8-27-2009

Distributed load balancing over directed network topologies

Alejandro Gonzalez Ruiz

ALEJANDRO GONZALEZ RUIZ

Candidate

ELECTRICAL AND COMPUTER ENGINEERING

Department

This thesis is approved, and it is acceptable in quality and form for publication on microfilm:

Approved by the Thesis Committee:

, Chairperson

Accepted:

Dean, Graduate School

Date

Distributed Load Balancing over Directed Network Topologies

by

Alejandro González Ruiz

Electronics Engineer,
Universidad del Valle de Guatemala, 2006

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

August, 2009

Dedication

To my mother Emma

Acknowledgments

I would like to thank my advisor, Professor Yasamin Mostofi for everything she has allowed me to learn. It has been an honor to be able to work with her. Her knowledge and her enthusiasm for research have been truly inspiring and an example to follow.

I also thank my family, especially my mother Emma, my sisters Paola and Matty, their families and my father Rodolfo for their love and support and for always being there for me.

My gratitude also goes to Jorge Pezoa and Dr. Majeed Hayat for countless hours of useful discussions and for increasing my interest in load balancing. I would also like to thank Dr. Chaouki Abdallah for accepting to be in my thesis committee and for his helpful comments.

I would also like to express my gratitude to the Fulbright Foreign Student Program for giving me the opportunity and support to come to the United States to earn my Master's degree.

Finally, I would like to thank my friends Luis Rivera and José Velasquez, my colleagues at the Cooperative Network Lab and everyone who in one way or another has helped me develop this work.

Distributed Load Balancing over Directed Network Topologies

by

Alejandro González Ruiz

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

August, 2009

Distributed Load Balancing over Directed Network Topologies

by

Alejandro González Ruiz

Electronics Engineer,
Universidad del Valle de Guatemala, 2006

M.S., Electrical Engineering, University of New Mexico, 2009

Abstract

Due to the increasing demand for high performance computing and the increasing availability of high speed networks, it has become possible to interconnect various geographically distributed computational elements (nodes) so that they can work cooperatively and obtain a performance not attainable by individual nodes. In the literature, distributing the total computation load across available processors is referred to as load balancing.

This thesis considers the problem of distributed load balancing over directed graphs that are not fully connected. The impact of network topology on the stability and balance of distributed computing is studied. Furthermore, Informed Load Balancing (I-LB) is proposed. This is an approach in which the nodes first reach an agreement over the balanced state, by using a consensus-seeking protocol, before proceeding to redistribute their tasks. The performance of I-LB is compared with the performance of the Original Load Balancing approach (O-LB) in terms of speed of convergence and bandwidth usage. A proof is

given that shows that the O-LB approach can guarantee convergence to a balanced state if the underlying graph is strongly connected while I-LB may not converge. However, I-LB can increase the speed of convergence and/or reduce the bandwidth usage especially for low-connectivity graphs. Finally, a third protocol called Consensus-Based Load Balancing (C-LB) is studied and its convergence characteristics and tradeoffs are discussed.

Contents

List of Figures **xii**

Glossary **xiv**

1 Introduction **1**

1.1 Problem Description and Motivation 1

1.2 Objective of this Thesis 3

1.3 Overview of Thesis 4

2 Overview of Load Balancing Policies and Previous Work **5**

2.1 Types of Load Balancing Policies 5

2.1.1 Centralized or Distributed Load Balancing 5

2.1.2 Local or Global Load Balancing 6

2.1.3 Static or Dynamic Load Balancing 6

2.1.4 Sender-initiated and Receiver-initiated Load Balancing 7

2.1.5 When to do Load Balancing? 7

2.2 Related Work 8

2.2.1 Load Balancing in a Computer Network 8

2.2.2 Load Balancing in the Presence of Delays 9

2.2.3 Load Balancing using Regeneration Theory 10

2.2.4 Agent-Based Load Balancing 10

2.3 Differences Between this Work and Previous Load Balancing Approaches 11

3 Overview of Discrete-Time Distributed Consensus Approaches 13

3.1 Algebraic Graph Theory 14

3.1.1 Properties of the Laplacian Matrix 15

3.2 Discrete-Time Consensus 16

3.3 Quantized Consensus 17

4 Analysis of Load Balancing over Directed Network Topologies 19

4.1 Problem Formulation 19

4.1.1 Link Symmetry and the Need for Considering Directed Graphs . . 20

4.2 Description of the Distributed Load Balancing Algorithm 21

4.3 Impact of Network Topology on Distributed Load Balancing 23

4.4 Convergence to a Balanced State 27

4.5 On the Necessity of Having a Strongly Connected Graph for Achieving a Balanced State 32

5 A Consensus Approach to Load Balancing **33**

5.1 Distributed Load Balancing with a priori knowledge of the Balanced State (I-LB) 33

5.1.1 Discrete-Time Consensus Problem 34

5.1.2 Consensus over the Global Average and Informed Load Balancing 34

5.1.3 Underlying Tradeoffs Between O-LB and I-LB 35

5.1.4 Lack of Asymptotic Performance Guarantee for I-LB 37

5.2 Consensus-Based Load Balancing over Directed Graphs (C-LB) 38

5.2.1 Underlying Tradeoffs Between O-LB and C-LB 38

6 Conclusions and Further Extensions **45**

Appendices **46**

References **47**

List of Figures

4.1 An example of a directed network topology that contains no spanning tree. 23

4.2 Queue dynamics for distributed load balancing over the network of Fig. 4.1. 24

4.3 An example of a network topology with an underlying spanning tree (solid arrows only) and a network topology with an underlying strongly-connected graph (dashed and solid arrows). 25

4.4 Queue dynamics for distributed load balancing over the graph topology of Fig. 4.3 (considering only the solid arrows). The queues cannot reach a balanced state even though there is an underlying spanning tree. 26

4.5 Queue dynamics for distributed load balancing over the strongly connected graph of Fig. 4.3 (considering both solid and dashed arrows). The queues reach a balanced state. 27

4.6 An example of a directed network topology that contains a spanning tree. 27

4.7 Queue dynamics for distributed load balancing over the network of Fig. 4.6, which is not strongly connected but has a spanning tree. The distributed system reaches the balanced state for the initial distribution of $x(0) = [951\ 707\ 486\ 332]^T$. 28

5.1 Network topology of a path network. I-LB reaches a balanced state considerably faster for an initial load distribution $x(0) = [707\ 486\ 332\ 951]^T$. 36

5.2 Graph configurations studied for the comparison of O-LB and I-LB. . . 40

5.3 Comparison of the time steps required to reach the balanced state for various undirected network topologies using O-LB and I-LB with 4 nodes (see Fig. 5.1). 41

5.4 Comparison of the bandwidth usage (total number of transmitted packets) required to reach the balanced state for various undirected network topologies with 4 nodes (see Fig. 5.1). 42

5.5 Queue dynamics for O-LB over the network of Fig. 5.1 and with $x(0) = [20\ 19\ 20\ 17]^T$. The queues reach a balanced state. 43

5.6 Queue dynamics for I-LB over the network of Fig. 5.1 and with $x(0) = [20\ 19\ 20\ 17]^T$. The queues cannot reach the balanced state. 43

5.7 Comparison of the time steps required to reach the balanced state for various network topologies using O-LB and C-LB with 4 nodes. 44

Glossary

N	Set of nodes representing processors $N = \{1, ..., n\}$.
E	Set of edges connecting nodes.
$x_i(k)$	Load on processor i at time k.
$s_{ij}(k)$	Load sent from node i to node j.
$J_i(k)$	External tasks arriving to node i at time k.
$C_i(k)$	Tasks serviced by node i at time k.
$\mathcal{N}_i$	Neighborhood set of node i.
$L_i^{ex}(k)$	Excess load of node i at time k.
$L_{j(i)}^{ex}(k)$	Excess load of node j with respect to the neighborhood average of node i with the convention that $L_{i(i)}^{ex}(k) = L_i^{ex}(k)$.
$p_{ij}(k)$	Fraction of $L_i^{ex}(k)$ to be sent from node i to node j.
$\mathcal{M}_i(k)$	Set of neighbors of node i whose load is below ave$_i(k)$.
A	Adjacency matrix $A = [a_{ij}]$ with $a_{ii} = 0$ and $a_{ij} > 0$ if there exists a directed edge from node j to node i.
L	Graph Laplacian.

Chapter 1

Introduction

1.1 Problem Description and Motivation

Everyday there is an increasing demand for high performance computing. High speed networks have become more common, allowing for interconnecting various geographically distributed Computational Elements (CEs). This has enabled cooperative operation among the nodes, which can result in obtaining an overall better performance than the one achieved by a single CE. Grid-computing is an example of a system that can benefit from cooperative computing [1].

Distributing the total computational load across available processors is referred to as load balancing in the literature. The goal of a load balancing policy is to ensure an optimal use of the available resources so that each CE ends up with a "fair" share of the overall job, thus allowing the overall load to be completed as fast as possible. Load balancing can be implemented in a centralized [2] or a distributed manner [1]. In this work, we are interested in distributed load balancing, where each node polls other processors in its neighborhood and uses this information to decide on a load transfer.

There has been extensive research in the development of effective load balancing policies. In [3–7] various distributed load balancing schemes that use concepts such as queuing- and regeneration-theory have been proposed and analyzed. A common factor in these approaches is that the amount of load to be exchanged is based on the weighted averages of the loads of the neighboring nodes. In [8–10] the proposed policies use concepts such as graph coloring and gossip algorithms. These algorithms do pairwise balancing of loads, i.e., two adjacent nodes are randomly chosen to exchange their loads at a given time step.

In the load balancing literature, a common assumption is to take the topology of the underlying graph to be undirected. However, most realistic wireless networks will have asymmetric uplink and downlink, resulting in directed graphs. This is due to the fact that wireless transmissions in uplink and downlink typically occur in two different and uncorrelated pieces of bandwidth [11], resulting in different and uncorrelated link qualities. Furthermore, different nodes may have different transmission powers resulting in different reception qualities (even if the links were the same) and as a result directed graphs. Another scenario where the network topology is directed occurs when there are firewalls between certain nodes that allow incoming connections but block outgoing ones. Therefore, in this thesis we mainly focus on directed graphs. We study the impact of network topology on the stability and balance of distributed computing. For the case of a one-time arrival of the loads, we show that a proposed load balancing scheme (O-LB) over a strongly connected graph reaches a balanced state, independent of the initial load distribution, while having a spanning tree is not a sufficient condition for reaching a balanced state. We furthermore propose Informed Load Balancing (I-LB), an approach in which the nodes first reach an agreement over the global balanced state before starting the actual load balancing process. We show that while I-LB lacks asymptotic performance guarantees, it has the potential of increasing the speed of convergence and/or reducing the bandwidth usage, especially for low-connectivity graphs.

Finally, we study Consensus-Based Load Balancing (C-LB) where at every time step each node keeps a fixed fraction of its load while the rest of the load gets equally partitioned among all of its neighbors. The advantage of this approach is that its convergence can be analyzed in the same manner as the convergence of a discrete-time consensus protocol to average consensus.

1.2 Objective of this Thesis

The main goal of this thesis is to study the effect that the underlying graph topology of a network has on the dynamics and stability of load balancing. In the existing literature, balancing policies have been developed where the network is assumed to be fully connected and/or undirected. However, there are real-world scenarios where this is not true due to, for example, power constraints or link asymmetries. Therefore, there is a need to study the conditions where stability of load balancing over these networks can be guaranteed for any initial condition.

