Elements of the Theory of Dynamic Networks

The Challenge of Computing in a Highly Dynamic Environment

Othon Michail
Department of Computer Science, University of Liverpool, Liverpool, UK &
Computer Technology Institute, Patras, Greece
Othon.Michail@liverpool.ac.uk

Paul G. Spirakis
Department of Computer Science, University of Liverpool, Liverpool, UK &
Computer Technology Institute, Patras, Greece
P.Spirakis@liverpool.ac.uk

A dynamic network is a network that changes with time. Nature, society, and the modern communications landscape abound with examples. Molecular interactions, chemical reactions, social relationships and interactions in human and animal populations, transportation networks, mobile wireless devices, and robot collectives, form only a small subset of the systems whose dynamics can be naturally modeled and analyzed by some sort of dynamic network. Though many of these systems have always existed, it was not until recently that the need for a formal treatment that would consider *time* as an integral part of the network has been identified. Computer science is leading this major shift, mainly driven by the advent of low-cost wireless communication devices and the development of efficient wireless communication protocols.

The early years of computing could be characterized as the era of *staticity* and of the relatively *predictable*; centralized algorithms for (combinatorial optimization) problems concerning static instances, as is that of finding a minimum cost traveling salesman tour in a complete weighted graph, computability questions in cellular automata, and protocols for distributed tasks in a static network. Even when changes were considered, as is the case in fault-tolerant distributed computing, the dynamics were usually sufficiently slow to be handled by conservative approaches, in principle too weak to be useful for highly dynamic systems. An exception is the area of online algorithms, where the input is not known in advance and is instead revealed to the algorithm during its course. Though the original motivation and context of online algorithms is not related to dynamic networks, the existing techniques and body of knowledge of the former may prove very useful in tackling the high unpredictability inherent in the latter.

In contrast, we are rapidly approaching, if not already there, the era of *dynamicity* and of the highly *unpredictable*. According to some latest reports, the number of mobile-only Internet users has already exceeded the number of desktop-only Internet users and more than 75% of all digital consumers are now using both desktop and mobile platforms to access the Internet. The *Internet of Things*, envisioning a vast number of objects and devices equipped with a variety of sensors and being connected to the Internet, and *smart cities* [37] are becoming a reality (an indicative example is the recent £40M investment of the UK government on these technologies). Computer scientists, nanoscientists, and engineers are joining their forces towards the development of *programmable matter*, that is, matter that can algorithmically change its physical properties, and have already produced the first impressive outcomes, such as programmed DNA molecules that self-assemble into desired structures [16] and large collectives of tiny identical robots that orchestrate resembling a single multi-robot organism [39]. Other ambitious long-term applications include molecular computers, collectives of nanorobots injected into the human circulatory system for monitoring and treating diseases, or even self-reproducing and self-healing machines. What all of these systems have in common, is their characteristic of typically being highly dynamic both in space and time.

The theoretical and analytic approach, prominent in computer science research from the very beginning, has been invaluable in modeling real-world systems and problems, abstracting their essential properties, and answering what can or cannot be done in ideal, extreme, or average conditions. Its findings have constantly enlightened and reshaped applied research and it has revealed some of the deepest and most outstanding models, notions, problems, and theorems of modern mathematics, such as the Turing Machine and Turing's proof on the Entscheidungsproblem, the **P** vs. **NP** question, the theory of **NP**-completeness, the four color theorem, the traveling salesman problem, primality testing, Lamport's causality [27], and the FLP impossibility of distributed consensus [21], to name just a few.

Theory will continue lying at the center of progress in our science and its necessity towards our understanding of dynamic networks is already evident. We have reached a point at which a large gap has been formed between existing systems and applications on one side and our fundamental understanding of their underlying principles on the other. Though theory has already identified some first core questions and has provided some preliminary answers to them, it has to run faster in order to bridge the gap and catch up to practice. What computations can be performed by a collection of automata, such as nanodevices or even molecules, that cannot control their own interactions? Even if the computing entities are powerful devices, like smartphones or tablets, can they still carry out basic distributed tasks, such as leader election or counting the size of the system, and with what algorithmic techniques and under what required guarantees about the network's dynamics? Are the traditional network measures adequate for dynamic networks? If not, how can we represent and measure basic quantities, like the speed of information propagation or the diameter, in a network that changes perpetually? Can we continue proving rigorous average-case or even worst-case guarantees and limitations as we have very successfully done for static systems? How can we design a dynamic network that satisfies some desired connectivity properties while minimizing some cost constraints (e.g., associated with the fact that creating and maintaining a connection does not come for free)? Which structural and algorithmic properties of static graphs carry over to temporal graphs (an invaluable abstraction of dynamic topologies) and which need a radically new per-

spective? All these are questions whose ultimate answers are to be provided by the theoreticians.

The existing literature can be roughly partitioned into three clearly distinguishable but also closely interrelated subareas: Population Protocols, Powerful Dynamic Distributed Systems, and Temporal Graphs. The population protocol model, proposed by Angluin *et al.* in 2004 [4], was originally motivated by highly dynamic networks of simple sensor nodes that cannot control their mobility. As the work of Angluin back in 1980 [3] that is generally accepted as a landmark study for distributed computing in static networks, population protocols could be considered as the starting point of distributed computing in dynamic networks. The other main sub-area originates from the 2005 work of O'Dell and Wattenhofer [38], and later the work of Kuhn, Lynch, and Oshman [25], who reconsidered classical distributed tasks, like leader election, counting, and information dissemination, in a network whose dynamics are captured by a worst-case temporal graph. In parallel, starting from the studies of Berman [9] and Kempe *et al.* [24], an increasing number of groups are interested in investigating the structural and algorithmic properties of temporal graphs, which are, roughly speaking, graphs that evolve over time, with the aim at developing a temporal extension of graph theory. In what follows, we will have the opportunity to look deeper into each one of these lines of research. We encourage the interested reader to complement his/her reading of the present article with some of the existing technical introductory texts [7, 32, 26, 11, 30].

