The Role of Groups in Smart Camera Networks

by

Jacky Mallett

Submitted to the Program of Media Arts and Sciences,
School of Architecture,
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2006

Author_____

Program of Media Arts and Sciences
October 7, 2005

Certified by_____

V. Michael Bove, Jr.
Principal Research Scientist
MIT Media Laboratory
Thesis Supervisor

Accepted by_____

Andrew B. Lippman
Graduate Officer
Departmental Committee on Graduate Students

The Role of Groups in Smart Camera Networks

by

Jacky Mallett

Submitted to the Program of Media Arts and Sciences,
School of Architecture,
on October 7, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Recent research in sensor networks has made it possible to deploy networks of sensors with significant local processing. These sensor networks are revolutionising information collection and processing in many different environments. Often the amount of local data produced by these devices, and their sheer number, makes centralised data processing infeasible. Smart camera networks represent a particular challenge in this regard, partly because of the amount of data produced by each camera, but also because many high level vision algorithms require data from more than one camera.

Many distributed algorithms exist that work locally to produce results from a collection of nodes, but as this number grows the algorithm's performance is quickly crippled by the resulting exponential increase in communication overhead. This thesis examines the limits this puts on peer-to-peer cooperation between nodes, and demonstrates how for large networks these can only be circumvented by locally formed organisations of nodes. A local group forming protocol is described that provides a method for nodes to create a bottom-up organisation based purely on local conditions. This allows the formation of a dynamic information network of cooperating nodes, in which a distributed algorithm can organise the communications of its nodes using purely local knowledge to maintain its global network performance.

Building on recent work using SIFT feature detection, this protocol is demonstrated in a network of smart cameras. Local groups with shared views are established, which allow each camera to locally determine their relative position with others in the network. The result partitions the network into groups of cameras with known visual relationships, which can then be used for further analysis.

Thesis Supervisor: V. Michael Bove, Jr.
Title: Principal Research Scientist, MIT Media Laboratory

The Role of Groups in Smart Camera Networks

by

Jacky Mallett

The following people served as readers for this thesis:

Thesis Reader_____

Ted Selker
Associate Professor
MIT Media Lab

Thesis Reader_____

Wayne Wolf
Professor of Electrical Engineering
Princeton University

Contents

1 Introduction **15**

1.1 Issues 17

1.1.1 Distributing Group Processing 17

1.1.2 Camera Networks 20

1.1.3 View Matching 21

1.2 Motivation 21

1.3 Goals and contributions 21

2 Background **23**

2.1 Introduction 23

2.2 Smart Cameras 27

2.3 Sensor Networks 28

2.4 Smart Camera Networks 30

2.5 Distributed Groups 32

2.5.1 Reliable Process Groups 32

2.5.2 Peer-to-Peer Networks 34

2.5.3 Distributed Task Management 35

2.6 Surveillance 36

2.7 Multi-view Analysis 38

3 Bottom-Up Organisation **41**

3.1 Introduction 41

3.2 The importance of Organising Groups 46

3.2.1 The limits on local communications within a Group 49
3.2.2 The Advantages of Locally formed Organisations 53
3.2.3 Changing the Requirements for Group Memberships 54

4 Local Group Forming Protocol **57**
4.1 Introduction 57
4.1.1 Protocol Elements 58
4.2 Local Group Interface 63
4.3 Forming Layered Groups 65
4.4 Examples 66
4.4.1 Group calibration of Light Levels in a distributed camera network 66
4.4.2 Tracking People or Objects 67

5 Forming Groups based on Image Matching **69**
5.1 Introduction 69
5.2 SIFT Based Image Matching - Theory 71
5.3 SIFT Implementation 72
5.4 Distributed Image Matching 74
5.4.1 Determining relative position within a Camera Grid 75
5.5 Network Analysis 78
5.5.1 General Case 79
5.5.2 The Effects of Group Organisation on Network Performance . 81

6 Conclusion **85**

7 Future Work **91**
7.1 Smart Camera Networks 91
7.2 Distributed Social Systems 92

8 Acknowledgements **95**

A Implementation **97**
A.1 Smart Camera Network Platform 97

A.2 Software Architecture 98
A.3 The Sensored Tile 99
A.4 Eye Society Robot 100
A.4.1 Camera Capability 100
A.4.2 Communication Capability 101
A.4.3 Operating System 102
A.4.4 Performance 102

List of Figures

3-1 Group Topologies 49

4-1 Message Exchange for Group Creation 59
4-2 Message Exchange for Group Info 60
4-3 Message Exchange for Group Send 61
4-4 Message Exchange for Group Leave 62
4-5 Message Exchange for Group Ending 63

5-1 Difference of Gaussian Pyramid 72
5-2 SIFT points extracted from Camera Tile Images 73
5-3 Conference Room Tile Installation 75
5-4 Local view of alignment groups in Tile Array 76
5-5 Example of camera groups in an unstructured environment 81
5-6 General limits on Organisational Size 82

A-1 View from the Mobile Robots 97
A-2 Software Architecture 99
A-3 Tile Internals 100
A-4 Eye Society Robot on overhead track 101

List of Tables

4.1 Inter-Node Group Management Protocol 57
4.2 Local Group Interface 64

Chapter 1

Introduction

The ability to organise individuals into structured task groups is regarded by sociologists as one of the distinguishing features of the human species[1]. With the recent development of widespread wireless data communication in conjunction with a plethora of cheap, portable, and variously abled computing devices, applications can now be constructed from large numbers of cooperating, heterogeneous elements. Providing organisational methods for communications within such applications has consequently become a research problem of considerable interest. Ssensor networks based on smart cameras provide one of the most demanding testbeds currently available for research into the problems of communication and organisation within such networks, in particular for applications that require significant amounts of data to be shared between local nodes.

Consider as an example, a smart camera network, where each camera is able to locally process video and communicate over a network with its fellow cameras. It is not uncommon in today's cities to have thousands of cameras networked back to a single data centre for real-time monitoring, but these installations are as a practical matter only able to monitor a small subset of the video feeds available[2]. Extracting useful data from these networks presents a considerable challenge when they encompass any significant number of cameras, both because of the sheer size of the data-streams being captured, and the computational demands of the analysis. These networks should be capable of providing far more sophisticated functions than that provided by a set

of singleton cameras reporting back to a centralised, and severely overloaded, data centre. To resolve this problem, methods need to be found that allow processing and communication to be better distributed within the network. Within camera networks in particular, advantage can be taken of locally based organisations arising from connected or overlapping views, by using peer-to-peer communications between the cameras. In fact, often by combining views from multiple cameras, more information such as depth can be extracted from a scene than can be obtained from a single camera.

Ideally it should be possible to form and destroy ad hoc associations of cameras continuously throughout the network in response to particular requests for data, with local task specific considerations being used automatically to determine which cameras participate in any given problem. However, support for such dynamic peer-to-peer task organisation is completely absent in the protocols supporting the predominantly client-server distributed applications in use today.

Peer-to-peer applications have been gaining credibility for some time, as the limitations of centralised schemes under certain conditions have become apparent. Client-server based applications have been extremely successful under the right conditions, but fail badly if the amount of data to be sent and processed by the server exceeds the capacity of either the network or the server. This is particularly likely to be the case for applications that involve data sharing between local nodes, whether because of locally relevant information and power conservation, which are the typical issues addressed in sensor network work, or the need to distribute processing and network resources more equitably as demonstrated by popular peer-to-peer file sharing applications such as Gnutella[3]. However, as the number of nodes in these applications increases, their efficiency drops dramatically as the overhead of supporting large numbers of connections at each node increases near exponentially.

In this thesis the problem of organising communications between the individual nodes within peer-to-peer applications is examined. This problem can be characterised as partially a problem of selection - for example determining which nodes can usefully share data with each other, and also one of efficiency - avoiding situations

where individual nodes are overloaded by excessive numbers of peer-to-peer connections, or data requirements. To assist applications in resolving these issues, a group forming protocol is presented that allows locally autonomous nodes to independently determine their suitability for shared tasks, and to participate in these tasks as appropriate for them. The ramifications of this approach for large scale peer-to-peer distributed applications, and the particular characteristics of the groups formed by it are also explored. This protocol is demonstrated in a smart camera network, in which cameras resolve one of the critical group calibration problems of such an environment, namely how the cameras views are physically related to each other.

1.1 Issues

1.1.1 Distributing Group Processing

Conceptually moving processing out into the device may help solve one set of problems, explicitly those involved with centralised processing of large quantities of data. As this particular problem is often otherwise intractable, local processing is often seen as offering the only solution. However intractable the original problem may be though, the issues associated with building an application distributed over a large number of independent units are also not inconsiderable. To date, most distributed applications are either built on a strict client-server basis, which is the very problem we are trying to solve; or in a very regimented fashion, where homogeneous individual units perform a small number of strictly defined operations that have been empirically determined to result in a predictable emergent outcome.

Leaving these problems momentarily aside, client-server type models for distributed applications work best when the underlying problem consists of small amounts of simple data on the client side being communicated to an overdesigned centralised point. Neither direct client to client communication, nor any large degree of behavioural heterogeneity within the clients are well supported by the client-server model.

It is this lack of peer-to-peer communication that is particularly problematic in a network of cameras, where many higher level visual processing algorithms, for example depth recovery, require input from more than one camera. Conceptually a camera network can be regarded as consisting of a set of groups of cameras within which useful information can be shared, because the cameras in the groups all have relevant information to the task in hand. These groups of cameras may have physically overlapping views or be showing related objects, the cameras themselves may be known to be in close geographic proximity (for applications such as tracking), or they may have purely logical relationships - such as all cameras monitoring town centre car parks, for an application that is attempting to advise seekers of parking spaces. Although it may be practical to provide some of this organisational structure statically, in practice many applications within such a network will need dynamic data driven, layered types of organisations, which allow them to take advantage of cross communications between some but not all nodes in the network.

An example of the importance of this approach can be found in human activity from the various forms of military organisation that have been developed over several millennia. A large body of men can be ordered to advance against an enemy by ordering each individually to advance - this would be analogous to the client-server model of computer organisation. Alternatively the same individuals can be organised into groups, sections and brigades, and orders distributed to each unit. Leaving aside the advantages to the central point of being able to differentially manoeuvre blocks of men on the battlefield according to terrain and emerging battle factors, it is also the case with the latter form that there is considerably more scope for independent adaption by and within the grouped units as the fog of war descends. [1] The level of independent adaption available to the individuals in the first form of organisation is mainly restricted to deciding which way to run.

In fact, what is most noteworthy about military organisation, from the perspective

[1] "By Method and Discipline are to be understood the marshaling of the army in its proper subdivisions, the graduations of rank among the officers, the maintenance of roads by which suppliers may reach the army, and the control of military expenditure." Sun Tzu[4]

of network organisation, is that it uses a fractal form of localised centralisation, with successive organisational layers each breaking down into one central point of command, with multiple groups beneath it, and with a very clearly defined message flow between them. The time and training it takes to establish this organisation can also be regarded, from a network communications perspective, as a way of increasing the available bandwidth during operations by performing the communications overhead of establishing the organisation outside of the main activity.

This structure has notoriously had its own problems with allowing sideways communication, but it must be recognised that it has now been in existence for several millenia without substantial alterations[4]. As a side note, it can be observed that a considerable number of military disasters can be traced to blind obedience to orders received, in combination with a breakdown, absence of, or wilful disregard of return communications from the active units to a centralised command, or to their accompanying units.

Dixon[5] provides an impressive collation of failures in military leadership in his work on the psychological causes of military incompetance. However there are many instances which could equally well be attributed to failures arising directly from the underlying communication structure, rather than the behaviour of the individuals immediately involved. An established hierarchical message passing organisation is extremely efficient at delivering messages quickly from a small number of sources to a large numbers of end nodes. It is extremely inefficient in providing return routes for information from any significant number of nodes. It is also consequently lacking in mechanisms to correct any message corruption, particularly in networks with high latency. From the perspective of data communication models, the occasional misunderstanding about which valley to charge into, is the price paid for being able to quickly mobilise the cavalry to charge at all. Later in this thesis it will be shown that these organisational issues are equally critical for distributed algorithms running in large scale sensor networks. [2]

[2]The notorious Charge of the Light Brigade directly into an emplaced set of Russian cannons at the battle of Balaclava during the Crimean War in 1854, was attributed to misunderstood com-

Large scale distributed applications are victims of the same considerations when questions of data organisation and sharing are addressed within the larger system. Strictly hierarchical systems without feedback or cross-level communications have shown a demonstrated fragility in many settings, but finding scalable methods which allow this cross-level communication to occur, especially dynamically, presents its own considerable challenges.

1.1.2 Camera Networks

The problem of distributing a task over several nodes, according to their individual ability to contribute their local data to a desired global solution, is particularly acute for large scale camera networks. Viewed solely as a data sensor cameras produce extremely dense and rich data, and many different algorithms currently exist to extract this information with greatly varying degrees of accuracy, computational demands and dependencies on local visual conditions. Smart camera networks where each camera has on-board visual processing and data communication capability have become an increasing research interest as the limitations of the current centralised camera networks have become apparent.

Such networks are not required to be of fixed position. A robot with an on-board camera for sensing can equally be regarded as a mobile smart camera. Group robotics where several cooperating small cheap robots collaborate on problems is currently an active research field, with many of the same issues.

In both cases, the problem of locally determining how a camera shares its views with other local cameras can be regarded as a fundamental group calibration problem for working on shared tasks.

munications between Lord Lucan and Lord Cardigan, the Brigade's commanding officer, on which valley the Brigade was to use for the attack.

1.1.3 View Matching

Matching views from different cameras, or from the same camera at different positions, is an important precondition for many higher level computer vision problems. A number of different approaches to the problem exist, whose individual suitability depends on a number of factors including available computation, visual environment, the presence or absence of calibrated cameras, and the types of images being analysed.

For the types of applications envisioned by this thesis, it is important that the algorithm used be capable of real-time performance, works well across a varied visual environment, and can successfully detect non-overlapping as well as overlapping views. In other words, the algorithm should have a very low false positive rate, even if this means rejecting some correct matches.

1.2 Motivation

Now that we have ubiquitous and widespread data communication, as well as a plethora of cheap, portable, and variously abled computing devices, applications involving large numbers of scattered objects have become realisable. Mass production of simple, cheap objects is straightforward; finding software methods that allow us to build applications using these components then becomes a way of moving complexity from hardware into software, where it is hopefully more tractable. It is certainly true that designing complex software presents its own challenges. However, it is equally true that there are no practical solutions at this time to some of the physical problems of complex hardware, for example the adversarial relationship between a robot's weight and its available battery power. Forming task groups of cooperating simple components is consequently high attractive as a solution to some of these problems.

1.3 Goals and contributions

The principle objective of this thesis is to demonstrate a simple and practical method of forming interest groups to work on collaborative tasks. New and significant con-

tributions include:

- A new group protocol which allows task groups to be determined purely locally using task dependent information, thereby providing a generalisable approach to distributed information applications in an environment with requirements for significant amounts of peer-to-peer information exchange.
- A new distributed algorithm for locally determining relative camera position based on shared views for large collections of cameras in real-time, using the recently discovered SIFT algorithm.
- Insight into the problems network communication issues of data flow, network latency, and message load can cause in human social and economic groups.

Chapter 2

Background

2.1 Introduction

Many network communications approaches, whether at the application level or within the network itself, are based on preplanned topologies or fixed communication paths. However, as networks, and the distributed systems they support have grown larger, there has been considerable research into adaptive protocols for purposes such as routeing, or node selection, in order to make the resulting systems more robust to local conditions.

Work in sensor networks in particular has looked at the problems of designing locally adaptive communication strategies in order to overcome problems of bandwidth, and energy conservation in such systems. Camera based sensor networks are at present allowed to ignore problems of power, but face considerable bandwidth issues owing to the amount of data represented by video streams. In this thesis the problem of establishing locally shared views in a large network of camera sensors is examined, and a solution is proposed that allows the construction of a data driven organisation of the nodes within such a network.

Normally in a data communications network, the physical topology, which is to say the design and engineering of the connections between the individual nodes of a data network, is determined partly by the physical placement of the nodes, and by their individual and combined traffic requirements. This is typically pre-planned

as part of network installation, and then manually modified in response to changes in the network as time progresses. Design and implementation of the protocols that allow these networks to function has over the last forty years allowed the creation of a planetary data network, which for the most part relies on decentralised, or nearly so algorithms to function. This decentralised approach within the network has not generally been reflected in the architecture of the distributed applications running over it.These are typically centralised on at most a few distinguished points of control.

The sheer size of visual data, combined with the need for input from multiple cameras for many high level visual processing algorithms, means that camera networks are an exemplar for distributed systems with high local traffic and processing requirements. Consequently they provide an excellent example of a distributed wide area application whose requirements for decentralised programming and control are unavoidable.

Existing vision research on large numbers of cameras is generally limited to work in a controlled, laboratory environment with a pre-defined configuration of cameras. Relationships between cameras are provided to the program or algorithm as part of the environment, and are not typically considered to be part of the problem being investigated. Usually images from these cameras are communicated to one or more centralised processors for analysis. However, owing to the high dimensionality of video data, this has often led to installations with large numbers of cameras where analysis is only performed on the output from a few selected cameras[6], and typically this selection has been pre-determined.

Hitherto, digital cameras have been individually expensive sensors, and the CPU and data communications resources required to do even elementary processing on their pictures equally so. However the relentless impact of technological progress is in the process of removing each of these barriers to wide scale deployment of cameras that are both networked, and with enough local computational power to be termed 'smart'. It is anticipated that these visual sensor networks, which are a part of a growing field of research into the problems presented by sensor networks, will be widely deployed for a variety of surveillance applications, and the already sizable

deployment of simple passive cameras certainly supports this argument.

The scenario envisioned for these visual sensor networks is demanding. It is expected that cameras will be unevenly distributed in the environment, with initial positions unknown, at least as far as precise orientation, and scene of view. Coordinates derived from GPS or manual installation may be useful for coarse localisation, but will not assist in determining the precise view of each camera, or how it relates to that of other cameras, particularly with respect to shared fields of view. There may well be too many cameras in the network to make manual calibration practical, and cameras can also be partially mobile and able to pan and tilt in order to better observe their larger environment. In time, developments in robotics can also be expected to introduce an element of greater mobility into these networks.

