To Appear, First International Conference on Nano-Networks

Self-Assembled Networks: Control vs. Complexity

Jaidev P. Patwardhan[†], Chris Dwyer[‡], and Alvin R. Lebeck[†]
{jaidev,alvy}@cs.duke.edu, dwyer@ee.duke.edu

†Department of Computer Science
Duke University
Durham, NC 27708

‡Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708

Abstract

DNA-based self-assembly of nanoelectronic devices is an emerging technology that has the potential to enable tera- to peta-scale device integration. However, self-assembly currently is limited to manufacturing small computing blocks (nodes) which must then be interconnected to build a larger computing system. In this paper, we study node networks created by varying control over three aspects of the self-assembly process (node placement, node orientation, and inter-node link creation). In particular, we examine the tradeoff between node complexity and control required during self-assembly to maximize the number of connected nodes in the network. As the level of control decreases, we find that node communication hardware needs to be augmented to allow link sharing between several transceivers. This also results in better network connectivity in the presence of defective nodes and links. Finally, we show that for a data parallel architecture with enough available nodes, the specific network topology has a negligible effect on performance.

1 Introduction

Self-assembly of nanoelectronic devices has the potential to emerge as a lower cost alternative to top-down manufacturing. DNA-based self-assembly [14] is a bottom-up manufacturing process that uses the precise binding rules of DNA with nanoscale devices to build computing systems. Previously, we proposed a circuit architecture [9] to place electronic circuits on a DNA lattice. A key requirement of the process is the ability to control the placement of electronic devices (e.g., self-assembled nanowire transistors [16]) at specific points on a DNA scaffold [2,8,13,15]. This enables building DNA scaffolded electronic circuits (nodes) with limited storage, compute and communication capabilities which could then be used to build computing systems [10,12]. Self-assembly is currently limited to producing small sized DNA lattices thus limiting circuit size. However, the parallel nature of self-assembly enables the construction of a large number (~10^9-10^{12}) of nodes that may be linked together by self-assembled conducting nanowires [19]. These self-assembled computing systems assume either a mesh or a random network of nodes. However, these two topologies lie at opposite ends of a spectrum defined by the amount of control exercised over the self-assembly process.

In this paper, we study network properties as we exercise varying degrees of control over how nodes are placed and oriented, and how inter-node links are created during self-assembly. We examine a range of networks, from a mesh (full control) to a random network of nodes (no control). For each network type, we determine the connectivity of the network, and the need for any additional hardware in each node's communication logic to maximize the number of connected nodes. We evaluate the performance of a data parallel architecture [10] built on top of these networks using a simple benchmark and find that system performance is independent of the underlying network, as long as sufficient nodes are available for computation. Finally, we show that the introduction of defects in nodes and links can exacerbate the poor connectivity found in networks with low control during self-assembly.

Recently, researchers have been actively developing nanoelectronic devices and architectures that could potentially replace CMOS in the future. Most designs either assume the ability to create regular structures [1,4], or unstructured interconnect [18] (within a computing block). Since the networks we study are highly dependent on physical node locations, we cannot leverage the large body of work on generating [17] and analyzing internet topologies [7]. Future developments that allow embedding radio transceivers in nodes could potentially allow researchers to leverage this work. To our knowledge, ours is the first attempt at characterizing networks of nanoscale devices.

The rest of this paper is organized as follows. First, we provide a brief overview of our node and system architecture (Section 2). Next, we describe the different networks that are created as we exercise various levels of control over self-assembly (Section 3). Finally, we evaluate the connectivity of these networks with and without defects, and determine their impact on system performance (Section 4).

2 Node and System Architecture

Each node is assumed to have basic functionality that enables the organization of the network of nodes into a high-performance computing system. A detailed description of node and system architecture can be found in [10] and we summarize the relevant details here.

