Chapter 1

Adaptive Social Hierarchies: From Nature to Networks

Andrew Markham
Department of Electrical Engineering, University of Cape Town
Rosebank 7701, South Africa,
acmarkham@gmail.com

Social hierarchies are a common feature of the Animal Kingdom and control access to resources according to the fitness of the individual. We use a similar concept to form an Adaptive Social Hierarchy amongst nodes in a heterogeneous wireless network so that they can discover their role in terms of their base attributes (such as energy or connectivity). Three different methods of forming the hierarchy are presented (pairwise, one-way and agent based). With Agent ASH we show that the time taken for the hierarchy to converge decreases with increasing N, leading to good scalability. The ranked attributes are used as a network underlay to enhance the behaviour of existing routing protocols. We also present an example of a cross-layer protocol using ranked connectivity and energy. The ASH hierarchy provides an abstraction of real world, absolute values onto a relative framework, and thus leads to simpler and more general protocol design.

1.1. Introduction

A group of network nodes can be regarded as a society which is performing a task (such as sensing). How well the group performs this task affects information delivery and network longevity. Nodes in a network generally form a heterogeneous set, differentiated by explicit attributes (such as size of energy reserve, amount of buffer space) and implicit attributes (such as connectivity). The network must efficiently transfer information from source to sink, generally through intermediate nodes in the absence of direct connectivity between source and sink. Choosing which intermediates to transfer through is the subject of routing protocols, of which there exist a plethora.

Rather than introduce yet another routing protocol, we consider how a network could manage itself, given that nodes 'knew' their role within the group. For example, a node with low remaining energy level is likely to be exhausted soon and should thus adopt a less active role in the network than a node with a large amount of battery energy. What is meant by a less active role is application dependent, but could range from avoiding routing other node's packets to disabling energy hungry sensors or altering their duty cycle. In a network where all the nodes are initialized with identical sized energy reserves, determining whether a node should have a low or high role is simply related to the remaining proportion of energy.

Consider now the case where nodes are not introduced with the same initial amount of energy. Such would be the case in a widely heterogeneous network. We have previously introduced the example scenario of an animal tracking network, where the size of the host animal dictates the maximum weight of the tracking collar.[1] This results in a large degree of diversity in initial energy reserves. Assessing when a node is a low energy node now becomes dependent on the composition of the group, and no longer related to the absolute amount of energy. For example, in a network with nodes with energy levels of 100J, 200J and 300J, a 100J node has the lowest energy and thus should adopt a low role. Conversely, in a network composed of nodes with energy levels 20J, 50J and 100J, the 100J node is the 'best' in terms of energy and thus should be the most active. Although the energy of the node is the same, its behaviour is dictated by the energy reserves of its peers.

This simple example demonstrates that some sort of network wide discovery system is required in order to determine the role of nodes in the network. Nodes could send their attributes to a central server which would gather network information and from this instruct each node to adopt a certain behaviour. Whilst this is simple, it presents a single point of failure and does not scale well to large numbers of nodes in the network. A simple and lightweight localized discovery scheme is thus required so that each node can decide on its own level, based on information acquired from its peers.

We turn to the Animal Kingdom for inspiration, and observe that animals are able to determine their role within a society without any centralized control. Animals form collective groups, with an order of precedence, where some individuals are superior to others in their preferential access to resources (such as mates and food). We transpose this method of social organization onto our wireless network, so that nodes can determine their

role and access scarce resources (such as network bandwidth and energy).

In this chapter, we first examine social hierarchies in nature, to understand how and why they form and adapt to changes. The application and construction of social hierarchies to wireless networks are discussed in Sections 1.3 and 1.3.1 respectively. A method of creating a stable social hierarchy through pairwise exchange is presented in Section 1.4. Based on the observation that pairwise exchanges place a high burden on the MAC layer, a broadcast version of social hierarchy formation is elaborated upon in Section 1.5. To remove the restriction that the network be mobile in order for the hierarchy to correctly form, pseudo mobility is introduced through the action of random network agents in Section 1.6. This section also examines how mobility models impact on the rate of convergence and adaption. The focus of the chapter then turns to suitable network attributes to rank in Section 1.7, followed by a presentation of some example scenarios showing the use of the social hierarchy approach. Lastly, our work is contrasted with related work in Section 1.9, before conclusions and future direction are posed in Section 1.10. We now examine how social hierarchies are formed in nature.

1.2. Social Hierarchies in Nature

Many animals live in societies or groups of individuals. The size and composition of these groups vary, but most have a common feature of the imposition of a social hierarchy. A social hierarchy can be regarded as the organisation of individuals of a collective into roles as either leaders or followers with respect to each other. In biological terms, leading and following are referred to as dominance and submission respectively. A individual dominant over another typically has preferential access to resources such as food, water or mating rights. Note that although one animal may be dominant over another, it itself may be dominated by yet another member of the group. Social hierarchies are such a pervasive natural formation that words describing their behaviour have taken on colloquial usage, such as 'pecking order', 'leader of the pack' and 'alpha male'.

Some hierarchies are statically created through gross physiological differences, where it is impossible for certain individuals to dominate others. Such an example would be the relationship between sterile worker bees (drones) and the queen bee. It is impossible for a drone to become a queen as it lacks the required reproductive organs. We will not consider static hierarchies here, but are more interested in how hierarchies can be dynami-

cally formed based on differences between the members in the group. This, for example, would be how the queen bee dominates her sisters for the right to lay eggs in a hive .

We examine how these hierarchies form and adapt and consider how they result in a cohesive structure.

1.2.1. *Formation and maintenance of hierarchies*

Different species form hierarchies in different ways, but there is a common feature to all: *communication* of relative dominance or submission. Based on the received communication, animals subsequently update their perception of position within the hierarchy. Communication can be implicit, such as another animal observing the physical size of another or it can be explicit in the form of an interaction, such as an aggressive fight. Communication can take on many forms, depending on the capabilities of the particular species and can be tactile, olfactory, acoustic/vocal or visual. Whatever its form, the communication can be regarded as a stimulus emitted by the sender which alters the receiver's behaviour. We refer to the communication exchange followed by the hierarchy update as a dyadic (pairwise) *tournament*. Tournaments generally result in a winner and a loser, with the winner increasing its role in the hierarchy upwards and the loser downwards. An animal which frequently wins encounters with other animals is likely to have a high role in the hierarchy. We now consider some attributes which are used to construct the social hierarchy and how they are communicated.

In a hive of bees, there is only one queen. This individual is responsible for the laying of all the eggs within the hive. The queen is tended to by her sister bees, and the queen emits a hormone which suppresses the formation of ovaries in her peers.[2] This is an example of a totalitarian or despotic hierarchy.[2] If the queen bee leaves to form a new nest or dies, in the absence of the hormone, the sister bees start to regrow their ovaries. One bee will dominate the rest, becoming the new queen. A similar structure also occurs in other eusocial organisms such as wasps,[3] ants[4] and termites.[5]

The formation of dominance structures in chickens is a well studied area,[6].[7] Hens form a linear dominance structure, with a dominant individual meting out pecks to subordinates, hence the term 'pecking order'.[8] When a group of hens are assembled, there are frequent fights and changes in rank initially. Over time, as the structure becomes known, there are fewer and fewer fights as the hens are aware of their role or position within

the hierarchy.[2] If a new hen is introduced into the flock, the other hens will attack it as a group.[9] In a linear dominance hierarchy, an alpha or super-dominant individual dominates all others. The next individual in the hierarchy, the beta individual dominates all bar the alpha and so on. Linear dominance hierarchies are a common feature in many species, such as crayfish,[10] anemones,[11] goats[12] and ibex.[13]

Hierarchies are also a common system in primates, such as baboons and rhesus monkeys.[14] Baboons have a complex culture of social interaction which cements the troop together. Within human societies, hierarchies are also pervasive theme. Political and business structures are organized along the lines of dominance hierarchies, with presidents, vice presidents and so forth. Even the academic world is an example of a social hierarchy, with professors, lecturers and students being its constituents.

There are many attributes which are hypothesized to be important in the formation of social hierarchies, and these depend on the individual species. In crayfish, there is a strong correlation between claw length and social status.[10] In anemones the length of the feelers results in a difference in ranking in individuals.[11] In hens, there was a strong correlation between rank and comb size.[9]

The exact mechanism which is behind the formation of a social hierarchy in natural systems is unclear.[15] In repeated experiments with the same fish, researchers demonstrated that the rank order formed in successive trials was not identical, as would be the case under the assumption that rank order was solely based on observed differences between individuals.[16] There is a strong correspondence between the rank of the attribute and the resulting social hierarchy ranking, but it does not appear to be the only process at work.[17] It has been hypothesized that there is a form of positive reinforcement, where an individual which has recently won a tournament is more likely to win a subsequent tournament.[16] The reasons for this are unclear, but it is assumed to be as a result of increased hormone production, leading to the individual becoming more aggressive.

It must be made clear that social hierarchies are formed along the lines of comparative or relative differences between individuals, rather than absolute attributes. For example, in one group of animals, an individual may be ranked at the top, but in a fitter group of animals, it may only be ranked in the middle. Thus, an individual's role within the hierarchy is not only dependent on its own attributes, but also the composition of the group itself. Social hierarchies are not static structures and dynamically react to changes as the result of insertion or removal of other creatures. They also

alter if the attributes of a creature change, such as a result of injury or age.

1.2.2. *Purpose of social hierarchies*

Why do animals and insects form social hierarchies? The strongest driver for social agglomeration and organisation is thought to be the avoidance or minimization of predation. This is the 'selfish herd' hypothesis of Hamilton.[18] Essentially, it states that an animal is less likely to be a target for predation if its chances of being picked by the predator are minimized. Hence, if an animal is alone, it is more likely to be chosen by a predator. However, if it is part of a large group, it is less likely to be attacked. Some social animals form a 'mob' which collectively acts to drive away a predator, giving further evidence for the purpose of the formation of the group.[19]

Some predators also form societies in order to hunt more effectively. Wild dogs form packs that can bring down a large animal which one dog could not do by itself.[20] Although the prey needs to be shared amongst more animals, the individual effort and risk required to obtain the food is reduced. There is also a strong correlation between the size of the wild dog pack and their hunting success.[21]

In essence, the role or position of an animal within its society controls its preferential access to resources with respect to its peers. A super-dominant individual generally has first choice of mate, food and other resources and thus is more likely to propagate its genes onto a future generation. This comes at the cost of maintaining its position at the apex of the hierarchy, and a deposition can often result in death or serious injury. However, social structures provide for an organized access to resources, and are thought to lead to less competition over resources, as each individual is aware of its position.[2]

1.3. Using adaptive social hierarchies in wireless networks

Wireless networks are by their very nature amenable to the imposition of a social hierarchy. A hierarchy is a relative ranking system, based on measurable differences between individual nodes. Essentially, a hierarchy provides an abstraction of the real world resources onto a framework which indicates relative performance (such as poor, good, best). The purpose of a network is to transfer information from a source to a sink, through multiple nodes. To conserve resources (such as bandwidth and energy), the information is sent along the shortest path (as measured by some metric

such as hop-count or latency) between the sink and the source, through intermediate nodes. The aim of a routing protocol is in essence to determine the 'best' path for the information to take. This means that the set of all possible paths can be cast into a hierarchy, ordered by the desired metric, and the routing algorithm decides which is the best path. All routing protocols which are based on minimizing some metric are implicitly forming a hierarchy amongst the nodes in the network and basing routing decisions upon differences between individuals. However, explicitly constructing and adapting a hierarchy is an area which yet to be investigated fully.