Benign and obvious as this scenario sounds, it poses a number of extremely challenging research problems. Perhaps the most fundamental is simply establishing the relationship between views of cameras in the network. Work in this area ranges from determining the overlap of views between calibrated and closely coupled, stereo cameras to stitching panoramic views together[7]. The precise methods used can depend greatly on the types of cameras and scenes being observed. Many high level visual algorithms require input from more than one camera observing a single scene, in order to resolve information such as depth, which requires images from two cameras with related views of the same object in order to successfully perform triangulation. Other generalised problems such as tracking an object moving through an area, similarly require shared information between multiple cameras for whom the relationship between their views is known. Because of the large number of cameras expected in these networks, or equally the large number of changing views - in the case of a smaller, but mobile, robotic camera network - centrally processing camera images in order to extract this information is regarded as impractical.

Automatically establishing these relationships between cameras in a large network may seem at first sight straightforward. In fact it is an extremely challenging problem, both in terms of the visual analysis, and in distributing the necessary information between some, but preferably not all of the cameras in the network. Most current

distributed applications are based on the client-server approach where processing is concentrated at a central point, and direct peer-to-peer communication is explicitly not supported. Distributed applications without a central point of control are a very open area of research, the issues include design of algorithms in this environment, since simply obtaining debugging information is non-problematic when multiple nodes are involved. Algorithms must demonstrate properties such as self-stabilisation[8], where it can be shown that from any initialisation point, any arbitrary configuration of the distributed algorithm on all nodes will eventually exhibit desired "legal" behaviour. Such behaviour is not allowed to include flooding the network, for example by entering a state where message production within the distributed algorithm increases exponentially, which was an issue seen frequently in first generation peer-to-peer file sharing applications. The problem that is not as well recognised as perhaps it should be, is that neither centralisation nor peer-to-peer approaches, can adequately address these issues as the number of nodes within the application increases, without some kind of larger organisation of the nodes being imposed.

Multi-view analysis is an active research topic, where a number of different methods have been proposed for determining correspondences between separate images. Determining this correspondence between uncalibrated cameras is still an open research problem, although recent research based on the SIFT algorithm developed by Lowe[9] is proving promising. This is a feature extraction algorithm with relatively good computational performance, and the extracted features are invariant to image scaling, rotation, and partially invariant to changes in illumination and camera viewpoint. Extraneous keypoints are culled by the algorithm, and each keypoint is a compact representation of the data, such that the amount of data shared between cameras when exchanging keypoints is relatively small, particularly so with respect to the size of the original picture.

Smart cameras themselves are also an active research topic as the limitations of scale with the existing heavily centralised camera surveillance networks have become apparent, and low cost embedded platforms have become more capable of supporting the required computation demands. Approaches vary from specialised designs, with

custom embedded signal processors, to more general approaches based on physically small, but computationally powerful microprocessors.

2.2 Smart Cameras

The large amount of data produced by a single digital camera makes processing information from multiple cameras computationally challenging. Although on one side of this problem network transmission and CPU processing speeds are still increasing, on the other digital cameras currently have relatively low resolutions, and this can also be expected to improve. As doubling the resolution of a camera quadruples the amount of raw data being produced, incremental hardware improvements in either network or processing speeds are unlikely to ameliorate the problem. The resulting bottleneck at the point of processing makes working with multiple video streams extremely challenging. This has been amply demonstrated by the simple surveillance camera networks deployed during the late twentieth century[2]. It is impractical to monitor all but a fraction of the resulting live feeds in real time, and searching for data in the resulting video archive is prohibitively labour intensive for anything other than information that is highly specific to time and location.

One partial solution to this problem is to move at least part of the image processing out to the camera itself, allowing advantage to be taken of the recent improvement in embedded systems by creating a smart camera with on-board processing. As Wolf[10] describes, it is already possible to perform low level processing, with a number of different algorithms such as region extraction and contour filling on these platforms.

A number of design issues surround smart cameras, and these are being actively researched as the underlying technology improves. Complete integration is seen as a desirable goal for reasons of power and inter-processor computational efficiency, but it is hard to define a good multi-purpose solution given the broad range of requirements for such systems and the large number of competing visual algorithms. The technology is of interest to a number of different fields, with varying approaches being explored, from general purpose cameras, to those highly specialised for one particular

application or algorithm such as described by Caarls[11].

Specific VLSI architectures for a RISC based smart camera implementing human activity classification algorithms are discussed by Lv[12], as well as a distributed smart camera system with two cameras and multiple processors. Bramberger[13] presents a multi-processor design for a smart camera which uses two 600MHz Texas Instruments DSP's with an XScale processor running embedded Linux used to provide system control and communication. Kogut[14] presents a prototype military mobile smart camera network, where each of the mobile nodes carries two cameras, and Mallett[15] describes a general purpose approach based on 400MHz XScale boards running Linux.

2.3 Sensor Networks

Physical developments in integrated, low-power, low-cost communication and computation devices have created a range of networked sensors of varying degrees of capability, and with it the considerable challenge of deploying and extracting data from large networks of the same. The size and complexity of the proposed networks has caused many in the field to agree that a self-organising approach is required for the numbers, capabilities and deployment profile of the devices involved, and a number of protocols and architectural approaches are currently being explored[16].

Current work in sensor networks is often concerned with issues that do not impact directly on the work here, such as economically obtaining and transmitting sensing information in the presence of energy constraints; or moving data through the network without explicit identification of data origin. General issues of organisation and information extraction however, are directly relevant, and there has been considerable work on the theoretical capacity of these networks under different assumptions. Scaglione[17] presents a useful description of the limits on information sharing in such networks, and possible strategies to overcome them.

Many of these issues are also shared in large part with data networks, and work in this area precedes the present generation of sensor networks by several decades, as

part of research, amongst other projects, into the early warning radar network over North America. Yemini's[18] paper in the 1978 Distributed Sensor Nets workshop presents an overview of the general issues that is still quite relevant, including the challenges represented by the problem of finding efficient organisational structures to perform tasks in these networks. Tilak[19] provides a more recent overview of networking issues specific to today's view of micro-sensor networks.

The recent Berkeley motes work, Hill et. al[16], has created one of the best known small scale examples of the micro-sensor based smart networks. This initially used an 178 byte sized operating system, (TinyOS) which provided support for elementary scheduling and communication capabilities. TinyOs was designed specifically as a small footprint operating system for wireless embedded sensor networks, and has now been ported to many different platforms. It relies on an event-driven execution model with modular components that allows swapping of system components to support required functionality [20].

This work has led to a succession of devices of increasing power and mobility. Sinopoli[21] provides an overview of research activity on the problem of distributed control using this platform, and propose a hierarchical approach, with a low level of control that is driven by time based interrupts, and a higher level that is event driven.

Approaches to the problem of organisation within these types of networks often take the form of a predefined structure that is then implemented by the algorithms at each node. Typically this involves one or more layers of control, with successive concentration of data and control into groups or clusters of nodes. Identifying or tasking specific nodes to a place within these clusters is usually seen as specific to the particular application, or something that can be pre-planned as part of installation.

In practice this form of approach can involve greater and lessor degrees of predestined structure. Huang and Khendry[22] propose a scheme named Layernet, which creates an initial sparse tree topology of connected nodes, and then builds out to a better connected network over time. A less explicitly organised scheme termed *Directed Diffusion* is also being proposed as a general data-centric approach to self-organising network paths in sensor networks[23]. Directed diffusion constructs a data

influenced set of indicators that nodes use locally in order to structure their immediate networking environment, based on their and their neighbours data values with respect to a shared task. Individual nodes broadcast information requests, and their neighbours set up an interest gradient which allows the requested data to be routed back to the originating node. This method allows the network to avoid explicitly identifying each node with a global identifier, although GPS co-ordinates do appear to enjoy a slightly suspicious prominence in some of the diffused data packets being propagated.

There is also considerable interest in peer-to-peer protocols, drawing on recent experience with peer-to-peer information sharing applications running over the Internet. In particular gossip-based protocols, which essentially rely on probabilistically creating random links between nodes in the network to obtain sufficient global connectivity[24], are viewed as a possible solution to some of the scaling issues of these networks. Although these approaches are undoubtably capable of creating sufficient connectivity - as much because of the six degrees of separation phenomena[25] as anything else - it is not clear that organisation, as the word is generally understood, is really the correct term here.

An explicitly hierarchical approach to the problem, in the context of controlling unmanned vehicles in a pursuit-evasion task is discussed by Kim[26]. An off-vehicle strategy planner and map builder is used to communicate control commands to the vehicles, and analyse sensor information from them. This approach does not use any peer-to-peer communication between the vehicles. A solution including a sensor network that is organised using a simple peer-to-peer supernode structure and some consideration of network transmission problems is discussed by Schenato[27] in the context of space based pursuit-evasion.

2.4 Smart Camera Networks

Most current research environments with large numbers of cameras simultaneously observing a scene are highly controlled laboratory systems, in well calibrated envi-

ronments. These include the Virtualized Reality project at Carnegie Mellon[6], or the Intelligent Environment Research Complex at the University of California, San Diego.[28]

The type of wide area camera network being described in this thesis is beginning to enjoy the term 'Visual Sensor Network' in some papers, as more attention is being given to the specific problems shared by sensor networks where the dominant sensor is a camera. Obracska[29] provides an overview of the class of problems to be expected in this field, and discusses some of the problems of managing information flow in such networks. The suggested solution is IP protocol replacements designed specifically for camera networks. However, although application level protocols are certainly needed for these networks, it is hard to see what is specific to smart cameras that would justify the cost of a network level protocol replacement, especially given that IP based networks are well established and widely available.

As an extension of Wolfe's work on grouping cameras of different types[30], a similar proposal has been made by Kulkarni[31] with the SensEye project for building multi-tier, multi-modal camera networks. In this work cameras are hierarchically organised, with cheap low level cameras performing simple motion and heat detection triggering more expensive high resolution cameras with shared scene views on demand.

Interest in the problem of calibrating large camera networks in order to organise the views of related camera is relatively recent. Devarajan[32] considers the problem of forming clusters of related cameras using the exchange of pre-computed scene points in a simulated environment, and provides a useful discussion of the vision aspects to the problem, under an assumption of perfect networking conditions. The specific problem of localising cameras in a wide area network of cameras is considered by Mantzel, with a theoretical algorithm for localisation based on Faugeras work on shared feature points which relies on broadcasts of localisation estimates to local wireless clusters[33].

Discussion on specific problems within such networks is also beginning, particularly within the robotics community[34]. Gerkey provides a general review of the various approaches to task scheduling within a group of robots engaged on a coop-

erative task[35] whilst Navarro-Serment[36] simulates the problem of scheduling a group of robots to obtain the best collective view of a shared target, and Isler[37] discusses assigning cameras to targets in situations where the camera's view relationships are already known. Dantu[38] presents a 'Super Sensor' system, where the output of a number of visual sensors is completely shared within the sensor group in order to perform histogram calculations. The algorithms used showed a performance improvement up to the point at which communication overhead began to dominate performance at slightly over eight nodes.

2.5 Distributed Groups

2.5.1 Reliable Process Groups

The concept of organising separate computer processes into a group is well established in computer science, and has been extensively explored, primarily with the goal of providing fault tolerance in real-time applications, using hardware redundancy. The traditional definition of these communications groups is a collection of processes that can communicate directly with each other using 1-to-many messages[39]. Distributed process groups, reliable process groups, or virtually synchronous process groups as they have been variously termed, provide automatically managed membership for application programs with guarantees that messages to a process group are reliably delivered to all members in the same order. Birman[40] provides a general review of the field with emphasis on its use for real-time control problems.

The process groups created by these schemes have rigidly enforced state and membership. The underlying software levels manage messaging such that without using a synchronised real-time clock, participating processes are unable to distinguish between actual and nearly synchronous executions, hence the 'virtual' in the above description. The major benefit claimed for this approach is to make the execution model much easier for application builders, and in the environments for which it was intended - distributed, real-time systems - large differences in execution times between

processes were unlikely to be an issue, except in instances of actual failure.

However this approach is not without its difficulties, a major one being the resources and overhead associated with attempting to provide this kind of controlled messaging. Attempting to provide alternatives with reduced overhead led to several different models for the reliability of the messaging, and a complicated set of configurable options to the underlying protocol stacks, management of which in and of itself began to present a noted obstacle to deployment. More significantly, Birman[40] reports on scalability issues that appear to be intrinsic to the model itself, and suggests a limit of one hundred as the number of group members that can be realistically supported by this approach. It is also apparent that considerable homogeneity is required from the clients, with reports from an implementation at the Swiss Stock Exchange that a single slow client would cause measurable performance impacts across the entire group. This particular problem would appear to be a property of the design of these protocols. It should also be noted however, that the application concerned - reliable, time-stamped delivery of information concerning stock market trades to different trading companies - has legal and fiduciary requirements for reliable and sequential message delivery to its clients.

Another issue has been provability of the claims for reliable message delivery being made for what was in essence, a complex set of message passing protocol stacks. Also perhaps a little outside the scope of this particular line of research lies the question of what would determine group membership? Why would a process want to be in a particular group, and what, besides reliable message delivery, would it expect? In terms of software process, the Horus/Isis/Ensemble approach effectively allows rigid control of a set of processes using simultaneously presented messages, the result having much more in common with the maintenance of the illusion that the distributed application is running on a single host, rather than on multiple hosts, than the kind of loosely coupled, peer-to-peer distributed applications being discussed here.

2.5.2 Peer-to-Peer Networks

Probably one of the more successful uses of unreliable group protocols are the peer-to-peer file sharing applications currently being used for a variety of purposes, such as the rapid distribution of Linux source releases [41]. Originally these applications operated on a scheme where nodes forwarded requests indiscriminately to other nodes within the application for a specified number of hops. It is worth noting that these applications emerged on the Internet relatively soon after the introduction of wide spread fibreoptic cabling had increased the available bandwidth to the Internet by several orders of magnitude. Their use quickly grew to consume an appreciable amount of available network resources, reports from the University of Wisconsin circa 2000[42], suggested that Napster traffic at that time was consuming 23% of available bandwidth, (Web based traffic for comparison was measured at 21%). Later work by Azzouna[43] in 2004 found that almost 80% of total traffic within a commercial IP network, (France Telecom) could be attributed to peer-to-peer applications.

Since Napster itself relied on peers registering with a set of central servers to support the actual file searching function, and then allowed peers to communicate directly for file transfers, it in fact supported a hybrid client-server-peer model rather than the true peer-to-peer model as it is currently recognised. The forcible termination of the searching service provided by the central servers, for legal reasons, also demonstrated the problem with having a single point of failure, which is probably the most significant drawback for any distributed application using the client-server based model. However, it was also evident before this occurred that there were scaling problems in the Napster network, as the central server farm responsible for handling search requests became overloaded[44].

Consequently for legal and technical reasons, Napster was rapidly superceded by purely peer-to-peer applications such as Gnutella, which initially relied on randomly organising connections between peers to provide network connectivity[45]. These applications also ran into scaling problems, the precise nature of which is still a matter of active research, although the emerging consensus as discussed by Schollmeier[46],

is that they scale much better than expected. A node belonging to a Gnutella, or similar application's network, can not expect to be able to send search requests to every single node in the network, but should be able to send its request to enough of them to be able to find the desired file. The systems effectively not only rely on there being multiple copies of the file for efficient copying reasons, but also for the initial searching function to succeed, and various suggestions have been made for hashing schemes to improve the search function itself[47].

Initially no attempt was made to organise the peers in these applications which led to significant scaling problems as they grew in popularity. Some of these scaling problems have been addressed by the introduction of an organisational layer of super-peers, which are chosen for their higher bandwidth and processing capability, and which act as an aggregating point for less capable peers[48]. Super-peers are then connected directly with other super-peers, and effectively act as routeing intermediaries for their peers. In the terms used in this thesis, this introduced a simple group based structure into the network, and consequently for the reasons discussed in Chapter 5, enabled the networks to scale to considerably larger sizes than was demonstrated by the initial unorganised peer-to-peer structure.

The potential of peer-to-peer protocols has also been recognised for other purposes besides file sharing. Manufacturer support of reliable multi-cast is still absent, and in the interests of global network stability likely to remain so. The highly varying degrees of reliability in the unreliable multi-cast support currently being provided, is motivating attempts to find alternatives using peer-to-peer like approaches. Gossip based protocols based on probabilistic models, where each member of a group randomly sends messages to other members of the group are recently attracting attention as an alternative to the absence of reliable multi-cast[24].

2.5.3 Distributed Task Management

Distributing tasks to multiple nodes has been extensively researched, from both the perspective of using some form of centralised manager or managers responsible for breaking the task up to subordinate nodes, and from the emergent results of co-

ordination arising from sophisticated local decision making[49]. Bidding schemes for deciding task allocation between participating nodes have also been studied, and were proposed as an early solution to sensor network problems by Smith [50] as part of the contract net framework in 1978.

2.6 Surveillance

Surveillance is currently one of the more popular applications for research in smart cameras, in particular due to the noticeable shortcomings of the existing camera networks which use a 1:N aggregation scheme where feeds from cameras in the entire network are monitored at a central point. Unsurprisingly this breaks down for any significant number of cameras, and the resulting data overload severely limits their effectiveness in detecting anomalous behaviour, unless specific information is available about when and where this behaviour has occurred. This makes them almost completely useless for their stated purpose - preventing crime - and only practical for solving crime when the crime is well localised in time and space, or regarded as sufficiently extreme to attract enough resources to allow an exhaustive search through the data. Consequently their most effective application so far has been in casinos.

Analysis of dynamic scenes for surveillance purposes is an active research topic, covering a number of distinct areas. A recent survey has been provided by Hu[51], which provides a high level view of the many techniques being explored for this such as motion segmentation, object classification, etc.