We propose a protocol where each node compares its load with the load of the nodes in its neighborhood, then it determines whether it is overloaded and if it is, it sends a fraction of its excess load only to those nodes that are underloaded with respect to its local average. For that protocol, we show the sufficient conditions to reach a balanced state, independent of the initial distribution. We furthermore propose a novel modification to this protocol whose main objective is to reduce the bandwidth usage and the time required for the system to reach the balanced state. This modification is based on the theory of consensus and consists in allowing the nodes to reach a consensus over the global average before starting the task redistribution. We study the performance and tradeoffs of this modification with respect to the original load balancing scheme.

1.3 Overview of Thesis

This thesis is organized as follows: in Chapter 2 we present an overview of load balancing and several of the approaches and models that have been presented in the literature. In Chapter 3 we do a brief survey of the main results of the discrete-time consensus problem. Specifically, we focus on the approaches based on the Laplacian of the underlying graph. We list the theorems regarding the conditions to reach average-consensus. In Chapter 4 we present our system model and proposed load balancing strategy, we explore the impact of graph topology on distributed computing and show the asymptotic convergence of the distributed load balancing algorithm to the balanced state. Chapter 5 introduces I-LB, an alternative load balancing approach. We explore the underling tradeoffs between I-LB and the original load balancing approach and compare both approaches in terms of bandwidth usage and time steps required to reach the balanced state. Finally, we also study C-LB, whose convergence analysis can be done in the same way as the convergence analysis of the discrete-time consensus protocol to average consensus.

Chapter 2

Overview of Load Balancing Policies and Previous Work

This chapter presents an overview of the different types of load balancing, followed by a brief review of the related work in this area.

2.1 Types of Load Balancing Policies

The load balancing policies can be categorized according to whether they are centralized or distributed, local or global, static or dynamic, and sender-initiated or receiver-initiated [12]. Next, we provide a brief explanation of each category.

2.1.1 Centralized or Distributed Load Balancing

In centralized load balancing, there is a designated node that controls if or how much load balancing should be done as discussed in [2]. All sender nodes (or receiver nodes) access the designated node to calculate the amount of load to be transfered as well as to identify

where tasks are to be sent to (or received from). The main drawback of this type of policy is that it is paralyzed if the central node fails.

On the other hand, if every node executes balancing autonomously, then load balancing is said to be distributed [1,3,4,6,13]. In this case each node makes the decisions regarding the amount of load (if any) to be sent to other nodes as well as the nodes that will receive that load. The policies that will be proposed and analyzed in this thesis belong to the category of distributed local load balancing, which are more robust and less susceptible to failure.

2.1.2 Local or Global Load Balancing

In a local load balancing scheme, each processor polls other processors in its small neighborhood and uses this local information to decide upon a load transfer, thereby minimizing remote communications. At every step a processor communicates with its nearest neighbors in order to achieve a local balance. The primary objective is to efficiently balance the load on the processors as well as to minimize remote communications. In contrast, for a global balancing scheme, a certain amount of global information is used to initiate the load balancing [14]. The schemes proposed in this thesis are all local schemes since they only use information of the processors in their neighborhood. However, in one of the schemes (I-LB) the nodes first try to estimate some global knowledge based on local information.

2.1.3 Static or Dynamic Load Balancing

In a static load balancing policy, load is assigned to nodes without consideration of runtime events. This scheme has a limited application in realistic distributed systems since it is generally impossible to make predictions of arrival times of loads and processing times required for future loads, as indicated in [4]. Static load balancing policies can also be

affected by the existence of communication delays.

In dynamic load balancing, the scheduling decisions are made at runtime according to the current status of the system [3]. A dynamic load balancing policy can be made adaptive to changes in system parameters, such as traffic and delays in the channel and the unknown characteristics of the incoming loads. State information exchange is a necessary part of this approach. Dynamic methods generally react better to changes in the system state compared to the static methods and as a result, have better performance.

2.1.4 Sender-initiated and Receiver-initiated Load Balancing

In a sender-initiated load balancing policy, the overloaded nodes transfer one or more of their tasks to the under-loaded nodes [6], while in a receiver-initiated load balancing policy, the lightly loaded nodes request loads from the overloaded nodes as shown in [15]. All the policies proposed and studied in this thesis belong to the class of sender-initiated load balancing.

2.1.5 When to do Load Balancing?

Load balancing can be done continuously, periodically, at one time, or at a limited number of times. In [6], the authors suggest that for realistic distributed systems, it is better to perform load balancing more than once or periodically during run-time such that there is a closer match to the available computational resources. In [3] the authors study a one-shot load balancing policy, where load balancing is done only once at run-time. They find that if the distributed system has random communication delays, limiting the number of balancing instants and optimizing the performance over the choice of the balancing times as well as the load balancing gain can result in significant improvement in the computing efficiency. They also propose a policy where at every external load arrival, the receiver node

is triggered to perform load balancing. The authors show experimentally that the proposed policy minimizes the average completion time per task when the delays are random. In this thesis we will consider load balancing that is performed continuously.

2.2 Related Work

In this section, we describe some of the previous work on load balancing that is relevant to the problem that we will study in this thesis.

2.2.1 Load Balancing in a Computer Network

In [16], the authors study load balancing over a network of n processors with an underlying connected undirected graph $G = (N, E)$. Load exchange takes place among the processors asynchronously and the load can be described by a continuous variable. The load handled by processor i at time t is denoted by $x_i(t) \geq 0$. The total load in the network is L and $E(i)$ is the set of neighbors of the ith processor. Each processor i makes an estimate of the load of its neighbor(s) j: $x_j^i(t)$. Since there is a communication delay, this estimate can be outdated so that $x_j^i(t) = x_j(\tau_j^i(t))$ where $\tau_j^i(t)$ is an integer variable satisfying $0 \leq \tau_j^i(t) \leq t$. At certain load balancing times, which are not necessarily periodic, a processor compares its load to its neighbors and if it is overloaded it transfers a nonnegative amount of its load ($s_{ij}(t)$) to node j.

The authors focus on proving that, given certain assumptions and following a given algorithm, a balance will be reached i.e. $\lim_{t\to\infty} x_i(t) = \frac{L}{n}$. They start by listing a set of assumptions that are sufficient for the load balancing algorithm to converge for any initial distribution. The first three assumptions are related to the frequency in which load balancing has to be done, the maximum allowable information delay and the time it takes for the load sent from node i to reach node j. The next assumption states that when a processor i

detects a load imbalance, it will transfer a nonnegligible portion of its excess load to some lightest loaded processor in its neighborhood, and possibly to other neighbors, as long as these processors are not carrying a larger load. The final assumption prevents a processor i from transferring a very large amount of load and creating a load imbalance in the opposite direction.

Their main result is the proof showing that any algorithm that satisfies the given assumptions will converge to a balanced state i.e., $\lim_{t\to\infty} x_i(t) = L/n$ for any initial distribution. The authors however, consider undirected graphs only. In Chapter 4 we will extend these results to consider directed graphs and find the sufficient condition on the topology that guarantees convergence of distributed load balancing for any initial distribution.

2.2.2 Load Balancing in the Presence of Delays

In [6], the authors consider an arbitrary number of distributed nodes with different queue sizes. Initially, each node broadcasts its queue-size information, which will be delayed by some time (referred to as communication delay) while reaching other nodes. All the nodes execute load balancing together at common balancing instants (called load balancing instants). At the load balancing instant, each node calculates its excess load by comparing its load to the total load of the system and partitions its positive excess load among other nodes. Then, each partition is scaled by multiplying with a common balancing gain $K \in [0; 1]$ before transferring it to the receiving node. In [17], performance of this protocol is demonstrated through simulation. The authors show that a large value of K (closer to one) produces a high degree of fluctuations in the tails of the queues. Furthermore, they show how to optimize balancing times and load balancing gain. In [4], the authors study a one-shot load balancing policy. In particular, once nodes are initially assigned certain number of tasks, all nodes execute load balancing together only at one instant, using a common load balancing gain K. The authors verified theoretically and by simulation that

for a given initial load and average processing rates, there exists an optimal load balancing gain and an optimal balancing instant associated with the one-shot load balancing policy, which together minimize the average overall completion time.

2.2.3 Load Balancing using Regeneration Theory

In [1] and [3] a queuing approach to load balancing, based on stochastic regeneration, is formulated to analyze the joint evolution of the queues. The model specifically considers the randomness and heterogeneity in processing times of the nodes, randomness in delays in the communication network, and uncertainty in the number of functional nodes. Coupled renewal equations are derived for certain classes of static and dynamic load balancing policies. The performance of the proposed load balancing policies are evaluated using analytical, experimental and Monte-Carlo simulation methods. In particular, the interplay between the optimal amount of load-transfers between nodes, node-failure/recovery rates, and the average load-transfer delays are rigorously investigated. The performance of the proposed dynamic load balancing policy is compared to that of existing static and dynamic load balancing policies. Additionally, the theory is applied to a distributed wireless-sensor network and the interplay between the total service time and the energy consumption of each sensor is shown.

2.2.4 Agent-Based Load Balancing

In [18], a load balancing algorithm inspired by the study of ants is analyzed. Agents are used to "carry" tasks. Immediately after a task is submitted to a node, an agent will be automatically dispatched to the task. The agent carries the task to search for appropriate agent teams to queue. An agent team is a group of agents queuing at a certain node. An agent can decide to leave a node, wander on the network or join a team in a visited node. The collection of neighbors is highly dynamic and the strategies are determined from own

experience and communication from other agents. The scheduling mechanism is decentralized. The following assumptions are made: processors can be added or removed from the system at runtime and all tasks can be divided into independent pieces with the same size. Furthermore, the network is homogeneous, i.e. all nodes have the same processing speed, memory size, etc and it is undirected. The authors show that load balancing with different initial distributions will converge to the same distribution finally. However, it is proven that the final distribution does not always correspond to perfect balancing. However, if agents have complete information about nodes on the system, then the steady state corresponds to perfect balancing.

In [19] and [20], the authors study a multiagent system whose agents move across a network of discrete locations (in a load balancing context, processing nodes) competing for resources they obtain from such locations (processing power for load balancing). The resources they compete for are continuous while the movements of agents in the network are discrete. The environment is modeled as a graph where the nodes represent the discrete locations and the edges represent the paths that the agents use to obtain information about other nodes and to move between locations. The ultimate objective is to be able to control agents and nodes such that their interacting dynamics converge to a point that optimizes the usage of the resources of the network (a balanced state in load balancing context). The authors propose a hybrid optimization framework that can guarantee stability for all possible configurations of agents and nodes.

2.3 Differences Between this Work and Previous Load Balancing Approaches

The class of load balancing schemes that we study and propose belong to the category of distributed, local, sender-initiated, dynamic load balancing. As the previous sections have shown, a common practice in the load balancing literature has been to assume that the

underlying graphs are undirected. In this work, however, we study a more general case of networks whose underlying graph is directed. We show the impact that directed graphs have on the convergence characteristics of distributed load balancing. We also show the conditions on the topology that guarantee convergence to a balanced state for any initial distribution.