Hierarchies are a common feature of network protocols, but many are imposed at design time, based on physical differences between nodes (for example, the Data Mule system is composed of a three tier static structure: stationary nodes ; mobile (higher power nodes) and base-station nodes[22]). A network should be able to learn the differences between nodes, rather than have the differences specified prior to deployment. In this way, the network will be able to dynamically adapt to insertions and removals. Thus, we refer to the structure as an *adaptive social hierarchy* to reflect the fact that the hierarchy adapts to changes in attributes and is not static.

A hierarchy can be thought of as a mapping from an absolute domain (such as 1 joule of energy, 3 hops from sink) to a relative domain (e.g. best node in network). An example of this mapping is shown in Fig. 1.1. Although the real world values of the nodes in in network B are ten times as large as the metrics in network A, they both map to the same system when viewed from a relative ranking perspective. In the presented protocol, the rank is scaled to lie within [0;1], where 0 corresponds to the lowest ranked node and 1 corresponds to the highest ranked node.

Node management and routing is based on the relative parameters, rather than the absolute. This means that network decisions are based on factors which are independent of the real values, leading to scalability in the resource domain. In a relative domain, network decisions do not need to take into account real values, with the implication that a network protocol can be used in vastly different application scenarios, without having to alter network tuning parameters. In addition, any parameter that can be measured can be placed into a social hierarchy. Thus, the design of network protocols can be separated from the real world scenario. For example, in a static network, hop count is an important metric, whereas in a mobile network other metrics such as utility are more useful. Traditionally, each of these would require a different routing protocol, whereas if a social hierarchy is used, a single network protocol can be used for both problems. This

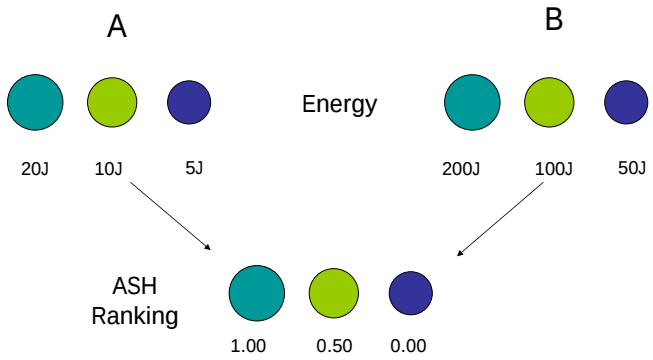


Fig. 1.1. Illustration of ASH ranking procedure. Although the networks in A and B have different energy levels, they rank to the same network when viewed in the ASH sense. This resource abstraction makes network design independent of real world values.

is illustrated in Fig. 1.2 for a stationary and a mobile system, showing how they both map into the same relative network, simplifying network design. In the stationary network, hop count is used as a measure of 'goodness', whereas in the mobile network, transitive connectivity is used. The fixed network would require different routing rules to the mobile network, as the range of the values of the absolute parameters is different. However, if a ranking system is used, the absolute values of the parameters are mapped onto relative values which are between 0 and 1. In this way, the same routing rules can be used if a social hierarchy is constructed.

Thus, there are now two different areas in network protocol design: constructing the social hierarchy and the formulation of protocol rules. The hierarchy can be regarded as an underlay which provides the resource abstraction onto a relative structure. Various methods of creating a social hierarchy are discussed in the following section.

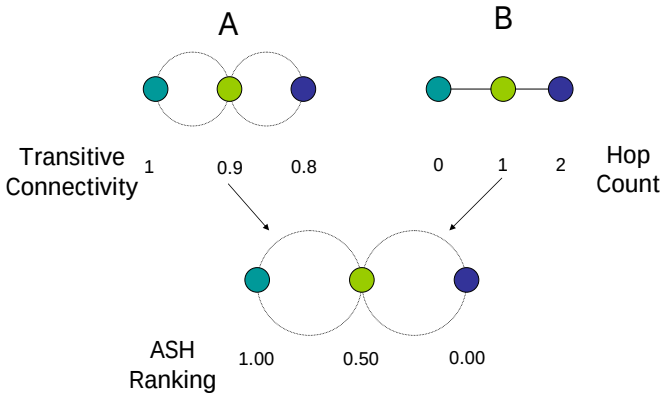


Fig. 1.2. ASH working on two different connectivity metrics. In mobile network A, connectivity is measured by the transitive connectivity, whereas in the stationary network B, connectivity is measured using the traditional hop count. Note that both networks map to the same network when ranked

1.3.1. *Constructing an Adaptive Social Hierarchy*

As previously stated, a hierarchy can be viewed as a mapping from an absolute parameter space to a relative parameter space. This mapping can be undertaken in many different ways. Ranking needs to be determined in a distributed fashion amongst all the nodes, as opposed to a centralized system which has a single point of failure. As nodes are typically resource constrained, methods to determine ranking should not result in a high level of overhead in the form of ASH ranking and updating. In addition, the ranking needs to be scalable not only with the number of nodes in the network, but also with the range of parameter values. Before various ranking systems are discussed, some terminology is introduced.

Let S be the set of a particular attribute A of the N nodes in the network

$$S = \{A_1, A_2, \ldots, A_N\}\,. \tag{1.1}$$

When the order of the elements in this set is taken into account, an

ordered set is formed. For example, consider the four element set of attributes $S = \{10, 4, 8, 6\}$. The ordered set, $\phi = [4, 6, 8, 10]$ is the original set, sorted into order of size. The rank order of the sorted set is given by $R = [4, 1, 3, 2]$ where the rank refers to the index of the original element. In our case, however, we are interested in the scaled or normalized rank, which is expressed as

$$\widehat{R} = \frac{R - 1}{N - 1}. \tag{1.2}$$

Thus, for the given example with rank order $R = [4, 1, 3, 2]$, the normalized rank is found to be $\widehat{R} = [1, 0, \frac{2}{3}, \frac{1}{3}]$. By normalizing the rank, the rank is made independent of the number of nodes in the network, leading to scalability. This is because it is known that the maximum rank is 1 and the minimum rank is 0. Thus a node with an attribute that has a rank of 1 can be regarded as the 'best' node in the network, in terms of that attribute.

The goal of our work is to present various methods of evaluating the ranking in a distributed fashion, much like animals form social hierarchies based on measurable or perceived differences. We denote the rank as estimated by a method of ranking as $\widehat{E}$. First though, we consider how to measure how well a method performs in the task of ranking a set of attributes.

There are a number of factors which we are interested in with respect to the formation and maintenance of the social hierarchies. Probably the most important factor is the rate of convergence or settling time of the social hierarchy, from initialization to a point when it is deemed to be settled. Another factor is the stability of the hierarchy - whether nodes remain in the correct order, or whether there are time varying errors in the ranking. Lastly, we are also interested in the adaptibility of the social hierarchy - how long it takes the system to react to disturbances, such as node insertion or removal or a change in a node's attribute. With all of these performance metrics, it is necessary to determine whether they are scalable in terms of increasing N, the number of nodes in the network.

To determine when the hierarchy has converged, a metric is required that reflects the degree of order of the rankings of the nodes in the system, and whether they are in concordance with the order of the absolute parameters. The simplest metric is one which indicates when the estimated ranks are perfectly ordered. In this case, we can say the system is correctly ordered when $\widehat{E} = \widehat{R}$

In a network of this nature, subject to frequent changes and rank reversals, it is not necessary for the hierarchy to be perfectly sorted in order for the nodes to know their approximate role in the network. One possible measure of error is the Mean Squared Error (MSE). This reflects how close the system is to its desired steady state values, but does not weight incorrect orderings highly. Another metric, formulated especially for determining ordinality of the ranking, is the Kendall Correlation Coefficient, denoted by τ.[23] This indicates the normalized number of pairwise exchanges or switches required for the set of estimated ranks to be perfectly sorted. A Kendall τ of -1 corresponds to perfectly reversed ranking (for example [4;3;2;1]), whereas a τ of +1 reflects that the system is perfectly ordered.

However, we are interested in the worst case performance, and this would be expected from the node with the greatest error in ranking. For example if a node has a small amount of energy, and it is erroneously ranked highly relative to its peers, then depending on the application, it could expire rapidly. Thus, we define another convergence time as the expected time taken for the maximum rank error in the network to drop below a certain threshold. We define the maximum (absolute) rank error as $e_{max} = max\left|\widehat{E} - \widehat{R}\right|$, and we say that our system is converged when $e_{max} < \epsilon$, where ϵ is the acceptable percentage error tolerance. The time taken for this to occur is denoted as $T_{\pm\epsilon}$. Note that T_0 corresponds to the expected time to be perfectly settled. We use $T_{\pm 20\%}$, $T_{\pm 10\%}$, $T_{\pm 5\%}$ and T_0 (corresponding to a maximum error of 20%, 10%, 5% and 0 respectively) to measure Rate of Convergence. Different applications will require different error thresholds depending on their requirements.

Closely related to the initial Rate of Convergence is the performance of the system when nodes are subject to changes in their attributes or nodes are inserted or removed. In control theory terms, the Rate of Adaption (ROA) is a measure of disturbance rejection. A system which takes a long time to recover from perturbations will not be suitable for use in a wireless network, which is characterised by frequent variations in resources and connectivity. To measure the ROA, we vary 10% of the node's attributes to new, random values once the network has converged to the specified error threshold. The ROA is defined as the time taken for the system to re-achieve $T_{\pm\epsilon}$ after the pertubation.

1.4. Pairwise ASH

Much like the way animal dominance hierarchies form through dyadic interactions, we start with a simple pairwise switch or exchange method of sorting the ranks of the nodes, in concordance to their attributes. When a pair of nodes meet, they trade their attributes and ranks. If the order of their ranks are in contradiction to the order of their attributes, their ranks are switched. If there is no contradiction, their ranks are left unchanged. The rules for this method of ranking are shown in 1.3. There are four possible combinations - two correspond to ranks and attributes with the same order and the other two deal with contradictions.

	$A_j > A_k$	$A_j < A_k$
$E_j > E_k$	$E_j \leftarrow E_j$ (unchanged)	$E_j \leftarrow E_k$ (exchange)
$E_j < E_k$	$E_j \leftarrow E_k$ (exchange)	$E_j \leftarrow E_j$ (unchanged)

Fig. 1.3. Rank update rules for pairwise ASH. Both nodes perform the same rules at the same time, which will lead to a simultaneous exchange of ranks in the event of a contradiction

A demonstration of the sorting action is shown in Fig. 1.4 for a 10 node network. Initially, the nodes were assigned to have ranks in perfect reverse order. After 89 meetings, the network is perfectly ordered. Note the frequency of switches with respect to the number of meetings – initially, most meetings result in rank alteration. However, as the system becomes more ordered, fewer node meetings result in a rank exchange. This is similar to what is observed in many animal social hierarchies which are characterized by a high initial competition rate which decreases as the hierarchy becomes ordered. The corresponding maximum rank error plot for the trajectory of Fig. 1.4 is shown in Fig. 1.5. It can be seen that the error decreases rapidly, with the maximum error of any node being less than 30% after 30 meetings. However, to reach perfect order, it takes a further 60 meetings.