Marcenaro also performs a fairly extensive analysis of the wide scale surveillance problem[52], and presents a hierarchical organisation of intelligent cameras, hubs and control rooms as a proposed third generation solution. He acknowledges the limitations of such 'third generation surveillance systems' and points to the need for more flexible architectures and 'fourth generation surveillance' which involve intelligent and migratory agents as part of the solution, such as described by Orwell, et al.[53]. Although Orwell describes how high level agents could be employed in conjunction with smart cameras on analysis tasks, he does not describe how they should be or-

ganised. Collins[54] also presents a classic centralised model, where smart cameras perform localised detection and provide data to a central point for further processing. There is no direct communication between the cameras in this model, but data can be combined from multiple cameras at the data centre. Similarly Hampapur et al.[55] present a detailed analysis of the problem of video analysis in the context of the "multiscale spatiotemporal challenge", and develop an integrated approach using centralised databases in some detail, but do not discuss either the transmission and storage challenges of data distributed in such a network, or the potential benefits of allowing of peer-to-peer communication and processing. Their list of challenges to deployment of such networks is mainly concerned with technical issues at the level of the individual camera, wiring, calibration, etc. and although they allude to the need for developing system management tools, do not explore the ramifications for a system of any size.

Any kind of centralised scheme is obviously limited by the computational power of the supervisory nodes and the connectivity allowed between them. Karuppiah[56] amongst others has proposed more involved architectures which allow communication between the second and third level supervisory elements. In this system a set of resource managers provide an interface between cameras and user agents which seek to obtain information from appropriate cameras about a task. This allows the semi-intelligent cameras in the system to be dedicated to feature detection and extraction, and gives the user agents a controlled point of access to the resulting data.

A similar approach involving static camera surveillance, with active high resolution cameras available on request by the static cameras is described by Micheloni[57] in the context of a parking lot surveillance application, using a multicast based communication group to handle communications between cameras in the system.

There are other ways to structure solutions to this problem, Pinhanez has demonstrated a system where task control has been moved to sit between the cameras, and a world model manager[58], and Focken has similarly described a model where tracking agents interface with camera models and cameras to perform people tracking[59]. Foresti has proposed a simpler tree level structure where cameras are progressively

grouped into area based clusters, and extra levels are added as required.[60] Wolf, Ozer, and Lv[30] have proposed explicitly grouping several different types of camera at a local level, with for example wide angle cameras backing up telephoto cameras so that the wide angle cameras can provide extra scene information to track movement between the telephoto cameras.

Mittal[61] has demonstrated an example of using small groups of cameras in a relatively cluttered scene, where pairs of cameras use region based analysis in conjunction with Bayesian based segmentation and occlusion analysis is used to track people. The system uses a 450MHz Pentium II for each camera, but also uses a separate controller PC to synchronise frames within the system.

2.7 Multi-view Analysis

Several groups are working on scene analysis using multiple smart cameras. Collins[54] describes a system of three smart cameras which tracks a single person moving through an environment. The cameras broadcast information about their current views and tracking confidence every second, and the system as a whole produces a synchronised sequence of the person being tracked from multiple viewpoints. Mittal[61] has also demonstrated a similar scheme capable of tracking multiple individuals in a crowded environment, and Lin[62] describes a peer-to-peer multiple camera architecture for distributed real-time gesture recognition based on Windows using cameras with overlapping views whose relative position to each other is already known.

A great deal of image analysis work is performed using calibrated cameras since this makes view geometry much easier to extract. However work with uncalibrated cameras is a research area of considerable interest, particularly as cameras become cheaper. In practice this either becomes a problem of self-calibration using information extracted from the scene, or of successfully extracting information from the image(s) that is not dependent on calibration.

Dailey[63] has shown that traffic speed can be estimated from uncalibrated cameras, using a sequence of video images and geometric relationships within the image in

conjunction with some reasonable assumptions about traffic flow, and vehicle length. Similarly Taylor [64] uses stereo vision to perform lane extraction, although the algorithm used is specific to lane extraction.

Determination of information from camera movement, known as ego-motion, has been researched extensively especially by the robotics community. Tian[65] has compared six different approaches in simulation, and analysed their performance, effect of image velocities, and field of view but this work was done only in simulation. Dornaika[66] shows that stereo geometry can be extracted between two moving cameras using only motion correspondences.

Chapter 3

Bottom-Up Organisation

3.1 Introduction

Distributed camera networks then, present an interesting example of a potential class of applications and algorithms that must by necessity be distributed over multiple nodes, and whose organisation with respect to exchanging peer-to-peer data can be at least partially determined by analysis of local camera information to determine shared views. In most realistic network scenarios this organisation needs to occur dynamically, both as the views of the cameras themselves change, and in the face of the demands over time from different applications and algorithms.

One high level way to think of distributed systems with peer-to-peer communications is in terms of the nature of the information transferred between the different independent components of the system. Under this taxonomy, the simplest examples of distributed systems in fact involve neither direct nor indirect communication between individual components of the system. Rather each component implements an identical algorithm which runs independent of any inputs, the emergent outcome of which causes the system as a whole to behave in a desired fashion. The patterns displayed by finite state autonoma such as demonstrated by Conway's game of Life[67], are a well known example of this kind of system.

It is worth noting that even for this simple form of dynamic distributed behaviour there is at present no satisfactory general mathematical approach for analysing its

behaviour[68].

A rather more complex system arises with the introduction of implicit communication between the distributed components. Here the results of each component's algorithm, and hence its behaviour depends in part on information about, and by implication arising from, the behaviour of other components of the system. Examples of this form of system include schooling or herding behaviour in fish and animals, as exemplified by Craig Reynolds with his example of simulated bird flocking with the boids animation at SIGGRAPH '87[69].

Complex as the behaviour exhibited by these systems can be, they are still comparatively simple to the results when explicit, message based communication is introduced into a distributed system. In these systems several interrelated problems arise:

- Message reliability, or designing for lost, delayed or corrupted messages.
- Inability to guarantee eventual consensus between nodes in the system.
- The limits on each individual node's ability to receive and process messages.
- The limit on the number of other nodes in the system each individual node can maintain direct contact with.

It is instructive to consider how these problems interact with each other in the underlying message based systems that distributed applications are constructed over.

However robust the underlying physical layer of the network is in terms of message transmission and delivery, and today's optical fibre networks are extremely reliable compared to the copper based networks that preceded them, if the number of messages transmitted to a single node exceeds that node's ability to momentarily process them, then there will be delay and ultimately loss of messages at that node. How great an impact this will have on the larger distributed system is application dependent, but especially in instances of prolonged overload there is a high possibility that this will have an impact well beyond the original node.

This effect is most likely to be magnified in any system where the total number of components exceeds the number of direct connections supported by any individual node, such that communications between the nodes must be organised in a way where at least some messages are relayed between intermediary nodes. A delay at a single node will necessarily propagate back to any intermediary nodes as message buffers start to fill with delayed messages. Whether this effects just the node with the delay, or all message delivery from the intermediary node represents a design tradeoff on how message processing is organised at that node. However, even in the case where each node has independent buffers for its messages, other nodes in the system will be affected if either the network or the application layers start to trigger retransmissions in order to recover dropped messages.

Nor are any of these limits particularly hard and fast. The limit on the number of nodes that an individual node can maintain a direct link to, is partially dependent on the amount of processing being performed at that node, which is in turn a factor of the number of messages being received, and the complexity of the processing the node is performing either as a result of, or in spite of, the messages being received at the node.

How problematic these issues are for any given distributed system depends partly on the components of the system, and the number of these components in the system. There has traditionally been considerable debate within the networking community as to how, or even if, these problems should be exposed to the application writer. Although the nature of the distributed application itself also plays a part, it can be somewhat naive to assume that distributed systems can in fact be written without an awareness of these issues. The classic example is the transmission of real time video frames across a TCP/IP based network. The application programmer does not understand why they cannot simply transmit a single frame of video data, and the data network programmer does not understand why anyone would try. [1]

[1]The problem presented by video frames in particular arises from the packet sizes used by routers and switches within the network. Typically in a TCP/IP based network this is 1500 bytes, and a fairly frequent receive buffer size for a desk top system at this time is 65,635 bytes. If the application sends larger amounts of data than this, it is automatically broken down into a set of packets by the

One solution to the Gordian's Knot of problems listed above is to use small numbers of highly reliable components and carefully control the messaging between them. In a system such as that being considered by this thesis, where there are large numbers of unreliable components with a need for local cooperation on an ad hoc basis, it consequently follows that a possible generalisable approach is to find methods that reduce the number of components in the system, or more properly sub-system, below the limits outlined above.

Consider specifically the problem of extracting information from a large network of information dense sensors. If routing large amounts of raw information to a centralised point for analysis is impossible for any significant number of nodes, then the alternative is to distribute the problem of analysis over the devices that are producing the data - providing them with a sense of the goal or goals of the overall process as described by Bove[70]. If this problem involves shared information between the devices, then each individual sensor needs to determine on as local a basis as practicable, which other sensors have information that is relevant to its current view of the goal. However this shared information discovery problem also operates within the constraints of a distributed system as described above. Nodes cannot simply broadcast their requirements to all other nodes, (for any significant number of nodes), because this will overload their individual ability to receive and process messages[17].

Camera networks provide an extreme example of what is fundamentally a general problem with distributed systems, as they produce far too much data for central processing to be practical, but many data discovery problems within them require information from multiple nodes. Take the problem of finding the nest of a rare bird. Individual cameras may well capture and be able via image processing to recognise the bird, however in order to locate at least the general area of the bird's nesting

underlying network layers, transmitted and re-assembled at the other end. An uncompressed 320x240 YUV video frame is 115,200 bytes, so any attempt to send and receive it as a single frame, and there is nothing stopping the application from attempting this in most protocol stacks, will inevitably overflow the receive buffer at the destination machine. Increasing the receive and transmit buffer sizes, although it may well provide some local performance improvements has yet to prove to be a general solution since historically the size of the presented data packet immediately increases as well. In fact, within the communications equipment which provides the underlying data network the trend has generally been towards smaller transmission packets.

place, multiple sightings across time and space have to be correlated.

Being able to link individual nodes together into coordinated communicating groups to work on shared tasks is a very powerful technique, as evidenced on many scales by human activity. In order to implement this approach in software though, it can be seen that many of the problems of distributed message handling and organisation that have hitherto been more or less successfully hidden from the application programmer by the network[2] , have now re-manifested themselves *at the application level itself.* Selectively sharing information between sensors, immediately implies a need for some form of communication organisation. Arranging for every sensor to receive the unfiltered messages of every other sensor will not scale beyond very small networks of sensors. This implies that some kind of organisation within the distributed members of the application must be constructed, such that only those sensors with relevant information to each other share data.

As observed by Dantu[38] amongst many others, any improvements attributable to shared processing amongst a large number of nodes will be quickly defeated by the overheads of communication in any scheme where data is simplistically shared between all nodes.

This thesis defines a method that allows sensors to build a task specific, group based communications organisation, that is constructed locally and which is driven by local information relevant to the task being performed. Quite simply, each node in the network is allowed to broadcast a set of task based group membership requirements to other nodes when it needs to find group members. Each node that receives these requirements makes a decision whether or not to join this group based purely on its local information and the requirements provided by the membership message. Broadcast of these requirements can be performed within groups, so that the messaging overhead of the protocol can be appropriately controlled by the application. For example, a membership message may invite nodes to join its group if the node receiving the message is within a certain distance from the originating node; broadcast of this invitation being restricted to the group of nodes using a subnet, or wireless access

[2]This should not be a particular surprise since said network is itself a highly distributed system.

point. However, no attempt is made to guarantee that all nodes in a group share a common view of group membership. This very loose and purely local definition of a group can allow the creation of application specific network structures that allow nodes to dynamically collaborate on shared problems.

It is assumed that the communications environment being created is resource and data rich, so that one of the critical problems of the distributed application being constructed is not one of economic execution in order to extract as much computational value as possible from a given hardware platform, but rather one of selecting the most appropriate individuals, whether in terms of access to data, availability, or capability for the problem. It is this movement from computational scarcity to plenty that has so dramatically changed the ground rules for group computation in the last decade, and created the conditions for the questions being discussed here.

3.2 The importance of Organising Groups

There is an important distinction in the definition of a group that should be touched upon before discussing group forming in distributed networks more deeply. When the behaviour of collections of people is typically discussed, whether they are organised or not, there is usually an implicit assumption that the group in question can be regarded as a single entity. As Berger and Luckman describe it[71]

> "For man, the group exists as an autonomous, abstract reality, a social construct that is apparently independent of him"

Similarly Csányi[1] notes,

> "man habitually abstracts groups into autonomous, separate entities, that are regarded as being independent and wholly representative of their members."

Although this is a useful linguistic shortcut, in reality a group of even semi-autonomous nodes is always a collection of individuals that share some method for reaching a

greater or lesser degree of consensus on a shared goal. The problem with the abstraction is that it ignores a multitude of details such as how the consensus is achieved, the limits of communication between individuals that constrains achieving this consensus, and how changes on the nature of this consensus could if necessary be negotiated. These details are very apparent to the programmer of any distributed application, but are not infrequently assumed into non-existence in more theoretical discussions on high level applications, when groups of nodes are discussed as single entities.

Here a group is defined flexibly, as an ad hoc association of individual nodes that has identified that they are potentially useful to each other, either working on the same problem, or needing inputs from other group members, but not necessarily performing exactly the same task in the process. By implication too, members of a group have qualities with respect to the other group members that distinguish them from nodes that are not members of the group. It is worth noting though, that the implication of the limits on communication within a sensor network identified by Scaglione[17] amongst others, are that almost any division into identifiable groups that succeeds in restricting the amount of communication to nodes outside the group is beneficial in terms of ameliorating the communication overhead on the network as a whole.

The older reliable process groups created by systems such as Horus[40], were far closer to the model of a single centralised task that happens to be running on some distributed hardware, than those having any kind of local control or selection over their members. If there was a choice about which group a process should join it was usually in the form of which server to choose from, or whether a process should be part of the active or standby system. These types of decisions tended to be implicitly answered as the system was designed, and the question of group selection itself rarely arose since engineering the resources required for the application was part of the design problem. There was no need to tolerate flexible group memberships, since there simply weren't any spare members. Now there are potentially millions.

In moving outside the specialised world of reliable real-time computing, some limitations have to be recognised. Reliable messaging for example, is in some senses

an application trap, since it implicitly assumes that the message senders are reliable, and reductio ad absurdum that the underlying hardware and the application software is reliable. The latter in particular being provably incorrect as a variant of the halting problem.

One of the premises of this thesis is that local information can be used to create organisational structures within the application that are globally useful. So here it is assumed that the messages being sent between sensors carry information that may be useful to a local calculation. There is no requirement that each sensor in the group finds every single message it receives from the other sensors to be useful, nor are there any guarantees that it will receive all the information it needs from them. No attempt is being made to solve the consensus problem, rather it is being unconditionally surrendered to.

What is required is that for any given group, all the nodes share the same functional definition for that group, even though their individual views of its membership may differ. So from the purely local perspective, membership of a group is a guarantee to the local member that messages from that group meet certain shared criteria that the local node has specified in order to define the group. Each node has its own view of the group's membership, and a local set of parameters that applied to the group's functional definition, provide the membership criteria for that group at that node. Groups are identified on the network using a name, which corresponds to the groups agreed function. The name identifies the group uniquely at all nodes, however the membership criteria can be adjusted dynamically at each node, such that each node in the group can have varying views of the group's membership at that node.

It may seem at first sight unlikely that relaxing the concept of group membership in this way could be useful. In order to understand the benefits of this approach, the peculiar advantage groups offer when the particular problem of multi-directional communication is concerned should be considered.

3.2.1 The limits on local communications within a Group

If a group of nodes is to communicate with each other using messages, then there must be links between the nodes to allow these messages to be sent. These links may be fiberoptic cables between cities, obstinately straight roman roads, or chance meetings that have allowed the creation of social connections and the resulting exchange of contact information. Invariably though, there is a cost in either time or material to establishing and maintaining these links, and a consequent limit on the number of of links any single node can support. This cost - however it manifests itself - represents one of the fundamental structural constraints on any network.

Consider that for any given group of nodes, the number of links required to ensure that every node is contactable by other nodes within the group is determined by the topology used to connect the individual nodes together. If the total number of links between nodes is regarded as a group constraint, then the possible topologies for the group effectively reduce to three, as shown in 3-1.

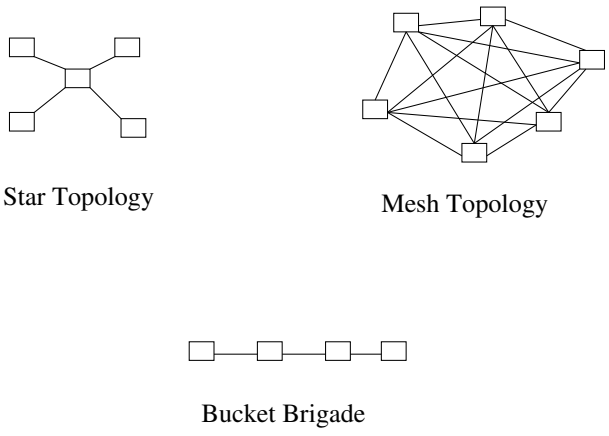

Figure 3-1: Group Topologies

If all messages between nodes are sent directly, that is without using any intermediaries, then the total number of links required to completely connect every node with every other node is

$$\sum_{i=1}^{N}(N-1) = N(N-1)$$

where N is the total number of nodes that are completely interconnected. This is the mesh topology shown in Figure 3-1 This places a relatively small upper limit, dependent on the communication capability of each node and the amount of communication required, on the number of nodes that can be directly interconnected, and consequently on the number of nodes that can exist in a completely interconnected group.[3]

Alternatively, if intermediaries are allowed, that is nodes are allowed to pass messages on for other nodes, each node does not have to be connected to every other node. In this case the minimum number of links, required to completely connect a group is *N-1*, arrived at by connecting the nodes in series, (also known as the bucket brigade). However, this is highly inefficient in terms of the number of hops a message must go through in order to be passed from one end of the networked group to the other. By adding only one link, and also accepting that one node in the network has far greater individual connectivity than the others then the maximum number of hops for complete connectivity within the group is reduced to two. In data networking this creates the widely used client-server model. Although the total size of the group, or the number of clients, is still limited by the capacity of the server, the individual communications load on each client is greatly reduced.

