Turing's original approach. We assert that these formal methods that stem from Turing's foundational work, generally lead to the connection to the Internet as outlined in Hypothesis 1.

This connection between Turing and architecture is not expected to silence the community in awe but to stimulate a line of thought that needs further exploration, concretization, adjustment, and finally proof. We believe that such line of thought provides appealing connections between novel insights in optimization theory and the understanding of other systems in, e.g., biology, cyberphysical systems as well as neuroscience and the fundamentally theoretical work that Turing provided more than 65 years ago. Finally, it timely positions Turing as one of the fathers of the Internet and system design in general.

Let us fast-forward to 2026. We envision that engineering distributed systems, ranging from nano- to global-scale types in fields that spread from communication over cyberphysical to neuroscience and biology, will have entered the mainstream of engineering. The underlying 'theory of design' will have found its entry in well-accepted best practices for system design, while allowing for concise statements of the constraints for which these systems have been built and the reasoning over its functioning. It will have been 90 years since Turing published his groundbreaking work. It is finally established as the cornerstone to building ANY distributed system.

4. ACKNOWLEDGEMENTS

The creation of this tale is based on work and numerous discussions with members of the wider research community. Special thanks go to John Doyle, Srinivasan Keshav, John Wroclawski, Karen Sollins, Jon Crowcroft and Dana Pavel for their input and critical review of this article.

5. BACKGROUND READING

[1] D. Hilbert, "Mathematical Problems", Bulletin of the American Mathematical Society, vol. 8, no. 10, pp. 437-479, 1902

[2] K. Goedel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I.", Monatshefte für Mathematik und Physik 38: 173-98, 1931

[3] A. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem", Proceedings of the London Mathematical Society. 2 42: 230–65. 1936–37

[4] A. Church, "An unsolvable problem of elementary number theory", American Journal of Mathematics, 58 (1936), pp. 345–363

[5] A. Turing, "Computing Machinery and Intelligence", Mind LIX (236): 433–460, Oct 1950

[6] Saltzer, J. H., D. P. Reed, and D. D. Clark (1981) "End-to-End Arguments in System Design", Proc. of the 2nd Intl. Conference on Distributed Computing Systems, April 8–10, 1981. IEEE Computer Society, pp. 509-512.

[7] D. Clark, M. Blumenthal, "The End-to-End Argument and Application Design: The Role of Trust," Conference on Communication, Information, & Internet Policy (TPRC), 2007

[8] Day, J., "Patterns in Network Architecture - A Return to Fundamentals", Prentice Hall, 2008.

[9] D. Clark, J. Wroclawski, K. R. Sollins, R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet," in IEEE/ACM Transactions on Networking, Vol. 13, No. 3, 2005

[10] Barabási, Albert-László; Albert, Réka. "Emergence of scaling in random networks". Science 286 (5439): 509–512, Oct 1999

[11] Li, L., Alderson, D., Willinger, W., and Doyle, J. "A First-Principles Approach to Understanding the Internet's Router-Level Topology", Proc. ACM SIGCOMM, 2004.

[12] S. Akhshabi, C. Dovrolis, "The Evolution of Layered Protocol Stacks Leads to an Hourglass-Shaped Architecture", ACM SIGCOMM, 2011

[13] Gerhart J, Kirschner M., "The theory of facilitated variation", Proceedings of the National Academy of Sciences of the United States of America. 2007

[14] J. Doyle, D. L. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, W. Willinger, "The robust yet fragile nature of the Internet", Proceedings of the National Academy of Science, 2005

[15] Wei, David X.; Jin, Cheng; Low, Steven H. and Hegde, Sanjay (2006). "FAST TCP: motivation, architecture, algorithms, performance". IEEE/ACM Trans. on Networking 14 (6): 1246–1259.

[16] Chiang M, Low SH, Calderbank AR, Doyle JC, "Layering as optimization de-composition: A mathematical theory of architecture", Proc IEEE 95:52–56, 2007

[17] B. C. Greenwald, J. E. Stiglitz, "Externalities in Economies with Imperfect Information and Incomplete Markets", Quarterly Journal of Economics, no. 90, May 1986

[18] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, A. Singla, J. Wilcox, "Intelligent Design Enables Architectural Evolution", Proc. of ACM HotNets Workshop, 2011

[19] W. Willinger, J. Doyle (2004), Robustness and the Internet: Design and Evolution. In *Robust design: A Repertoire of Biological, Ecological, and Engineering Case Studies*, E. Jen (Editor), Oxford University Press

[20] J. Gage, "The Network is the Computer", http://en.wikipedia.org/wiki/John_Gage, 2012

[21] J. Crowcroft, "Turing Switches: Turing machines for all-optical Internet routing", Technical Report UCAM-CL-TR-556, Cambridge University, 2003

[22] I. Piumarta, "An association-based model of dynamic behaviour", Workshop on Free Composition at ECOOP, 2011

[23] N. Fotiou, G. Polyzos, D. Trossen, "Illustrating a Publish-Subscribe Internet Architecture", Telecommunication Systems, Springer, Special Issue on 'Future Internet Services and Architectures: Trends and Visions', Mar. 2013, at http://www.springerlink.com/content/t6m0k022042088t5/fulltext.pdf

[24] T. Koponen et al., "A Data-Oriented Network Architecture", ACM SIGCOMM, 2007

[25] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, R. Braynard, "Networking Named Context", Proceedings of ACM CoNext conference, December 2009

[26] D. Trossen, M. Sarela, K. Sollins, "Arguments for an Information-Centric Internetworking Architecture", ACM Computer Communication Review, April 2010

[27] L. Popa, A. Ghodsi, I. Stoica, "HTTP as the Narrow Waist of the Future Internet", Proc. of HotNets workshop, 2010

[28] PSIRP, "Update on the Architecture and Report on Security Analysis", Deliverable 2.4 at http://www.psirp.org, 2009

[29] PURSUIT, "Architecture Definition, Component Descriptions and Requirements", Deliverable 2.3 at http://www.fp7-pursuit.eu, 2011