We use this simple method of sorting as a baseline to compare other methods against. The possible transitions between all the possible orderings, as well as the probability of the transition are shown in Fig. 1.6 for

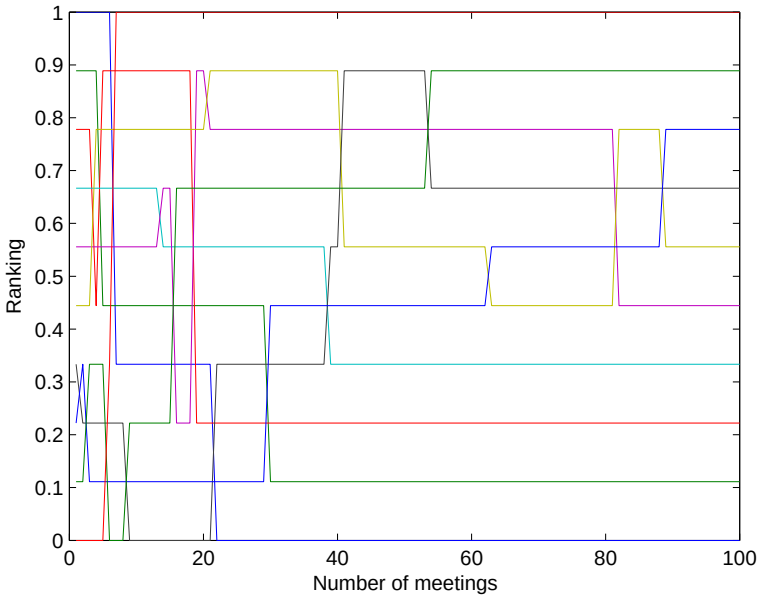


Fig. 1.4. Rank trajectories for a 10 node network, starting from perfect reverse order.

a three node system. Note that the transition graph is a Directed Acyclic Graph (DAG), as it can be topologically sorted. In addition, the Kendall correlation coefficients are strictly increasing with the transitions. This has the implication that any meeting that results in a transition will result in a more ordered network. It is impossible for a meeting to result in a network which is less ordered.

We can place an upper bound on the expected number of meetings by examining how long it will take if the system traverses the maximum number of switches from perfectly disordered to perfectly ordered. The maximum number of switches is given by $N_s = \frac{N(N-1)}{2}$, which is the same as the number of possible combinations in which two unique nodes can be picked at random. For a three node system, this results in the maximum number of switches being 3.

It will now be shown how to calculate the upper bound, given that the system starts from perfectly disordered, i.e [3;2;1]. From this ordering, every possible meeting results in a transition to a new sequence. Thus the time that the system remains in this sequence before leaving is $3/3 = 1$

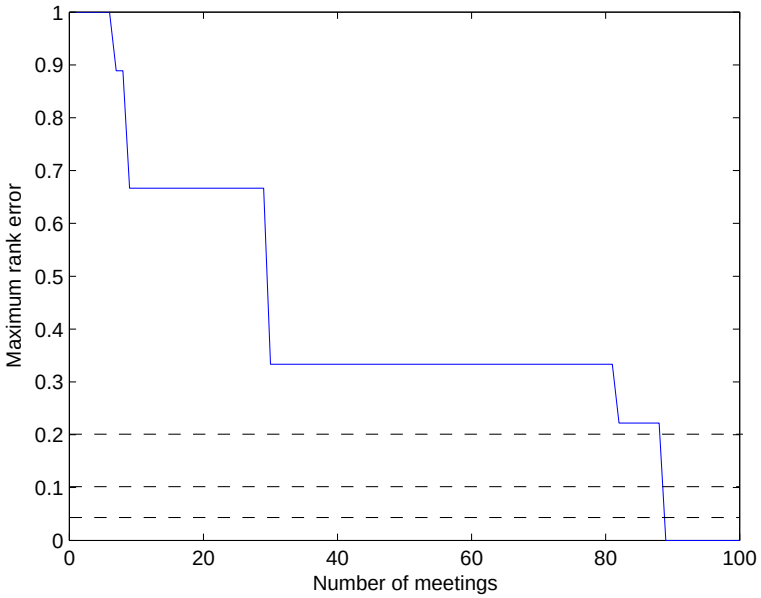


Fig. 1.5. Maximum rank error for the trajectories shown in Fig. 1.4. Note that the network is perfectly ordered after 89 meetings. The error thresholds corresponding to the determination of $T_{\pm 20\%}$, $T_{\pm 10\%}$ and $T_{\pm 5\%}$ are shown on the diagram as dotted lines.

meeting. Consider if the next ordering chosen is [3;1;2]. From this point, two out of a possible three meetings will result in a transition to a new ordering. Hence, we can expect the system to remain in this ordering for 3/2 meetings. Lastly, if the rank ordering is [1;3;2], only one meeting (one between nodes 2 and 3) will result in a transition to [1;2;3]. The system will be expected to take 3/1 meetings on average to leave. Once in the final state, the system will never leave. Thus, the total expected time to transition from completely disordered to perfectly disordered is the sum of the number of meetings required to reach this state, which is $3/3+3/2+3/1 = 5.5$ meetings. In general the upper bound on the expected number of meetings can be written as

$$T_{max} = \sum_{i=1}^{N_s} \frac{N_s}{i}, \tag{1.3}$$

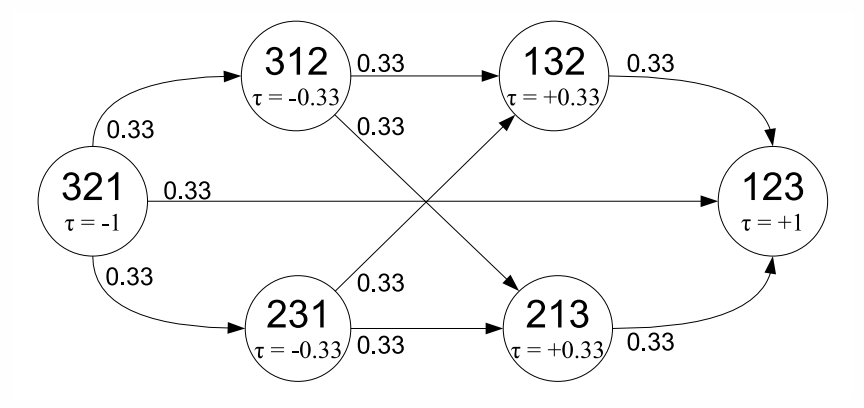


Fig. 1.6. Transition diagram for the possible sequences of a three node network. Probabilities next to each arrow are the exit probabilities from each state. The τ value shown is the Kendall rank correlation coefficient. It should be observed that τ is monotonically increasing from left to right, and that the transition graph results in strictly increasing values of τ.

where $N_s = \frac{N(N-1)}{2}$.

For large N, this can be expressed as

$$T_{max} = N_s(\ln(N_s) + \gamma), \tag{1.4}$$

where $\gamma = 0.57721...$ is the Euler-Mascheroni constant.

Thus, in the limit, this shows that the upper bound on the settling time varies as $O(N_s \ln(N_s))$, or in terms of N as $O(N^2 \log(N))$. This has the implication that the required number of meetings for the system to converge perfectly appears to be prohibitively large as N increases. Later, in Section 1.6.2, we show that in the more realistic scenario where more than one node can meet per time interval, the effect of increasing node density counteracts the polynomial convergence time, resulting in a decrease in convergence time with increasing N.

However, whilst the result of Eq. 1.4 is useful to place an upper bound on the performance of the switching scheme in terms of the expected number of meetings, in reality the meetings are random and thus the system is unlikely to pass through every possible combination to reach the ordered state. In addition, initial node rankings are likely to be random, not in perfect disorder.

To calculate the expected settling time, we can formulate this problem as a Markov Chain. The terminal state, corresponding to perfectly ordered,

is called an absorbing state, and the ROC is equivalent to the expected absorption time. The possible state transitions are expressed as a transition matrix. This indicates the probability of moving from one state to another. The transition matrix for a system with N nodes is denoted by P_N. Thus, the transition matrix for the case of N = 3 is given by

$$P_3 = \begin{array}{c|cccccc} & 321 & 312 & 231 & 132 & 213 & 123 \\ \hline 321 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 312 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ 231 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 132 & 0 & 0 & 0 & \frac{2}{3} & 0 & \frac{1}{3} \\ 213 & 0 & 0 & 0 & 0 & \frac{2}{3} & \frac{1}{3} \\ 123 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \tag{1.5}$$

There are some things to notice about the P matrix, which we will later use to simplify our calculations. Firstly, it is of size $N! \times N!$, which has the implication that large N results in state space explosion. It is strictly upper triangular, as it is impossible for two nodes to meet and result in the system becoming more disordered. The first element on the diagonal is 0, as any meeting will result in an exit from this state. The last element on the diagonal is 1, as once the system is in this state it can never exit, corresponding to the absorbing state.

To determine the expected time to absorption, we express P in canonical form as

$$P = \left[\begin{array}{c|c} Q & R \\ \hline 0 & I \end{array}\right]. \tag{1.6}$$

Using a standard result from the theory of Markov Chains, the expected time from each state to absorption is given by

$$D = (I - Q)^{-1} = I + Q + Q^2 + Q^3 + \ldots. \tag{1.7}$$

D is a vector of dwell times, and the mean time to absorption from any state (assuming there is an equal chance of starting in any state, including starting off perfectly ordered) can be calculated as

$$t_0 = \frac{\sum D}{N!} \tag{1.8}$$

For P_3, the mean time to absorption is 3.167, which means that on average, just over three pairwise meetings are required to perfectly sort the system.

To prevent state explosion, we attempt to reduce the size of the state universe. We note, that although the number of possible states varies as $N!$, the number of possible exchanges only grows as $\frac{N(N-1)}{2}$. We thus redefine our states as the expected number of exchanges to reach perfect order. This smaller matrix will be numerically easier to invert in order to find the expected time to absorption.

We define the reduced state transition matrix as S_N. The reduced transition matrix for a three node network is given by

$$S_3 = \begin{array}{c|cccc} & 3 & 2 & 1 & 0 \\ \hline 3 & 0 & \frac{2}{3} & 0 & \frac{1}{3} \\ 2 & 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 1 & 0 & 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 1 \end{array} \tag{1.9}$$

The expected time to absorption still evaluates as 3.167, indicating that this approach of reducing the state universe is valid. However, even constructing the smaller matrix and inverting it is tedious. Table 1.7 shows the expected mean time to absorption as N varies for the various methods. Also shown is the convergence time as evaluated by the Monte Carlo simulation of the ranking process.

Fig. 1.7. Convergence times as predicted by the various methods for different N. Also shown is the upper bound on the expected settling time, T_{max}.

N	T_0 (Full Markov)	T_0 (Reduced Markov)	T_0 (Simulated)	T_{max}
3	3.17	3.17	3.18	4.50
4	9.13	9.14	9.13	11.20
5	18.90	18.93	18.89	21.67
6	32.79	32.82	32.81	36.24
7	51.02	51.04	51.15	55.13

To this end, we have obtained an approximation to the expected convergence time for large N

$$T_0 = \frac{N_s}{2} + N_s(ln(N) + \gamma) \tag{1.10}$$

A diagram showing the Monte Carlo simulated convergence time compared against the approximation given by Eq. 1.10 is shown in Fig. 1.8. This

shows that the approximation achieves a good correspondence to the Monte Carlo simulated convergence times. For comparison, the upper bound of expected convergence time, T_{max} is also shown.