In an information network where the exchange of data between nodes is being considered, there are other considerations when the flow or balance of traffic between nodes is considered. It is clear that the client-server model provides the most efficient solution when message flow is purely in one direction, from the server to the clients, with very limited traffic coming back from the clients; especially when large numbers are involved. Equally the mesh network is the most efficient way to share information bi-directionally between a group of nodes, but it can only support a limited number of nodes within a completely connected group. It can though be regarded as more robust, since there is no single point of failure in the system. Perhaps most interestingly, by

[3]The Dunbar number of 150[72] is the most frequently quoted maximum level on the size of human groups, however this itself tends to be a larger community composed of smaller task groups in the 5-9 range.

adding or removing links, each topology can ameliorate its particular set of problems by at least partially adopting features of the other topology.

It can be observed from this that for many of the communication patterns required between the components of a distributed system, there may often be no perfect organisational topological solution. All networks, whether biological, software, or social exist as the sum of the compromises that have to be made in the face of these fundamental structures to local data needs. Each topology has a solution to the other's problem: introducing peer-to-peer links into the client-server model allows clients to communicate better, and removes the server as single point of failure; whilst introducing centralised routeing modules into the mesh network allows it to support more nodes. Consequently a certain degree of oscillation between the two extremes can often by observed as distributed systems evolve over time.

In both cases it also follows that once the limit on the number of links into a node has been reached for that particular topology, the network must then be further structured into multiple, interconnected groups. These inter-group connections are also subject to the same topological constraints discussed previously. Viewed from this perspective, the function of a group in an information network is simply that it allows large collections of nodes to collectively circumvent the limits on the number of network links they can individually support. In doing so it considerably increases the amount of information that can be efficiently accessed by any individual throughout the network.

The problem faced by a local node in a sensor network though is essentially one of directed navigation. It needs to be able to identify which nodes in the network have information that it needs, and in order to be able do this at all in a network of any size, it needs some kind of organisational structure that allows it to avoid having to exhaustively communicate with all the other nodes in the network. Indeed it follows from this observation that almost any organisational structure will be an improvement on having none at all, and that in fact local consistency between views of that structure may not be particularly relevant, or even useful. Especially if the cost of maintaining them is significantly increased network traffic or program complexity.

An example of this aspect of group organisation was demonstrated in human networks by an experiment performed by Stanley Milgram[73] on social distances. Milgram demonstrated that a set of simple instructions, together with the implicit structure provided by human social relationships was enough to perform long distance message routing without end to end knowledge of the ultimate destination. The first level of participants in Milgram's experiment were instructed to attempt to route a letter to a person who was unknown to them by sending it to the person they felt most likely to know the person's correct address. This letter contained instructions that the recipients should repeatedly iterate this process. In many cases these simple instructions were enough to allow the letter to be delivered.

Although the most infamous result of the experiment was to demonstrate that remarkably few attempts were required to route a letter to its ultimate destination, the well known "6 degrees of separation effect", a less remarked result as observed by Kleinberg[25] was that it also demonstrated that in general people knew how to navigate their social networks in order to obtain useful information. It also seems to have passed without mention that the experiment itself took unremarked advantage of a millennia old system of physical mail delivery, which functions transparently, at least to its users, between regions and countries on a global basis. Indeed, the underlying mechanisms of the postal system are precisely the same as those Milgram demonstrated the existence of with his experiment. The district sorting office at Paddington in England does not need to know the precise location of an address in Mumbai, only how to get a letter over to India. It is the problem of the sorting office in India to then route the letter correctly to its eventual recipient, which they, equally ignorant of the precise details, achieve by sending it to the closest local delivery office. [4]

Although organising groups appear at all levels of the postal system no attempt has ever been made to ensure consistency in membership. Indeed quite the contrary. The

[4]Dating back at least to the Roman Empire. Milgram's actual success rate however, was only 30% of all the letters sent out, which may also demonstrate the problem that end to end connectivity in such networks cannot be guaranteed on a purely local basis.

global postal system has shamelessly exploited inconsistencies in group membership views in order to serve the larger purpose of mail delivery, by using neutral countries such as Switzerland to route mail during times of major wars and minor diplomatic hostilities. In the process it has demonstrated impressive resilience.

Indeed, the widespread use of these locally formed groups with inconsistent membership views in human society, along with the evidence (from systems such as the Post Office), that they have been in use for several thousand years, suggests that they may well provide a useful organising concept for corresponding problems involving cooperating nodes in distributed sensor applications.

3.2.2 The Advantages of Locally formed Organisations

Clearly some of the advantages outlined above apply equally if groups are imposed upon the individual nodes in the network, or arrived at by some local process. There are, as described, significant advantages to the individual nodes attendant from their operating within any communications structure that reduces the individual load on the nodes from network communications.

Still there are some advantages peculiar to local groups, that is groups where no particular effort is made to enforce consistency of membership views across the group's members. A not inconsiderable one being that it is, in an asynchronous network, provably impossible to reliably do so. This is a problem known as the Fischer consensus problem[74], which states that it is impossible for a set of processors in an asynchronously connected distributed system to be guaranteed to agree on a binary value in the presence of even one failure.[5]

Changes in group membership as members join and leave a group are inevitable, and even without actual failure these changes require time to propagate across the

[5]In a related piece of work Lamport[75] showed that in the presence of n failures, where failures are lost, delayed or deliberately altered messages, (3n+1) nodes are required to identify the nodes which share an agreed value for a message. This is not strictly a solution to the consensus problem, rather than a way of partitioning the network into groups with consistent agreed values. In large sensor networks, knowledge of the correct value of m is obviously rather more problematic, than in systems designed to have 5 processors for redundancy and verification such as the United State's Space Shuttle.

system. The consensus problem simply assures us that it is impossible to ever guarantee that a stable system will emerge from these changes. Although in practice there will be periods where the nodes share a common view of group membership, there will also be periods when they won't. Attempting to resolve this problem programmatically typically induces successive degrees of complexity in the application, as the problem repeatedly occurs at the different levels of consequent protocol abstraction.

Locally organised groups also have the advantage that they can adapt much more easily to local conditions at the nodes, and within the network. However, centrally imposed group memberships do enjoy an advantage when it comes to changing the parameters of group membership, since it is easier to broadcast the change to all group members from a central point. Whether or not autonomous members of a group will accept such a centrally imposed change is of course another question, particularly if the imposed change is insensitive to local conditions.

3.2.3 Changing the Requirements for Group Memberships

Or why Martial Arts and Religious Organisations have a tendency to schism.

Network topology presents a set of communication constraints on groups that are fundamentally the same, whether the network is designed, or develops out of the aggregate of local decisions. However, in networks where group membership is determined at a local level, as opposed to those where it is imposed hierarchically, there is an interesting difference when it comes to attempts to change the group membership requirements or the purpose of the group.

Consider a group of locally autonomous nodes that has been established to work on a problem. Each node has local information that is needed by other group members to solve their part of the problem, and it is this information that in fact determines their suitability for membership of the group. Each node can only solve the problem successfully by combining local information with information from the other group members, (for example tracking the direction of a moving object). What happens if in order to solve the problem at at least one node, the global group membership requirements need to be changed?

Changing group membership requirements for the entire group, means that all the other group members must receive, and agree to use the new requirements. Because of the consensus problem it can never actually be guaranteed that this will in fact occur, but there is also the problem that other group members may not agree with the change of parameters, since their local requirements are still met by the original set of group requirements. It is of course possible to set up a negotiation protocol to reach agreement between all nodes on a change of parameters, with some degree of certainty, but it must also be acknowledged that from the perspective of the individual node this represents a lot of work in terms of communication overhead. Moreover as the size of the group increases, the effort this process induces within the network increases near exponentially.

Alternatively, the individual node can withdraw from the existing group, and establish a new group by broadcasting a list of the new requirements and waiting for other nodes to join it. It is in fact a lot less effort, in terms of communication overhead both for the individual node and for the network, to leave the original group and create a new one, than it is to negotiate a change of parameters within an existing group. The precise effect on the actual membership of both groups will depend on how radical the change in parameters is.

In computer networks this is in fact extremely useful - it means that group parameters can readily be changed as the task requires, with relatively little overhead either locally or on the network. Since there is a guaranteed way to destroy a group, there is no danger of the older group hanging around after it has been superseded.

In human social networks similar phenomena can be observed in many situations which involve coordinated group effort, particularly in organisations that have been established for long periods of time. The result, particularly where the organisation in question can be characterised by a certain resistance to changes in group parameters, for example those with religious, political or martial arts goals seems to be an observable tendency to schism into new groups with divided memberships, rather than attempt to negotiate the large scale change in group function that would otherwise be required.

Chapter 4

Local Group Forming Protocol

4.1 Introduction

One advantage of not enforcing consistent group membership views across nodes in a group, is that the resulting group membership protocol becomes quite straightforward.

The protocol description below is divided into two sections, control of local group conditions by the application using the group, and group setup and management between nodes within the group. No assumptions are made about the underlying protocol stack, and in particular about its reliability. Limits on the number of groups an application may use, multiple process membership of groups at a single node, the number of groups supported at a node are left to the implementation.

The primitives for the group management protocol are shown in table 4.1.

Primitive	*Description*
Create Group	Create a group
Group Information	Send group membership requirements
Send to Group	Send a message to a group
Send to Group Member	Send a message to a group member
Leave Group	Leave the group
End a Group	*End the group, by removing the terminating node*

Table 4.1: Inter-Node Group Management Protocol

Note on Terminology

In the following, membership criteria are the general requirements a node has to meet in order to qualify for group membership at another node. This might be an exact value, a single value being within a particular range, a reliability or time commitment from the other node, etc. This value or values, there is no specific restriction on the number of requirements, as instantiated at a node, is referred to as the membership qualification for the group, and it is this value in conjunction with the membership criteria received from other nodes that is used by a node to determine whether or not they qualify for group membership at those nodes.

4.1.1 Protocol Elements

Group Creation

Under the scheme presented here, each node that needs to belong to a particular group, 'creates' it locally, and then attempts to find other members to belong to it. Amongst autonomous elements and an assumption that the individual node's view of group membership does not have to be consistent, group creation then becomes the problem of providing all the information required for each node to decide whether or not they can usefully contribute to another node's group-view.

From the global perspective of many tasks it is reasonable to assume that this information will necessarily be incomplete, since the desirability of group membership may be partially dependent on other individual's decisions to participate or the number of nodes already present in that node's local group-view. These criteria make group creation a potentially more dynamic and continuous process than typically seen under other assumptions, particularly in a heterogeneous network of nodes.

The message exchange between nodes for group creation is shown in Figure 4-1. Node 1 is attempting to form a group, and broadcasts the group creation message to Nodes 2 and 3. Node 2 meets the group membership requirements in the message and responds with its local group membership criteria, Node 3 does not and ignores it. On receipt of Node 2's group information message, and assuming that its own

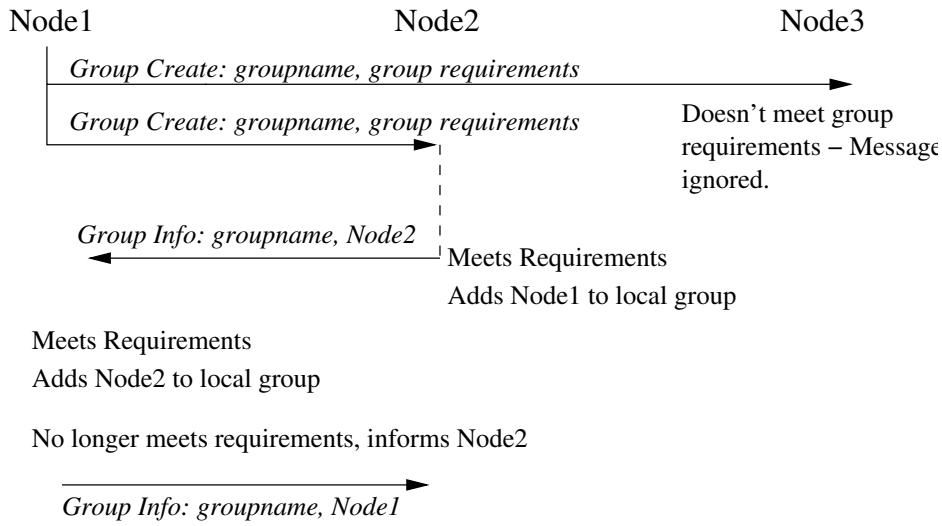

Figure 4-1: Message Exchange for Group Creation

local group qualification value has not changed in the time since its original message was sent, Node 1 similarly adds Node 2 to its local membership list for the group. If the local membership requirements have sufficiently changed during the message exchange to now exclude the qualification value provided by Node 2 in the *Group Info* message then it replies with a *Group Info* message to Node 2 containing its new group criteria.

There is nothing to stop an application establishing differing membership criteria at independent nodes such that for example, Node1 is a member of Node2's local group view, but Node2 is not a member of Node1's group view. This would be one way to implement a directed diffusion[23] scheme for example.

Group Info

The *Group Info* message provides a node's local group membership value(s) to another node. It is strictly the response to the *Group Creation* message from another node, and is also sent to other nodes when a node's local group value changes. There is an obvious implementation issue with this particular message, since it is potentially sent with every local change of group qualification value(s), and could therefore

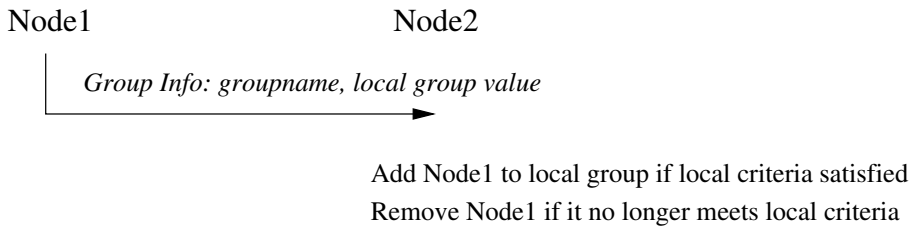

Figure 4-2: Message Exchange for Group Info

cause considerable local load on the other group members. However, not all group membership criteria need necessarily be determined by the application, the group management layer can impose its own conditions, and exclude nodes that say fail to have a minimum available bandwidth, or exceed a locally determined messaging allowance in a measured period.

Sending Messages to Group Members

There are two forms of message sending, a simple one which sends a message directly to the specified member of a group, and a slightly more complex form which results in a message being sent to all the members of the group at the local node.

There is one parameter, besides the message contents, a hop count for the message to be forwarded. If a node receives a message with a hop count greater than zero, then it decrements the hop count on the message, and forwards it to all group members in its local view, except for the originating node. There are no guarantees that subsequent forwarding doesn't result in duplicate messages arriving from different sources at a node, this will depend on the higher level structure of group organisation constructed by the group membership parameters at each node.

If a message is received for a group which the local group is no longer a member of, it is ignored. There is again a potential timing problem here where announcements of changes in local group qualification arrive after messages have been sent to group members. If this a critical issue for the application then one solution would be to include group membership qualification values in the send message header.

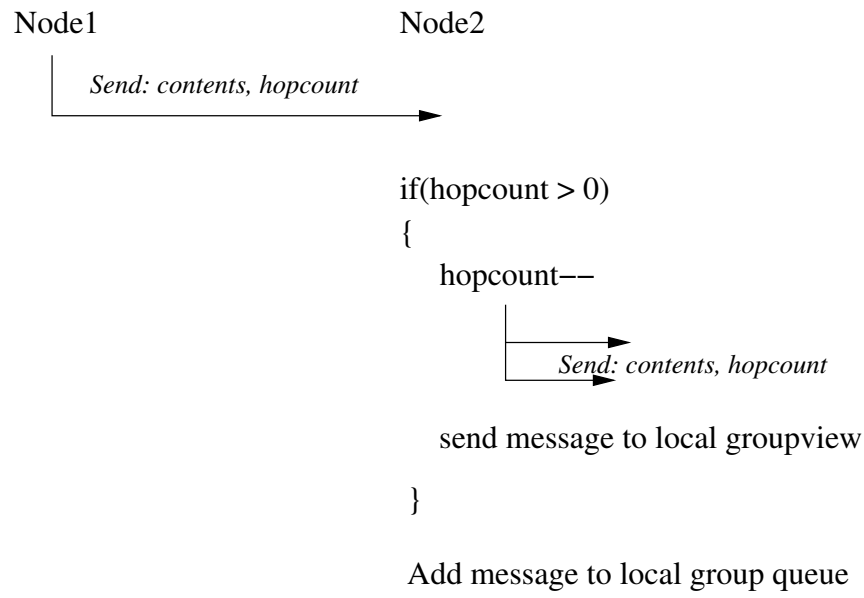

Figure 4-3: Message Exchange for Group Send

Leaving a Group

Group membership is determined at all times by the ability of any particular member to meet the local group membership criteria, and is managed locally by each node. Consequently there are in effect three ways a node can leave a group, its local membership qualification can change to remove it from the group-view of other members of the group, their group criteria can change so that it is excluded, or it can deliberately leave the group. The first two require that the task provide the group management layers either with a method for periodically evaluating performance, or periodic updates on its local evaluation of the group criteria. In practice the value is controlled by the application, and consequent group membership changes by the group management layer. Changes to the value that determines group membership are propagated to other members of the group at that node, using the Group Info message.

The Leave Group message shown in Figure 4.1.1 is available as a method for terminating group membership by the local node or application unilaterally.