Furthermore, we also propose a novel approach (I-LB) that makes use of consensus protocols to improve the speed of convergence and bandwidth usage. To the best of our knowledge, the underlying tradeoffs between I-LB and traditional load balancing approaches are not characterized, which is one of the goals of this thesis. To emphasize on the impact of the network topology, we will assume that the communication delays are negligible and that no new tasks arrive or leave the system after the time 0.

Chapter 3

Overview of Discrete-Time Distributed Consensus Approaches

This chapter provides an overview of the basics of distributed consensus, which will serve as a basis for the Informed Load Balancing (I-LB) and Consensus-Based Load Balancing (C-LB) techniques, discussed in chapter 5.

In consensus problems, a group of agents such as mobile robots, unmanned air vehicles, satellites or possessors try to reach an agreement over a certain value. They exchange information with their neighboring agents and update their values according to a given update protocol. The goal is to ensure that as time progresses, all agents converge to the same value [21]. Average consensus seeking protocols aim to make the final value (the group decision) the average of the initial values of the agents. In [21–26] distributed consensus has been studied using concepts from algebraic graph theory, for which we will provide a brief overview in this section.

3.1 Algebraic Graph Theory

It is common to model the information exchange between agents of a cooperative network as a graph. A directed graph G consists of a pair (N, E), where N is finite nonempty set of nodes and $E \in N^2$ is the set of edges. An edge of G will be denoted by $e_{ij} = (i, j)$. The adjacency matrix $A = [a_{ij}]$ of a graph is then defined as $a_{ii} = 0$ and $a_{ij} > 0$ if $(j, i) \in E$ where $i \neq j$.[1] The Laplacian of the graph is then defined as $L = [l_{ij}]$ with:

$$l_{ij} = \begin{cases} \sum_{k=1,k\neq i}^{n} a_{ik}, & , \quad j = i \\ -a_{ij}, & , \quad j \neq i \end{cases} \tag{3.1}$$

Then a neighborhood set $\mathcal{N}_i$ of node i is given by all the nodes $j \in N$ such that there is a directed link from i to j. In other words, $\mathcal{N}_i = \{j \in N | (i, j) \in E\}$ with $|\mathcal{N}_i|$ representing its cardinality. The in-degree and out-degree of node i are then defined as follows:

$$\deg_{\text{in}}(i) = \sum_{j=1}^{n} a_{ij} \qquad \text{and} \qquad \deg_{\text{out}}(i) = \sum_{j=1}^{n} a_{ji}.$$

For a graph with 0-1 adjacency elements, $\deg_{\text{out}}(i) = |\mathcal{N}_i|$. A graph is undirected if $(i, j) \in E$ implies that $(j, i) \in E$ for all $i, j \in N$. In terms of connectivity, we will classify graphs as in [21]:

- Fully connected graph: A graph where each node has a direct connection to all other nodes in the network.
- Strongly connected graph: A graph where there exists a directed path from any node to any other node of the network. A direct connection to all other nodes is not necessary, but information flow from any node must reach the others.

[1] It should be noted that some authors e.g. [23, 27], define A as the transpose of the matrix defined here. This means that $a_{ij} > 0$ if and only if there is a directed edge from node i to node j.

- Spanning tree: A directed graph contains a spanning tree if there exists a node that has a directed path to all the other nodes.

- Balanced graph: A directed graph where the out-degree of each node equals its in-degree, i.e., the number of directed edges coming into a node is the same as the number of edges leaving that node for all nodes in the graph.

3.1.1 Properties of the Laplacian Matrix

Graph Laplacian and its spectral properties play a key role in the convergence behavior of consensus algorithms. Let $\lambda_i(L)$ represent the ith smallest eigenvalue of the Lapalacian matrix with $1 \leq i$. The second smallest eigenvalue of a graph Laplacian ($\lambda_2(L)$) is called the algebraic connectivity or Fiedler eigenvalue of L. In [28], it is shown that $\lambda_2(L)$ is a good measure of the speed of convergence of consensus algorithms. Every Laplacian has row sum equal to zero, i.e., $\sum_j l_{ij} = 0 \ \ \forall i$. As a result, L always has zero as its eigenvalue: $\lambda_1 = 0$. This zero eigenvalue corresponds to the eigenvector $\mathbf{1} = [1, 1, ..., 1]^T$: $L\mathbf{1} = 0$. Therefore, $x^* = (\alpha, ..., \alpha)^T = \alpha\mathbf{1}$ is an equilibrium of a system of the form $\dot{x} = -Lx$, which is a consensus state [28]. This also means that rank$(L) \leq n - 1$. In [23] it is shown that if G is a strongly connected directed graph, then rank$(L) = n - 1$, i.e. $x^* = (\alpha, ..., \alpha)^T = \alpha\mathbf{1}$ is the only equilibrium state, which guarantees consensus.

Spectral properties of the Laplacian are instrumental in the convergence analysis of linear consensus algorithms. According to Gershgorin's disc theorem, all the eigenvalues of L are located in a closed disc centered at $\max_i d_i + 0j$ in the complex plane with a radius of $\max_i d_i$, where d_i represents the maximum in-degree of a graph [28]. For undirected graphs, L is symmetric with real eigenvalues. For connected graphs, $\lambda_2 > 0$, in other words, the zero eigenvalue is isolated.

3.2 Discrete-Time Consensus

In the discrete-time consensus problem, each node has an associated value $y_i \in \mathbb{R}$ representing the information on which the team must come to an agreement. The set of nodes $\{1, ..., n\}$ is said to be in consensus if $y_i = y_j$ for all i, j. When the final value of each y_i is $y_i = \frac{1}{n}\sum_j y_j[0]$, the team is said to have reached average consensus [21]. Consensus protocols define how a node should update its value (y_i) based on the values of its neighbors. In [29] the authors propose the following discrete-time consensus protocol:

$$y_i[k+1] = \sum_{j \in \mathcal{N}_i \cup \{i\}} \beta_{ij}[k] y_j[k],$$

where $\mathcal{N}_i[k]$ represents the set of agents whose information is available to agent i at time step k, $\sum_{j \in \mathcal{N}_i[k] \cup \{i\}} \beta_{ij}[k] = 1$, and $\beta_{ij}[k] > 0$ for $j \in \mathcal{N}_i[k] \cup \{i\}$. In other words, the next state of each agent is updated as a weighted average of its current state and the current states of its neighbors. This protocol can be written in matrix form as $y[k+1] = D[k]y[k]$. In [21] and [23] the following discrete-time protocol in terms of the Laplacian is proposed and analyzed:

$$y[k+1] = (I - \epsilon L)y[k], \tag{3.2}$$

where $\epsilon \in (0, \frac{1}{\max_i l_{ii}})$. Larger values of ϵ can furthermore increase the convergence rate. The matrix $P = I - \epsilon L$ is referred to as the Perron matrix of a graph G with parameter ϵ. Since the row sums of L are all zero by construction, the row sums of $A = I - \epsilon L$ are all one. In [28], the following properties of the Perron matrix are proved:

1. P is a row stochastic nonnegative matrix with an eigenvalue equal to 1. This is due to the fact that $P = I - \epsilon L$, so $P\mathbf{1} = \mathbf{1} - \epsilon L\mathbf{1} = \mathbf{1}$.

2. All eigenvalues of P are on or inside the unit circle;

3. If G is a balanced graph, then P is a doubly stochastic matrix;

4. If G is strongly connected and $0 < \epsilon < \frac{1}{\max_i l_{ii}}$, then P is a primitive matrix.

In [28], the authors show that running the update protocol of Eq. (3.2) over a strongly connected topology guarantees achieving consensus, for any initial condition. The proof makes use of Perron-Frobenius theorem [30] which states that if P is a primitive non-negative matrix with left and right eigenvectors w and v respectively, satisfying $Pv = v$, $w^T P = w^T$, and $v^T w = 1$, then $\lim_{k\to\infty} P^k = vw^T$. It should however, be noted that average consensus may not be reached. In [21] and [23] it is then shown that average-consensus can be reached if and only if the graph is balanced and strongly connected. This result is justified by noticing that when the graph is balanced, then the matrix P becomes doubly stochastic with left eigenvector $w = (1/n)\mathbf{1}$.

3.3 Quantized Consensus

In [9], the authors study the problem of discrete-time consensus over undirected graphs. They propose a randomized distributed algorithm based on quantized gossip. The bounds on the convergence time of these algorithms are also derived. Furthermore, they suggest load balancing as one possible application of quantized consensus.

Each node is assigned a value (or in terms of load balancing, a load) at a time t given by $x[t]_i$. The algorithms are devised so that: the value at each node is always an integer, the sum of the values in the network does not change with time (i.e., there is no incoming or outgoing load) and that for any $x[0]$ there is a convergence time such that for any time t greater than or equal to that convergence time, $x[t]_i \in \{L, L+1\}$ for all i and L integer. Notice that since the values of $x[t]_i$ are quantized, the algorithms do not exactly converge to consensus. It is proved that for any two nodes, their values end up within a step size of each other for undirected connected graphs. The step size is assumed to be equal to one. Finally, the bounds on the expected convergence time to a quantized-consensus distribution are

found for fully connected networks and linear networks. However, these bounds are not tight.

Chapter 4

Analysis of Load Balancing over Directed Network Topologies

4.1 Problem Formulation

Consider a distributed computing system of n nodes connected over a wireless network with a directed topology that can be described by a graph $G = (N, E)$, where N is the set of nodes $N = \{1, ..., n\}$ and E is the set of edges connecting the processors. Let $x_i(k)$ represent the load on processor i at time $k \geq 0$. The goal is to spread the subtasks among all n processors as evenly as possible such that no node is overburdened while other nodes are idle. At every time step the nodes assess how overburdened they are and exchange loads in order to increase the overall computational efficiency.

For the purpose of mathematical analysis, we take the load of each processor to be infinitely divisible such that $x_i(k)$ can be described by a continuous nonnegative real number. We furthermore assume that: (i) tasks are independent so that they can be executed by any processor; (ii) processors are homogeneous, i.e. they have the same processing speed and (iii) the link-level delay in the exchange of loads between nodes is negligible.

This last assumption is justified when the product of the queue length and the service time (execution time per task) is considerably greater than the time it takes to transfer a group of tasks from one node to the other. We also assume that processors are synchronized i.e., all the processors perform load balancing at the same time. As indicated in [14], assuming synchrony yields similar results, in terms of the final load distribution, to the ones obtained by working with asynchronous algorithms.

4.1.1 Link Symmetry and the Need for Considering Directed Graphs

Consider the wireless link from node j to node i and that from node i to node j. In wireless communication, these two transmissions typically occur at two different and uncorrelated pieces of bandwidth [11]. As a result, the link quality in the transmission from node i to node j can be considerably different from that of the transmission from node j to i, resulting in an asymmetry. Therefore, in this work we take G to be a directed graph. Then a neighborhood set $\mathcal{N}_i$ of node i is defined as in Section 3.1, i.e., all the nodes $j \in N$ such that there is a directed link from i to j and $|\mathcal{N}_i|$ represents the cardinality of the neighborhood.

The nodes furthermore exchange their queue lengths with their neighbors continuously in order to assess how overloaded their local neighborhood is. Since this information has a considerably lower volume than the actual loads, it can be transmitted over a separate lower bandwidth channel or with higher transmission power. As a result, it can experience better reception quality and higher probability of symmetry. Therefore, in this work we assume that the queue length information is transmitted over an undirected version of graph G.