Node Architecture. Each node has three primary components: a) communication logic, b) configuration logic, and c) compute logic. The communication logic has four transceivers that allow the node to communicate with other nodes over single wire links. In case more than two transceivers

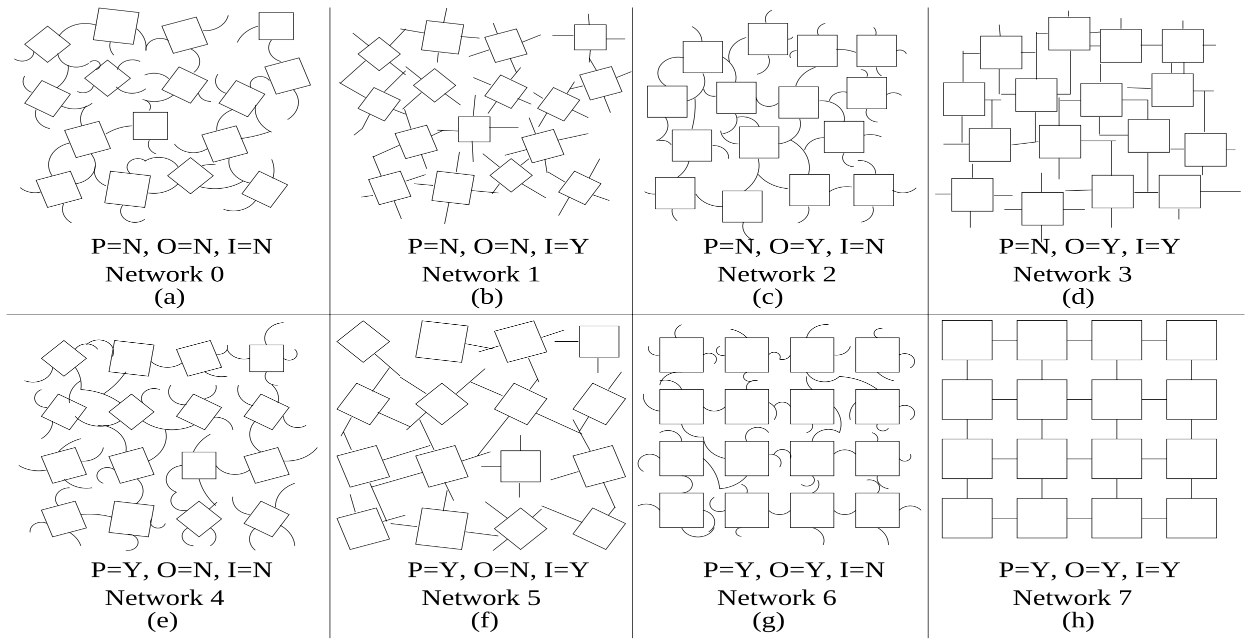

FIGURE 1. Examples of eight networks with varying control over placement (P), orientation (O), and inter-node link creation (I). (a-d): no control over P - nodes can get isolated due to large distances, control over O and I improve connectivity, (e-h): control over P improves connectivity, but can still result in isolated nodes without control over O and I.

share a single wire, our baseline transceiver model implements an infinite backoff mechanism that permits only two active transceivers on that link. When the node is powered up each component undergoes a simple test to ensure correct operation [11]. If the transceivers pass this test they attempt to signal neighboring nodes over their wire link. If a transceiver detects more than one other transceiver signal over the link, it shuts down. This can potentially affect network connectivity in cases where the transceiver that shuts down provides access to a region of the network.

System Architecture. To maximize node utilization we implement a data parallel architecture on the network of nodes. Since one node does not have the necessary compute power, we logically group together sets of nodes to form SIMD style processing elements (PEs) connected in a logical ring [10]. This system architecture supports a data parallel programming model. The topology of the underlying network affects the logical organization of PEs in the logical ring, which in turn can have an effect on system performance. In Section 4.3 we evaluate the performance of one data parallel program on various networks.