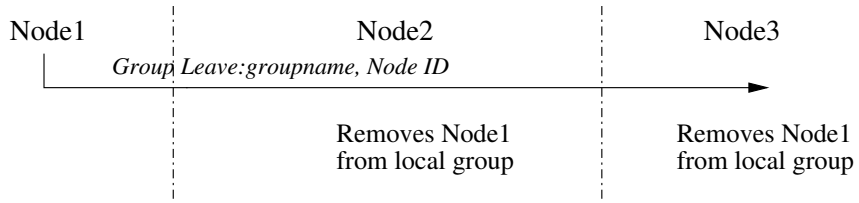

Figure 4-4: Message Exchange for Group Leave

Ending a Group

Completely ending a group, that is removing it at all nodes, is a more interesting problem than it may at first appear. Since membership of the group is distributed, any attempt to end the group must be successful at all group members, otherwise the group will continue - and quite possibly be re-populated as membership criteria broadcasts solicit new members. For groups that are not intended to be permanent this may be a problem. There is unfortunately only one way to guarantee complete group destruction when required, and that is to create an association of a group with a single endpoint, either the group creator, or another fixed point. It can be envisioned that a large network could use dedicated group destruction servers for this purpose, but in many cases this would simply be the originating node of the distributed application.

The end group primitive itself then becomes a request to the fixed point to end the group. How this is done is implementation dependent, it could simply drop its TCP/IP connection for all group nodes, or it could stop responding to periodic 'are you alive' messages from individual group members. Both solutions have the advantage of failing safely, since group destruction and failure of the end point are in effect the same thing. This one instance of centralisation can be justified on the grounds that it is relatively straightforward to build redundancy into the task/application that will prevent task failure arising from a single endpoint failure, but otherwise impossible to solve the problem of immortal groups consuming local and network resources.

As an implementation consideration, it is straightforward to require that only

certain nodes are allowed to request group termination.

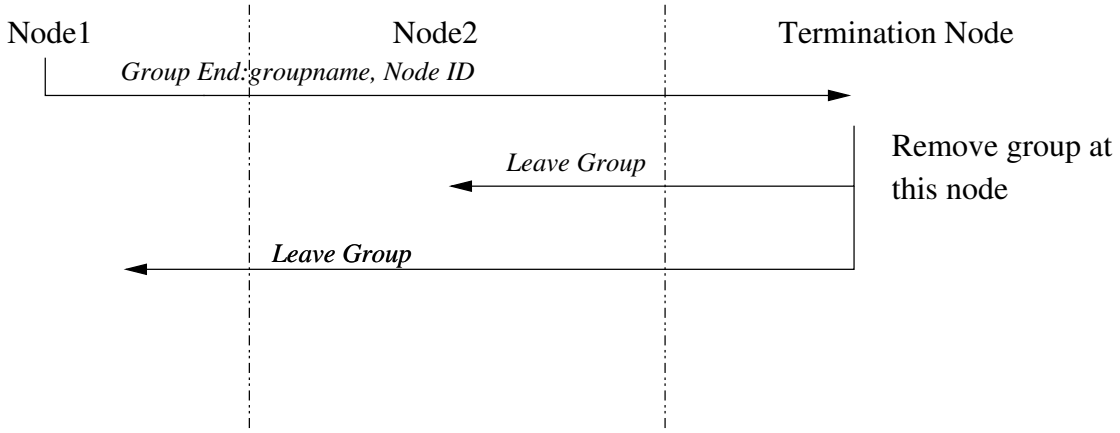

Figure 4-5: Message Exchange for Group Ending

4.2 Local Group Interface

The local group interface provides the application interface to the group management layer described above. Applications create a group locally, with specific parameters, and the group management layer handles advertising the group's existence to other nodes, providing qualified members, receiving and holding messages from the group members until the application is ready for them. The interface primitives are shown in Table 4.2. In general they map directly to the group management protocols above.

Create Group

The create group command specifies the group name being created, and the local membership criteria for the group being created.

Set Membership Value

Sets the local membership value or values for this group.

Primitive	*Description*
Create Group	Create a group
Set Membership Value	Set the current membership value at this node
Group Information	Get information on local group
Send to Group	Send a message to a group
Send to Group Member	Send a message to a group member
Receive Message	Receive a message from a group member
Leave Group	Leave the group
End a Group	End the group, by removing the terminating node

Table 4.2: Local Group Interface

Group Information

Gets information from the group management layer about the group, including the number of members currently in the local group view, state of message queue, etc.

Send to Group

Send a message to the all the group's members in the local group view.

Send to Group Member

Send a message to a specific group member.

Receive Message

Receive a message for a particular group.

Leave Group

Remove this node from the group.

End Group

End the group at all nodes. As discussed under group management, the application may choose to restrict this action.

4.3 Forming Layered Groups

The organising power of the type of group being discussed here, comes from the ability to create multi-layer groups with partially shared memberships. Once this is done, intra-group messaging can be used as a powerful application specific method of re-organising the network by using the existing groups, and the known relationships between their membership criteria and their members, to solicit for members for new groups. However, this does beg the question of how to handle the initial state of the network where there are no defined groups to provide any organisation.

One obvious way to solve this problem, is to provide nodes with a predefined group structure, and use this to bootstrap into a set of group memberships more suited to the application. In any network based on TCP/IP the existing network structure can be re-used to supply this, by building up groups based on the existing IP address subnet structure. This has the advantage that in most cases the resulting group structure will have some relationship to geography. Similarly in a wireless network, group organisation could be initially formed around the base stations.

Alternatively a simple startup procedure can be provided by the application as follows. Each node joining the network is supplied with the same destination node as a connection point. The designated destination node forms a general group of all nodes that connect to it. Each node that joins this group then broadcasts its particular, application dependent, group requirements to this list of nodes; and sets up the initial groups it requires for its application. Once the local node is satisfied that there is sufficient connectivity into the network from these groups it can drop out of the initial mesh group.

4.4 Examples

4.4.1 Group calibration of Light Levels in a distributed camera network

A simple problem for a group of calibrating cameras is to coordinate light levels. Automatic gain control(AGC) for example can easily be confused by spurious reflections. Group membership for this task is complicated by an implicit requirement that the cameras have shared lighting conditions, since light level measurements are highly specific to local conditions. If for example, the cameras are scattered about a relatively large room with variable lighting, some of them near a window, whilst others are under artificial light, then determining which light readings are relevant for comparison can be challenging.

One possible solution, would be that if a camera detects that its AGC is overloaded then it will initiate a light level calibration group. Membership of the group is initially be restricted to the first camera to respond, with the proviso that this camera does not have a light level problem. Once this camera has joined the group, the membership criteria is changed to all cameras with a light level within a specified range of its own. After membership of the group has stabilised, the initial camera can readily obtain an average light level which it can use to manually set its AGC. At this point there are two potential states, the first camera may have had an acceptable agc value for this camera, in which case it can maintain the group to get periodic updates, or shut it down as required.

If it does not have an acceptable agc value, it can shut down this AGC group, and start a new one using the same process as above, but with criteria that exclude the unhelpful light level, and any others it may have accumulated throughout the process. In this way it can iterate through the different light conditions in the room until it determines one that is locally suitable.

4.4.2 Tracking People or Objects

A task of increased interest is the detection and tracking of objects or people through a monitored area. Approaches to it typically involve a set of visual analysis tasks performed by camera or cameras in the area, which are then analysed for a probabilistic match.

Tracking can be regarded as a multistage process, where first successive frames are analysed for movement, moving objects are segmented, and then one or more heuristics are applied to obtain items of interest. In a collaborative environment, two additional problems present themselves: tracking objects continuously as they move from camera to camera, and distributing the tracking of several objects amongst the set of available cameras in order to load share. Particularly in a large scale distributed environment, such as envisioned by Marcenaro[52], dynamic ways need to be found to pass tasks between cameras as objects move through the area, and to reduce the load on any centralised elements in the system such as reporting hubs.

A possible approach to this problem using the group forming protocol, would as a first step initiate a group with members capable of performing movement detection. This might be done with criteria that specified a particular area or common view. Any member that detected a moving object would then start a new group from the existing members specifying criteria that matched this object. Other cameras with a suspected match could join that group, and perform more involved comparisons. Since cameras are not constrained in the number of groups they join, this has the advantage that a camera may end up belonging to more than one group as the cameras sort out a crowded environment.

Finally, by setting up a group for reporting, with criteria based on the results of the tracking phase, a simple method is provided for cameras to report results above a defined but potentially dynamic threshold.

This approach has several advantages for the problem. As group membership is dynamic, objects can be readily tracked by multiple cameras dropping in and out of a an object tracking group. If guarantees of group persistence are required then higher

level reporting groups can be added to the architecture. By adjusting the membership criteria of the tracking groups, load balancing and priorities of tasks and objects can also be readily achieved.

Chapter 5

Forming Groups based on Image Matching

5.1 Introduction

Finding corresponding points between images is a fundamental aspect of many problems in computer vision, for example obtaining 3D structure from multiple images, stereo correspondence and motion tracking. For group robotics in particular it represents a critical group calibration task, since it is difficult for group co-operation to occur on any task if the robots cannot determine their relative position with respect to each other. Localisation, where cameras within the network dynamically and automatically determine their scene relationship with other local cameras, is a necessary first step for successful deployment in any large scale camera network, since manual calibration on such a large scale is at best impractical, and in some environments - such as air deployment over wilderness areas - impossible.

Due to the high dimensionality of image data, local processing must be used to reduce the volume of image data being exchanged. However even after this has been done, the resulting information exchange between nodes and the associated processing is still significant enough that attempts to exchange data between any significant number of nodes will rapidly overwhelm their individual network handling capacity. As such it makes an ideal problem for examining the problems of organisation in

communicating networks of peers.

There are several different approaches to the problem of scene matching between cameras. Very close and well aligned images, can sometimes be matched by subtracting a segment of one image from the other using a sliding window across the image, and taking the lowest result. This can be a useful approach for stereo vision problems. A popular approach using calibrated cameras is to use the epipolar constraint, which constrains corresponding points between the two images to lie on their respective epipolar lines in the other image, thereby reducing the matching problem to a single dimension.

A relatively recent approach, and the one used by this thesis, is to attempt to identify distinct image features that can be compared between images, and thereby establish matching co-ordinates between separate views. In order to be successful this approach requires image features that are invariant, or mostly so, to reasonable transformations between the two images; that can be reliably matched; and are sufficiently distinctive to be able to provide unique correspondence between the feature points. Particularly in an application attempting to find matches between large numbers of cameras, it is critical that whichever method is chosen have an extremely low false positive rate.

Generally features that meet these criteria tend to be based on the topological characteristics of the arithmetic representation of the image, rather than characteristics that are apparent to the human visual system. The Harris Corner detector and the Hough transform have both been used for this purpose, the latter in particular leading to a certain preference for distinctive straight lines in many sample images. Other approaches include using the phase, rather than the magnitude of local spatial image frequencies, which may assist with illumination invariance. Schiele and Crowley[76] have suggested combining several local features and using multi-dimensional histograms to summarise the distribution of these measurements within image regions. This approach allows local features to be combined relatively inexpensively, and may be particularly useful for problems that can incorporate motion detection with other cues.

This thesis uses the Scale Invariant Feature Transform (SIFT), as proposed by Lowe[9] The SIFT approach suggested itself for this work as a consequence of its success when applied by Brown[7] to the problem of creating panorama images from a collection of individual overlapping shots. This approach relies on detecting scale-space extrema, and computing a keypoint location with associated gradient orientation information. Subsets of related keypoints are then used to determine correct correspondences between the images.

5.2 SIFT Based Image Matching - Theory

Gaussian scale-space is created from images when the image, I(x,y) is convolved with the Gaussian, $G(x, y, \sigma)$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

where * is the convolution operation in x and y. Lowe's[9] was to use the difference of Gaussian function convolved with the image $D(x, y, \sigma)$ to detect scale-space extrema in the image. The difference of Gaussian is calculated from the difference of two nearby scales which are separated by a constant multiplier, k.

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{5.1}$$

The topological ideas behind scale-space theory as used by the image community are generally attributed to Koenderink who in his 1984[77], paper showed that the scale-space representation described by Witkin[78] must satisfy the linear diffusion equation

$$\delta_t I = \bigtriangledown I \tag{5.2}$$

This is the standard physical equation which describes the evolution over time t of a heat distribution I in a homogeneous medium with uniform conductivity. The Gaussian Kernel is the Green's function of the diffusion equation at an infinite domain, and so it follows that it is also the unique kernel for generating scale space. This result

was also derived a littler earlier in Japan by Taizo Iijima in 1959[79][80].

5.3 SIFT Implementation

The implementation of the SIFT algorithm in this thesis is taken directly from Brown's[7] work on panorama recognition. As described previously, the algorithm proceeds by identifying a large set of SIFT features, and deriving a small set of match points from them.

The first step of the algorithm is to calculate the difference of Gaussian pyramids, used to locate the scale space extrema. The calculation is based on a blurred image, and calculated a total of 5 times for each image, as showed in Figure 5.3. SIFT features

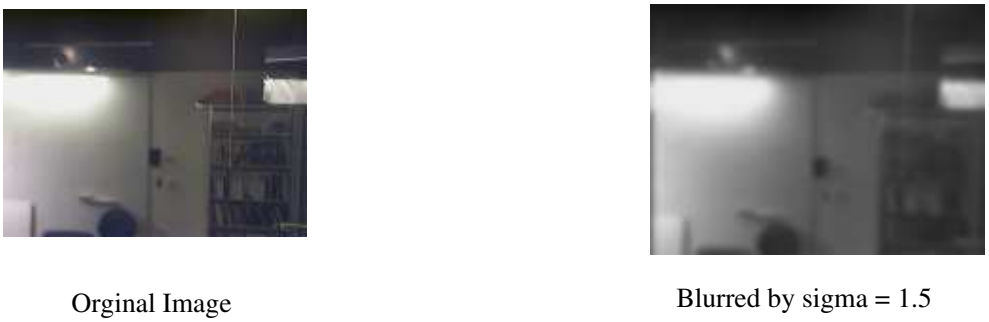

Figure 5-1: Difference of Gaussian Pyramid

are located at the maxima/minima points of the difference of Gaussian function. After identification a characteristic scale and orientation is established for each SIFT feature, and used to establish an invariant descriptor based on accumulating local

gradients in orientation histograms. This allows some leeway for edges to shift without altering the descriptor vector, thereby allowing a degree of robustness to affine changes in the images. Critically, since the descriptor vector consists of differences in intensity values, it is invariant to affine changes in intensity.

Once the features have been extracted from the images, they are filtered with low peak magnitude values being removed. This typically produces a set of approximately 150 keypoints for the 160x120 resolution images used for matching purposes. These keypoints are then examined for matches between the two images, by consolidating the sets of keypoints into a single K-D tree, where each point is matched to its k nearest neighbours, (for this work, k=4 after Brown[7]). A typical set of results at this resolution from two of the cameras in Figure 5-3 is shown in Figure 5-2.

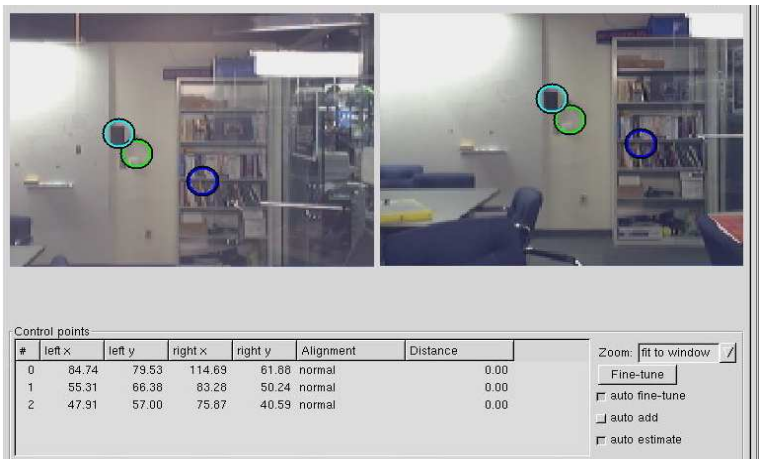

Figure 5-2: SIFT points extracted from Camera Tile Images

One of the advantages of the algorithm is that it reduces to the calculation of a set of keypoints for each image, which can then be distributed to each node to be matched with the local set of keypoints. The keypoints only need to be computed once by each node, and the matching is a separate operation which can be performed independently against each individual set of received keypoints.

5.4 Distributed Image Matching

As a result of computing the SIFT feature points, a set of keypoints is derived for each node, that when locally compared against another node's feature points can be used to produce a small set of match points if there is a shared area between their views. Each match point is a set of two x,y co-ordinates representing the corresponding points in each image. Given a set of keypoints from another node, each node can compute the set of match-points that provide information about the overlapping areas, if any of the two images. From the perspective of a distributed task running over the nodes, the question then becomes how to use this information to create an organisation that is locally useful to the task at each node.

It can reasonably assumed that in a wide area network some network based method can be found to divide the cameras into tractable locally interconnected groups, over which a view based topology can be derived. In a TCP/IP based network this could be readily derived from the subnetwork addressing, in a wireless network base stations could be used. As a worst case, GPS or human input could be used to indicate broad geographical groups. Assuming that based on this information a local mesh network has been established between cameras in these groups, then the problem becomes one of determining the camera correspondences within these groups, and then extending this information to the entire network.

For example, consider the constrained environment provided by the Sensored Tile installation at the MIT Media Laboratory. Here a number of tiles have been installed along two walls. (A description of the tiles capabilities is provided in Appendix 1). These tiles can be manually moved into different configurations, but typically occupy the two blocks shown in Figure 5-3.

For applications such as gesture tracking, or displaying a picture over the entire area, the tiles need to be able to automatically determine their position with respect to each other in a way that can then be used by a distributed application running on the tiles. This can be regarded as a highly constrained version of the more general problem of localising cameras in a camera network.