4.2 Description of the Distributed Load Balancing Algorithm

Let $s_{ij}(k)$ denote the load that node i sends to node j at time k. Let $J_i(k)$ and $C_i(k)$ represent the number of external tasks arriving to node i and the number of tasks serviced by node i respectively at time k. We will have the following for the dynamics of the queue length of node i:

$$x_i(k+1) = x_i(k) - \sum_{j \in \mathcal{N}_i} s_{ij}(k) + \sum_{j \in \{l | i \in \mathcal{N}_l\}} s_{ji}(k) + J_i(k) - C_i(k). \tag{4.1}$$

Using an approach similar to the one presented in [1] and [3], the amount of load to be transferred from node i to node j can be calculated based on the excess load of node i. Define $\text{ave}_i(k)$ as the local average of node i, i.e. the average load of node i and its neighbors:

$$\text{ave}_i(k) = \frac{1}{|\mathcal{N}_i| + 1} \left(x_i(k) + \sum_{j \in \mathcal{N}_i} x_j(k) \right). \tag{4.2}$$

Note that although the graph for the exchange of loads is directed, we assumed an undirected graph for the exchange of queue length information. Therefore the ith node has access to the value of $x_j(k)$ for all $j \in \mathcal{N}_i$ and can therefore calculate its local average.

The excess load of node i at time k ($L_i^{ex}(k)$) is then given by the difference between its load and local average:

$$\begin{aligned} L_i^{ex}(k) &= x_i(k) - \text{ave}_i(k) \\ &= x_i(k) - \frac{x_i(k) + \sum_{j \in \mathcal{N}_i} x_j(k)}{|\mathcal{N}_i| + 1}. \end{aligned} \tag{4.3}$$

This quantity represents how overloaded node i is with respect to its own neighborhood. Next, node i evaluates how overloaded its neighbors are with respect to its local average. Let $L^{ex}_{j(i)}(k)$ be the excess load of node j with respect to the neighborhood average of node i, namely:

$$L^{ex}_{j(i)}(k) = x_j(k) - \text{ave}_i(k) \;\text{ for }\; j \in \mathcal{N}_i. \tag{4.4}$$

$L^{ex}_{j(i)}(k)$ represents how overloaded the jth node "appears" to the ith one, based on the partial information available at the ith node. It can be easily verified that

$$\sum_{j \in \{\mathcal{N}_i \cup\, i\}} L^{ex}_{j(i)}(k) = 0, \tag{4.5}$$

with the convention that $L^{ex}_{i(i)}(k) = L^{ex}_i(k)$. Define $p_{ij}(k)$ as the fraction of $L^{ex}_i(k)$ to be sent from node i to node j. Then we will have the following for $s_{ij}(k)$, the amount of load that the ith node sends to the jth one at the kth time step [3]:

$$s_{ij}(k) = p_{ij}(k)\left(L^{ex}_i(k)\right)^+, \tag{4.6}$$

where $(x)^+ = \max(x, 0)$ and:

$$p_{ij}(k) = \begin{cases} \frac{L^{ex}_{j(i)}(k)}{\sum_{l \in \mathcal{M}_i(k)} L^{ex}_{l(i)}(k)} & j \in \mathcal{M}_i(k) \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

with

$$\mathcal{M}_i(k) = \{j | j \;\in\; \mathcal{N}_i,\; \text{ave}_i(k) > x_j(k)\}. \tag{4.8}$$

In other words, node i will send part of its excess load to node j at time k only if $x_j(k)$ is below $\text{ave}_i(k)$.

4.3 Impact of Network Topology on Distributed Load Balancing

In order to motivate the mathematical derivations of the next section, we first consider the impact of different network topologies on distributed load balancing through simulations. Two concepts that are frequently used throughout the thesis are that of spanning trees and strongly connected graphs. As indicated in Section 3.1, a directed graph has a spanning tree if there exists a node that has a directed path to all the other nodes [22]. Furthermore, a strongly connected graph is a directed graph that has a directed path from every node to every other one.

Consider the directed graph of Fig. 4.1, which has no underlying spanning tree. The processing speed of each node is 20 tasks per time step (i.e. $C_i(k) = 20,\ \forall i$). Let $J(k) = [J_i(k)] = [5\ 60\ 8\ 2]$, where the ith element represents the incoming rate of loads at the ith node. Note that the overall incoming load rate of the whole system is less than the overall system's processing rate $\sum_i C_i(k) > \sum_i J_i(k)$, a necessary condition for successful load balancing.

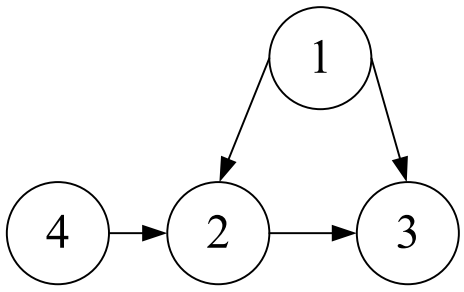


Figure 4.1: An example of a directed network topology that contains no spanning tree.

Figure 4.2 shows the dynamics of the queues in the distributed system. As expected from the topology, the system becomes unstable, i.e. as $k \to \infty$ nodes 1 and 4 become idle with queue lengths of 0, while $x_3(k) \to \infty$. This example highlights the importance of under-

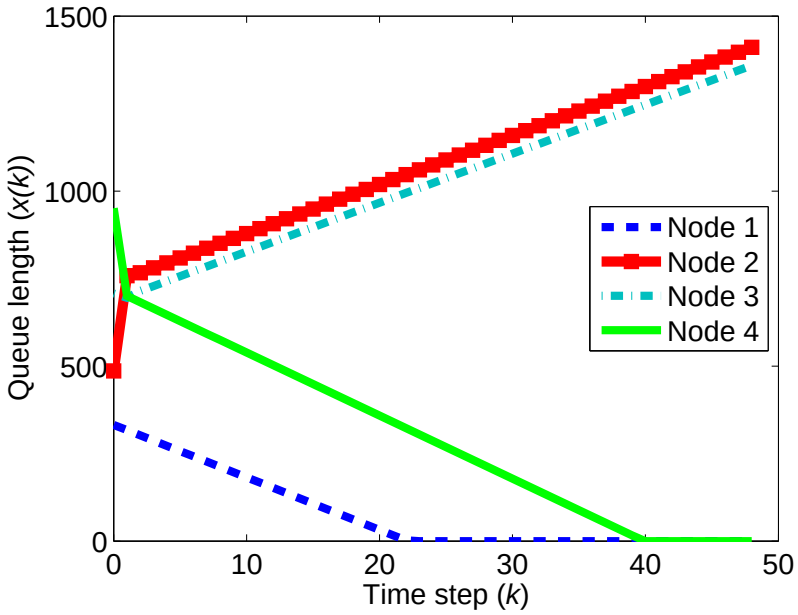


Figure 4.2: Queue dynamics for distributed load balancing over the network of Fig. 4.1.

standing and characterizing the impact of graph topology on distributed load balancing, one of the goals of this thesis.

Since a mathematical analysis of the impact of graph topology on load balancing is considerably challenging, we make a number of simplifying assumptions for the sake of mathematical derivations. We assume a one-time arrival of loads, i.e. $J_i(k) = 0$ for $k \neq 0$ and $\forall i$. We also assume that no tasks will leave the system, i.e. $C_i(k) = 0, \ \forall i$. These assumptions allow us to solely analyze the impact of the underlying graph topology on distributed load balancing. In other words, given the initial loads of the nodes, we seek to understand the conditions under which all the nodes will have a "fair" share of the overall load after a number of load balancing instants. Let L represent the total load of the network, which is given by: $L = \sum_{i=1}^{n} x_i(0)$. Since all the nodes are assumed to have the same processing speed, the optimum load balancing should result in the load of each node

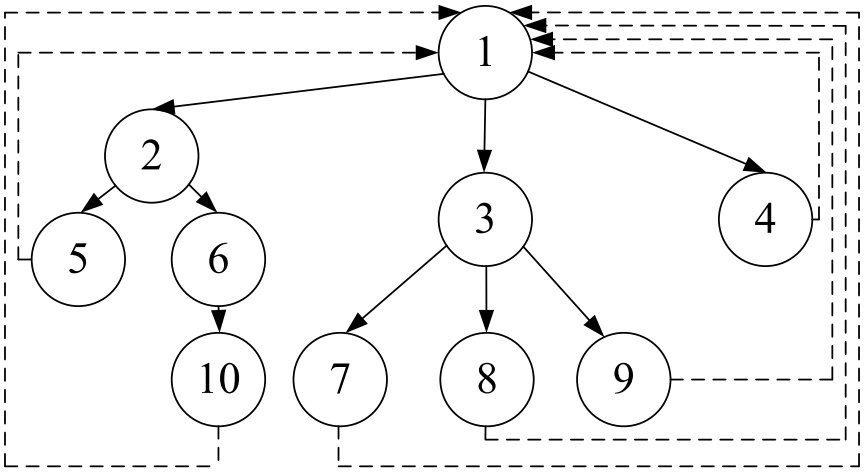


Figure 4.3: An example of a network topology with an underlying spanning tree (solid arrows only) and a network topology with an underlying strongly-connected graph (dashed and solid arrows).

approaching the average of the system ϕ given by:

$$\phi = \frac{1}{n}\sum_{i=1}^{n} x_i(0) = \frac{L}{n}. \tag{4.9}$$

We say that the load balancing algorithm converges if $\lim_{k\to\infty} x_i(k) = f_i$ for all i, where f_i is a nonnegative constant. Similarly, we define *balanced state* as the state in which all the queue lengths converge to the same value ϕ.

Consider the graph topology that is formed by only considering the solid arrows of Fig. 4.3. This corresponds to a directed network of ten nodes with an underlying spanning tree. The distributed load balancing algorithm is executed according to (4.1)-(4.8) with $J_i(k) = 0$ for $k \neq 0$ and $C_i(k) = 0$. Initially, there is a total of 5000 tasks, all located at node 1. Fig. 4.4 shows the dynamics of the queues. It can be seen that a balanced state is not reached since the leaf nodes have no way of distributing their excess loads.

By adding the additional links, represented by the dashed lines in Fig. 4.3, a strongly connected graph is formed. Figure 4.5 shows the dynamics of the queues for load balancing over this graph. It can be seen that a balanced state of $\phi = 500$ is reached. In the

next section, we show that having an underlying strongly connected graph is a sufficient condition for reaching a balanced state asymptotically.