3 Networks of Self-Assembled Processing Elements

The topology of the network of nodes depends on the level of control exercised during node self-assembly, and during the creation of inter-node links. As self-assembly technology matures, it might be possible to create three-dimensional topologies as well, but we limit ourselves to the analysis of two-dimensional topologies in this work. We explore topologies created as we vary control over three aspects of the manufacturing process:

- placement of nodes (P)
- orientation of nodes (O)
- creation of inter-node links (I)

In each case, to limit the parameter space to be explored, we consider two alternatives: 1) full control, and 2) no control. This results in eight network types, ranging from a random planar network to a mesh. Table 1 lists the networks by the type of control necessary to create them and Figure 1 shows examples of these networks. The goal is to identify the level of control necessary to maximize the number of connected nodes. Next, we describe how we could potentially control P, O or I, and the implications of that control on the number of connected nodes in the network.

TABLE 1. Classification of network topologies based on control over P, O and I.

Name	Control			Example
	Placement (P)	Orientation (O)	Link (I)	
N0	No	No	No	Figure 1a
N1	No	No	Yes	Figure 1b
N2	No	Yes	No	Figure 1c
N3	No	Yes	Yes	Figure 1d
N4	Yes	No	No	Figure 1e
N5	Yes	No	Yes	Figure 1f
N6	Yes	Yes	No	Figure 1g
N7	Yes	Yes	Yes	Figure 1h

Placement (P). Control over node placement enables uniformly spaced nodes. We expect uniform spacing to improve network connectivity by reducing the number of nodes that require long links to connect to the rest of the network. Control over node placement can be achieved in two ways: 1) pick and place techniques, and 2) placing DNA tags on the underlying substrate to control the locations where nodes self-assemble. While each node is large enough to enable the use of pick and place strategies, they are not practical for systems with a large number of nodes. We can minimize external intervention by placing DNA tags on the substrate to initiate node growth at tag locations. The greater the number of tags per node, the greater the chance that nodes form at the right locations. However, increasing the number of tags increases the effort required in preparing the substrate for self-assembly. Examples of the network types created with node placement can be found in Figure 1e-h.

Orientation (O). Control over orientation aligns node faces, which can increase the chances of links intersecting (as depicted in Figure 1c, Figure 1d, Figure 1g, and Figure 1h), potentially improving network connectivity. The techniques to control node placement could also be extended to control node orientation by increasing the number of tags per node. In addition to using multiple tags on the substrate, nodes could be aligned using an external electric field, or using fluid flow [5].

Inter-node Link Creation (I). Control over inter-node link creation implies control over the shape of links. Without creating a mesh network (and linear links), there is still a chance that more than two transceivers are connected by a link. Linear links cannot loop back on themselves and are useful in improving network connectivity. Researchers have demonstrated the creation of mostly linear wire structures [3,6]. Networks with linear links are shown in Figure 1b, Figure 1d, Figure 1f, and Figure 1h.

Maximizing Reachable Nodes. In any non-mesh topology, more than two links (and transceivers) can potentially be connected. The baseline node design deals with such links by implementing an inifinite backoff mechanism (see Section 2). However, disabling a transceiver can partition the network. This problem can be mitigated by treating each link like a shared medium (bus). However, this requires extending transceiver functionality to enable arbitration for link access, as well as the use of source/destination identifiers for each transfer on the link. We evaluate the potential benefits of one method for link sharing in Section 4. As self-assembly matures, it might be possible to create larger nodes that incorporate this extra functionality.

4 Experimental Setup and Evaluation

We begin with a description of our custom network topology generator and the methodology used to model infinite backoff and link sharing between transceivers (Section 4.1). The goal of this evaluation is to study the connectivity characteristics of each network type and determine if the baseline node design needs to be augmented to maintain good system connectivity (Section 4.2). We also want to determine the impact of the different networks on system performance (Section 4.3). Finally, we want to study the effect of node and link defects on connectivity (Section 4.4).