Population Protocols: A Soup of Automata

Imagine a population of nanodevices, interacting randomly with each other in a well-mixed solution, like a boiling liquid. Each device has a small memory, whose size does not depend on the size of the population, and it has no control over its own mobility, which stems solely from the dynamicity of the environment. The only things that these devices can do, is to obtain an input, e.g., by performing a sensing measurement, to update their local state during an interaction with some other device (by applying to their local states a common simple program, called *protocol*, executed by all the devices), and to give an output. An interaction may simply occur when two devices come sufficiently close to each other to establish some sort of communication.

At this point, the reader may be wondering the same that Angluin *et al.* [4, 6] asked themselves: Can such a soup of automata really compute anything useful? Angluin *et al.* proved that actually they can, but not that much, compared to what one is used to expect from modern computing systems. First of all, non-trivial terminating computations are impossible in this model. Indeed, if a node (i.e., a device) terminates in some execution, then the same local execution may also result as part of another execution of the same protocol on a larger population, in which case the node terminates and decides without having heard from all the other nodes (imagine a node deciding that there is an even number of nodes with input 1, without knowing all the inputs). One important consequence of this fundamental inability, is that, in this model, we cannot sequentially compose protocols, which makes very challenging the development of protocols for composite tasks. Moreover, we can only hope for computations that *stabilize eventually*, in the sense that the nodes always manage to reach a point at which their outputs cannot change any more, even though the nodes cannot actually tell that this has happened. For example, to (stably) compute the parity of the 1s in the whole distributed input, whenever an odd number of nodes have input 1, all nodes must eventually stabilize their output to 1, and to 0, otherwise. This is the parity predicate, which is true if and only if the number of 1s in the input is odd.

But in order to hope for such global computations, we must also say something about the pattern of interactions between the nodes. Imagine, for example, that two parts of the system never influence each other or that some nodes always interact at inconvenient times. In the first case, the system consists of two isolated sub-systems and in the second the environment has the power to enforce some inconvenient symmetries that the protocol cannot break. So, we have to restrict ourselves to environments that are "connected" and "random" enough to not suffer from such inconveniences. There are two main ways to satisfy this: either by assuming that the interactions happen in a *fair* manner, essentially meaning that they do not forever avoid an always reachable configuration of the system, or that they happen uniformly at random from all possible interactions. The former way is very handy for answering computability questions, while the latter is usually preferred when one wants to analyze the running time of a protocol (i.e., the expected number of interactions until stability). See Figure 1 for an example of the model in action.

Angluin *et al.* managed to give an exact characterization of the computational capabilities of such systems. They proved that if the environment is fair, then the devices can stably compute precisely the *semilinear* predicates, and that this is also true for several interesting variations of the model. But what does this mean exactly? Let us give a simple illustration. Assume that when a device senses its environment, it either sees an a or a b, and denote by N_a and N_b the total number of as and bs sensed by all the devices, respectively. Then there is a protocol that, on any population and any combination of sensed inputs, can stably compute whether at least $1/3$ of the nodes have seen an a. In other words, the predicate which is true whenever $N_a \geq (N_a + N_b)/3 \Leftrightarrow 2N_a - N_b \geq 0$, is stably computable. The semilinear predicates are precisely those predicates that can be expressed in the form of a linear combination of input variables compared to a constant, i.e., $\sum_{i=1}^{k} \gamma_i N_i < c$ (where the inequality can be of any type, equality inclusive, and can also be replaced by equivalence modulo an integer constant μ). So, for example, the characterization tells us that we can stably compute whether the as are a strict majority (as in Figure 1), whether the bs have been sensed by at least 5% of the nodes, or whether the size of the population is odd. On the other hand, we cannot compute even the simplest expressions involving multiplications of input variables and expressions requiring any form of global iterative sub-computations, such as whether the number of cs is the product of the number of as and the number of bs or whether the number of bs is a power of 2. The positive part of the characterization is constructive, which means that there is a generic protocol that can be adjusted to compute any semilinear predicate.

Now that we know exactly what can be computed in this setting, we may ask: *how fast can it be computed*? Angluin *et al.* [4] proved that if the interactions happen uniformly at random, one at a time, then the aforementioned generic protocol stabilizes in an expected number of $O(n^2 \log n)$ interactions, where n is the size of the population. Can we do much better than this? Angluin, Aspnes, and Eisenstat [5] showed that, if there is a pre-elected unique leader in