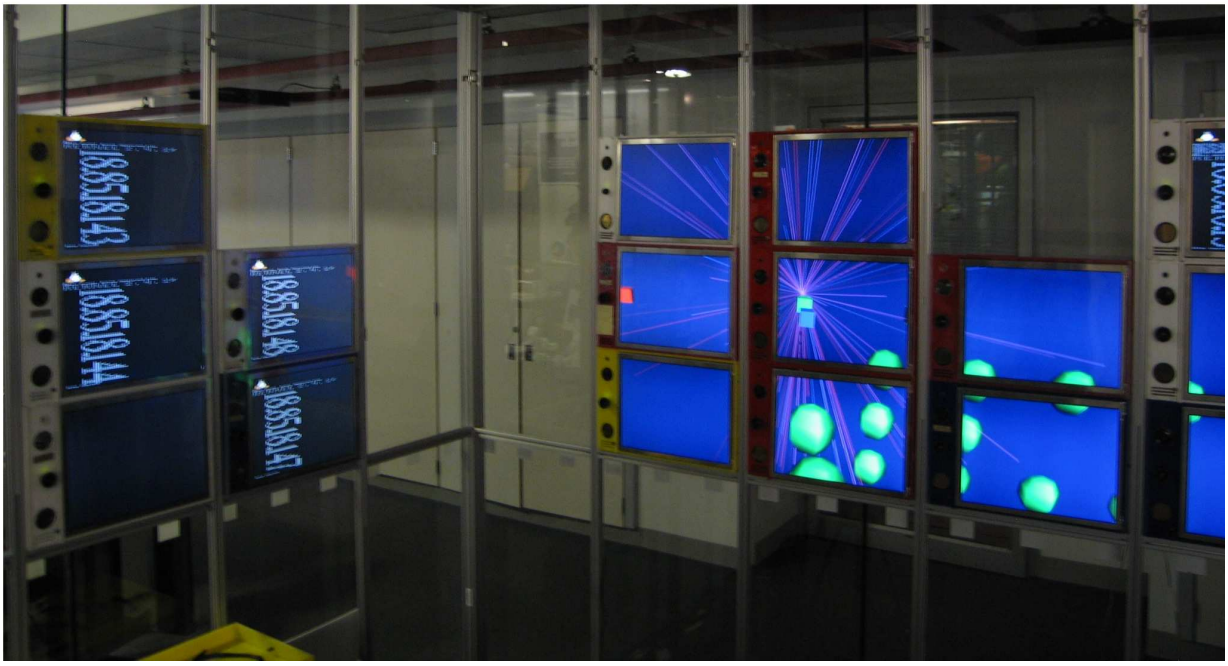

Figure 5-3: Conference Room Tile Installation

5.4.1 Determining relative position within a Camera Grid

The cameras used for the sensored tile have a wide angle field of view, and are laid out in an adjustable grid formation, as shown in Figure 5-3. There is considerable overlap between their views. By using the explicit assumption that the cameras are positioned in a grid, the problem of determining relative positions amongst the cameras can be presented as determining the membership of the horizontal and vertical rows of the grid. To achieve this the array executes the algorithm shown in pseudo-code under Algorithm 1. Group names used by the algorithm are shown in *italics*.

Algorithm 1 causes nodes to establish a local horizontal group if their computed position with respect to the starting Node is approximately horizontal, i.e the differ-

Algorithm 1 Local procedure for relative position in camera array

```
1: Compute SIFT feature points at each local camera, and join the UNASSIGNED group
2: Select a single node to distribute its' keypoints to the UNASSIGNED group
3: for all Received Keypoint sets at each local node do
4:    Compute SIFT matches, and determine relative position with originating node.
5:    if position is Horizontal wrt originating tile then
6:       Join Horizontal Group
7:    else if position is Vertical wrt originating tile then
8:       Join Vertical Group
9:    end if
10:   if Member(Horizontal) OR Member(Vertical) then
11:      Leave UNASSIGNED group.
12:   end if
13: end for
```

ence between the calculated match y points for the local and received keypoints is less than an allowed ε; and a vertical group if they are vertically aligned. Because this is done with respect to a single starting node, and assuming the tiles and their cameras are aligned within the selected error tolerance, the global result of the first pass of the algorithm is two sets of shared group memberships, as shown in Figure 5-4.

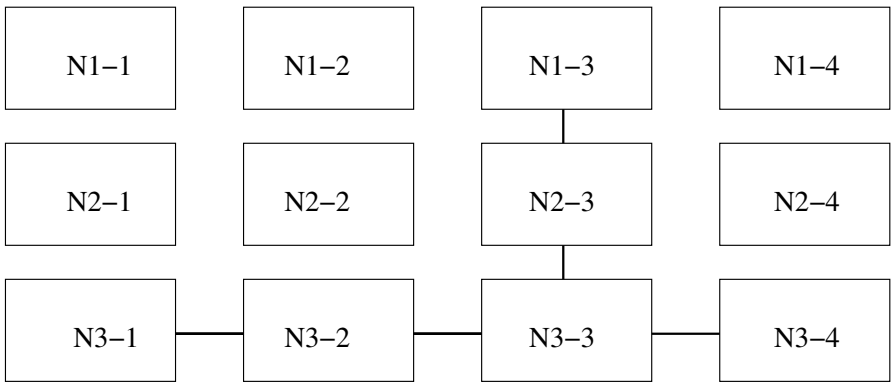

Figure 5-4: Local view of alignment groups in Tile Array

If there were only one wall of tiles, then the simplest way to proceed from this point would be for the starting tile to use the *HORIZONTAL* group and send a command to all tiles in that group to broadcast their Keypoints to the *UNASSIGNED* group.

This would result in all tiles belonging to either the *HORIZONTAL* or *VERTICAL* groups.

With two walls of tiles a slightly more involved procedure is required as outlined in Algorithm 2. Here nodes are repeatedly selected from the *UNASSIGNED* group until there are no nodes left, or all nodes remaining have broadcast their keypoints to the nodes in the *UNASSIGNED* group up to a predetermined limit on the number of broadcasts.

Algorithm 2 Starting node procedure for relative position in Tile Array

```
1: repeat
2:    Select node from UNASSIGNED
3:    Broadcast keypoints of new node to HORIZONTAL
4:    if Node is still a member of UNASSIGNED then
5:       Increment keypoint broadcast count on node.
6:       Randomly select node from UNASSIGNED
7:       Node becomes new starting node.
8:       Node broadcasts its keypoints to UNASSIGNED
9:    end if
10: until No nodes left in UNASSIGNED or keypoint broadcast limit reached
```

After the second algorithm terminates, there will be two independent groups of nodes, with horizontal and vertical group membership providing connections between the nodes. Note that at this point the nodes are not fully connected, the algorithm only guarantees that every node is a member of at least one vertical or horizontal group. Once these partial relationships have been established however, it is straightforward to extend the group memberships at each node, in order to pull in the entire row and column, and for each node to then use the resulting information to determine its grid position in the array.

The result of running this algorithm on all of the tiles on the two separate walls in the conference room is two collections of nodes, each with geometry based links to other nodes in their group, but with no connections between the groups themselves. In most cases for scene analysis this is what is required, but for ancillary tasks such as network management, or collecting debugging information, it is useful to have a unified way of addressing the two groups. This can be obtained quite simply by

recognising the status of 'starting node' within the distributed application, and using it as a group membership requirement for a management group that can be used to route general requests and information to both groups.

It should be noted that the above approach only works if the tiles are reasonably well aligned with each other, and in particular the cameras are positioned in the same perpendicular plane; otherwise the SIFT feature analysis will be performed on images which are effectively cross-eyed, and do not provide the correct information on relative position. The value of ε can be used as a local group member control, and progressively adjusted at each node in order to allow for differences in positioning between the cameras.

Although all tiles will end up being a member of at least one of the two positioning groups, if the grid is not a complete filled rectangle, as for example in the configuration shown in figure 5-3, then simply using hop counts to traverse the rows or columns of the grid will not necessarily provide the correct tile.

5.5 Network Analysis

Work on the theoretical transportation capacity of unorganised large-scale wireless sensor networks has provided a theoretical basis for the unsurprising result that per-node throughput within these networks is severely constrained by the number of nodes within the network[81]. Scaglione[17] presents an analysis of this problem based on bit transmissions in a wireless sensor network, that shows that the total transport capacity of such a network cannot be larger than $O(L\sqrt{N})$ where L is the bit capacity of the links, since it follows from the max flow, min cut theorem[82], that an upper bound on the flow within a network is the capacity of any cut through the network's links. Although Scaglione's analysis is based on the bit transmission capacity between nodes in the network, it can be equally applied to the message processing capacity at each node in a message passing network.

Considered purely as a problem of data transmission, the worst case for a camera network occurs when there are no shared views with any other camera, a state which

without any organisation requires each camera to perform a total of $N-1$ comparisons of its scenes match-points with every other camera. This represents a total load within the network of $N(N-1)$ keypoint exchanges, assuming perfect communications.

In Algorithms 1 and 2 all nodes in the network are treated as initially belonging to a single group of unassigned nodes. One broadcast is made to all the nodes in this group of a single node's keypoints, which may cause nodes to be removed from the unassigned group, and placed in two positional groups, horizontal and vertical. In the example of the sensored tiles installation, where nodes are known to be positioned horizontally and vertically to each other, each broadcast removes all or part of a row and a column of nodes from the unassigned group depending on the amount of view overlap. For the best case, where a complete row is removed, the number of comparisons performed by any particular node in the array is reduced to $O(\sqrt{N})$.

The example of the Sensored Tiles with its useful artificial geometry is clearly a constructed example, but the advantages of being able to introduce a data centric, application specific organisation into the messaging between the nodes are nonetheless apparent. The overhead of the group management behind the algorithm on the system is relatively small, and nodes are only required to be a member of at least one positional group in order to be successfully incorporated into the resulting organisational structure.

In a more realistic environment, it would be expected that there would be a group of cameras with no shared scenes with any other cameras - access to these cameras and their status is incorporated into the algorithm through the *UNASSIGNED* group.

5.5.1 General Case

Consider the more general case of camera group assignment for cameras randomly placed in an environment. As above, the worst case occurs when there are no cameras with shared views. To determine that this is in fact the case in an unorganised environment requires each camera to attempt to match its views with every other camera, producing once again a network load of $N(N-1)$ comparisons.

For this scenario, Algorithms 1 and 2 above, are modified as shown in Algorithm

3. Instead of forming horizontal and vertical groups at each nodes, the nodes form groups based purely on whether or not there is a shared view with other nodes. In

Algorithm 3 Generalised procedure for determining shared views

```
1: Compute SIFT feature points at each local node, and join UNASSIGNED group
2: Select a single node to distribute its keypoints to the UNASSIGNED group
3: At each local node on receipt of keypoints
4: if local view is shared with received view then
5:     Join shared view group with originating node
6:     Distribute local keypoints only within that group
7:     Leave UNASSIGNED group
8: end if
```

the worst case where there are no shared views between the cameras, the algorithm proceeds as follows, each node broadcasts its keypoints to the pool of unassigned nodes, and waits for a determined period for the other nodes to compute their shared match-points. If at the end of this period there are no members in its shared group, it selects a node at random from the unassigned group, and leaves the unassigned group. The new node repeats the process.

When there are no shared views between any of the nodes, the result is eventually no members of the unassigned groups, and a set of nodes with no members in their local shared views group. Analysing the message load on the system, it can be observed that after each node exchanges its keypoints with the nodes in the unassigned group it is removed from the set of nodes performing the comparisons. This results in the total number of keypoint exchanges performed within the network being reduced from $N(N-1)$ to:

$$\begin{aligned} \sum_{i}^{N} \alpha_k &= \frac{1}{2} n(\alpha_1 + \alpha_n) \\ &= \frac{n(n-1)}{2} \end{aligned} \tag{5.3}$$

where $\alpha_k = (n - i)$. Effectively, introducing a simple organisation of nodes into two groups, has removed the overhead of comparisons of match sets being performed identically at both match set's nodes.

This is the worst case limit on the number of comparisons that must be performed by the network which is however still above the $O(L\sqrt{N})$ comparisons available for any appreciable size of N. As an example, consider the situation shown in Figure 5-5.

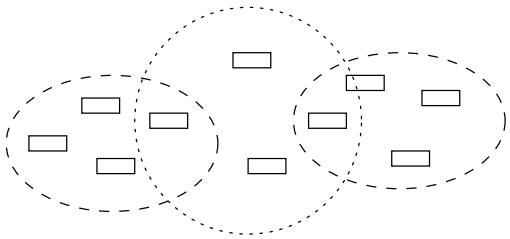

Figure 5-5: Example of camera groups in an unstructured environment

If the nodes in fact have the shared viewpoints as indicated by the dashed-circles, then executing Algorithm 3 results in three groups of shared views, with precise membership of the groups for some nodes being indeterminent since the order of broadcast will determine initial membership of shared groups.

As a side effect of the initial algorithm, all members within each independent group have a copy of the match-point's of all other members within that group. If a member of each group is then chosen at random, each group's set of match-points can be compared with the other groups in order to identify members with shared group membership. Although this increases the total number of comparisons being performed within the network, it does so in a form that is tractable to controlled engineering - for example in a camera network the node in the group with the least current processing node could be chosen to perform the comparisons.

5.5.2 The Effects of Group Organisation on Network Performance

Consider the different values for link processing capacity (L), and the number of nodes in a network (N), shown in Figure 5-6. This figure demonstrates the underlying networking issue for any distributed application which requires some degree of peer-to-peer communication between its nodes. As remarked by Gupta[81] amongst others,

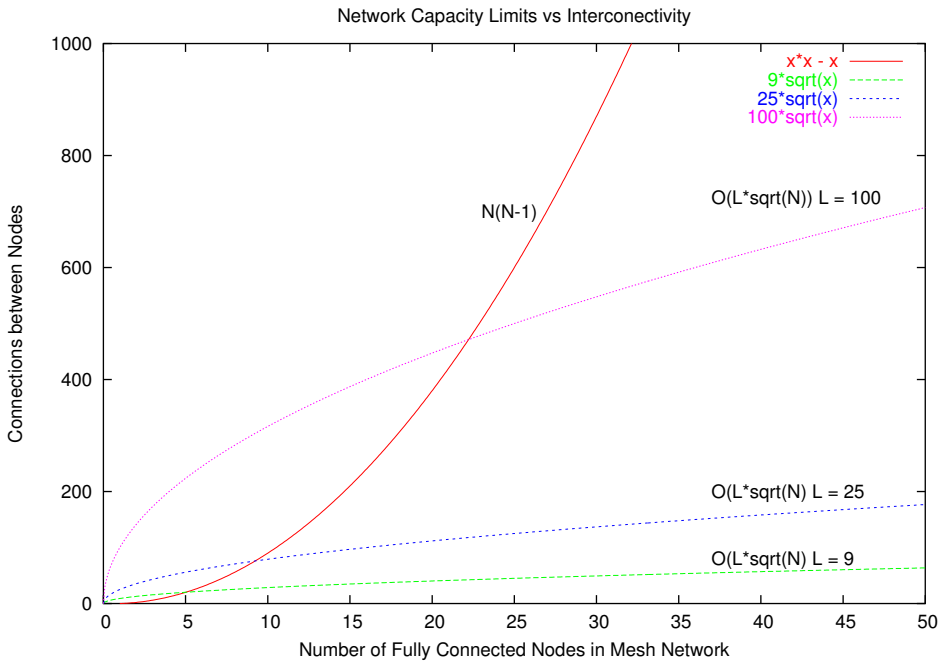

Figure 5-6: General limits on Organisational Size

in a fully connected network adding nodes increases the throughput capacity of the network up to a limit, but beyond that the impact of every additional node is a very rapid decrease in total network throughput. However as shown, for low values of N and reasonable values of L, link capacity dominates so that there is sufficient capacity in the network for all the peers to communicate.

This situation applies equally to any wholly or partially connected group within a larger network of nodes. As long as the load on each node within the group can be kept on the left hand side of the graph where the influence of L dominates, the group will have sufficient communication resources to handle its inter-node communications. However as the size of the group increases beyond that point, its overall communication performance will drop very quickly as delayed message processing triggers dropped messages, retries, and other inefficiencies that increase the overall load on the network. It is not just that adding additional nodes slows down communication within the group, but that it in fact reduces performance of the whole group below what it was prior to the introduction of the new nodes. This phenomena

has been widely observed, and is known in software projects as Brooks' law, "Adding manpower to a late software project makes it later" [83].

The networking capacity of a node for which L is an approximation, depends on both the available bandwidth and the local node processing capacity, which in turn depends on the algorithm or applications being run. Consequently for applications with any significant CPU requirement it is often very low.

For example, the average size of a SIFT Keypoint set is 15k bytes, which with a receive buffer of 65k on the tiles, and an approximately 45s processing times, puts $L \approx 4$. If a cooperative interconnected processing group is established of nodes running this algorithm, then even with only four nodes and a resulting total of 12 inter-connections, the actual limit on efficient network throughput is of $O(10)$ inter-connections. Above this limit network communications degrade rapidly, with a consequent effect on the application. This has been born out practically with experience with message loss within the tile environment. Notice from the graph that even substantially larger values of L than are currently available technologically, will not greatly increase the effective limit on the maximum number of completely interconnected nodes in any network.

However, if total communication between all nodes is not possible for large numbers of N, or even small numbers if their inter-node communication requirements are fairly high, partial connectivity can be arranged, provided that this connectivity is managed so that the load on individual nodes in the system is maintained below the $O(L\sqrt{N})$ limit.

So any system requiring peer-to-peer communication that contains sufficient nodes to exceed the local limits on communication must be sub-divided into separate groups in order to maintain communication throughput. For a static system this can eventually be achieved centrally, but in any dynamic system the argument above applies equally well to the exchange of data required to determine the necessary organisation - depending on the messaging and processing requirements of the application - but inevitably for large values of N, the central organising point will be overwhelmed. This implies that in any instance where a dynamic or adaptive organisation is re-

quired from any significant number of nodes, it can only be maintained by using local organisation. Further, to maximise efficient use of the available resources, connectivity at each node must be kept below the $O(L\sqrt{N})$ limit. It follows directly from the argument above, that this can only be successfully achieved on a purely local basis. [1] Notice that the impact of this effect in practice can be quite subtle. An organisation may function just below the threshold without experiencing any communication problems, absorb a small number of extra nodes which pushes it over the threshold, and then start to experience general delays in communication processing without the cause of the problem being immediately obvious.