4.1 Topology Generator

The topology generator's inputs include the number of nodes, total area, type of control over placement (P), orientation (O), interconnect (I), minimum distance between nodes, and an optional parameter that decelerates interconnect growth with time. It also accepts a random seed which allows the creation of distinct topologies. For networks with no control over node placement (P=N in Figure 1), it generates a random location for the node and places it there if all constraints are met (no overlap, minimum distance, within area). The program attempts to place each node a maximum of 10^6 times. For networks with control over node placement (P=Y in Figure 1), a simple check of the area and number of nodes allows the program to determine if the nodes fit. If O=N, each node is rotated (about its center) through a random angle before being placed.

After placing all nodes, the simulator models link growth. For random growth (I=N in Figure 1) we use a random number generator and a probability distribution function (PDF) for the angle and distance by which the link grows to perform a directed random walk. If we model linear growth (I=Y in Figure 1), we grow the link by a random length (<=50nm). Each link is iteratively grown by this random length until one of two conditions is satisfied: 1) it collides with another node or link, or 2) the simulation terminates as a user-defined condition is satisfied. Once growth of all links terminates, the simulator generates a graph corresponding to the node network created by the links and generates connectivity statistics for the graph. Next, we describe how we modify the generated graph to model infinite backoff or shared links.

Modeling Infinite Backoff. To model infinite backoff we identify links with more than two transceivers, randomly pick two transceivers to be active, and disconnect the rest. There are multiple ways of picking a pair of transceivers and we generate multiple networks by randomly picking different pairs of transceivers.

Modeling Links as Buses. We model one possible implementation of shared links, where the N transceivers connected by a single link are divided into pairs that communicate with each other. If N is odd, one transceiver is not used. There are multiple ways in which the transceivers can be paired and we try to capture the effect of such pairing by creating multiple networks with different transceiver

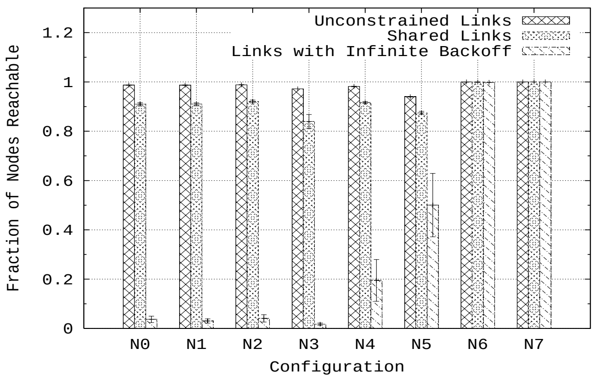

FIGURE 2. Fraction of Reachable Nodes

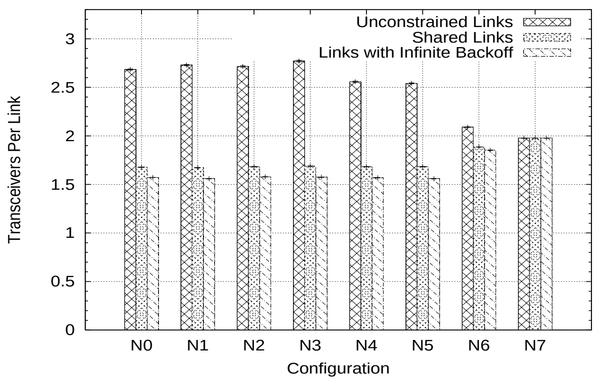

FIGURE 3. Transceivers Per Link

pairs. Next, we evaluate each network type to determine the fraction of reachable nodes for a range of network sizes.

4.2 Reachable Nodes

For each of eight network types, we generate 100 topologies for different network sizes. We use network sizes of 10,000 nodes and the smallest square mesh required to run 8x8 (1,296 nodes), 16x16 (4,900 nodes) and 32x32 (21,025 nodes) matrix multiplication. In Figure 2, we plot the size of the largest connected group of nodes as a fraction of total nodes for networks with 21,025 nodes (we get similar results for other network sizes). For each network type, we plot three bars, the first representing the unconstrained network, the second corresponding to links modelled as buses, and the third with transceivers implementing infinite backoff. From Figure 2, we see that if connectivity is unconstrained, all networks are able to connect in excess of 95% of the nodes. However, when modeling realistic hardware the fraction of reachable nodes decreases. The decrease is small when links are treated as shared media. However, if we model infinite backoff on links, for networks without control over placement and orientation, less than 50% nodes are reachable.

