Chapter 9
C2, Networks, and Self-Synchronization

A. H. Dekker
University of Ballarat, Australia

ABSTRACT

This chapter examines the connection between network theory and C2, particularly as it relates to self-synchronization, which requires a rich network structure. The richness of the network can be measured by the average degree, the average path length, and the average node connectivity. The chapter explores the connection between these measures and the speed of self-synchronization, together with other network properties, which can affect self-synchronization, resilience, and responsiveness. Two important network structures (random and scale-free) are described in the context of self-synchronization. Experimental data relating network topology to self-synchronization speed is also explored. In particular, the chapter notes the connection between average path length and self-synchronization speed, as well as the importance of good networking between sub-networks.

INTRODUCTION

The past few decades have seen an increasing awareness of the importance of C2 networks. There has also been an exploration of non-traditional designs for C2 networks, both in terms of network topology and in terms of network-enabled styles of operation, in order to achieve greater effectiveness in the face of modern threats (Alberts & Hayes, 2003, 2006). The application of network theory has obvious benefits here. But what can network theory tell us about C2 network design? What measures and metrics from network theory characterize "good" networks? There has also been an increasing awareness of the importance of *agility* in military forces. Agility can be broken down into the attributes of *robustness*, *resilience*, *responsiveness*, *flexibility*, *innovation*, and *adaptation* (Alberts & Hayes, 2003, p. 128). Which networks make a military force more agile?

At the same time, recent decades have seen important advances in network theory, and the concept of self-synchronization has become a meeting-point, approached both from inside the network science community (Watts & Strogatz, 1998; Watts, 2003; Strogatz, 2003) and the military community (Alberts & Hayes, 2003, 2006; Orr & Nissen, 2006; Brehmer, 2009). Detailed

DOI: 10.4018/978-1-4666-6058-8.ch009

mathematical analysis of network attributes has been conducted (Bollobás, 2001; Chung & Lu, 2003), and this has been complemented by experiments studying the ability of human beings to self-synchronize in practice (Kearns *et al*., 2006; Thunholm *et al*., 2009). Computer simulation experiments (Watts & Strogatz, 1998; Dekker 2005, 2006, 2007a, 2007b, 2010a, 2011; Gateau *et al*., 2007) have further illuminated this meeting-point between network theory and military science.

In this chapter, we explore the connection between C2, networks, and self-synchronization, in order to address the question of which network topologies are best. In particular, we examine how several network measures and attributes relate to the ability of a networked system to self-synchronize. We do this by surveying the relevant theoretical literature as well as reporting the results of some experiments with an abstract model of synchronization. We begin with the factors that influence an organization's choice between a centralized and a decentralized structure, and continue with evidence for the importance of networks with low average path length, high average node connectivity, and a priority on networking across a whole force, rather than simply within subnetworks.

NETWORK TOPOLOGY AND PROBLEM TYPE

Much of Command and Control (C2) consists of addressing challenging resource allocation problems – often under conditions of uncertainty and risk. It is true that there is a core part of C2 which is essentially creative, and involves outlining a conceptual framework or way of thinking about the problem at hand (Builder *et al*., 1999). However, a large part of C2 involves the allocation of people and platforms (on the one hand) to places and tasks (on the other). A good network topology will facilitate this process.

In studying C2-related resource allocation problems, we can divide them into three categories, which we will call "easy," "difficult," and "fiendish." An example of an "easy" problem is finding the largest number in a set. The effort required to solve such an "easy" problem will be at most proportional to the size of the problem, since the problem can be solved by a single scan through the set. Technically, such problems are known as linear-time or sub-linear-time problems.

"Difficult" problems include finding the Minimum Spanning Tree (MST) of a network (Cormen *et al*., 1990). Figure 1 shows an example. If the network is understood to be a network of cities connected by roads of various lengths, then the minimum spanning tree is the shortest network of cables which will connect all the cities, on the assumption that the cables must be strung alongside the roads. This problem can be solved with a computer, but large instances require minutes (or even hours) of computation. In general, the effort required to perfectly solve "difficult" problems will be proportional to some more-than-linear polynomial function of the problem size. Technically, such problems (together with the "easy" class) are known as polynomial-time problems.

"Fiendish" problems (technically, NP-hard problems) include the Travelling Salesman Problem (TSP), which requires finding the shortest loop visiting all nodes in the network exactly once (Cormen *et al*., 1990). Again, Figure 1 provides an example. Perfectly solving "fiendish" problems requires effort proportional to some exponential function of the problem size, which makes problem instances of even moderate size impossible to solve. Typically, the best that can be hoped for is finding reasonably good solution, and doing so may fall into either the "easy" or "difficult" categories.

In a more military context a similar distinction arises. The Assignment Problem (AP) – the simple assignment of units to tasks, where each task requires one unit, and each unit can carry out only one task – is a "difficult" problem, requiring