Hence the conditions under which the kind of group based organisation described here are required. Note that as proposed by this thesis, groups are formed purely locally, with membership requirements being partially adaptable at each node, on a basis that can include control over the number of members allowed in that local node's group-view. As long as each node locally maintains its group size underneath the $O(L\sqrt{N})$ limit, additional nodes can be introduced into the application without adversely effecting throughput. It is also possible for the underlying group management level to provide statistics on the memberships present at each node, which makes problems which occur from applications attempts to exceed these limits more transparent.

Perhaps most importantly, by adjusting allowable group sizes, and by controlling multiple group memberships at each node, the overall connectivity and consequent access to information, within the algorithm, application or network can be maintained for numbers of nodes that considerably exceed the limit imposed on fully connected peer-to-peer applications.

[1] This would seem to suggest that centrally planned economies are ultimately doomed to fail as societies grow in size and complexity, regardless of any ideological considerations, simply because the required information flow within the society will cause these limits on connectivity to be exceeded.

Chapter 6

Conclusion

Work in distributed applications often implicitly assumes that components in the application are organised into related groups, without exploring how in a large and dynamic network of nodes this organisation could be created. When small numbers of nodes are involved in a static application it is entirely appropriate that this structure be designed into the system. However for systems constructed of larger numbers of nodes, and in particular systems where multiple distributed tasks may be in execution on a dynamic and changing basis a static approach will not work. The significance of the approach described here, is that it describes an automatic and localised way of determining task relevant organisations in such a system.

There is a large class of distributed applications where the standard client-server approach to distributed organisation cannot be used because of the need for significant peer-to-peer information exchange between individual nodes of the application. Unfortunately pure peer-to-peer systems scale extremely badly with respect to the number of nodes within the application. Quite how badly they scale depends in part on the capacity of the underlying network, but largely on the individual capacity of each node to receive and process network communications. This can be broadly characterised by a simple order statement, $O(L\sqrt{N})$, where N is the total number of nodes in direct communication with each other, and L is their individual capacity for network communications.

It is important to realise in this context, that L is not simply a measure of the

bandwidth available to the node, but also the cost of any consequent processing on receiving information from other nodes. A 1 byte message that creates several seconds of processing at each node may well be worse in terms of overall network overhead than a far larger message which creates mere micro-seconds of associated processing. Consequently the precise value of L is dependent in large part on the distributed algorithm being run by the nodes. For most distributed systems of any reasonable complexity though, it is surprisingly low. In human society for example, there is considerable evidence that it lies in the 7-15[72] range, with the combination of low power embedded computers and SIFT algorithm used in this thesis it was approximately 3-4 nodes. Whatever the actual value is though, there is no escaping the fundamental problem faced by peer-to-peer applications, that without any organisation the overall network load increases near exponentially as nodes are added. Sooner or later, the centre cannot hold.

This creates a subtle trap for many distributed applications. With small numbers of nodes the available network capacity, $O(L\sqrt{N})$, exceeds the demand on the network, $O(N(N-1))$ fairly comfortably. However as nodes are added to the network, this capacity is exceeded. Unfortunately the result of this is that overall traffic within the network increases, due to the extra processing and repeated communication caused by individual nodes locally dropping messages. Consequently the overall capacity of the network in fact drops as the throughput threshold is exceeded. This creates an anomalous situation where adding a small number of nodes to a network at its throughput threshold, may well have a network wide effect that dramatically lowers overall throughput. [1]

So there are in fact fairly severe limits to the size of communicating groups, and unpleasant consequences for the overall application when those limits are exceeded. This then limits the number of nodes a distributed algorithm with any communication between its nodes can use, and in most cases these limits will be fairly low. Even

[1]The precise size of group that a given algorithm can support is clearly dependent on a number of factors. But it can be seen from the examples provided that not infrequently in real world applications, the size of the system used to test the application before deployment will be below this limit, and that of the deployed system above it. Hence the trap.

worse, from the perspective of system control, there may not be much warning as the system grows from a size where it is stable and functions without problems, to one where its performance is suddenly degraded network wide.

The solution to this problem proposed here is to introduce organisation between the nodes based on local information, and allow the nodes to individually manage this organisation so that overall connectivity within the application is maintained below the critical point. Critically it is shown in this thesis, that it is not necessary in such a system that group membership is maintained consistently at each node. Besides the known issue presented by the Fisher consensus problem, this is important because any attempt to determine the appropriate organisation centrally will run into the same set of problems as described above - for some value of N, the central point will fail to keep up with the associated processing and communications load. In a large network with multiple task groups and dynamic associations between them, only the local node can be guaranteed to have all the information necessary to manage its group memberships such that the network's overall connectivity stays below the critical threshold.

Network throughput itself can be maximised using this scheme, if nodes are allowed to be members of multiple groups. If two groups of separate nodes are connected by a shared membership of one node, and all nodes are still operating below their maximum number of supported connections (L), it follows that if the nodes have similar values for L the nodes which do not have a shared membership have at least one extra link available to the network. Peer-to-peer based applications can at least theoretically take advantage of this extra capacity. By contrast, any client-server based organisation where the individual clients are capable of more communication than is required to keep up with the server clearly cannot. This helps to explain why the peer-to-peer file sharing applications have been so successful, they were able to take advantage of the increased performance of personal computers to provide additional network capacity.

Almost any kind of group organisation, can in fact provide a dramatic reduction in overall messaging within the network, and thereby allow the throughput of the entire

system to be protected, provided that discovering and maintaining that organisation does not in and of itself impose a prohibitively large cost in terms of network processing and communications. Consequently how group membership views themselves are maintained becomes a critical part of the problem.

It is not that the Fischer consensus problem presents an insurmountable barrier to constructing cooperative applications, rather that attempting to solve it, or perhaps more precisely program around it, leads to an unpleasant degree of complexity in what is already a difficult arena in which to construct applications. This complexity manifests itself in a number of ways at different levels of the system, but one of the more important is in terms of additional load on the network. When these issues are not properly addressed, as was the case in the first generation of peer-to-peer file sharing applications, then the size of the resulting application is severely restricted. Moreover, such applications running in an environment such as the Internet can have a global effect on all application traffic using the underlying transmission equipment.

The amount of communication that any particular node can support with other nodes in the network is a combination of its own individual capabilities and the load caused by communicating with other nodes. For large scale distributed applications this can only be measured locally, and consequently can only be adjusted locally. As soon as the number of nodes in a distributed application exceeds the limits on peer-to-peer communications for its own peculiar combination of network and local processing requirements, some kind of organisation must be introduced to allow these peer-to-peer communications to be appropriately constrained. For small numbers of nodes this can be and often is performed centrally, but this approach fails for dynamic organisations beyond a certain size.

By constructing group membership as a completely local problem, it becomes possible to design ad hoc organisational structures amongst much larger numbers of nodes. These organisations may often look extremely similar to the centrally imposed ones, but are in fact arising from decisions based on local data. They can also be transient within the network as different tasks are presented and completed. Perhaps most importantly this permits local group size and memberships to be independently

adjusted to allow the creation of hierarchies of group organisations, which allow the limits on peer-to-peer communications to be at least partially circumvented.

Chapter 7

Future Work

Large scale sensor networks present considerable challenges in terms of constructing, and in particular debugging, their applications and algorithms. The only computer systems that provide a source of comparison in terms of complexity and also of potential scale are today's packet switched networks, and these are themselves the result of several decades of research and development. Viewed as an application, there is certainly a great deal of overlap between the evolution of packet switched networks as various issues of scale were successfully tackled, and the current development of distributed peer-to-peer applications running within them. New researchers to this field would probably find it useful to spend some time becoming familiar with the historical research and development of these networks.

7.1 Smart Camera Networks

The groupmgr process developed for this work is in effect a specialised, albeit somewhat limited, software router and message handler. There are two main areas of enhancement currently needed. One is to provide a built in way to organise the underlying communications within a large network of groupmgr processes, in a similar way to that currently used by routers within the Internet. Ideally this would use the existing group mechanisms, but be sensitive to underlying network conditions such as bandwidth, latency and load. Simulation of how local adjustment to these conditions

changes overall performance may well be useful here.

The other deficiency is with respect to more complicated group requirements. As far as possible, group membership considerations are handled by the groupmgr process. This is trivial for simple requirements such as distance, or range, but algorithms such as SIFT require computation results performed by the application. Ideally, it should be possible for the application to provide a dynamically loaded function for execution by the groupmgr process to determine membership, as well as the existing simpler criteria.

Whether or not support for group memberships as described in this thesis could ever be turned over to the underlying network switching layer is also an interesting question.

7.2 Distributed Social Systems

Initially, work for this thesis involved an examination of human and animal group forming in order to try and gain insight into the structure of naturally occurring groups and their behaviour. It later became apparent that there were structural problems encountered by asynchronously communicating nodes in a larger network that appeared to be also manifesting themselves within human society. Human social and economic groups seem to share some characteristics with the groups described in this thesis, in particular the problem of group destruction and the implications of the resulting network load from attempting to change a group's global function.

As the network analysis of the simple examples presented here shows, there are considerable advantages in terms of the communication overhead experienced by individual group members, and ultimately for communications within the entire network, to almost any form of group based organisation. Consequently it would be expected that mechanisms that create groups, and control the amount of communications generated between groups, would have been subjected to evolutionary selection once humans evolved speech. Indeed, on reflection it would seem fairly obvious that communication issues between autonomous, message based cooperating computer processes

would manifest themselves in human society. It is after all the result of large numbers of autonomous, message based nodes attempting to perform quite considerable amounts of cooperation for many millennia, with varying degrees of success.

Although there has been considerable recent interest in scale-free network phenomena within human communication networks, there appears to be relatively little work to date on applying other insights from packet switched networking such as latency, communication topologies, and the limits on network throughput discussed here. Clearly though, these limits constrain the effectiveness of human organisations, and deserve to be better explored.

Communications within human networks have extremely long latency compared to computer networks, and as such temporal considerations become more significant for the resulting systems. So the introduction of new communication technologies, such as reliable postal services which reduce latency within the network, or provide broadcast mechanisms to large numbers of nodes, such as radio, could be expected to be extremely disruptive purely on network communication principles. Similarly rapid increases in group size or population, or attempts to impose a new group structure centrally without reference to the existing communications organisation, may create network wide instability as group organisations adjust to the new conditions independent of any other considerations. If network effects such as this are occurring it may be possible to detect them from an historical examination of regional population, social organisations and the extant communication technologies.

In computer networks, centralised systems can be extremely efficient when they are operating within available network capacity, and will degrade badly when operating beyond it. One aspect of the group organisation described here is that it provides a mechanism for repeated locally centralised organisations to be built, thereby allowing the entire network to stay within its communication threshold. Looked at from this perspective, the fractal nature of military organisation, and many similar social structures, can be understood as a solution to network communications problems.

A better understanding of these problems as they apply to human networks may well assist in improving these structures, or at least understanding better the inherent

dangers in trying to rapidly change them. A considerable amount of progress has been made in understanding stability and traffic issues in packet switched networks, which may well be usefully applied to understanding the constraints and behaviour of large scale human social systems.

Chapter 8

Acknowledgements

In no particular order:

My committee for their wise and constructive guidance, my advisor Mike Bove in particular for realising the significance of this work long before i did.

The Media Lab 'Sensor Network Crowd' - Ari Benbasat, Bill Butera, Dean Christakos and Josh Lifton, Ari in particular for his extraordinary Latex skills and Dean for several useful discussions on the historical development of organised Christianity.

The MIT UROPS who have helped with Tile and Robot design and construction, including Eric Varady, Brian Mullins, Sam Kesner, Jose Martinez and Stephen Oney.

Diane Hirsh, Jim Barabas and Arnaud Pilpre, who have contributed to work on the sensored tile and truly appreciate it for what it is.

The Media Lab's long suffering, and extremely helpful network administrators Jane Wojcik and Will Glesnes.

And last, but of course by no means least, my parents.

This research was sponsored in part by the Digital Life, Things That Think and CELab consortia, a joint research program with the Information and Communications University of Korea, as well as the unfortunate state of current teaching in applied Information Theory and Probability.

Appendix A

Implementation

A.1 Smart Camera Network Platform

For the work in this thesis a smart camera network consisting of a total of 25 independent smart cameras was constructed. Communication between cameras is performed using 802.11b wireless networking. The majority of the cameras are located in a single room, fixed statically with two groups at right angles to each other, whilst a small group of 4 mobile robotic cameras are located in a separate area.

View from 3 overhead robots attempting to cooperatively track bright yellow objects.

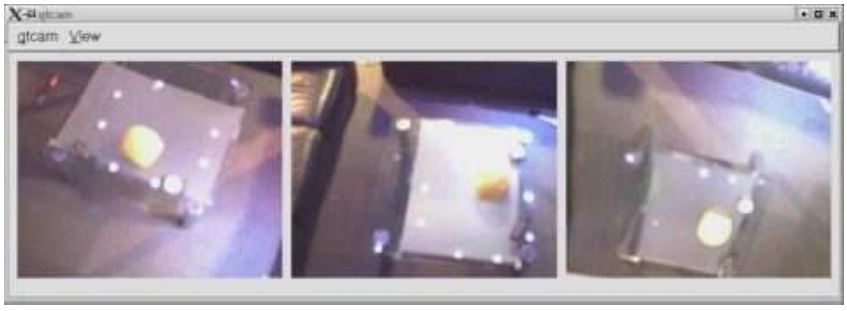

Figure A-1: View from the Mobile Robots

Both systems in the network, the Sensored tile and the Eye Society robot, are designed around the Applied Data Systems XScale Bitsy device, a 400MHz PDA sized board, with 64Mb of on-board memory and a wireless PCMCIA card, which provides a USB hub to which a Logitech Phillips Web cam is connected. The board runs a standard Linux kernel using the ARM patches, and the camera is supported using the public domain pwc driver.

The XScale was chosen for a number of reasons, size and its relatively low cost being significant ones. Although it is somewhat low powered, this was was seen as an advantage in a test network, as it was hoped, and this indeed proved to be the case, that this would provide a closer approximation to effects that would be present in a larger system. For similar reasons, the cameras are maintained as relatively anonymous devices on the network, IP addresses for example are dynamically assigned through DHCP, although for logging purposes the MAC address is occasionally used as an unique identifying characteristic for each tile. Programs for the platform are compiled using the gcc cross compiler, and are loaded by the devices over an NFS mounted file system.

A.2 Software Architecture

The underlying software architecture used for group communication between the cameras is shown in Figure A.2. It is essentially a small footprint message router, and was deliberately developed to be applicable as a generalised solution. Communications between the cameras and debugging and monitoring applications running on local workstations for example, also use the groupmgr process shown below.

Communication with other nodes is handled by a separate groupmgr process running on each node, which maintains a shared memory area for local communication with processes on the node. Communication is message based with messages first being written to the shared area memory for handling by the groupmgr or local process. The groupmgr process handles all group membership communication and decisions, determining which other nodes are members of the local group, and routeing messages

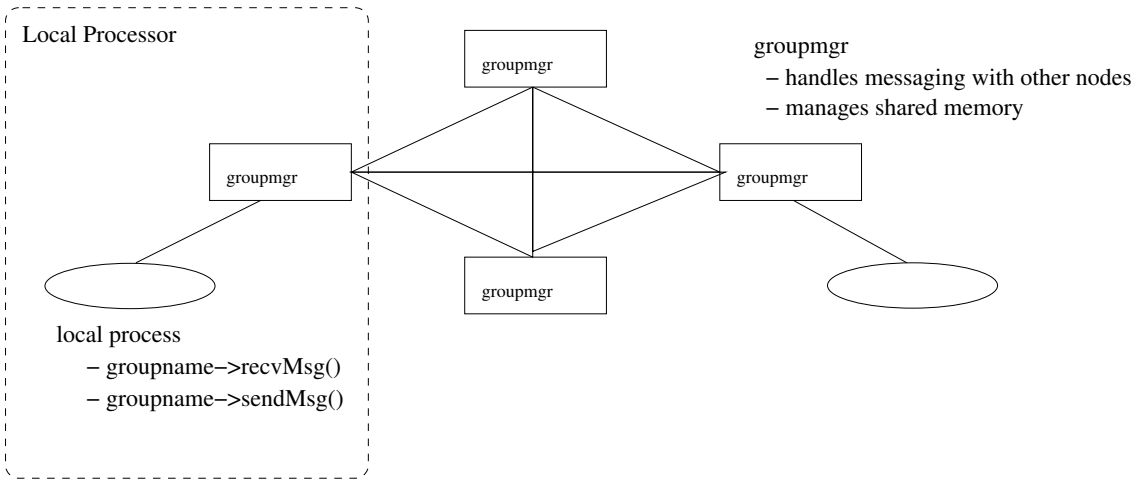

Figure A-2: Software Architecture

accordingly. It equally handles messages originating from other nodes, and determines whether they should be held in the shared memory area for local processes.

The shared memory area is maintained as an overwriting FIFO buffer of messages, with messages being accessed by the local processes on demand. In the event of overflow, earlier messages are overwritten, with error statistics being available to the local process also through shared memory.

A complete set of classes is provided to allow the local process to create groups, and send and receive messages to them. All messaging is done in ASCII, primarily to avoid marshalling problems, and also for ease of debugging in a research environment.

A.3 The Sensored Tile

The tile is a multi-sensor display device, with a 17" LCD panel, which provides access to multiple sensors, including a microphone, camera, and smoke detector. The SmartIO chip on ADS Bitsy is used to provide analog and digital inputs for the sensors, and the camera is accessed using a USB hub. Wireless communication(802.11b) is provided using a PCMCIA card. Figure A-3 shows the components in an opened tile.