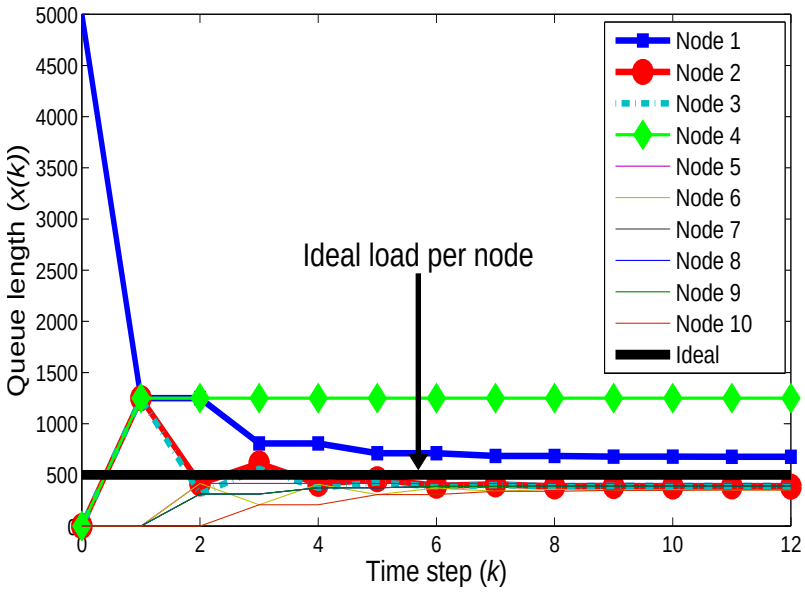


Figure 4.4: Queue dynamics for distributed load balancing over the graph topology of Fig. 4.3 (considering only the solid arrows). The queues cannot reach a balanced state even though there is an underlying spanning tree.

The initial distribution of the loads also plays a key role in reaching the balanced state. Consider the topology shown in Fig. 4.6, which has a spanning tree. It can be confirmed that for the initial distribution $x(0) = [332\ 486\ 707\ 951]^T$, there will be no convergence to a balanced state as the final values are $[332\ 596.5\ 596.5\ 951]^T$. Consider the same graph but with the following initial distribution $x(0) = [951\ 707\ 486\ 332]^T$. As seen from Fig. 4.7, the system reaches the balanced state. This example shows the impact of the initial load distribution on the convergence to the balanced state. In the next section, we will formally prove that for any initial load distribution, having a strongly connected topology is a sufficient condition to guarantee that distributed load balancing according to Eq. (4.1) will reach the balanced state.

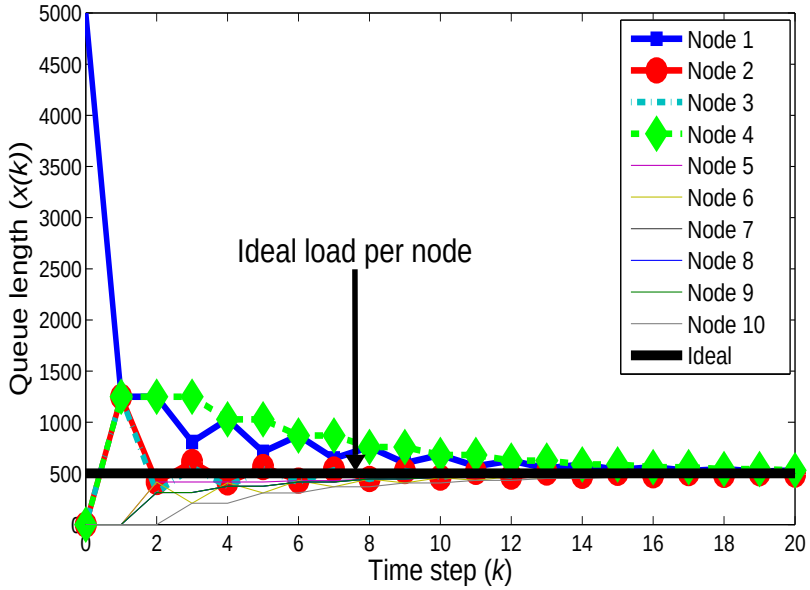


Figure 4.5: Queue dynamics for distributed load balancing over the strongly connected graph of Fig. 4.3 (considering both solid and dashed arrows). The queues reach a balanced state.

4.4 Convergence to a Balanced State

In [16] it was shown that a distributed load balancing algorithm converges to a balanced state if it complies with a number of conditions. While those conditions were described with an undirected graph in mind, they can be easily extended to distributed load balancing

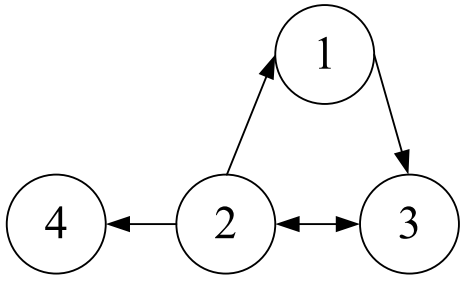


Figure 4.6: An example of a directed network topology that contains a spanning tree.

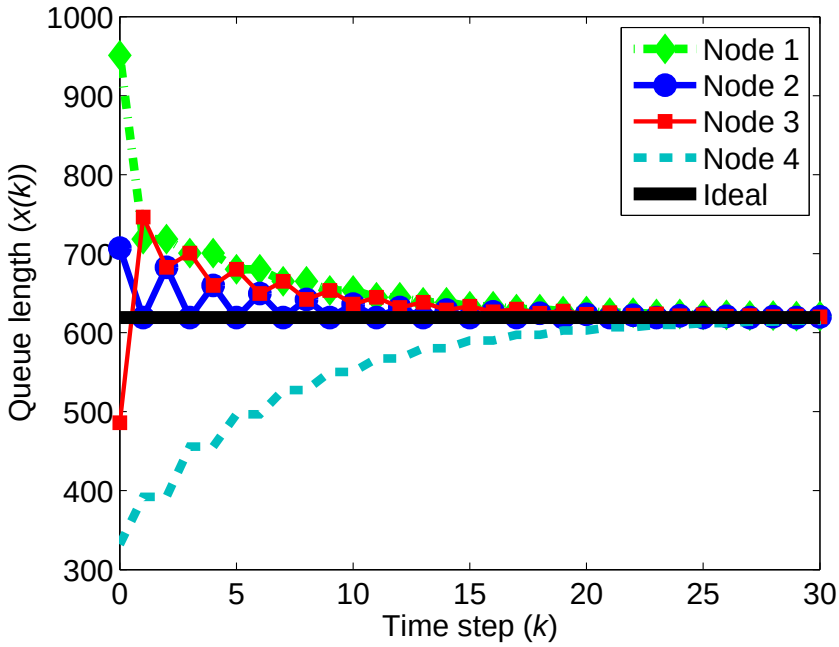


Figure 4.7: Queue dynamics for distributed load balancing over the network of Fig. 4.6, which is not strongly connected but has a spanning tree. The distributed system reaches the balanced state for the initial distribution of $x(0) = [951\ 707\ 486\ 332]^T$.

over directed graphs. In this part we show, following a similar approach to [16], that distributed load balancing according to (4.1)-(4.8) converges to the balanced state. First, note that our load balancing policy satisfies the following properties:

Property 1: If $x_i(k) > \text{ave}_i(k)$, there exists some $j^* \in \mathcal{N}_i$ such that $\text{ave}_i(k) > x_{j^*}(k)$ and $s_{ij^*}(k) > 0$.

Property 2: For any $j \in \mathcal{N}_i$ and any i such that $x_i(k) > x_j(k)$ and $\text{ave}_i(k) > x_j(k)$, the following can be easily confirmed:

$$x_i(k) - \sum_{l \in \mathcal{N}_i} s_{il}(k) \geq x_j(k) + s_{ij}(k). \tag{4.10}$$

If $x_i(k) \leq \text{ave}_i(k)$, Eq. (4.10) is obvious. If $x_i(k) > \text{ave}_i(k)$, the validity of this property

is based on the fact that: $L_i^{ex}(k)/\sum_{l\in\mathcal{M}_i(k)} L_{l(i)}^{ex}(k) \in [-1,0]$. To see this, note that,

$$L_i^{ex}(k) + \sum_{j\in\mathcal{N}_i} L_{j(i)}^{ex}(k) \;\; = \;\; 0.$$

Therefore,

$$L_i^{ex}(k) \leq -\sum_{j\in\mathcal{M}_i(k)} L_{j(i)}^{ex}(k),$$

and since $\sum_{j\in\mathcal{M}_i(k)} L_{j(i)}^{ex}(k) < 0$, then $L_i^{ex}(k)/\sum_{l\in\mathcal{M}_i(k)} L_{l(i)}^{ex}(k) \in [-1,0]$.

Therefore,

$$\begin{aligned} & x_i(k) - L_i^{ex}(k) - x_j(k) - s_{ij}(k) \\ = \;\; & -L_{j(i)}^{ex}(k) - \frac{L_{j(i)}^{ex}(k)}{\sum_{l\in\mathcal{M}_i(k)} L_{l(i)}^{ex}(k)} L_i^{ex}(k) \geq 0, \end{aligned}$$

which is equivalent to

$$x_i(k) - \sum_{l\in\mathcal{N}_i} s_{il}(k) \geq x_j(k) + s_{ij}(k).$$

This property implies that load balancing of node i cannot create a load imbalance in the receiving nodes $j \in \mathcal{N}_i$, i.e. the load sent from i cannot make a receiving node j become overloaded with respect to the neighborhood average of i. Notice, however, that a node $j \in \mathcal{N}_i$ can still become overloaded with respect to the local average of node i, since node i does not necessarily have the information of or control over the loads that other nodes may send to j.

Let $m(k)$ be defined as: $m(k) \triangleq \min_i x_i(k)$.

Lemma 1: There exists some $\beta \in (0,1)$ such that

$$x_i(k+1) \geq m(k) + \beta(x_i(k) - m(k)), \;\; \forall i \in N. \tag{4.11}$$

Proof. We will follow an approach similar to the one in [16]. Without loss of generality, fix a processor i and a time step k. Also consider the set $\mathcal{W}_i(k) = \{j | j \in \mathcal{N}_i, x_i(k) > x_j(k)\}$ and denote its cardinality as: $|\mathcal{W}_i(k)|$. From (4.1) and (4.10), we have:

$$x_i(k+1) \geq x_j(k) + s_{ij}(k) \; \forall j \in \mathcal{W}_i(k).$$

Adding over all $j \in \mathcal{W}_i(k)$:

$$|\mathcal{W}_i(k)| \; x_i(k+1) \geq \sum_{j \in \mathcal{W}_i(k)} x_j(k) + \sum_{j \in \mathcal{W}_i(k)} s_{ij}(k). \tag{4.12}$$

By noting that $s_{ij}(k) = 0$ if $j \notin \mathcal{W}_i(k)$, we will have:

$$\sum_{j \in \mathcal{W}_i(k)} s_{ij}(k) = \sum_{j \in N_i} s_{ij}(k) \geq x_i(k) - x_i(k+1). \tag{4.13}$$

Combining (4.12) and (4.13) will then result in:

$$|\mathcal{W}_i(k)| \; x_i(k+1) \geq \sum_{j \in \mathcal{W}_i(k)} x_j(k) + x_i(k) - x_i(k+1),$$

which is equivalent to:

$$\begin{aligned} x_i(k+1) &\geq \frac{|\mathcal{W}_i(k)|}{|\mathcal{W}_i(k)|+1} m(k) + \frac{1}{|\mathcal{W}_i(k)|+1} x_i(k) \\ &= m(k) + \frac{1}{|\mathcal{W}_i(k)|+1}(x_i(k) - m(k)) \\ &\geq m(k) + \frac{1}{n}(x_i(k) - m(k)), \end{aligned}$$

which proves the inequality with $\beta = \frac{1}{n}$. □

Consequently, we will have the following lemma:

Lemma 2: The sequence $m(k)$ is upper bounded by L and is nondecreasing. Therefore it converges.