We plot the average number of transceivers connected per link in Figure 3. For a fully connected network of nodes there would be two transceivers per link and 1.97 for a mesh since the boundary transceivers are disconnected. For the unconstrained networks the value is over 2 indicating that multiple transceivers share links. The value drops to about 1.7 for shared links and about 1.55 for links with infinite backoff. This implies that only 55% of all links are connected to a second transceiver if we implement infinite backoff, which explains the poor network connectivity.

This highlights the tradeoff between simple nodes and the degree of control required during self-assembly to achieve good network connectivity. Simpler nodes require regular topologies to achieve good connectivity. If nodes can implement mechanisms to allow more than two transceivers to share a single link, the system is well connected even if there is no control over the manufacturing process. Next, we examine the effect of the underlying network topology on system performance.

4.3 System Performance

In an ideal system, performance would be independent of network topology. To quantify the effect of topology on system performance, we measure the running time of an application (matrix multiplication) on different networks using a simualtor for our data parallel architecture. We measure program run time for networks with at most two transceivers sharing links (infinite backoff), or pairs of transceivers sharing links (links as buses). We simulate matrix multiplication for three matrix sizes - 8x8, 16x16 and 32x32. We find that as long as enough PEs can be configured in the network there is very little variation in program running time. Detailed results are omitted due to space constraints.

4.4 Effect of Defects

To study the effect of defects on network connectivity, we apply a node failure model [11] with a range of device reliabilities. Table 2 lists the percentage of nodes that are reachable in a network of 21,025 nodes for shared links. The numbers in parantheses are the percentage of reachable nodes when modeling infinite backoff. We do not show this number if it is less than 10%. We see that the percentage of reachable nodes drops rapidly as device reliability or control over self-assembly decreases. System connectivity decreases since some regions get disconnected due to the loss of critical nodes/links to defects. This is reflected by a drop in the number of transceivers per link (between 22-50% drop) as the device reliability decreases from 100% to 99.99%.

The results highlight the benefit of link sharing over infinite backoff. Link sharing allows a larger number of nodes to remain connected as device reliability decreases. This is true even for configurations with low control during self-assembly (N0-N3). We can draw two conclusions from these results: 1) if device reliability is lower than 99.999%, we either need to control placement and orientation during self-assembly, or we need to implement link sharing, to maintain network connectivity, and 2) controlling placement and ori-

TABLE 2. Percentage of nodes reachable with varying device reliabilities when links are shared between multiple transceivers. The figure in parantheses is for nodes implementing infinite backoff.

% Device Reliability	Configuration							
	N0	N1	N2	N3	N4	N5	N6	N7
99.990	3.8	4.9	3.9	2.5	3.4	6.6	86 (84)	90 (90)
99.993	18	26	19	7.2	18	38	91 (91)	93 (93)
99.996	73	73	73	31	72	72	95 (95)	96 (96)
99.999	88	88	89	78	88.6 (11.7)	84.7 (35.5)	99 (99)	99 (99)
100.00	91	91	92	84	92 (19.5)	87 (50)	100 (100)	100 (100)

entation has a greater effect on network connectivity than link sharing. The two techniques can be combined to achieve greater effect.

5 Conclusions

In this paper, we evaluate the characteristics of a class of network topologies that could be created by exercising varying degrees of control during the self-assembly of simple nodes. The evaluation highlights the tradeoff between node complexity and the amount of control required during self-assembly to maximize the number of connected nodes in the network. We also see that so long as the network has enough nodes, system performance is not affected by the type of configuration created by self-assembly. Finally, we see that introducing defects has a greater effect on networks with a lower degree of control during self-assembly. However, this can be mitigated to some extent by allowing more than two transceivers to share a link.