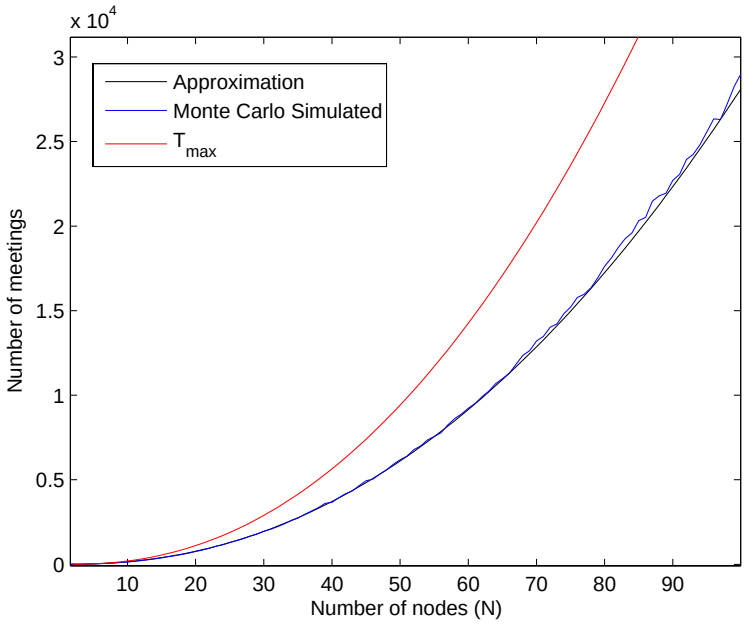
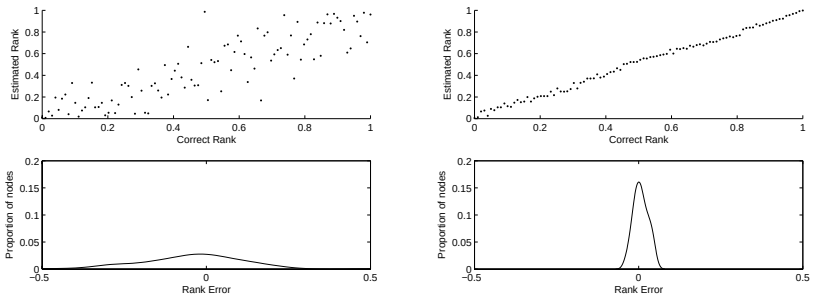


Fig. 1.8. Variation in Rate of Convergence against N for different error thresholds. The diagram shows that the approximation of 1.10 accurately predicts the convergence time as estimated by the Monte Carlo simulation. The upper bound on the convergence time is also shown.

Up to now, we have considered the time taken to be perfectly converged as a guideline. However, the question that must be posed is 'how close to correctly ordered, must the system be to behave well?' This is because once a node has determined its approximate position in the network, the decisions made by the node are generally not fine-grained. For example, the node behaviour is likely to be largely the same whether its rank is 0.16 or 0.17. Furthermore, attributes change with time, so a dynamic system is unlikely to be ever correctly ordered.

Error plots for the rank estimator are shown in Fig. 1.9 for two different error thresholds ($\epsilon = 0.5$ and $\epsilon = 0.05$) for a 100 node network. Fig. 1.9 (a) shows that a maximum rank error of 0.5 leads to a wide spread of the

ranks around the correct value. In most instances, this spread would make the network unusable. Fig. 1.9 (b) demonstrates that for a maximum rank error of 0.05, the correspondence between actual rank and estimated rank is very close to linear, with the majority of nodes having ranks that differ only a few percent from the correct values.



(a) (b)

Fig. 1.9. Ranks and the distribution of errors for two different error thresholds. Note how the distribution of the error becomes more peaked as the ϵ value is decreased, and how the distribution is approximately normal. The top plots show the scatter between estimated and correct rank and the bottom plots show the distribution of the error. (a) Error diagrams for $\epsilon = 0.5$ (b) Error diagrams for $\epsilon = 0.05$

Fig. 1.10 shows how $T_{\pm 20\%}$; $T_{\pm 10\%}$ and $T_{\pm 5\%}$ vary with increasing N. For comparison, T_0 is also shown. It can be seen that allowing for a looser definition of converged results in more rapid settling times. In addition, the growth of the convergence times with increasing N slows with increasing error tolerance. This can also be seen in the plot in Fig. 1.5 which shows the rapid drop in the initial error, followed by a long time to converge to perfectly settled. Based on our simulations and our prior results, it was found that the rate of convergence with N tended towards a log-linear relationship:

$$T_{\pm 20\%} = 3.32N(\ln(N)) \tag{1.11}$$

$$T_{\pm 10\%} = 7.08N(\ln(N)) \tag{1.12}$$

$$T_{\pm 5\%} = 14.2N(\ln(N)) \tag{1.13}$$

These equations show that the number of meetings required for the system to converge increases in sub-polynomial time, illustrating that this simple protocol will scale acceptably to large N. It should be noted that this is for the case when there is one pairwise meeting per unit time. In a realistic network scenario (which we encounter in Section 1.6.2), the number of meetings per unit time depends on the underlying mobility model. We deliberately decouple the mobility model from this analysis in order to provide general results that can be applied to many different application scenarios.

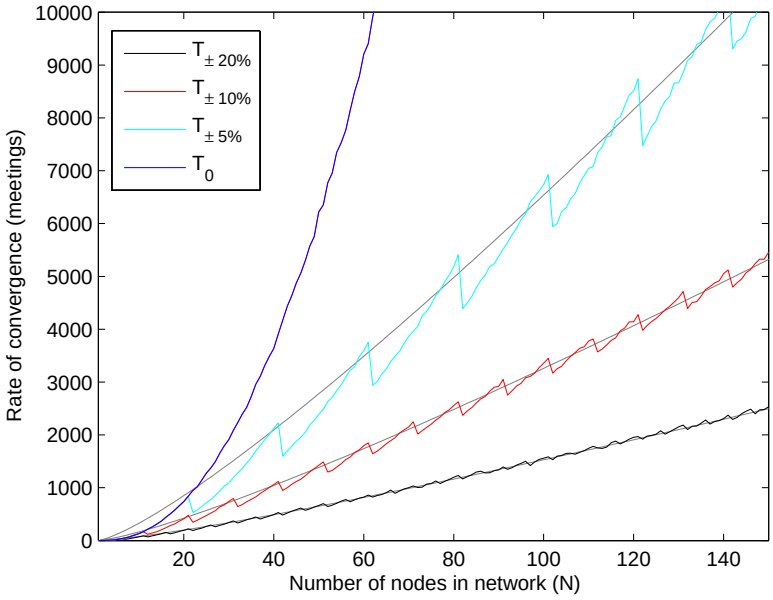


Fig. 1.10. Variation in Rate of Convergence against N for different error thresholds. The approximations to the ROC for the differing thresholds are shown in grey. Note the slow growth of $T_{\pm 20\%}$ compared with T_0.

1.4.1. *Pairwise ASH with reinforcement*

In Section 1.4, it was assumed that the initial ranks were equally spaced over the extent of the ranking interval. The restriction of equally spaced ranks makes initialization complex. Worse still, adaption to node insertions

and removals is impossible, as the former will result in duplicate values and the latter in gaps in the ranking order. This makes scaling to dynamically varying numbers of nodes in the network impossible, precluding its use in most scenarios.

To deal with this problem, nodes on startup choose a random normalized rank value between 0 and 1, with no input from their peers. Based on their meetings, nodes switch their rankings as in the prior section. However, in order that the ranks converge to be equally spaced over the interval [0;1], regardless of the number of nodes in the network, an additional update rule is introduced. If two nodes meet and their ranks are in agreement with their attributes, each node reinforces its rank. The node with the lower attribute reinforces its rank towards 0 and the node with the higher attribute reinforces its rank towards 1. Reinforcement upwards is defined as

$$R = R(1 - \alpha) + \alpha \tag{1.14}$$

and reinforcement downwards as

$$R = R(1 - \alpha), \tag{1.15}$$

both under the condition that $0 < \alpha < 1$, where α is the reinforcement parameter. The ranking update rules are shown graphically in Fig. 1.11. A large α results in rapid rank updates, but leads to rank instability, whereas a very small α leads to little success in spreading the ranks equally. Note that for the case when $\alpha = 0$, this method devolves to that presented in Section 1.4. Thus, this method can be regarded as a more generalized version of pairwise ASH.

	$A_j > A_k$	$A_j < A_k$
$E_j > E_k$	$E_j \leftarrow (1\text{-}\alpha)E_j + \alpha$ (reinforcement upwards)	$E_j \leftarrow E_k$ (exchange)
$E_j < E_k$	$E_j \leftarrow E_k$ (exchange)	$E_j \leftarrow (1\text{-}\alpha)E_j$ (reinforcement downwards)

Fig. 1.11. Rank update rules for pairwise ASH with reinforcement. This allows for dynamic insertion and removal of nodes, resulting in equal spacing of ranks over time. The reinforcement parameter, α controls the rate of adaption.

Rank trajectories highlighting how this method can handle node insertion are shown in Fig. 1.12, and the corresponding maximum rank error plot in Fig. 1.13. At the start of the simulation, 10 nodes with random ranks are placed into the network. The reinforcement parameter was set to be $\alpha = 0.01$. After 500 pairwise meetings, an additional 10 nodes with random ranks were injected into the network. This causes a spike in the maximum error, which is rapidly corrected through the switching (contradiction) action of the pairwise ASH protocol. The reinforcement action gradually causes the ranks to vary towards their correct values. The effect of random meetings can be seen as 'noise' in the ranks of the nodes.

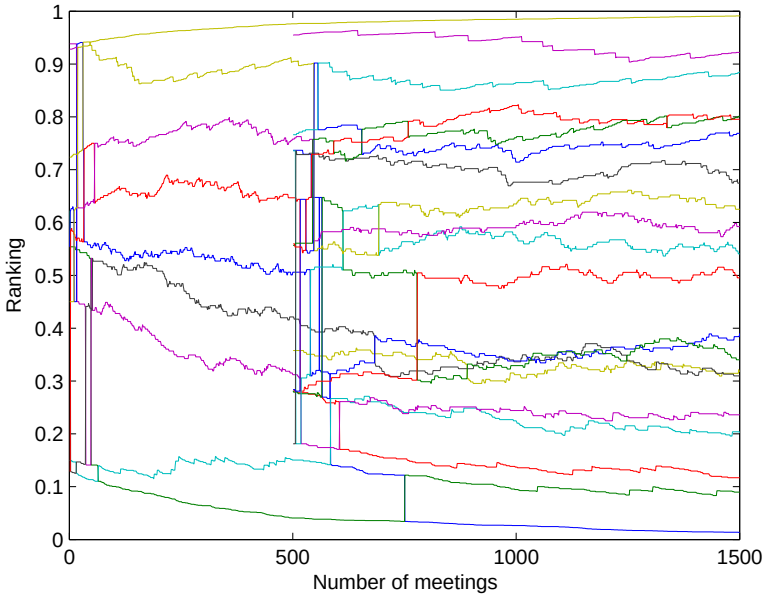


Fig. 1.12. Rank trajectories for pairwise ASH with reinforcement demonstrating node insertion. The reinforcement parameter was set to $\alpha = 0.01$. Initially ten nodes with randomly assigned ranks were present in the network. After 500 meetings, another ten nodes (also with randomly assigned ranks) were introduced into the network. Note how the trajectories spread out evenly.