Proof. From (4.11) we can easily see that $x_i(k+1) \geq m(k)$, resulting in $m(k+1) \geq m(k)$. Since $J_i(k) = 0$ for $k \neq 0$, $m(k) \leq L$. Therefore it converges. □

In [16], the following lemma is proved which establishes a lower bound on the queue length of any node j that can be reached from node i by traversing l edges. Since our load balancing algorithm satisfies properties 1 and 2, we will have the following lemma:

Lemma 3: Consider node i. For any node j that can be reached from i by traversing l edges, and for any $k \geq k_0 + 3ln$, we have

$$x_j(k) \geq m(k_0) + (\eta\beta^{k-k_0})^l(x_i(k_0) - m(k_0)), \tag{4.14}$$

where η is a nonnegative real number and β is as defined in Lemma 1.

Proof. See [16]. □

Using the previous lemmas, we can extend the convergence proof of [16] to the following:

Theorem 1: (Convergence of the load balancing policy to the balanced state) Consider the algorithm described in Eq. (4.1). If the graph G is strongly connected, then

$$\lim_{k\to\infty} x_i(k) = L/n. \qquad \forall i \in N \tag{4.15}$$

Proof. Consider a strongly connected graph. Then for a given node i, every other node is at a distance of at most $(n-1)$ from i. We can apply (4.14) and follow the proof in [16] to conclude that the difference between the highest load and the minimum load of the system $(\max_i x_i(k) - m(k))$ has to converge to 0. From Lemma 2 we have that $m(k)$ converges to a constant c. Therefore, $\lim_{k\to\infty} x_j(k) = c$ for all j. Since $\sum_{i=1}^{n} x_i(k) = L$, we have $c = L/n$. □

4.5 On the Necessity of Having a Strongly Connected Graph for Achieving a Balanced State

In this section we will give a brief qualitative discussion regarding how necessary it is for the underlying graph of the network to be strongly connected to achieve the balanced state for any initial distribution. For illustrative purposes, consider the topology in Fig. 4.3 (solid arrows only). As we saw in the previous sections, distributed load balancing over this topology does not converge to the balanced state. However, when adding the links represented by dashed arrows, the graph becomes strongly connected and hence it converges.

The effect of having a strongly connected graph is that it guarantees that when a node is overloaded, there will be a directed path to every other node in the system to which it can send its excess load. It may appear that, to reach the balanced state, it is sufficient to let the leaf nodes (7, 8, 9 and 10) have a directed link to some other nodes in the system, without having to make the graph strongly connected. However, this is not sufficient because a node that is not a leaf node can still become overloaded during the load balancing process since load balancing is done in a distributed manner. On the other hand, for some distributions, it is possible to reach a balanced state over graphs that are not strongly connected. However, as it is required to achieve a balanced state for any initial distribution, having a strongly connected graph becomes more crucial.

Chapter 5

A Consensus Approach to Load Balancing

In this chapter we propose two new load balancing strategies that make use of consensus approaches. We first propose Informed Load Balancing (I-LB), which allows the nodes to agree on the global fair load before starting to do the actual load balancing. We then study Consensus-Based Load Balancing (C-LB) in which the nodes follow a typical consensus algorithm to do load balancing.

5.1 Distributed Load Balancing with a priori knowledge of the Balanced State (I-LB)

In the distributed load balancing algorithm of the previous chapters, the nodes have to constantly distribute their "perceived" extra loads in order to reach a balanced state. Since the loads can have very large sizes, this can result in a considerable use of the available bandwidth. If the nodes could first reach an agreement over the global balanced state, it can

potentially reduce the overall time to reach the balanced state and the overall bandwidth usage. In this section we propose Informed Load Balancing (I-LB), a modification to the Original LB algorithm (O-LB) of the previous sections. The main idea of I-LB is to let the nodes exchange information about their queue lengths and reach an agreement over the global average before starting the redistribution of actual tasks. After reaching consensus over the global average, the nodes start exchanging tasks by comparing their own load with the global average (ϕ) instead of the local average (ave$_i(k)$). I-LB has the potential of reducing unnecessary transmissions and as a result the overall bandwidth usage. In this part, we compare the performance of I-LB with the O-LB approach and explore the underlying tradeoffs.

5.1.1 Discrete-Time Consensus Problem

In consensus problems, a group of nodes try to reach an agreement over a certain value (average of the initial queue lengths in our case). They exchange information with their neighboring agents and update their values according to a given update protocol. Define $y_i(k)$ as the status of node i at the kth instant of the consensus process. We have $y_i(0) = x_i(0),\ \forall i \in N$. The network is said to be in consensus if $y_i = y_j$ for all i, j. When each $y_i = \frac{1}{n}\Sigma_j y_j(0)$, the team has reached average consensus [21].

5.1.2 Consensus over the Global Average and Informed Load Balancing

Let A and L represent the adjacency matrix of the underlying graph and its Laplacian respectively, as defined in Section 3.1. Furthermore, let $y(k)$ be the information vector $y(k) = [y_1(k)...y_n(k)]^T$. The discrete-time consensus protocol will then be [21]:

$$y(k+1) = (I - \epsilon L)y(k), \tag{5.1}$$

where $\epsilon \in (0, \frac{1}{\max_i l_{ii}})$ and I represents the identity matrix.

In our case, it is desirable that the network reaches average consensus which corresponds to the system average (ϕ). As explained in Chapter 3, the authors in [21] and [23], proved that (5.1) can achieve average consensus if the graph is balanced and strongly connected. Larger values of ϵ can furthermore increase the convergence rate. Since we assumed that queue lengths were exchanged over undirected graphs in the previous section, we assume the same here for fair comparison. Once the nodes reach consensus and switch to redistributing the tasks, we take the graph over which they exchange the tasks to be directed.

Once consensus is reached, the redistribution of tasks starts. The original algorithm proposed in Chapter 4 needs to be modified to allow comparisons with the global estimated average ϕ, instead of the local average ave_i. The modifications to the original algorithm are as follows: $L_i^{ex}(k) = x_i(k) - \phi$, $L_{j(i)}^{ex}(k) = x_j(k) - \phi$ and, $\mathcal{M}_i(k) = \{j | j \in \mathcal{N}_i,\ \phi > x_j(k)\}$. Eqs. (4.6) and (4.7) remain unchanged. In practice, each node has to switch to exchanging loads after it senses that consensus is reached. In order to do so, it could monitor its value ($y_i(k)$) and those of its neighbors and declare that consensus is reached if those values do not change over a given time. Furthermore, the estimate of a node of ϕ may not be exact after it determines that consensus is reached.

5.1.3 Underlying Tradeoffs Between O-LB and I-LB

In this section we explore the underlying tradeoffs between O-LB and I-LB in terms of speed of convergence, bandwidth usage and performance guarantees. For I-LB, we take $\epsilon = \frac{1}{1.1\ \max_i l_{ii}}$, which will guarantee convergence. We also assume that the nodes can detect accurately when the consensus state is reached in order to proceed to redistributing loads. Consider the undirected path network of Fig. 5.1, with an initial load distribution of $x(0) = [707\ 486\ 332\ 951]^T$. If no information exchange is done, 49 load balancing steps

are required to reach the balanced state whereas for I-LB, 23 time steps are first required to reach consensus followed by 2 load balancing steps.

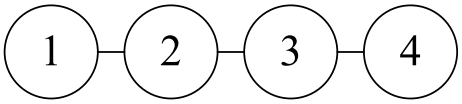


Figure 5.1: Network topology of a path network. I-LB reaches a balanced state considerably faster for an initial load distribution $x(0) = [707\ 486\ 332\ 951]^T$.

I-LB, however, does not always take fewer iterations than O-LB. As an example, consider the graph of Fig. 4.1 with the directed links replaced by undirected ones. Let the initial load distribution be $x(0) = [707\ 486\ 332\ 951]^T$. If O-LB is used, the system reaches the balanced state at time $k = 15$. For I-LB however, it takes 21 time steps to reach average consensus and 2 load balancing steps afterwards to reach the balanced state.

Figures 5.3 and 5.4 show a comparison of O-LB and I-LB for various configurations of an undirected network with 4 nodes and initial load $x(0) = [707\ 486\ 332\ 951]^T$. The studied configurations are shown in Fig. 5.2. The two approaches (O-LB and I-LB) are compared in terms of speed of convergence (number of time steps required to reach the balanced state) as well as bandwidth usage. The figures also indicate the second smallest eigenvalue of the underlying graph Laplacian as an indicator of the connectivity of each graph. The larger the second smallest eigenvalue is, the higher the graph connectivity will be. In Fig. 5.3, the bars representing I-LB include the time steps required to reach average consensus plus the actual load balancing steps required to get to the balanced state. By comparing the last two rows, it can be seen that for graphs with lower connectivity, I-LB can perform considerably better than O-LB in terms of the total time steps required to reach the balanced state. However, as the graph connectivity increases, O-LB can outperform I-LB.

Next, we compare the bandwidth usage of O-LB and I-LB. By bandwidth usage we mean the total number of packets that are transmitted in the network. We assume that each

unit of task requires 10 times the number of packets required for transmitting the queue length information of a node. Figure 5.4 shows the corresponding bandwidth usage of I-LB and O-LB for the initial distribution $x(0) = [707\ 486\ 332\ 951]^T$ and various configurations of an undirected network with 4 nodes. In most cases I-LB results in a better bandwidth usage. The difference will be more notable as the ratio of the size of task packets to information packets becomes larger. It should be noted that the second column of Fig. 5.3 and 5.4 corresponds to the graph of Fig. 5.1 while the first column is for the same graph but with a different ordering of the nodes. As a result, the nodes experience different loads in their neighborhood. It can be seen that this results in different performances, which highlights the impact of the initial distribution of the loads over the network on the overall behavior.

5.1.4 Lack of Asymptotic Performance Guarantee for I-LB

While I-LB has the potential of reducing the time and/or bandwidth required for reaching the balanced state, it lacks asymptotic performance guarantee. In Section 4.4, we showed that load balancing according to (4.1)-(4.8) and over a strongly connected graph can guarantee convergence to the balanced state. For I-LB, however, this is not the case. We show this with a counterexample. Consider the path network of Fig. 5.1, but with a different initial distribution $x(0) = [20\ 19\ 20\ 17]^T$. If no information exchange is done, i.e. O-LB is performed, the system reaches the balanced state, as proved by Theorem 1 (see Fig. 5.5). However, for I-LB, although the system reaches consensus, it is not able to reach the balanced state as seen in Fig. 5.6. To understand this, consider node 1 and 2. At $k = 0$, node 2 is already balanced but node 1 is still overloaded and can only balance itself with node 2. Since node 2 already reached the global average ϕ, it is not a candidate to receive any load, which results in the system not reaching the balanced state. Intuitively, we can see that as long as an overloaded node has a neighbor whose queue is below the global average, it can still reach the balanced state. The initial distribution of the loads, therefore,

plays a key role in the asymptotic behavior of I-LB.