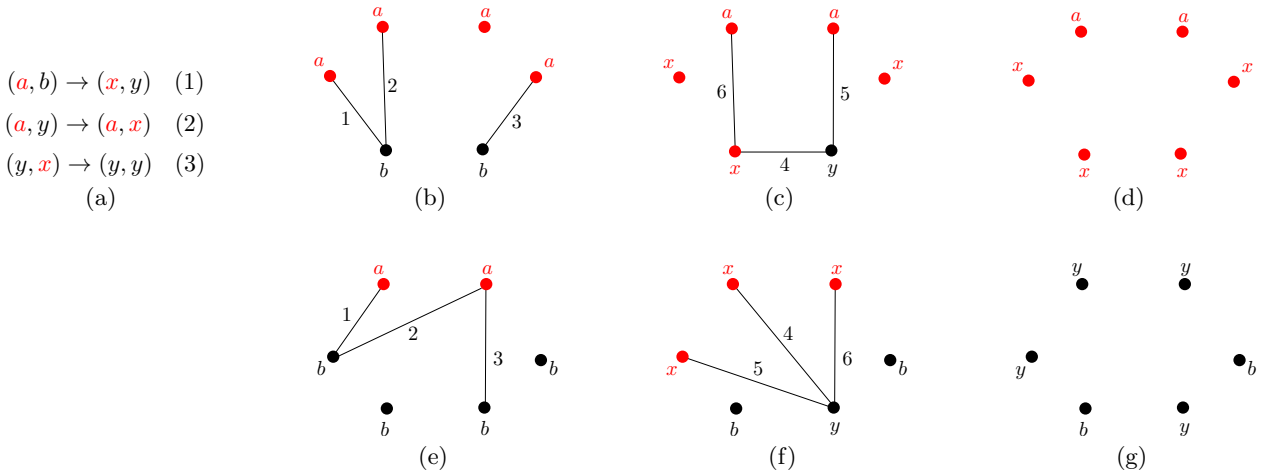

Figure 1: A population protocol computing whether the number of as in the input is a strict majority. Initially, each node is in an input-state a or b. Let N_a and N_b denote the initial number of as and bs, respectively. If $N_a > N_b$ we want *all* nodes to *stabilize* their output to 1 and to 0 otherwise. (a) The code of the protocol. The possible states are a, x (red states), b, and y (black states). The output of red states is 1 and the output of black states is 0. Rule (1) means that when an a interacts with a b, the former becomes x and the latter y (similarly for (2) and (3)). To see that this protocol is guaranteed to stabilize to a correct output, note that all rules preserve the difference $N_a - N_b$, and consider the last time (1) occurs (when the smaller of a or b disappears). If $N_a > N_b$, then some nodes remain in state a, so (2) and (3) compete to change y to x and back. However, (2) eventually wins with probability 1, which results in only red states present. This stabilizes the population to output 1, since all rules require a black state to execute. If $N_a \leq N_b$, then rules (1) and (2) are disabled once the final a is gone. The last occurrence of (1) ensures at least one y exists, so rule (3) then converts all xs to ys, resulting in only black nodes present, which stabilizes the population to output 0. (b-d) An example execution where $N_a > N_b$. The time-labeled edges indicate in which step the corresponding interaction occurs. (c) After 3 steps, all bs have been eliminated. (d) After another 3 steps there are only reds and the output 1 is stable. (e-g) An example execution where $N_a < N_b$.

the population, the time of computing any semilinear predicate can be reduced to $O(n \log^5 n)$. A natural next question was whether this speed-up can still be achieved by electing a leader instead of assuming it. Doty and Soloveichik [18] showed recently that it can't, by proving that an average of $\Omega(n^2)$ interactions have to be paid by any protocol that elects a leader [1].

Despite the indisputable fact that the semilinear predicates constitute a rather small class, this class is by no means trivially achieved. Actually, it can get much worse than semilinear, with apparently gentle additional restrictions. One such, studied by Chen *et al.* [13], is to restrict attention to protocols that never go through a "bottleneck" transition, meaning one that can only occur via an interaction between states that have low counts (constant) in the population. Such protocols have the nice property of always avoiding interactions that have a low probability to occur, and, thus, are slow. Unfortunately, it turns out that such protocols cannot count at all, and can only answer existence questions, asking whether a certain symbol is present or not in the input. Another, shows up when one tries to totally avoid the election of a leader, in an attempt to obtain inherently symmetric (i.e., parallel) protocols, that do not rely on some global symmetry-breaking process, and, thus, are more efficient and more resilient to faults (e.g., a crash failure of a processor). Formally defining what it really means to elect a leader in a distributed system is quite challenging, as it may be achieved implicitly and even sometimes in contrast to a protocol's intention. To this end, Michail and Spirakis [34] defined the *symmetry* of a protocol on a given population and input, as the minimum multiplicity of a state throughout an execution, in which the environment is as symmetric as possible for this protocol in the given setting. Then they proved that there are predicates, like parity, that cannot be computed if we require the symmetry of the protocol to be higher than a constant that depends on the size of the protocol. But enough of those weaknesses; let's see how minimal additional assumptions can allow the devices to cooperate in order to achieve collective complexity and enable much more powerful computations, in spite of the adversarial nature of the environment.

[1] Doty and Soloveichik call this a linear-time lower bound, as they perform their analysis in terms of parallel time, simply defined as sequential time divided by n. Both ways are almost equally used in the recent literature. In this article we have chosen to give all bounds in terms of sequential time.

Beyond Semilinearity

Semilinearity is the price that we pay for minimality: an amorphous system of computational entities that have only constant memory and that cannot infer a bound on the time it takes to hear from all the other entities. Relaxing any of these properties can dramatically increase the computational power. If, for example, the nodes are arranged in a line and the only interactions that can occur are between neighboring nodes in the line, then it is fairly straightforward to simulate a Turing machine of linear space. Similar improvements are possible if the pattern of interactions adheres to some probability distribution. Angluin *et al.* [4] showed that, if they happen uniformly at random, then the nodes can simulate a log-space Turing machine with high probability (w.h.p.).