The value of the reinforcement parameter affects how rapidly the system is able to adapt to changes in network composition. However, if the reinforcement parameter is too large, the ranks will 'chatter' and never

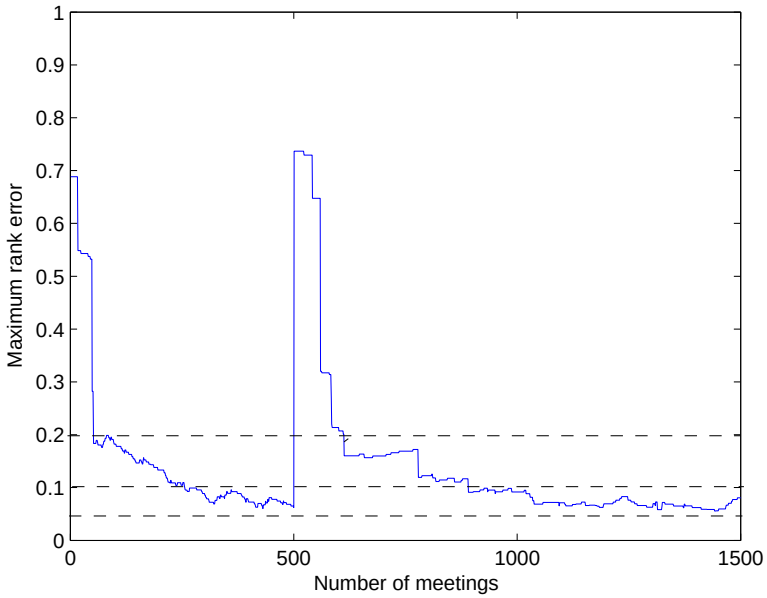


Fig. 1.13. Maximum rank error for the trajectories shown in Fig. 1.12. The error thresholds corresponding to the determination of $T_{\pm 20\%}$, $T_{\pm 10\%}$ and $T_{\pm 5\%}$ are shown on the diagram as dotted lines.

converge to their correct value. Consider the extreme case if $\alpha = 1$. In this case, reinforcement will result in a node either having a rank of 0 or 1, depending on the direction of reinforcement. Thus, there will be no other values for the ranks in the network, leading to rapid switchings from maximum to minimum rank. To determine suitable values of α the average rate of convergence (for different error thresholds) against α for a 50 node network has been plotted in Fig. 1.14.

1.5. One Way ASH (1-ASH)

The algorithms presented so far deal with the case when a pair of nodes meet and trade their attributes and ranks. In a more typical network scenario we need to deal with the nature of the radio medium - being broadcast - from one transmitter to many receivers.

For many routing protocols, nodes emit 'beacons' as network discovery packets (commonly termed 'Hello' packets[24]). ASH parameters can

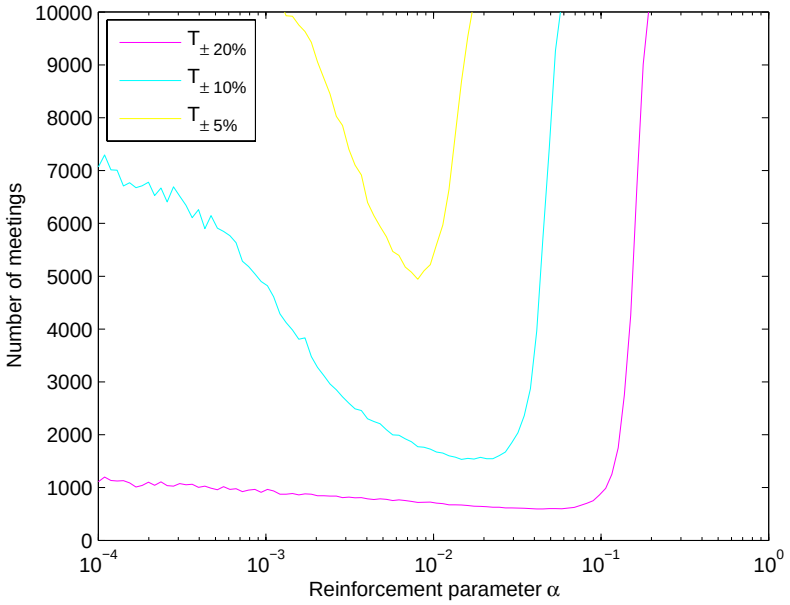


Fig. 1.14. Rate of convergence of a 50 node network with α for the various error thresholds. Note that large α results in excessively high (or infinite) convergence times.

piggyback on top of these 'Hello' packets such that they do not lead to a detrimental increase in network overhead. When nodes are in receive mode, discovering active nodes within their neighbourhood, they can update their ASH ranking according to the transmitted ASH attribute/rank pairs. The only issue is that although the receiver nodes can update their rankings depending on the newly acquired information, the transmitting node is unable to update its ranking until it later switches to receiver mode. This is essentially an asynchronous method of updating the ranks, and it can lead to slower convergence and more churn.

1.5.1. *Domination ASH*

A hypothesized method of forming linear dominance hierarchies in nature is thought to be the win/loss ratio.[17] In this method, each node tracks how many nodes it dominates (i.e. the number of nodes which it exceeds in the value of its attribute) relative to the total number of nodes it meets. The rank update rules are shown in 1.15. Every time a node is met, the

total number of observed nodes, M, is increased by 1. If the attribute of the receiving node dominates that of the transmitter, the win counter, W, is also increased by 1. Initially, W and M are both set to zero.

	$A_j > A_k$	$A_j < A_k$
$E_j > E_k$	$E_j \leftarrow (W{+}{+})/(M{+}{+})$ (domination)	$E_j \leftarrow W/(M{+}{+})$ (submission)
$E_j < E_k$	$E_j \leftarrow (W{+}{+})/(M{+}{+})$ (domination)	$E_j \leftarrow W/(M{+}{+})$ (submission)

Fig. 1.15. Rank update rules for domination ASH. As this is a one way process, only the receivers update their ranks in relation to the transmitted attributes. W is a node variable which records the number of nodes dominated and M is the total number of nodes met.

As this is a ratiometric measure, it is not dependent on the absolute number of nodes in the network, leading to good scalability. It is simple to compute, but adapts slowly to changes and converges slowly. However, each node's rank converges to the correct asymptotic value with increasing M. Essentially, the domination ratio can be thought of as the probability of dominating another node chosen at random. This is shown in Fig. 1.16 which demonstrates the long settling time coupled with the diminishing rank variation as the number of meetings increase for a 20 node network. After 1000 meetings, two nodes (corresponding to 10% of the nodes) attribute values are randomly changed. The recovery from this disturbance is slow as M is large. Fig. 1.17 shows the change in error with respect to time for the simulation conducted in Fig. 1.16.

1.5.2. *Domination ratio with switching*

There are two main problems with the approach of Section 1.5.1. The first drawback is that it reacts very slowly to node insertions and removals, especially for large M. The second issue is that it only uses the comparison between the attributes to update its rank. This is clear from Fig. 1.15, where it can be seen that the rank comparison plays no part in updating the ranks of the nodes. To address these two issues, we modify the rank update rules slightly, incorporating the idea of switching from Section 1.4, and limiting the maximum value of M (and hence W).

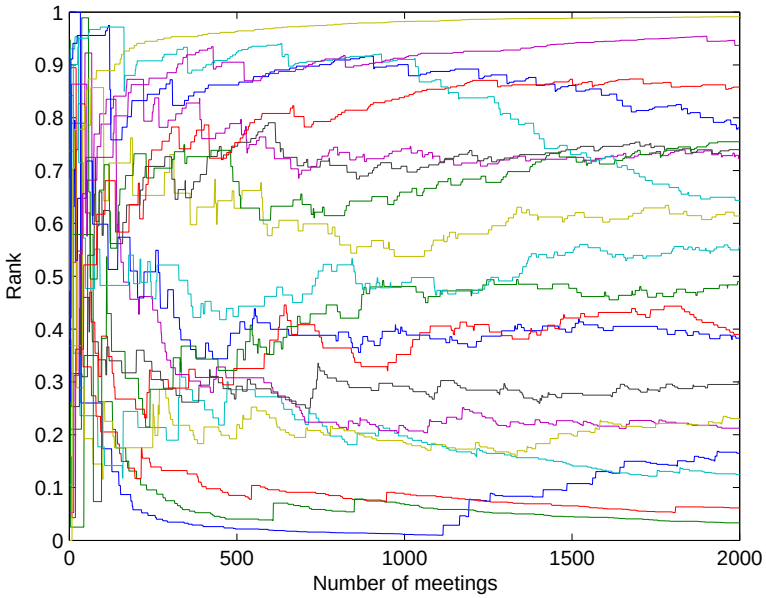


Fig. 1.16. Rank trajectories for a 20 node network. After 1000 meetings, two nodes' attributes are randomly changed.

These new rules are shown in Fig. 1.18.

Before the update rules are executed by the receiving node, the total number of nodes that have been observed is compared against a limit L. If M exceeds L, both M and W are multiplied by a factor $1 - 1/L$. This parameter controls the 'memory' of the rank update. A large value of L leads to slow convergence but stable ranks, whereas a value of L which is too small leads to excessive rank oscillation. Rank trajectories for a 20 node network with L = 100 are shown in 1.19. After 1000 meetings two nodes' attributes are randomly changed to new values. The corresponding error plot is shown in 1.20. Note how the system recovers much more rapidly from the perturbation than the previous domination algorithm with no switching.

The simulation plots shown so far deal with the case when only one node is listening to the ASH broadcast. We now consider how the ROC varies when multiple nodes listen to the same transmitter in each time interval. Fig. 1.21 demonstrates how increasing the number of receivers in

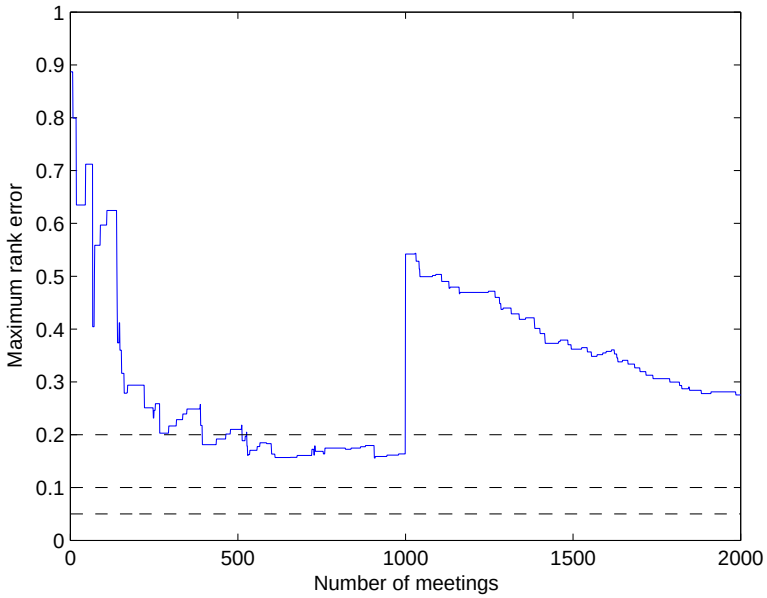


Fig. 1.17. Maximum rank error for the trajectories shown in Fig. 1.16. Note how the error 'spikes' at 1000 meetings when two nodes randomly change their attributes and that the recovery from this error is very slow. The error thresholds corresponding to the determination of $T_{\pm 20\%}$, $T_{\pm 10\%}$ and $T_{\pm 5\%}$ are shown on the plot as dotted lines.

a time window leads to a much more rapid rate of convergence. Increasing the number of receivers results in an increase in the dissemination of information across the network.