5.2 Consensus-Based Load Balancing over Directed Graphs (C-LB)

In this section we consider Consensus-Based Load Balancing (C-LB), which is a protocol where at every time step each node i keeps the same fraction $(1 - l_{ii}\epsilon)$ of its total load, where l_{ii} is the ith diagonal element of the graph Laplacian and $\epsilon \in (0, \frac{1}{\max_i l_{ii}})$. The rest of the load of the node gets equally partitioned among all of its neighbors. The queue dynamics for the ith node will then be:

$$x_i(k+1) = (1 - l_{ii}\epsilon)x_i(k) + \epsilon \sum_{j \in \{l | i \in \mathcal{N}_l\}} x_j(k). \tag{5.2}$$

It can be easily verified that the matrix form for the system is given by:

$$x(k+1) = (I - \epsilon L)x(k). \tag{5.3}$$

A similar protocol was proposed in [26] for undirected graphs. We are interested in the nodes reaching the balanced state, which is equivalent to the consensus protocol of Eq. (5.3) achieving average-consensus. As indicated in the previous section, the necessary and sufficient condition for this is for the underlying graph to be strongly connected and balanced.

5.2.1 Underlying Tradeoffs Between O-LB and C-LB

One of the motivations for considering C-LB is its ease of analysis. As shown above, the consensus protocol can be written in matrix form, for which the convergence properties are well known, as shown in Chapter 3. As Eq. (5.2) shows, C-LB does not take into

account the queue lengths of its neighbors when doing load balancing. Hence, it also does not take into account whether a neighboring node is already locally overloaded or not. As a result, it is expected that C-LB will perform worse than O-LB. C-LB has the advantage that the nodes only need to know the number their of neighbors. It should also be noted that the graph conditions for reaching the balanced state are more restrictive in C-LB than in O-LB. To reach the balanced state in C-LB, the graph should be balanced in addition to strongly connected.

Figures 5.7 shows a comparison of O-LB and C-LB for the same configurations of an undirected network with 4 nodes used for Figs. (5.3) and (5.4). The initial load is $x(0) = [707\ 486\ 332\ 951]^T$. The two approaches are compared in terms of speed of convergence (number of time steps required to reach the balanced state). As the graph connectivity increases, O-LB performs better than C-LB. This is due to the fact that the higher the connectivity, the closer the local averages (ave_i) will be to the real average (ϕ) and as a result, less exchanges of load will be necessary before the balanced state is reached. There are cases in which C-LB performs considerably worse than O-LB for example, for the seventh pair of columns C-LB required more than 100 steps to reach the balanced state whereas O-LB only required 9. For graphs with lower connectivity, C-LB can outperform O-LB in terms of the total time steps required to reach the balanced state. However, it should be noted that in all cases O-LB reaches a small ball of size δ around the balanced state very quickly, but extra steps are required where only a very small amount of tasks are exchanged until the exact value of the balanced is reached.

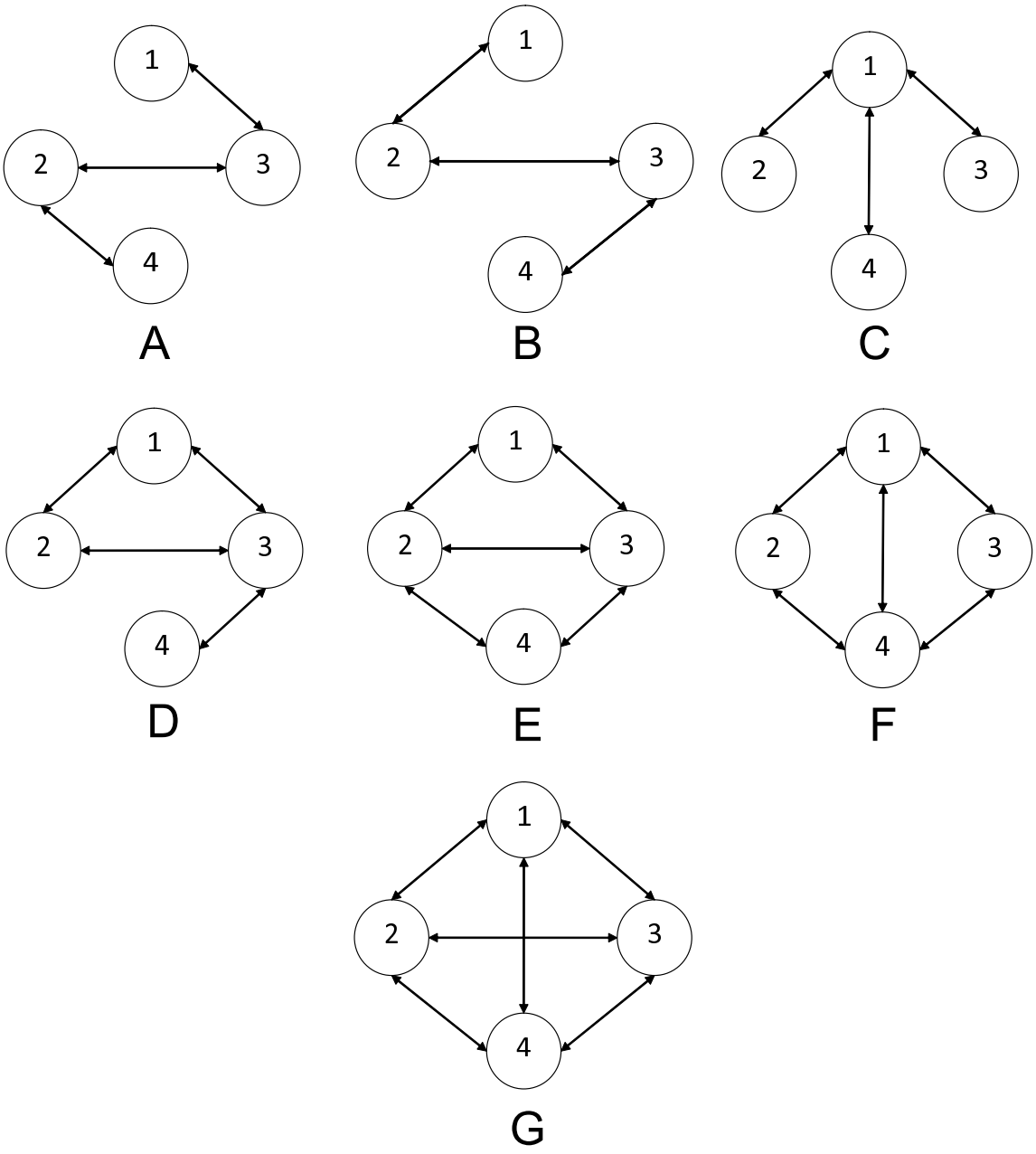
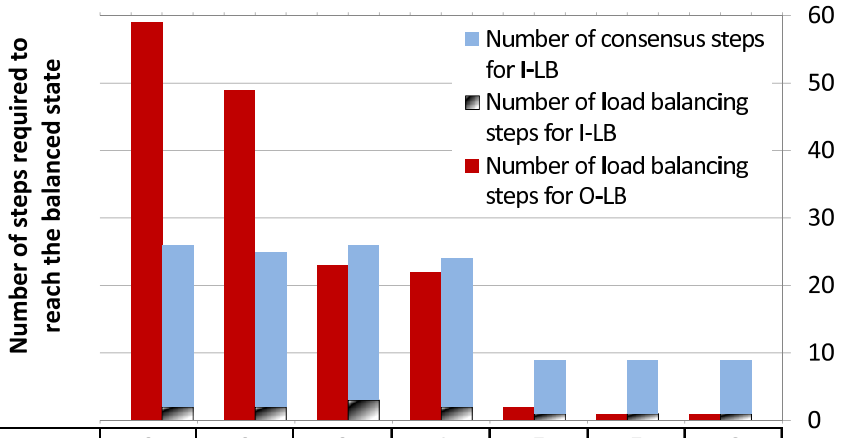
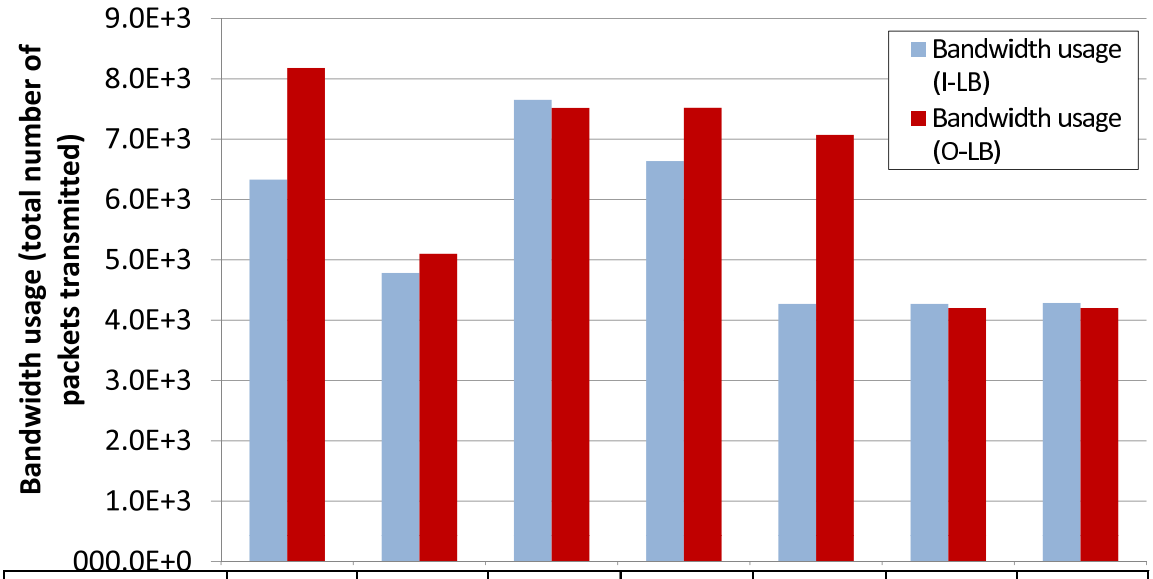


Figure 5.2: Graph configurations studied for the comparison of O-LB and I-LB.



Number of links	3	3	3	4	5	5	6
Second smallest eigenvalue of L	0.59	0.59	1.00	1.00	2.00	2.00	4.00
Topology	A	B	C	D	E	F	G
Load balancing steps for I-LB	2	2	3	2	1	1	1
Consensus steps for I-LB	24	23	23	22	8	8	8
Total steps (Consensus + load balancing) for I-LB	26	25	26	24	9	9	9
Load balancing steps for O-LB	59	49	23	22	2	1	1

Figure 5.3: Comparison of the time steps required to reach the balanced state for various undirected network topologies using O-LB and I-LB with 4 nodes (see Fig. 5.1).



Number of links	3	3	3	4	5	5	6
Second smallest eigenvalue of L	0.59	0.59	1.00	1.00	2.00	2.00	4.00
Topology	A	B	C	D	E	F	G
Bandwidth usage **(I-LB)**	**6.3E+3**	**4.8E+3**	**7.7E+3**	**6.6E+3**	**4.3E+3**	**4.3E+3**	**4.3E+3**
Bandwidth usage **(O-LB)**	**8.2E+3**	**5.1E+3**	**7.5E+3**	**7.5E+3**	**7.1E+3**	**4.2E+3**	**4.2E+3**

Figure 5.4: Comparison of the bandwidth usage (total number of transmitted packets) required to reach the balanced state for various undirected network topologies with 4 nodes (see Fig. 5.1).