The crucial role of memory in this type of systems has been extensively highlighted and has given some of the most impressive results in this area. One of the restrictions related to local memory that was early questioned, was *anonymity*, that is, the fact that nodes in the original model do not have and cannot ever obtain unique identifiers (ids), simply because there is not enough room in their memory to store them. However, in practice, it is reasonable to expect that even nanodevices will have access to ids, as several existing micro-controllers are set by the factory to store a unique serial number. Guerraoui and Ruppert [23] studied such a variant of the original model, and showed that it can simulate a pointer machine, yielding a computational power equal to that of a nondeterministic Turing machine of space $O(n \log n)$.

The effect of explicitly allowing to the devices a larger working memory, was first studied by Chatzigiannakis *et al.* [12]. Though for theoretical purposes it is quite reasonable to stick to memories that do not scale with the size of the system, this is quite an excessive requirement for real systems. Even for a population as large as 2^{273} nodes, which, by the way, is a number greater than the current estimates of the number of atoms in the observable universe, a logarithmic local memory is for most practical purposes as small as a few hundreds of cells, while most modern micro-controllers come with at least 16 KB of RAM. Chatzigiannakis *et al.* showed that $\Theta(\log \log n)$ local memory is a threshold, under which (asymptotically) semilinearity persists and at which the first non-semilinear predicates become feasible, like computing whether the multiplicity of an input symbol is a power of 2. They also proved that if the local memories have size $f(n) = \Omega(\log n)$, then the computational power is equivalent to that of a nondeterministic Turing machine of space $O(nf(n))$ and there is a space hierarchy, essentially meaning that protocols having access to more memory can compute more things.

If a moderate increase of local memory is additionally combined with a guarantee of a uniformly random interaction pattern, then even more fascinating tasks become feasible. Michail [29] showed that, in this case, a pre-elected unique leader with two n-counters can terminate and still count an upper bound on the size n of the system w.h.p.. The idea is to have the leader implement two competing processes, running in parallel. The first process counts the number of nodes that have been encountered once and the second process counts the number of nodes that have been encountered twice. The game ends when the second counter catches up the first. It can be proved that when this occurs, the leader will almost surely have already counted at least half of the nodes. Alistarh and Gelashvili [2] showed that $O(\log \log n)$ bits of memory per node are sufficient to elect a unique leader in an expected number of $O(n \log^3 n)$ interactions, a great improvement compared to the $\Omega(n^2)$ lower bound for the constant-memory case.

Natural Processes and Programmable Matter

Apart from being a model of computing in a highly dynamic environment, population protocols bear some striking similarities to several natural processes. They can be viewed as an abstraction of "fast-mixing" physical systems, such as chemical reaction networks, animal populations, and gene regulatory networks. The strong resemblance of population protocols to models of interacting molecules in theoretical chemistry had already been observed by Angluin *et al.* [4]. Assuming a fixed molecular population size and bi-molecular reactions, population protocols are formally equivalent to chemical reaction networks, a formal model of chemistry in a well-mixed solution, describing how certain species of molecules within a solution, such as DNA strands, react to produce new species [17]. An important consequence of this, is that bounds and characterizations for population protocols, apart from being useful for computer science applications, usually translate to inherent properties of natural systems. For example, the "molecular translation" of the aforementioned $\Omega(n^2)$ lower bound for leader election [18], is that it is essentially difficult to generate exact quantities of molecular species quickly (at least slower than destroying all molecules of the species, which takes $O(n \log n)$ time). There are even population protocols, like those for computing an approximate majority, that have been connected to biological networks [10]. Czyzowicz *et al.* [15] have recently studied the relation of population protocols to antagonism of species, with dynamics modeled by discrete Lotka-Volterra equations.

Another interesting possibility that was recently highlighted by Michail and Spirakis, is to use population protocols as a model of (algorithmic) distributed network construction and, consequently, as a potential model for programmable matter able to self-organize in a dynamic environment. Michail and Spirakis [35] studied an extension of population protocols, called network constructors, in which the devices can additionally establish bonds with each other (like a molecular bonding mechanism); [2] see Figure 2 for an example. One of their main results was that such systems can construct as complex stable networks as those that can be decided by a centralized algorithm. The idea is to program the nodes to organize themselves into a network that can serve as a memory of size $O(n^2)$, which is asymptotically maximum and can only be achieved by exploiting the presence or absence of bonds between nodes as the bits of the memory (if only the nodes' local space was used, then the total memory could not exceed $O(n)$). Then the population draws a random network and simulates on the distributed memory a Turing machine that decides whether the network belongs to the target ones. If yes, the population stabilizes to it, otherwise the random experiment and the simulation are repeated. What makes the construction intricate is that all the sub-routines have to be executed in parallel and potential errors due to this to be corrected by global resets throughout the course of the protocol. Michail [29] then studied a more applied version of this model, by adding geometric constraints (representing physical restrictions), according to which the formed network and the allowable interactions must respect the structure of the 2-dimensional (or 3-dimensional) grid network.

Powerful Dynamic Distributed Systems

As we have seen, population protocols and their variants, concern some rather specialized computing systems, operating in fairly extreme conditions. Typical dynamic distributed systems, usually consist of much more "gifted" devices, like smartphones or tablets, equipped, among other things, with ids, practically unbounded local memories, powerful processors, and wireless (radio) transceivers.