It should be noted that if the number of receiver nodes is increased from 1 to 2, the expected rate of convergence halves, regardless of the value of N. This is an important result, as it demonstrates that it is not the proportion of nodes receiving rank information, but rather the number of nodes. Thus, it would be expected that in a network with increasing N, the important factor is the average node degree, not the edge density. Lastly, in Fig. 1.22, we show how one-way ASH can converge more rapidly than pairwise ASH, for the situation where there are multiple receivers (in this case 3) to each node broadcast. For the case of N = 150, a node degree of 3 corresponds to a network density of 2%. This shows that one way ASH can converge quickly even in sparsely connected networks.

Fig. 1.22 demonstrates the performance of one-way ASH compared to

```
if (M > L):
    M = M(L-1)/L
    W = W(L-1)/L
fi
```

	$A_j > A_k$	$A_j < A_k$
$E_j > E_k$	$E_j \leftarrow (W{+}{+})/(M{+}{+})$ (domination)	$W = M \times E_k$, $E_j \leftarrow (W)/(M{+}{+})$ (exchange)
$E_j < E_k$	$W = M \times E_k$, $E_j \leftarrow (W{+}{+})/(M{+}{+})$ (exchange)	$E_j \leftarrow W/(M{+}{+})$ (submission)

Fig. 1.18. Rank update rules for domination ASH with switching. Before nodes update their rank, they first check the limit on M. If M exceeds the limit, both W and M are proportionally reduced. The nodes then run the update rules in the table. In the event of a contradiction, the receiver will adopt the transmitter's rank.

pairwise ASH for varying N. The Rate of Convergence for $T_{\pm 10\%}$ for both approaches is shown, when each transmitter in one-way ASH transmits to three receivers. This shows that although one-way ASH suffers from asynchronous updates, it can exceed the performance of pairwise ASH whilst being more suited to the broadcast nature of the wireless medium.

1.6. Dealing with mixed mobility: An agent based approach

The prior approaches to ranking nodes discussed in Sections 1.4 and 1.5 rely on the assumption that nodes meet at random (with a uniform probability) in order to percolate the attribute/rank information throughout the whole network. If nodes are stationary, the previously presented techniques can fail as a result of the restricted neighbour horizon, leading to limited node discovery.

We can introduce pseudo-mobility by recreating the effect of randomized meetings. Nodes listen to transmissions from nodes within their immediate radio range. If they repeat these transmissions to their neighbours, two nodes which do not have a direct connection can 'observe' each other and correctly update their ASH ranking. This has the effect of artificially increasing the probability of connection between any two nodes, increasing

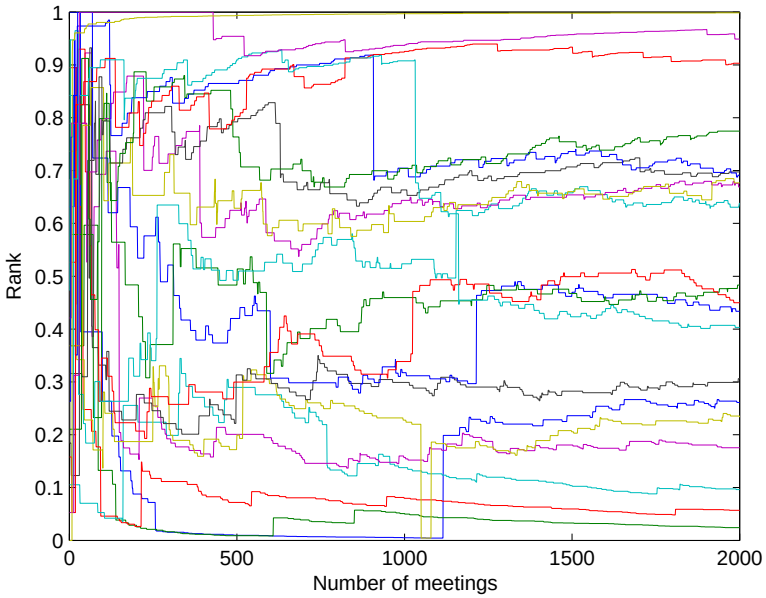


Fig. 1.19. Rank trajectories for a 20 node network with a limit L = 100. After 1000 meetings, two nodes' attributes are randomly changed. Note the rapid switching action at the start of the simulation due to the rank exchange.

the network connectivity. Thus, the goal of disseminating rank/attribute pairs to a large number of nodes in the network can be achieved, leading to a more accurate representation of the social hierarchy. We refer to these rebroadcasted rank/attribute pairs as agents, as they can be viewed as independent carriers of information. The problem with the agent based approach is that agents can be carrying outdated information. Thus churn (or rank variation with time) is expected to be higher in this scheme than in the other systems under purely random motion.

To prevent flooding and unacceptably high overhead, nodes only rebroadcast other node's data as part of the 'Hello' packet, as in the previous sections. Nodes select at random which rank/attributes to rebroadcast from a small local buffer, and upon overhearing new data, pick a rank/attribute pair in the buffer to replace. This way, ranks are randomly rebroadcast, without detrimentally loading nodes in the network. Agent ASH uses domination with switching as presented in Section 1.5.2, with the incorporation of additional rules which control agent creation and spreading.

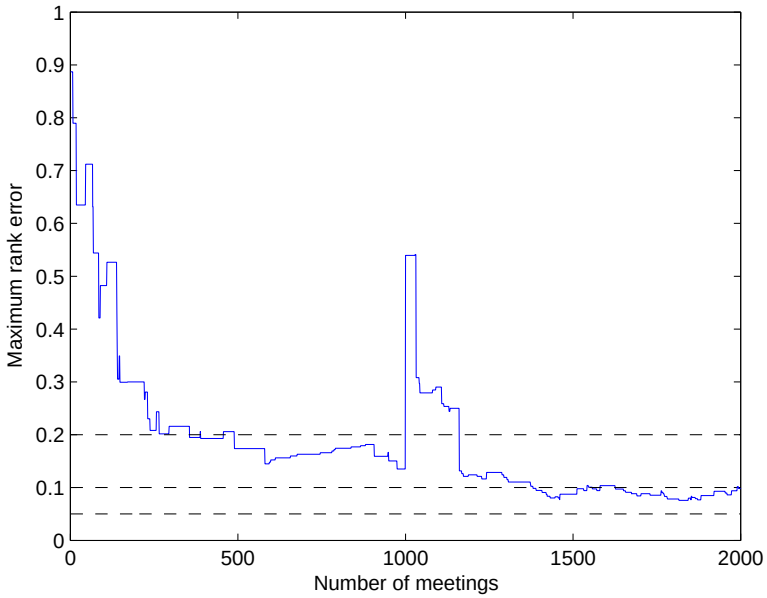


Fig. 1.20. Maximum rank error for the trajectories shown in Fig. 1.19. Note how the error 'spikes' at 1000 meetings when two nodes randomly change their attributes. The error thresholds corresponding to the determination of $T_{\pm 20\%}$, $T_{\pm 10\%}$ and $T_{\pm 5\%}$ are shown on the plot as dotted lines.

1.6.1. *Agent Rules*

Each node has a cache of length C entries. Each entry can store one rank/attribute pair. When nodes broadcast their 'Hello' packets, they send their own rank/attribute pairs as before. However, they now append G 'agents' or rebroadcast rank/attribute pairs from their local cache. Note that $G <= C$. For the case where $G < C$, not all entries in the cache are rebroadcast. Each entry is thus picked without replacement at random with a uniform probability of G/C. Obviously, when $G = C$, the probability of an entry being picked from the cache is 1.

When a node overhears a broadcasted 'Hello' packet, it can use this new information to update its cache. It randomly replaces entries in its local cache from the rank/attribute pairs sent in the message. There are $G + 1$ entries in the message, as each node sends its own rank/attribute pair followed by G agent entries. To populate its local cache with this new

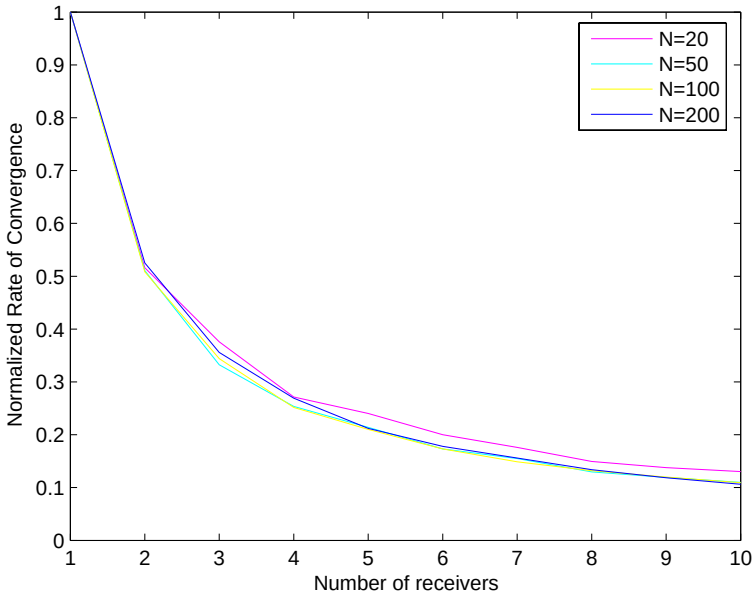


Fig. 1.21. Normalized Rate of Convergence ($T_{\pm 10\%}$) against number of receivers for varying number of nodes in the network. The time to converge for one receiver is taken as the base figure of 100%. Note how increasing the number of receivers leads to a rapid decrease in the normalized ROC.

information, the receiving node places each new piece of information into a random slot in the cache. Thus, the local cache is refreshed with new information as it arrives. To prevent the cache from containing repeated information, an entry is only replaced if that node has never seen that particular rank/attribute pair.

The receiving node updates its rank using only the agent information carried in the message. It does not use the neighbouring node's rank/attribute information, as this will lead to bias for frequently encountered peers, distorting the ranking process. This is not to say that a node will never use its neighbour's data. It is possible that it will observe this information indirectly through another node's agent data.