Acknowledgements

This work is supported by an NSF ITR grant CCR-0326157, the Duke University Provost's Common Fund, AFRL contract FA8750-05-2-0018, and equipment donations from IBM and Intel.

References

[1] A. DeHon. Array-Based Architecture for Molecular Electronics. In *Proc. of the First Workshop on Non-Silicon Computation (NSC-1)*, Feb. 2002.

[2] C. Dwyer et al. The Design and Fabrication of a Fully Addressable 8-tile DNA Lattice. In *Foundations of Nanoscience: Self-Assembled Architectures and Devices*, pages 187–191, Apr. 2005.

[3] Y. Fukunaka et al. Producing Shape-Controlled Metal Nanowires and Nanotubes by an Electrochemical Method. *Electrochemical and Solid-State Letters*, 3(9):C62–C64, 2006.

[4] S. C. Goldstein and M. Budiu. NanoFabrics: Spatial Computing Using Molecular Electronics. In *Proc. of the 28th Annual International Symposium on Computer Architecture*, pages 178–191, July 2001.

[5] Y. Huang et al. Directed Assembly of One-dimensional Nanostructures into Functional Networks. *Science*, 291:630–633, 2001.

[6] H. Kudo and M. Fujihira. DNA-Templated Copper Nanowire Fabrication by a Two-Step Process Involving Electroless Metallization. *IEEE Trans. on Nanotechnology*, 5(2):90–92, 2006.

[7] P. Mahadevan et al. A Basis for Systematic Analysis of Network Topologies. *ACM SIGCOMM Conference*, 2006.

[8] S. H. Park et al. Finite-size, Fully-Addressable DNA Tile Lattices Formed by Hierarchical Assembly Procedures. *Angewandte Chemie*, 45:735–739, Jan. 2006.

[9] J. P. Patwardhan et al. Circuit and System Architecture for DNA-Guided Self-Assembly of Nanoelectronics. In *Foundations of Nanoscience: Self-Assembled Architectures and Devices*, pages 344–358, Apr. 2004.

[10] J. P. Patwardhan et al. A Defect Tolerant Self-Organizing Nanoscale SIMD Architecture. *International Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Oct. 2006.

[11] J. P. Patwardhan et al. Design and Evaluation of Fail-Stop Self-Assembled Nanoscale Processing Elements. In *IEEE International Workshop on Design and Test of Defect-Tolerant Nanoscale Architectures (NANOARCH '06)*, June 2006.

[12] J. P. Patwardhan et al. NANA: A Nano-scale Active Network Architecture. *ACM Journal on Emerging Technologies in Computing Systems*, 2(1):1–30, 2006.

[13] C. Pistol et al. Design Automation for DNA Self-Assembled Nanostructures. *Proc. of the 43rd Design Automation Conference (DAC)*, July 2006.

[14] B. H. Robinson and N. C. Seeman. The Design of a Biochop: a Self-Assembling Molecular-Scale Memory Device. *Protein Engineering*, 1:295–300, Aug. 1987.

[15] P. W. K. Rothemund. Folding DNA to Create Nanoscale Shapes and Patterns. *Nature*, 440:297–302, 2006.

[16] K. Skinner et al. Nanowire Transistors, Gate Electrodes, and Their Directed Self-Assembly. *The 72nd Southeastern Section of the American Physical Society (SESAPS)*, Nov. 2005.

[17] H. Tangmunarunkit et al. Network Topology Generators: degree-based vs. structural. *SIGCOMM Comput. Commun. Rev.*, 32(4):147–159, 2002.

[18] J. M. Tour. Molecular Electronics. Synthesis and Testing of Components. *Accounts of Chemical Research*, 33(11):791–804, 2000.

[19] H. Yan et al. DNA Templated Self-Assembly of Protein Arrays and Highly Conductive Nanowires. *Science*, 301(5641):1882–1884, Sept. 2003.