One of the first formal models for such systems, developed by O'Dell and Wattenhofer, appeared in 2005 [38],

[2] A predecessor of this model had served as one of the first computationally powerful variants of population protocols, exploiting bond states to simulate an $O(n^2)$-space nondeterministic Turing machine [31].

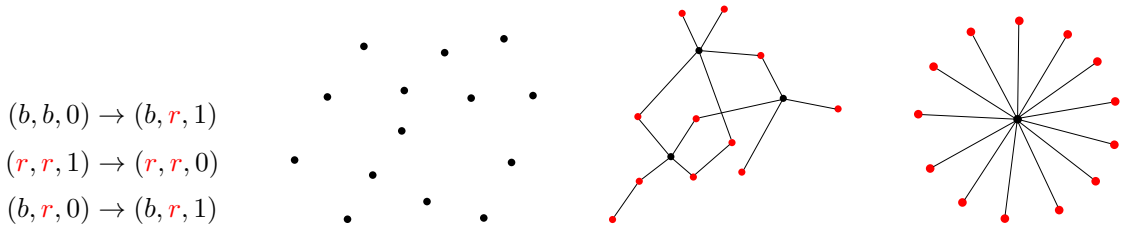

Figure 2: (a) A simple optimal protocol that allows the nodes to self-organize into a global star. Blacks eliminate each other, reds repel, and blacks attract reds. (b) Initially all nodes are black and no active connections exist. (c) After a while, only 3 blacks have survived each having a set of red neighbors. (d) A unique black has survived, it has attracted all reds, and all connections between reds have been deactivated. The construction is a stable global star.

just one year after the original paper of Angluin *et al.* on population protocols. Their model is, essentially, a generalization of classical networked message-passing distributed systems, where, instead of a static graph, the underlying network is now represented by an unknown and adversarially controlled (i.e., worst-case) *temporal graph* [3]. The nodes are Turing machines with unbounded tapes (it is the protocol designer's responsibility to minimize both the actual space used and the local processing time) that communicate with other nodes by interchanging messages over a wireless medium. As is always the case, the worst-case approach has the benefit that the results hold for all possible dynamic network topologies (of course, between those that make sense) and not just for some convenient special cases or distributions. In order to allow for bounded end-to-end communication, O'Dell and Wattenhofer imposed on the underlying dynamic network the restriction of being connected at any instant [4]. Such a simplification may sound artificial, as most real dynamic systems are expected to almost never be connected, still it is very convenient for the purpose of theoretical analysis and for establishing some first fundamental principles. More recent studies that we shall discuss later on, have developed ways of relaxing this restriction.

O'Dell and Wattenhofer defined their model in terms of *asynchronous* communication and studied the *token dissemination* and *routing* problems in this setting. In token dissemination, a token, i.e., a piece of information, is initially present on some source node and the goal is to distribute the token to the entire network and have all nodes terminate when dissemination has successfully completed (e.g., when a base station wants to disseminate to all nodes in a sensor network a global reset signal). In routing, the token has only to be delivered to a designated destination node.

Five years later, Kuhn, Lynch, and Oshman [25] proposed a *synchronous* version of the above model, which substantially simplified thinking and treating dynamic networks formally, and, thus, lead to numerous new insights and directions. The nodes operate now in lock-step, synchronized in discrete rounds either by having access to a global clock or by keeping local clocks synchronized. In every round, an adversary scheduler (modeling the worst-case nature of the network's dynamicity) selects a set of edges between the nodes and every node may communicate with its current neighbors, as selected by the adversary, usually by *broadcasting* a single message to be delivered to all its neighbors. As in the previous model, the network is revealed to the distributed algorithms in an online and totally unpredictable way and the nodes have no *a priori* knowledge about the network apart from the guarantee that its instances are connected. Despite the simplicity of the model, even the most basic distributed tasks no longer seem straightforward. For example, how can the nodes count the size n of the system and terminate (i.e., be able to detect that their task has successfully come to an end)?

To appreciate the difficulties, it is useful to see why a typical approach for static networks fails in dynamic networks. In static networks, the stability of paths is an invaluable implicit guarantee for the rate of global progress. In particular, if a node u broadcasts a message and every node that receives the message forwards it to all its neighbors, then u *knows* that, in every round, at least one more node receives the message for the first time. Moreover, if every node acknowledges the receipt by broadcasting an ack message containing its id, then if all nodes forward these acks, u *knows* that, in every second round, it must either hear from a new remote node or all nodes must have already received u's message. The first guarantee is still satisfied in the dynamic case, because if all nodes that have a piece of information broadcast it in every round, then connectivity of the instantaneous topology ensures that at least one of them will deliver it to a node that has not heard of it yet. However, the same is not true for the second guarantee. Imagine a star topology, with u lying on the center and being directly connected to all other nodes (the peripherals), apart from one node v that is connected to a peripheral node w but not directly to u. In round 1, u can learn about the existence of all nodes but v. Then, in round 2, the topology changes to a line spanning the nodes, with u lying on the left endpoint, v on the right, and w being the unique neighbor of v in the line, and remains static forever. As the only nodes that know about v are the two rightmost nodes of the line, it will take $n-2$ more rounds for u to realize that another node exists. Given that u does not know any estimate of n in advance, at first sight it seems that u has no means of determining how long it should wait. The good news is that it is still possible to *infer* such a bound.