This is a very simplistic approach to using agents to spread information through the network, using no state information carried in the agents to control their spread and route. However, these random agents improve the connectivity of the underlying graph by creating 'pseudo-edges'. This

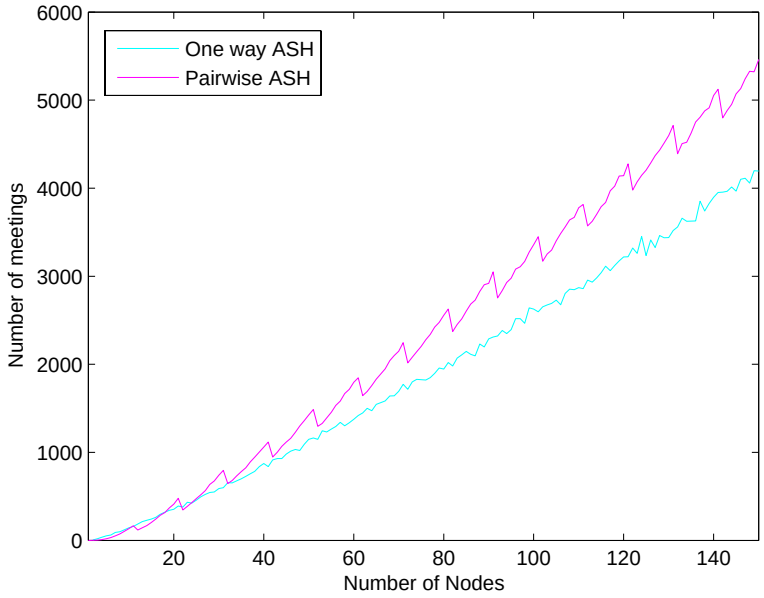
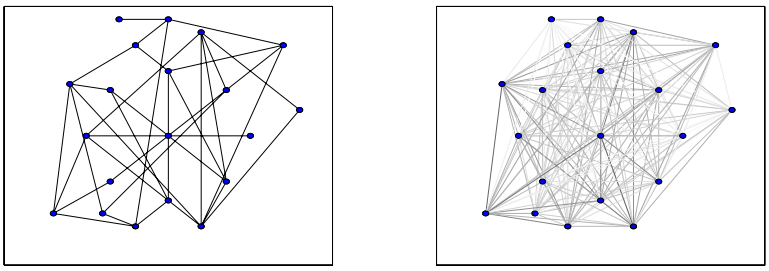


Fig. 1.22. Rate of Convergence ($T_{\pm 10\%}$) against number of nodes for pairwise ASH and one-way ASH. For one-way ASH, L = 100 and the number of receivers for each transmission was set to 3.

is shown graphically in Fig. 1.23 which shows the network edge density artificially increased for a 20 node stationary network.

In terms of ASH performance, we compare how Agent ASH is able to form the social hierarchy with how ordinary one-way ASH is unable to discover the nodes in the network, due to a limited horizon. This is shown in Fig. 1.24, where it can be seen that the Agent approach is able to result in a lower error and faster convergence for a 100 node stationary network. This network was generated randomly to be connected with an edge density of 5%. Nodes transmitted 'Hello' packets with a duty cycle of 5%. For the rest of the time, nodes were in receive mode. The cache size, C, was set to 3 entries, and the number of agents transmitted per message (G) was set to 1. This illustrates that Agent ASH can correctly discover the network ranking with a very modest increase in node resources, even in sparse stationary networks. Agent ASH is not restricted to stationary networks however – in Section 1.6.2 we demonstrate how it functions when nodes are mobile.

Ranking trajectories for a random 20 node connected network with edge



(a) (b)

Fig. 1.23. Graphs showing connectivity between nodes for a 20 node network. The graph on the right demonstrates the action of the agents which enable nodes which are not connected to each other to overhear their rank/attribute broadcasts. (a) Direct connections between nodes (indicated by solid lines) (b) Density of connectivity between nodes (line intensity denotes proportion of time edge is present)

density of 10% are shown in Fig. 1.25. In this example, both the cache and the number of agents per message were set slightly larger, to 4 and 2 respectively. The error plot for this simulation is shown in Fig. 1.26 which demonstrates the rapid recovery from the pertubation imposed after 1000 meetings. In this simulation, for fair comparison to the other protocols, one node was chosen at random to be the transmitter.

1.6.2. *Realistic meetings*

In the prior sections, we have assumed that only one node is active and transmitting at any one point. In reality, the way nodes meet is dependent on the underlying mobility model. In a realistic network scenario, it is possible that at any point in time, that no nodes meet or more than two meet (possibly in geographically distinct locations). To account for this, we incorporate the effect of the mobility model into the way nodes meet.

We examine how Agent ASH performs when subject to two commonly discussed mobility models - the random walk model and the random waypoint model. In the random walk model, at each point in time, a node chooses to move to another location that is one-step away. In the random waypoint model, nodes travel in a straight line at a randomly chosen speed until they reach their destination. Once the destination is reached, nodes choose a new destination and velocity and travel towards that. The random walk model exhibits a very slow mixing time, in the order $O(K^2)$ where K

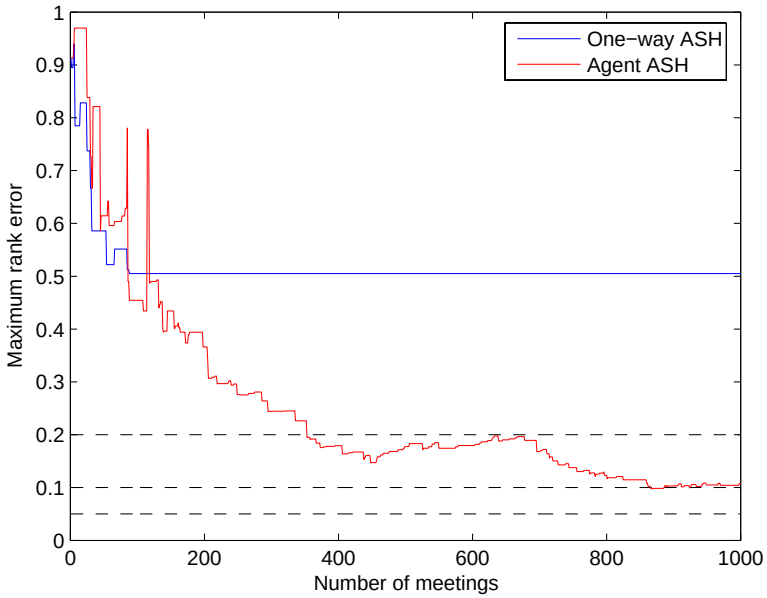


Fig. 1.24. Rank errors for one-way ASH and Agent ASH applied to the same 100 node stationary network. Observe how the error for one-way ASH never drops lower than 0.5. This is as a result of the limited discovery horizon. The error thresholds corresponding to the determination of $T_{\pm 20\%}$, $T_{\pm 10\%}$ and $T_{\pm 5\%}$ are shown on the plot as dotted lines. In the simulation, L = 500, G = 1, C = 3.

is the length of one side of the simulation area, whereas the random waypoint has mixing times of the order $O(K)$.[25] Thus, we would expect the performance of ASH to be worse when subject to the random walk model in comparison to the random waypoint mobility model.

For the random walk model, N nodes are placed on a $K \times K$ toroidal simulation area. The radio radius, U, is varied from 5 to 15. The transmission range (along with the number of nodes) controls the connectivity of the network. The rate of convergence is examined for N = 50; 100 and 200. We examine the time taken for the system to converge to $\pm 20\%$ of its final value, and denote this as $t_{\pm}20\%$, where the lower case t indicates real time results, as opposed to the number of meetings. To introduce some realism into the MAC layer, we assume that nodes transmit 'Hello' packets 10% of the time. For the remaining time, the nodes are in receive mode. Unless otherwise stated, the parameters for the Agent ASH model were chosen to

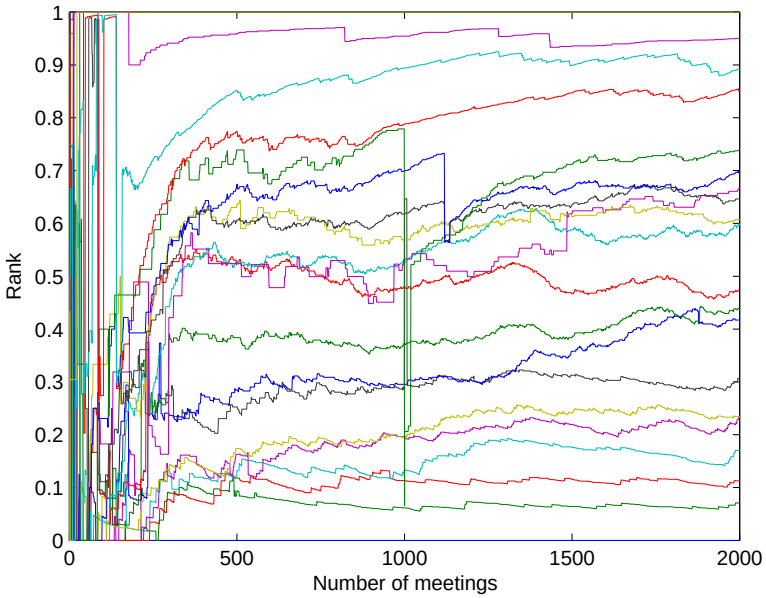


Fig. 1.25. Rank trajectories for a 20 node network with C = 4 and G = 2, using Agent ASH. After 1000 meetings, two nodes' attributes were randomly changed.

be G = 1; C = 3 and L = 500.

The simulation results for the Random Walk model are shown in Fig. 1.27. The graph shows that the radio range has a large effect on the Rate of Convergence. Increasing the number of nodes has the effect of reducing the simulation time. Contrast this to the previous results from Section 1.4 which showed that an increase in N resulted in an increase in the number of meetings for the system to converge. This demonstrates the scalability of the ranking system - an increase in N actually results in a faster convergence time, as it results in a more dense network.

In the simulation of the Random Waypoint Model, all common parameters are kept the same as the Random Walk. Nodes have a random speed uniformly chosen between 1 and 2 units/s. The results from this simulation are shown in Fig. 1.28. Comparing these results to those of the random walk simulation, it can be seen how the Random Waypoint model leads to faster convergence as it has a faster expected mixing time. Like the Random Walk results, it can be seen that an increase in N results in a decrease

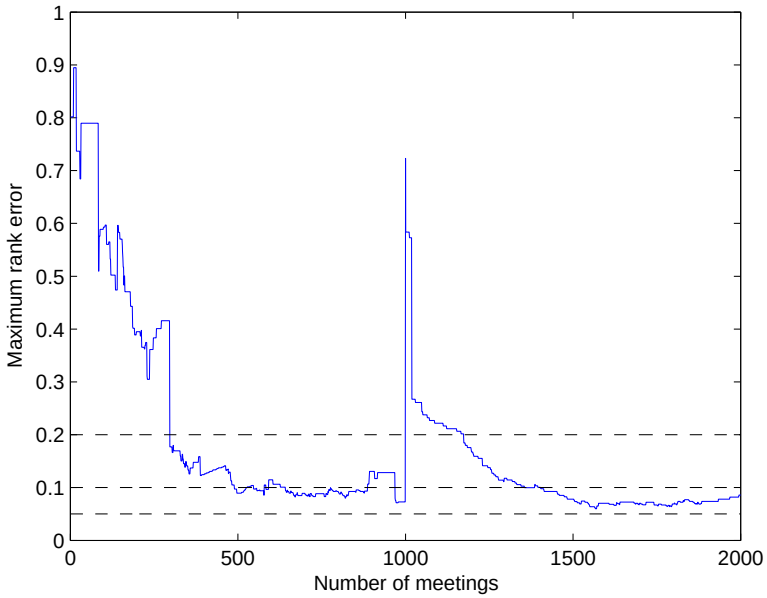


Fig. 1.26. Maximum rank error for the trajectories shown in Fig. 1.25. When the attributes are randomly changed after 1000 meetings, the error sharply peaks, followed by a rapid switching action to correct the erroneous ranks. The error thresholds corresponding to the determination of $T_{\pm 20\%}$, $T_{\pm 10\%}$ and $T_{\pm 5\%}$ are shown on the plot as dotted lines.

in the Rate of Convergence.