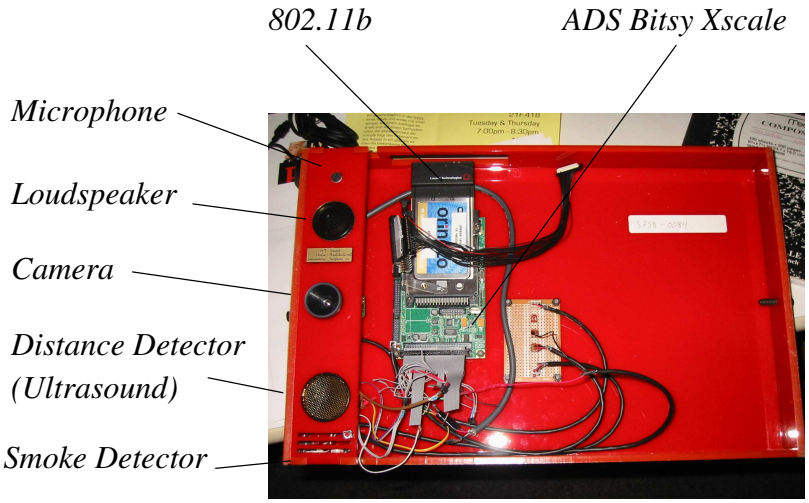

Figure A-3: Tile Internals

The tiles are installed in the Media Lab's Garden Conference room along two walls at right angles to each other, providing two distinct views for the cameras. As shown in figure X they are arranged in a grid, with their cameras being approximately horizontally and vertically aligned with each other. As shown in Figure X, each camera has an overlapping view of approximately 80% of the scene with its immediately adjacent tiles.

A.4 Eye Society Robot

The robots use the same embedded processing board as the tiles, but are also equipped with motors for movement along the overhead track, as well as the ability to pan and tilt their camera head. The pan and tilt mechanism is provided using two stepper motors, and allows 180° motion in two directions, as shown in Figure A-4.

A.4.1 Camera Capability

The web cams used for both the tiles and the robots are Logitech cameras based on the Philip's SAA8116HL02 chip. The camera supports a range of resolutions up to and including 640x480, and provides a colour picture in YUV format. On-board memory considerations make 320x240 the highest practical resolution for application

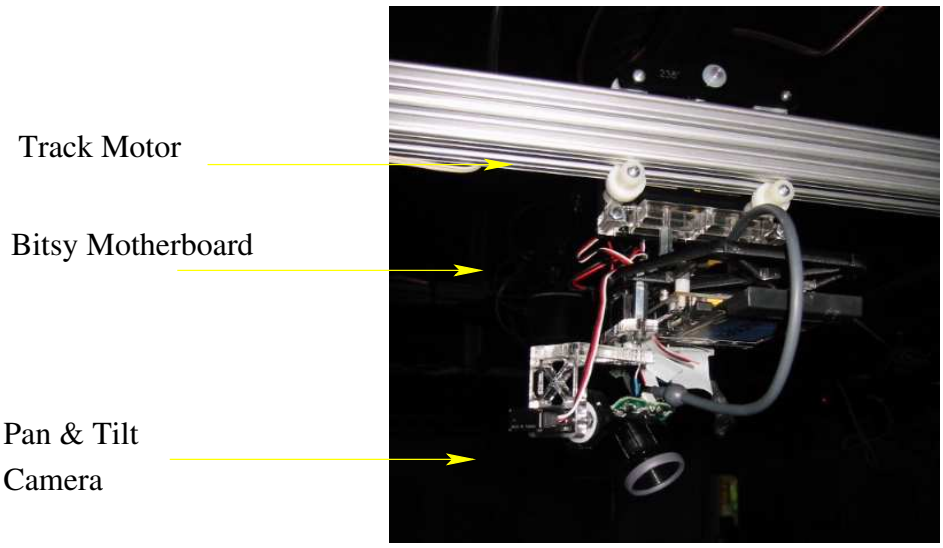

Figure A-4: Eye Society Robot on overhead track

use. On the Bitsy XScale architecture the cameras are capable of streaming at approximately 3-4 frames per second at 320x240 full colour resolution over USB by using on-board compression of the frame. They are reasonable quality cameras for their price, approximately $60, and have become quite popular with amateur astronomers for computer controlled exposures. Their main distinguishing characteristic is that they have a very wide field of view, and generally mediocre optics. Critically for this project, there is a full featured driver for these cameras already in the public domain.

A.4.2 Communication Capability

Standard TCP/IP communication is provided over an 802.11b network, and both devices provide Internet connectivity using telnet and ssh. Owing to a bus contention conflict between the USB and PCMCIA controllers on the Bitsy, the maximum streaming rate of the devices over the network is approximately 2 fps at a resolution of 320x240. Shared wireless networking availability is also an issue for some of the devices, since they are sufficiently concentrated to cause problems for the local hub if they simultaneously start to put a lot of data.

Some other idiosyncrasies have been observed with communication performance on this platform. Although the multicast protocol is nominally supported, under

very light traffic loads data losses of 50% or more have been observed, making it impractical for use, even though it would have been well suited for the group protocol proposed here.

There is no permanent on-board storage on the device, but up to 32MB is available for short term storage. Both devices have access to permanent file systems over NFS as required.

A.4.3 Operating System

The Linux kernel on the tiles and the robots is the 2.4.19 ARM branch. This is held in flash memory on the Bitsy XScale, and boots to terminal in approximately 30s. Owing to on-board memory constraints, and also for development speed the camera driver and software are loaded across the network, rather than being stored locally, although it would be possible to hold these in local flash memory if required.

A.4.4 Performance

The current generation of the Bitsy XScale is a 400Mhz device with no on-board floating point support. This has an adverse effect on the speed of floating point calculations, although this can often be ameliorated using precomputed tables, especially with trigonometric calculations.

Bibliography

[1] Vilmos Csányi. Single-person groups and globalisation. *The Hungarian Quarterly*, XLIII(167), Autumn 2002.

[2] Clive Norris and Gary Armstrong. *The Maximum Surveillance Society - The rise of CCTV*. Berg, 1999.

[3] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *First International Conference on Peer-to-Peer Computing*, pages 99–100, August 2001.

[4] Sun Tzu. *The Art of War*. 500BC.

[5] Norman Dixon. *On the Psychology of Military Incompetance*. Pimlico, 1976.

[6] T. Kanade, P. Rander, and P.J.Narrayanan. Virtualized reality: Constructing virutal worlds from real scenes. In *IEEE Multimieda*, volume 4(1), pages 34–47, 1997.

[7] M. Brown and D. G. Lowe. Recognising panoramas. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1218–1225. IEEE, 2003.

[8] E. W.Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the ACM*, 17:643–644, 1974.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60(2)*, pages 91–110, 2004.

[10] Wayne Wolf and Burak Ozer. Smart cameras as embedded systems. *Computer*, pages 48–53, 2002.

[11] W. Caarls, P. Jonker, and H. Corporaal. Smartcam: Devices for embedded intelligent cameras. In *PROGRESS 2002*, 2002.

[12] Tiehan Lv, I Burak Ozer, and Wayne Wolf. VLSI architectures for distributed smart cameras. *Computer*, 35:48–52, September 2002.

[13] Michael Bramberger, Josef Burnner, Bernhard Rinner, and Helmut Schwabach. Real-time video analysis on an embedded smart camera for traffic surveillance. *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 174–181, May 2004.

[14] Greg Kogut, Mike Blackburn, and H. R. Everett. Using video sensor networks to command and control ground vehicles. In *AUVSI Unmanned Systems in International Security (USIS '03)*, September 2003.

[15] Jacky Mallett and V. M. Bove Jr. Eye society. In *Proceedings of Multimedia and Expo 2003*, volume 2, pages 17–20, July 2003.

[16] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[17] Anna Scaglione and Sergio D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 140–147, New York, NY, USA, 2002. ACM Press.

[18] Yechiam Yemini. Distributed sensors networks: An attempt to define the issues. *Distributed Sensor Nets*, pages 53–60, 1978.

[19] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless microsensor network models. *ACM Mobile Computing and Communications Review*, 6:28–36, 2002.

[20] Philip Levis, Sam Madden, David Gay, Joseph Polastre, Robert Szewczyk, Alec Woo, Eric Brewer, and David Culler. The emergence of networking abstractions and techniques in TinyOS. In *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation(NSDI 2004)*, 2004.

[21] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. Sastry. Distributed control applications within sensor networks. *Proceedings of the IEEE, Special Issue on Sensor Networks and Applications*, August 2003.

[22] Hsin-Yuan Huang, A. Khendry, and T. G. Robertazzi. Layernet: a self-organizing protocol for small ad hoc networks. *Aerospace and Electronic Systems, IEEE Transactions on*, 38, April 2002.

[23] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.

[24] A. J. Ganesh, A-M. Kermarrec, and L. Massoulie. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52:139–149, february 2003.

[25] J. M. Kleinberg. Navigation in the small world. *Nature*, 406:845, 2000.

[26] H. Jin Kim, Rene Vidal, David H. Shim, Omid Shakernia, and Shankar Sastry. A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. In *40th IEEE Conference on Decision and Control*, pages 634–639, 2001.

[27] Luca Schenato, Songhwai Oh, Shankar Sastry, and Prasanta Bose. Swarm coordination for pursuit evasion games using sensor networks. In *International Conference on Robotics and Automation*, April 2005.

[28] Mohan Manubhai Trivedi, Kohsia Samuel Huang, and Ivana Mikic. Dynamic context capture and distributed video arrays for intelligent spaces. In *IEEE Transactions on Systems:Man and Cybernetics - Part A: Systems and Humans*, volume 35, January 2005.

[29] J. J. Garcia-Luna-Aveces K. Obraczka, R. Manduchi. Managing the information flow in visual sensor networks. In *The fifth International symposium on Wireless Personal Multimedia Communications*, volume 3, pages 1177–1181, October 2002.

[30] Wayne Wolf, Burak Ozer, , and Tiehan Lv. Architectures for distributed smart cameras. In *ICME 2003, International Conference on Multimedia*, volume 2, pages 5–8, 2003.

[31] Purushottam Kulkarni, Deepak Ganesan, Prashant Shenoy, and Qifeng Lu. The case for multi-tier camera sensor networks. In *International Workshop on Network and Operating Systems Support for Digital Audio and Video*, November 2005.

[32] Dhanya Devarajan and Richard J. Radke. Distributed metric calibration of large camera networks. In *Broadnets 2004 (Workshop Papers)*, 2004.

[33] W. E. Mantzel, Choi Hyeokho, and R. G. Baraniuk. Distributed camera network localization. In *The Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1381–1386, November 2004.

[34] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–23, March 1997.

[35] Brian P. Gerkey and Maja J Matarić. Multi-Robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures. In *Proc. of the IEEE Intl. Conference on Robotics and Automation (ICRA)*, May 2003.

[36] L.E. Navarro-Serment, J. M. Dolan, and P. K. Khosla. Optimal sensor placement for cooperative distributed vision. In *Proceedings of ICRA '04 Intelligent Robots and Systems*, volume 1, pages 939–944, 2004.

[37] V. Isler, J. Spletzer, S. Khanna, and C. J. Taylor. Target tracking with distributed sensors: the focus of attention problem. In *Proceedings of Intelligent Robots and Systems*, volume 1, pages 792–798, October 2003.

[38] Ram Dantu and Sachin P. Joglekar. Collaborative vision using networked sensors, 2004.

[39] M. Frans Kaashoek and Andrew S. Tanenbaum. An evaluation of the amoeba group communication system. In *International Conference on Distributed Computing Systems*, pages 436–448, 1996.

[40] K. Birman, B. Constable, M. Hayden, J. Hickey, C. Kreitz, R. Van Renesse, O. Rodeh, and W. Vogels. The Horus and Ensemble projects: accomplishments and limitations. In *DARPA Information Survivability Conference and Exposition*, volume 1, pages 149–161, 2000.

[41] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet:A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46, 2001.

[42] D. Plonka. An analysis of napster and other IP flow sizes. *Network Analysis Times*, April 2001.

[43] N. B. Azzouna and F. Guillemin. Impact of peer-to-peer applications on wide area network traffic: an experimental approach. In *Global Telecommunications Conference*, volume 3, pages 1544–1548, November 2004.

[44] Peter Biddle, Paul England, Marcus Peinado, and Bryan Willman. The darknet and the future of content distribution. In *2002 ACM Workshop on Digital Rights Management*, November 2002.

[45] E. P. Markatos. Tracing a large-scale peer to peer system: an hour in the life of gnutella. In *Cluster Computing and the Grid 2nd IEEE/ACM International Symposium*, pages 56–65, May 2002.

[46] R. Schollmeier and G. Schollmeier. Why peer-to-peer(p2p) does scale: An analysis of p2p traffic patterns. In *Proceedings of the Second International Conference on Peer-to-Peer Computing*, pages 112–119, September 2002.

[47] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *IEEE/ACM Transactions on Networking*, volume 11, pages 17–32, February 2003.

[48] Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, 2003.

[49] EdmundH. Durfee. Partial global planning: A coordination framework for distributed hypothesis formation. In *IEEE Transactions on Systems, Man and Cybernetics Vol 21, No. 5*, pages 1167–1183, Sept/Oct 1991.

[50] Reid G. Smith and Randell Davis. Applications of the contract net framework: Distributed Sensing. In *Distributed Sensor Nets*, pages 12–20, 1978.

[51] Weiming Hu, Toemoi Tam amd Liang Wang, and Steve Maybank. A survey on visual surveillance of object motion and behaviors. In *IEEE Transactions on Systems, Man, and Cybernetics - Part C:Applications and Reviews*, volume 34, pages 334–352, August 2004.

[52] Lucio Marcenaro, Franco Oberti, and Gian Luca Foresti. Distributed achitectures and logical-task decomposition in multimedia surveillance systems. In *Proceedings of the IEEE, Volume 89 Number 10*, October 2001.

[53] P. Remagnino, J. Orwell, D. Greenhill, G. A. Jones, and L. Marchesotti. An agent society for scene interpretation. *Multimedia Video-based Surveillance Systems: Requirements Issues and Solutions*, pages 108–117, 2000.

[54] Omead Amidi Robert T. Collins and Takeo Kanade. An active camera system for acquiring multi-view video. In *International Conference on Image Processing*, pages I–517 – I520, 2002.

[55] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti. Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. *Signal Processing Magazine*, pages 38–51, 2005.

[56] Deepak R. Karuppiah, Zhigang Zhu, Prashant Shenoy, and Edward M. Riseman. A fault-tolerant distributed vision system architecture for object tracking in a smart room. *Lecture Notes in Computer Science*, 2095, 2001.

[57] C. Micheloni, G. L. Foresti, and L. Snidaro. A network of co-operative cameras for visual surveillance. In *Vision, Image and Signal Processing*, volume 152, pages 205–212, April 2005.

[58] C. Pinhanez and A. Bobick. Intelligent studios: Using computer vision to control TV cameras. In *Proceedings of IJCAI'95 Workshop on Entertainment and AI/Alife, Montreal, Canada.*, pages 69–76, 1995.

[59] Dirk Focken and R. Stiefelhagen. Towards vision-based 3-d people tracking in a smart room. In *Fourth IEEE Proceedings on Multimodal Interfaces*, pages 400–405, October 2002.

[60] G. L. Foresti and L. Snidaro. *A Distributed sensor network for video surveillance of outdoors*. Kluwer Academic Publishers Group, 2003.

[61] Anurag Mittal and Larry S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision 51(3)*, pages 189–203, 2002.

[62] Chang Hong Lin, T. Lv, W. Wolf, and I.B.Ozer. A peer-to-peer architecture for distributed real-time gesture recognition. In *ICME 2004: IEEE International Conference on Multimedia*, volume 1, pages 57–60, June 2004.

[63] Daniel J. Dailey, F. W. Cathey, and Suree Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. In *IEEE transactions on Intelligent Transport Systems*, volume 1, Issue 2, pages 98–107, 2000.

[64] C. J. Taylor, Jitendra Malik, and Joseph Weber. A real-time approach to stereopsis and lane-finding. In *Intelligent Vehicles*, pages 207–212, 1996.

[65] T. Y. Tian, C. Tomasi, and D. J. Heeger. Comparison of approaches to egomotion computation. In *Proceedings of Computer Vision and Pattern Recognition*, 1996.

[66] Fadi Dornaika and C. R. Chung. Stereo geometry from 3-d ego-motion streams. *IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics*, 33, No. 2:308–323, 2003.

[67] Martin Gardner and John Conway. The fantastic combinations of john conway's new solitaire game 'life'. *Scientific American*, pages 120–123, October 1970.

[68] Robert B. Laughlin. *A Different Universe.* Basic Books, 2005.

[69] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87 Conference Proceedings*, volume 21(4), pages 25–34, 1987.

[70] V. M. Bove Jr. and Jacky Mallett. Eye society. *BT Technology Journal*, 22-4:45–51, 2005.

[71] P. J. Berger and T. Luckman. *The Social Construction of Reality.* New York Anchor Books, 1967.

[72] R. I. M. Dunbar. Coevolution of neocortical size, group size and language in humans. *Behavioral and Brain Sciences*, 16:681–735, 1993.

[73] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[74] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. In *Symposium on Principles of Database Systems*, pages 1–7, 1983.

[75] L. Lamport, M. Pease, and R. Shostak. The Byzantine generals problem. *ACM Transactions on PRogramming Languages and Systems 4*, pages 382–401, July 1982.

[76] Bernt Schiele and James L. Crowley. Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, 36(1):31–50, 2000.

[77] Jan J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[78] A. P. Witkin. Scale-space filtering. *IJCAI*, 8:1019–1022, 1983.

[79] Taizo Iijima. Basic theory of pattern observation. *Papers of Technical Group on Automata and Automatic Control, IECE, Japan*, December 1959.

[80] Joachim Weickert, Seiji Ishikawa, and Atsushi Imiya. Linear scale-space has first been proposed in Japan. *Journal of Mathematical Imaging and Vision*, 10:237–252, 1999.

[81] P. Gupta and P.R. Kumar. The capacity of wirelss networks. *IEEE Transactions on Information Theory*, 46:388–404, 2000.

[82] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 1989.

[83] Fred Brooks. *The Mythical Man-Month: Essays on Software Engineering*. IEEE, 1995.