Before showing how, let us first extract from the above discussion two very useful notions for capturing the spread of influence in a dynamic distributed system. Both are based on Lamport's *causal influence* [27], which formalizes the no-

[3] One way to define a temporal graph D is as a pair (V, A), where V is a static set of nodes and $A\colon \mathbb{N} \to \binom{V}{2}$ a mapping, such that $A(t)$ is the (possibly empty) set of all edges that appear at time t (*time-edges*). Then a *temporal path* or *journey* of D is a path of time-edges using increasing times.

[4] If the temporal graph of the dynamic network is $D = (V, A)$, then this means that, for all times $t \in \mathbb{N}$, the static graph $G = (V, A(t))$ is connected.

tion of one node "influencing" another through a chain of messages (possibly going through other nodes in between). The first one is the *future set* of a node u in a given time interval, containing all nodes that u has influenced in that interval. The second one is the *past set* of a node u in a given time interval, containing all nodes that have influenced u in the interval. Stated in the new terminology, the above discussion says that the cardinality of u's future set increases by at least one in every round, until it becomes equal to n. Though, as already highlighted above, the same is not true for the past set, still there is an alternative and equally useful guarantee on its rate of growth. The size of the past set of u is an upper bound on the number of rounds required for u to hear of a new node, i.e., for its past set to increase by at least one. This is because the set of nodes that know a new influence for u are initially those nodes not in the past set of u and, due to connectivity, in every round the former set increases by at least one, so in a number of rounds at most equal to the size of u's past set the whole past set will know a new influence, u inclusive.

By a simple induction on the number of rounds, we obtain that the size of u's past set must be either greater than the number of the current round or equal to n. This immediately gives to u a way for knowing when it has heard of all nodes: keep track of your past set in a list A (e.g., recording nodes' ids) and of the current round r; if it ever holds that $r \geq |A|$, then A contains all nodes in the system. This idea gives an $O(n)$-round distributed algorithm for counting the size of the system, and, by changing the output and the contents of the transmitted messages, is also an algorithm for many other basic distributed tasks, such as information dissemination, leader election, and computing arbitrary functions on inputs to the nodes. For these algorithms to work, all nodes must broadcast in every round all information that they know, which is not a desired property as it results in transmitting very large messages, i.e., of size $O(n \log n)$.

Kuhn, Lynch, and Oshman also developed an alternative approach that uses only $O(\log n)$ bits per message, a much more reasonable message overhead for real systems, paying a linear factor increase in termination time. The idea is as follows. The nodes have a guess k of the size of the system and then try to verify whether their guess was a correct upper bound on n. If it was, then it is possible for the nodes to terminate knowing the exact value of n, otherwise they double k and repeat. Assume, for simplicity, that the verification process is coordinated by a unique leader. What the leader does is to invite $k-1$ other nodes to join its committee. Each node that is not invited creates its own committee. As long as the guess is not correct, there must be at least two committees, and when it becomes correct for the first time, there will be a single committee (i.e., the leader's) containing all nodes. Then it is fairly simple for the nodes to verify whether there is precisely one committee. Having a leader coordinate the process is crucial for reducing the message overhead, as, in this way, the other nodes need only broadcast the information emanating from the leader. Fortunately, the leader need not be assumed, but can be elected in parallel with the above process, without increasing the size of the messages.

As alluded to above, continuous connectivity was one of the first assumptions of these models to be questioned. Michail, Chatzigiannakis, and Spirakis [33] replaced it by more general conditions of *temporal connectivity*, i.e., connectivity satisfied over time. To do this, they introduced metrics to capture the speed of influence propagation in networks that are possibly disconnected at all times. These metrics concern properties that do not necessarily hold in every round, but instead may require several rounds until they are satisfied. One such is the *connectivity time* of a dynamic network, which is the maximal time that the two parts of any cut of the network can remain disconnected. Another is the *outgoing influence time*, which is the maximal time until the state of a node at a given time (e.g., its initial state) influences the state of another node. They gave efficient distributed algorithms for counting and information dissemination, by exploiting a known upper bound on each of these metrics.

The *temporal diameter*, a measure of the time required for influence dissemination, generalizes the standard network diameter, as the latter is unsuitable for dynamic networks. It is defined as the minimum integer d for which it holds that the *temporal distance* (i.e., the duration of a journey of minimum arrival time) between every ordered pair of nodes at any given time is at most d. For an indicative example, consider a dynamic star in which all peripherals $(u_1, u_2, \ldots, u_{n-2})$ but two (u_{n-1}, u_n) go to the center one after the other in a modular way; that is, at any time $t \geq 0$, $u_{[t \bmod (n-2)]+1}$ is the center of the star and all the other nodes are peripherals. Then any message from u_{n-1} to u_n needs $n-1$ steps to be delivered, because u_n can only get the message if a node that has already obtained it becomes the center again. So, the temporal diameter of this network is $n-1$ even though its instantaneous diameter is at any given time just 2. This is a simplified version of a construction used by Avin, Koucký, and Lotker [8] to show that, in contrast to the cover time of a random walk on a static graph, which is always polynomial in n, the cover time of a random walk on a temporal graph may be exponential. The diameter is just one of those many network notions that have to be redefined to take time into account, in order to become suitable for dynamic networks.