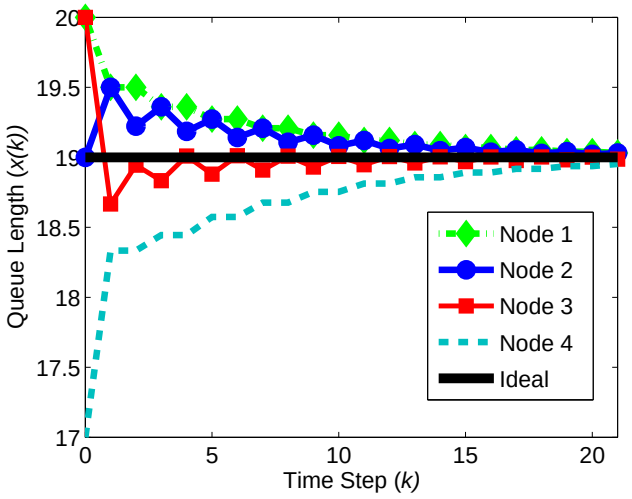


Figure 5.5: Queue dynamics for O-LB over the network of Fig. 5.1 and with $x(0) = [20\ 19\ 20\ 17]^T$. The queues reach a balanced state.

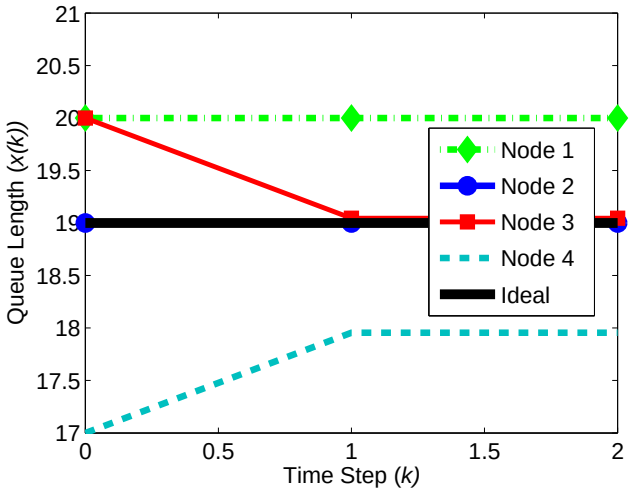
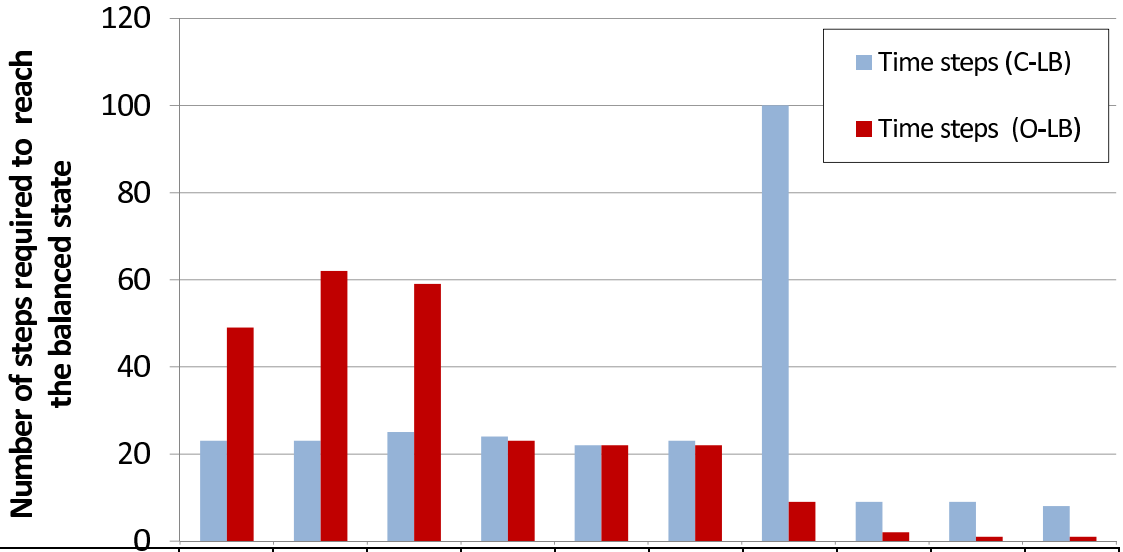


Figure 5.6: Queue dynamics for I-LB over the network of Fig. 5.1 and with $x(0) = [20\ 19\ 20\ 17]^T$. The queues cannot reach the balanced state.



Number of links	3	3	3	3	4	4	4	5	5	6
Second smallest eigenvalue of L	0.59	0.59	0.59	1.00	1.00	1.00	2.00	2.00	2.00	4.00
Time steps (C-LB)	**23**	**23**	**25**	**24**	**22**	**23**	**100**	**9**	**9**	**8**
Time steps (O-LB)	**49**	**62**	**59**	**23**	**22**	**22**	**9**	**2**	**1**	**1**

Figure 5.7: Comparison of the time steps required to reach the balanced state for various network topologies using O-LB and C-LB with 4 nodes.

Chapter 6

Conclusions and Further Extensions

In this thesis we considered the problem of distributed load balancing over a directed graph that is not fully connected. We studied the impact of network topology on the stability and balance of distributed computing. For the case of a one-time arrival of the loads, we showed that load balancing over a strongly connected graph reaches a balanced state, independent of the initial load distribution, while having a spanning tree is not a sufficient condition for reaching a balanced state. We furthermore proposed Informed Load Balancing (I-LB), an approach in which the nodes first reach an agreement over the global balanced state before proceeding to redistribute their tasks. We explored the underlying tradeoffs between I-LB and the Original Load Balancing (O-LB) approach. While I-LB lacks asymptotic performance guarantees of O-LB, it can increase the speed of convergence and/or reduce the bandwidth usage especially for low-connectivity graphs. Furthermore, we compared the performance of the proposed framework with that of a purely consensus-based approach (C-LB) with that of O-LB. C-LB is more restrictive than O-LB in terms of the connectivity requirements for reaching the balanced state. In addition to strong connectivity of the graph, C-LB requires the graph to be balanced. We also observed that O-LB can outperform C-LB in graphs with high connectivity. Even for low connectivity graphs, O-LB can reach a small ball of size δ around the balanced state

very quickly.

Further extensions could include considering incoming and outgoing loads as well as communication delays and heterogeneous processing speeds. Characterizing the speed of convergence is also a possible future extension. Furthermore, a switched linear system approach to the load balancing dynamics of Eq. (4.1) could be beneficial for analyzing convergence rate and stability.

References

[1] S. Dhakal, "Load balancing in communication constrained distributed systems," Ph.D. dissertation, University of New Mexico, 2006.

[2] M. Dobber, G. Koole, and R. van der Mei, "Dynamic load balancing experiments in a grid," in *IEEE International Symposium on Cluster Computing*, vol. 2, May 2005, pp. 1063–1070.

[3] S. Dhakal, M. Hayat, J. Pezoa, C. Yang, and D. Bader, "Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 485–497, April 2007.

[4] S. Dhakal, B. Paskaleva, M. Hayat, E. Schamiloglu, and C. Abdallah, "Dynamical discrete-time load balancing in distributed systems in the presence of time delays," in *Proceedings of 42nd IEEE Conference on Decision and Control*, vol. 5, Dec. 2003, pp. 5128–5134.

[5] J. Chiasson, Z. Tang, J. Ghanem, C. Abdallah, J. Birdwell, M. Hayat, and H. Jerez, "The effect of time delays on the stability of load balancing algorithms for parallel computations," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 6, pp. 932–942, Nov. 2005.

[6] S. Dhakal, "Load balancing in delay-limited distributed systems," Master's thesis, University of New Mexico, 2003.

[7] A. Anagnostopoulos, A. Kirsch, and E. Upfal, "Stability and efficiency of a random local load balancing protocol," in *Proceedings. 44th Annual IEEE Symposium on Foundations of Computer Science, 2003*, Oct. 2003, pp. 472–481.

[8] M. Franceschelli, A. Giua, and C. Seatzu, "Load balancing on networks with gossip-based distributed algorithms," in *Proceedings of the 46th IEEE Conference on Decision and Control*, Dec. 2007, pp. 500–505.

[9] A. Kashyap, T. Basar, and R. Srikant, "Consensus with quantized information updates," in *45th IEEE Conference on Decision and Control*, Dec. 2006, pp. 2728–2733.

[10] B. Joshi, S. Hosseini, and K. Vairavan, "Stability analysis of a load balancing algorithm," in *Proceedings of the Twenty-Eighth Southeastern Symposium on System Theory*, March-April 1996, pp. 412–415.

[11] W. C. Jakes, *Microwave Mobile Communications*. New York: Wiley-IEEE Press, 1974.

[12] T. Casavant and J. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Transactions on Software Engineering*, vol. 14, no. 2, pp. 141–154, Feb. 1988.

[13] R. Shah, B. Veeravalli, and M. Misra, "On the design of adaptive and decentralized load balancing algorithms with load estimation for computational grid environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1675–1686, Dec. 2007.

[14] A. Cortes, A. Ripoll, M. Senar, and E. Luque, "Performance comparison of dynamic load-balancing strategies for distributed computing," in *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, 1999.

[15] M.-C. Huang, S. Hosseini, and K. Vairavan, "A receiver-initiated load balancing method in computer networks using fuzzy logic control," in *IEEE Global Telecommunications Conference, GLOBECOM 2003.*, vol. 7, Dec. 2003, pp. 4028–4033 vol.7.

[16] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, New Jersey: Prentice-Hall, 1989, ch. Partially Asynchronous Iterative Methods.

[17] M. Hayat, S. Dhakal, C. Abdallah, J. Chiasson, and J. Birdwell, *Dynamic time delay models for load balancing. Part II: Stochastic analysis of the effect of delay uncertainty*, ser. Advances in Time Delay Systems, Springer Series on Lecture Notes in Computational Science and Engineering, K. Gu and S.-I. Niculescu, Eds. Springer: Berlin, 2004, vol. 38.

[18] J. Liu, X. Jin, and Y. Wang, "Agent-based load balancing on homogeneous minigrids: macroscopic modeling and characterization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 7, pp. 586–598, July 2005.

[19] J. Piovesan, C. Abdallah, and H. Tanner, "A hybrid framework for resource allocation among multiple agents moving on discrete environments," *Special Issue on Collective Behavior and Control of Multi-Agent Systems of the Asian Journal of Control*, March 2008.

[20] J. Piovesan, "Multiagent systems with hybrid interacting dynamics," Ph.D. dissertation, University of New Mexico, 2009.

[21] D. Kingston and R. Beard, "Discrete-time average-consensus under switching network topologies," in *Proceedings of the 2006 American Control Conference*, June 2006, pp. 3551–3556.

[22] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *Proceedings of the 2005 American Control Conference*, vol. 3, June 2005, pp. 1859–1864.

[23] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sept. 2004.

[24] Y. Mostofi, "Binary consensus with gaussian communication noise: A probabilistic approach," in *Proceedings of the 46th IEEE Conference on Decision and Control, 2007*, Dec. 2007, pp. 2528–2533.

[25] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," in *Proceedings of the 42nd IEEE Conference on Decision and Control, 2003.*, vol. 5, Dec. 2003, pp. 4997–5002 Vol.5.

[26] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computation*, vol. 67, no. 1, pp. 33–46, 2007.

[27] R. Diestel, *Graph Theory*. New York: Springer-Verlag, 2000, vol. 173.

[28] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[29] W. Ren and R. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[30] R. A. Horn and C. R. Johnson, *Matrix Analysis*. New York: Cambridge University Press, 1085.