In practice, the network dynamics may not always be totally unpredictable or irregular. The dynamicity patterns of many real-world systems, such as human interactions and transportation units, exhibit regularities and are to some extent predictable. In view of this, some authors considered network dynamics that are a result of randomness, while others deterministic network dynamics that are *recurrent* or *periodic*. Clementi *et al.* [14] studied the speed of information dissemination in the following type of edge-markovian dynamic networks: if an edge exists at time t then, at time $t+1$, it disappears with probability q, and if instead the edge does not exist at time t, then it appears at time $t+1$ with probability p. Flocchini, Mans, and Santoro [22] studied one type of periodic dynamic networks, called carrier networks, in which the dynamic network is defined by the periodic movements of some mobile entities, called carriers. This is a natural abstraction of several real-world systems like public transports with fixed timetables, low earth orbiting satellite systems, and security guards' tours. They studied the problem of exploring all nodes of the network by an agent who can only follow the route of a carrier (like a passenger) and can switch from one carrier to another.

Structural Properties of Temporal Graphs

Modern dynamic systems and applications, as well as the theoretical progress in dynamic distributed systems described so far, led several researchers to the realization that the underlying topology model of dynamic networks is not a mere generalization of graphs; rather, it manifests some essentially different structural and algorithmic properties. A

temporal extension of graph theory is already under development, with the aim at delivering a concrete set of results, tools, and techniques for temporal graphs. Graphs have proved to be an invaluable tool for representing and enabling the formal treatment of relatively stable networked systems. There are already strong indications that temporal graphs will play an equally important role for dynamic networks.

A temporal graph can be thought of as a special case of labeled graphs, where labels capture some measure of time, e.g., the precise times or time intervals at which each connection is available. But is there anything new here? Can't we just resort to traditional graph approaches to deal with this seemingly minor extension? A first indication that the answer might not be that obvious, is the richness that emerged from considering labels as colors and trying to solve conflict-free coloring problems (strongly motivated by real-world problems, like frequency assignment in cellular networks), in the classical and well-studied area of graph coloring. Indeed, the main message from existing research on temporal graphs is that many graph properties and problems become radically different and usually substantially more difficult when an extra time dimension is added to them.

This was first highlighted by Kempe, Kleinberg, and Kumar [24] in a minimal special case of temporal graphs, in which every edge is available only once. They proved that, in such temporal graphs, the classical formulation of Menger's theorem [5] is violated if applied to journeys and the computation of the number of node-disjoint s-z paths becomes **NP**-complete. A reformulation of Menger's theorem which is valid for all temporal graphs was recently achieved by Mertzios *et al.* [28].

The algorithmic problems of temporal graphs can be divided into two main types, depending on the algorithm's knowledge about the future evolution of the graph. *Online* algorithms have no knowledge about the future, while *offline* algorithms know the full evolution of the graph in advance.

An example of an online centralized problem on temporal graphs is *k-token dissemination*, which asks to disseminate to all nodes as fast as possible, k tokens that are initially assigned to some of the nodes. The only restriction on the algorithm's knowledge is that it has to make its selection of tokens to be forwarded without knowing the edges selected by the adversary in the current round. Kuhn, Lynch, and Oshman [25] showed by a potential function argument that any such deterministic centralized algorithm for the problem in continuously connected temporal graphs requires at least $\Omega(n \log k)$ rounds to complete in the worst case (their corresponding distributed upper bound, by the algorithm described in the previous section, is $O(nk)$). This lower bound was further improved to $\Omega(nk/\log n)$ by Dutta *et al.* [19], via the probabilistic method.

Even though offline centralized algorithms do not suffer from the unpredictability that characterizes dynamic network problems, still offline temporal versions of standard graph problems can be substantially more difficult to solve. One such example, studied by Michail and Spirakis [36], is the problem of exploring the nodes of a temporal graph as soon as possible. Though, in the static case, the decision version of the problem can be solved in linear time and the optimization version (known as Graphic TSP) can be satisfactorily approximated, in the temporal case the decision version becomes **NP**-complete and there exists some constant $c > 0$ such that the optimum solution cannot be approximated within cn (meaning that we cannot find a solution which is at most cn times worse than the optimum), unless $\mathbf{P} = \mathbf{NP}$ (things do not become any better, even if all instances are connected [36, 20]).

Mertzios *et al.* [28] also studied the problem of *designing* a cost-efficient temporal graph, given some requirements that the graph should meet. Briefly, we are provided with an underlying graph G and we are asked to assign labels to its edges so that the resulting temporal graph minimizes some parameter (related to the cost of making an edge available) while satisfying some connectivity constraint. Other authors have considered random temporal graphs, a succinctly representable model, in which the labels are chosen according to some probability distribution [1].

The Future

Do we really know how to compute in highly dynamic environments, how to represent and measure their core properties, or even how to efficiently solve centralized computational problems concerning them? Despite the considerable recent progress that we discussed in this article, the answer is: *not yet*. We are on the road to a unified theory of dynamic networks, but not yet there.

Though it is still quite early to anticipate the full range of potential applications, there is already strong evidence that there is room for the development of a rich theory. As is always the case, the groundwork will be laid by our ability to identify and formulate radically new problems and not just by studying adjusted versions of existing ones. Real dynamic systems is the natural place to look for such problems. Still, the existing literature has already identified some first challenging research directions and technical problems whose further investigation has the potential to push forward the area.

First, is there a general rule underlying the complexity increase of a network problem when that problem is extended in time? Moreover, most natural applications require an algorithm to operate on a dynamic network without knowing or being able to accurately predict the future evolution of the network. It might be the case that the right treatment of such settings is via online algorithms and analysis, however little effort has been devoted to this. We saw that there is a natural reformulation of Menger's theorem for temporal graphs. It would be very valuable to check the validity of many other fundamental results of graph theory, like Kuratowski's planarity theorem or Mantel's beautiful theorem on the existence of triangles. Another critical issue has to do with a long-standing problem in distributed computing theory: there is practically a different model for each setting and usually slight modifications of a model result in totally different formal properties. This multiplicity is expected to be even more intense in dynamic networks, due to the almost inexhaustible variety of different dynamicity patterns. Therefore, if possible, a unification of models for distributed computing in dynamic networks would be more than valuable. There is also a great need for progress in programming and verification of programmable matter protocols, probably the only sub-area of dynamic networks in which theory has grown faster than systems. We need many more real collectives of tiny devices, in order to identify which assumptions actually make sense and are worth studying and which would never show up in a real system and have to be abandoned. Finally, we should seriously take into account the recent advances in learning and statistical learning in par-

[5] *Menger's theorem* states that the maximum number of node-disjoint s-z paths is equal to the minimum number of nodes whose removal separates node s from node z.

ticular, as a rich source of powerful methods for a system to learn and, thus, be able to predict to some extent the pattern of the dynamic network and to adapt its algorithms in order to cope with the new conditions more effectively.

References

[1] E. C. Akrida, L. Gąsieniec, G. B. Mertzios, and P. G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *J. Parallel Distrib. Comput.*, 87:109–120, 2016.

[2] D. Alistarh and R. Gelashvili. Polylogarithmic-time leader election in population protocols. In *Proceedings of ICALP*, 479–491, 2015.

[3] D. Angluin. Local and global properties in networks of processors. In *Proceedings of STOC*, 82–93, 1980.

[4] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *Proceedings of PODC*, 290–299, 2004. Also in *Distributed Computing*, 2006.

[5] D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008.

[6] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

[7] J. Aspnes and E. Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, 97–120. 2009.

[8] C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Proceedings of ICALP*, 121–132, 2008.

[9] K. A. Berman. Vulnerability of scheduled networks and a generalization of Menger's theorem. *Networks*, 28(3):125–134, 1996.

[10] L. Cardelli. Morphisms of reaction networks that couple structure to function. *BMC systems biology*, 8(1):84, 2014.

[11] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *IJPEDS*, 27(5):387–408, 2012.

[12] I. Chatzigiannakis, O. Michail, S. Nikolaou, A. Pavlogiannis, and P. G. Spirakis. Passively mobile communicating machines that use restricted space. *Theor. Comput. Sci.*, 412(46):6469–6483, 2011.

[13] H.-L. Chen, R. Cummings, D. Doty, and D. Soloveichik. Speed faults in computation by chemical reaction networks. In *Proceedings of DISC*, 16–30, 2014. Also in *Distributed Computing*, 2015.

[14] A. E. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proceedings of PODC*, 213–222, 2008.

[15] J. Czyzowicz, L. Gąsieniec, A. Kosowski, E. Kranakis, P. G. Spirakis, and P. Uznański. On convergence and threshold properties of discrete lotka-volterra population protocols. In *Proceedings of ICALP*, 393–405. 2015.

[16] D. Doty. Theory of algorithmic self-assembly. *Commun. ACM*, 55:78–88, 2012.

[17] D. Doty. Timing in chemical reaction networks. In *Proceedings of SODA*, 772–784, 2014.

[18] D. Doty and D. Soloveichik. Stable leader election in population protocols requires linear time. In *Proceedings of DISC*, 602–616, 2015.

[19] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of SODA*, 717–736, 2013.

[20] T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *Proceedings of ICALP*, 444–455, 2015.

[21] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[22] P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. *Theoretical Computer Science*, 469:53–68, 2013.

[23] R. Guerraoui and E. Ruppert. Names trump malice: Tiny mobile agents can tolerate byzantine failures. In *Proceedings of ICALP*, 484–495, 2009.

[24] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of STOC*, 504–513, 2000.

[25] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of STOC*, 513–522, 2010.

[26] F. Kuhn and R. Oshman. Dynamic networks: models and algorithms. *SIGACT News*, 42:82–96, 2011.

[27] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[28] G. B. Mertzios, O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Temporal network optimization subject to connectivity constraints. In *Proceedings of ICALP*, 657–668, 2013.

[29] O. Michail. Terminating distributed construction of shapes and patterns in a fair solution of automata. In *Proceedings of PODC*, 37–46, 2015.

[30] O. Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

[31] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Mediated population protocols. *Theoretical Computer Science*, 412(22):2434–2450, 2011.

[32] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. *New Models for Population Protocols*. N. A. Lynch (Ed). Morgan & Claypool, 2011.

[33] O. Michail, I. Chatzigiannakis, and P. G. Spirakis. Causality, influence, and computation in possibly disconnected synchronous dynamic networks. *J. Parallel Distrib. Comput.*, 74(1):2016–2026, 2014.

[34] O. Michail and P. G. Spirakis. How many cooks spoil the soup? In *Proceedings of SIROCCO*, 3–18, 2016.

[35] O. Michail and P. G. Spirakis. Simple and efficient local codes for distributed stable network construction. *Distributed Computing*, 29(3):207–237, 2016.

[36] O. Michail and P. G. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016.

[37] G. Mone. The new smart cities. *Commun. ACM*, 58(7):20–21, June 2015.

[38] R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of DIALM-POMC*, 104–110, 2005.

[39] M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.