The Agent ASH algorithm demonstrates that it is possible to sort or order a network according to its attributes, even if the network is purely stationary or subject to mixed degrees of mobility. In addition, only small amounts of local information are used to infer global attribute distribution. ASH piggybacks on top of existing network discovery packets, so does not present a large overhead burden. Now that we have described various methods of forming a social hierarchy, we now consider how to use this information for network control, access, management and routing. First though, we examine some common network attributes.

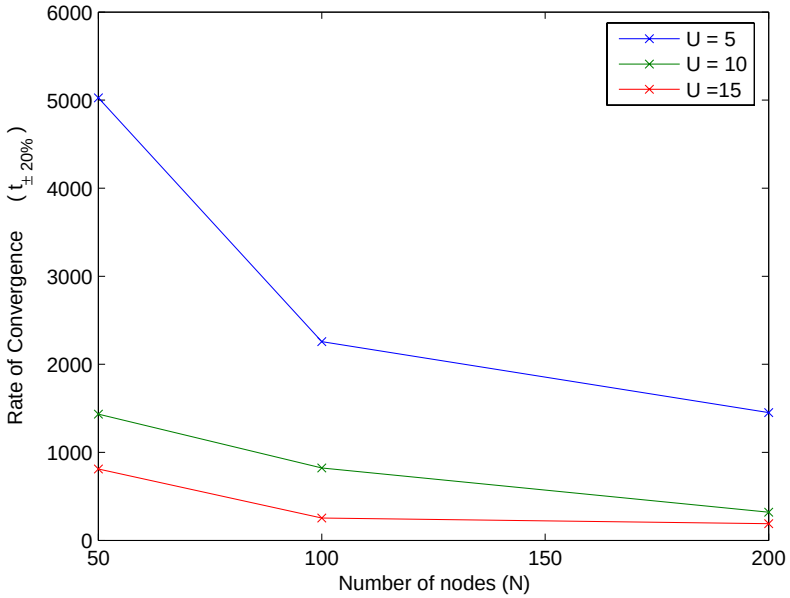


Fig. 1.27. Rate of convergence to $t_{\pm}20\%$ for the Random Walk Mobility model for different numbers of nodes. Note how the radio radius (U) has a strong impact on the ROC, as it alters the connectivity of the graph.

1.7. Suitable attributes to rank

Any network parameter which can be measured on a node-by-node basis can be ranked in the global sense using one of the ASH ranking methods. Suitable attributes are obviously ones which have a direct and useful impact on network performance and control. We present a few useful attributes, which we will later use in Section 1.8 to demonstrate the power and flexibility of the ASH philosophy.

1.7.1. *Energy/Lifetime*

Of prime importance in deeply embedded and remote networks is energy, and coupled to that, rate of use of energy. This is reflected in the vast number of energy aware network routing protocols (refer to[26] and[27] for more information). Nodes, depending on their hardware capabilities can measure their energy reserves and usage, in an absolute sense (such as 100J

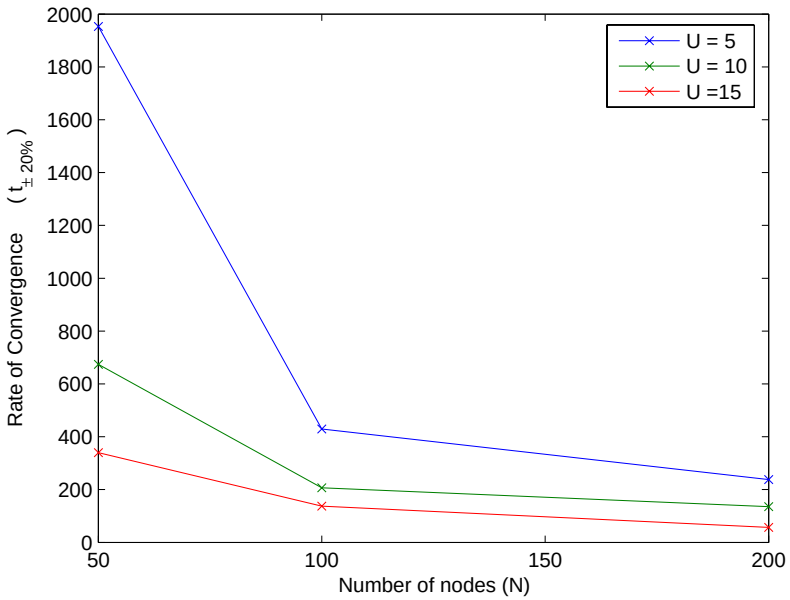


Fig. 1.28. Rate of convergence to $t_{\pm}20\%$ for the Random Waypoint Mobility model for different numbers of nodes.

remaining). However, it is also possible to estimate energy usage in software (for example using the Contiki Operating System[28]), and knowing the size of the battery that the unit is equipped with, determine the remaining amount of energy. A node with a large amount of energy in relation to its peers can be expected to survive longer in the network. However, this assumption of survival is made under the pretext that all nodes consume energy at the same rate. Thus, a more useful indicator is estimated node lifetime, based on a long term average of prior energy usage and current reserves.

A composite indicator of lifetime and connectivity can be formed from the lifetime of the path to the sink. The lifetime of the path can be regarded as the expected lifetime of the node with the lowest energy along the path, as this is the likely point of failure. To incorporate the distance from sink to source, the minimum path lifetime is decreased by a factor with each hop. The amount by which the minimum path lifetime is decreased controls whether shorter paths that travel through low energy nodes are favoured

over longer paths which avoid low energy nodes. This metric is similar to the Conditional Battery (MMCBR) protocol proposed by Toh.[29]

1.7.2. *Connectivity*

The purpose of a wireless network is to transfer information from source to sink, through intermediate nodes. The purpose of a routing protocol is to deliver the data along the 'best' possible path, where the desirability of each path depends on some underlying cost function of suitability (such as delay, hop count or redundancy). In a network for information delivery, nodes need to send information to base-station(s). At each hop, data should be sent to a node which is 'closer' to a base-station. Lindgren et al. present a method of evaluating transitive connectivity to a base-station,[30] which is based on the observation that if A and B frequently meet, and B and C frequently meet, then A and C can be regarded as being transitively connected through B. In static networks, the conventional hop-count metric can be used a measure of connectivity.

Another metric which impacts the delivery of information is the local traffic density. In regions of high traffic density, it would be expected that collisions will be more frequent. Nodes can assess the local error rate, possibly by monitoring how many of their transmissions fail. This information can be used to build a ranking of the expected congestion in the network. Data can be sent along paths where the expected congestion is low, thus balancing traffic across the network.

1.7.3. *Buffer space*

Depending on their physical memory size and their role within the network, nodes will have varying amounts of buffer space available for messages from other nodes. This can be ranked, although the ranking would be expected to be very dynamic as buffers are cleared upon successful delivery. A long term average of available buffer space would possibly be a better metric of delivery rate.

1.7.4. *Functions of attributes*

Ranked attributes do not need to be based on a single measure of network performance, but can be a composite of multiple attributes, weighted in various ways. There are many ways of constructing a combined attribute. Context Aware Routing (CAR) combines attributes to form a single weight-

ing.[31] Of particular interest with the CAR approach is that the availability and predictability of the various attributes is incorporated into the weighting procedure.[32] The ranking process can be performed on the combined attributes, or ranked attributes can be combined (and further ranked if necessary).

1.7.5. *Levels and loops*

In a practical network scenario, a node needs to decide whether to send a message to another node based on the difference in their ranks. If a node sends its data to another node with a greater rank, this can lead to an explosion in traffic. In addition, there is the possibility of data forming a loop as the rank of the host varies over time.

To address these issues, we quantize the rank into L discrete levels. As all the ASH methods presented in this Chapter generate linear hierarchies, it can be easily seen that each level (or bin) will contain N/L nodes. Assume that traffic is generated at a rate of λ messages per unit time, and nodes only send messages to nodes which have a greater level. The traffic density of the lowest nodes in the level is $D_1 = \lambda$ as they will only forward the packets they generate, and not route any other traffic. A level 1 node will send its messages to any node with a higher level. There are $L-1$ levels that are higher, and they will thus each expect to receive (and subsequently be responsible for forwarding) $\lambda/(L-1)$ messages. Thus, in general, we can express the traffic density at each level in the hierarchy using the recursive equation

$$D_k = D_{k-1}(1 + \frac{1}{L-k+1}), \tag{1.16}$$

where $1 \leq k \leq L$.[33]

This can be simplified to obtain an expression for the traffic volume at level k

$$D_k = \frac{\lambda L}{L-k+1}. \tag{1.17}$$

This important result shows that traffic density is *independent* of the number of nodes in the network, and only related to the number of levels in the quantized hierarchy. This demonstrates that the ASH framework results in good scalability by controlling traffic density.

To prevent packets from looping through the hierarchy, in the event of dynamic ranks, packets can be tagged with the rank of the node which sent the data to the current node. As packets can only ascend the hierarchy this means that loops cannot form.

1.8. Example scenarios of ASH

We now show how ASH can be used as an underlay to enhance existing protocols and also to form a cross-layer protocol in its own right. For reasons of space, we omit any simulations, but rather discuss how ASH can be used to improve the behaviour and functioning of protocols.

1.8.1. *Enhancing Spray and Focus*

Spray and Focus[34] is a multi-copy controlled replication routing protocol. Routing is divided into two phases. The first phase, Spraying, creates multiple copies of a message around a source. There are different ways of undertaking the spraying, but the authors show that binary spraying is optimal.[34] In this situation, given that L distinct copies are to be created, the source will hand over L/2 copies to the first node it meets, keeping L/2 for itself. At each point in time, a node can hand over half the remaining copies of the message, until there are L nodes in the network, each carrying a single copy. In the original version of this protocol, Spray and Wait,[35] nodes then performed direct delivery to the sink of the information. However, in the most recent version, nodes enter a Focus phase, where data is forwarded along a utility gradient towards the sink. A timer indicating the time of last contact with the destination is used as the utility parameter. To capture transitive connectivity, the timer value is updated if a node meets an intermediate node which was recently in contact with the destination. The use of the focus phase dramatically reduces delivery latency, whilst maintaining the benefits of controlled replication.

The details of the Focus phase are as follows. A node sends its local copy of a message to another node, if and only if the utility of the receiving node exceeds the sending node by a certain threshold, U_{th}. Clearly, the choice of this threshold is critical. If it is too large, then messages will rarely be forwarded closer to the destination, leading to increased delivery delay. Conversely, for a small threshold, a message can possibly be forwarded many times before reaching its destination, leading to resource exhaustion. Tuning the threshold parameter can be done at run time by the user, but

is likely to be time-consuming. Rather, by ranking the utility values, the choice of the threshold can be made independent of the underlying mobility model. Thus, using ASH as an underlay on this protocol will simplify the choice of the tuning parameter by making it scalable across wide attribute values.

1.8.2. *Enhanced Context Aware Routing*

Musolesi and Mascolo presented Context Aware Routing (CAR), which is a utility based routing protocol for intermittently connected networks.[31] In CAR, nodes evaluate their 'utility' which is a metric indicating the usefulness of a node in terms of network specific parameters such as connectivity or energy level. These different parameters are combined to give a single utility value for each node in the network. One of the main contributions of this work was the idea of predicting future values of the utility, using Kalman filters (or the reduced form of a Exponentially Weighted Moving Average).

A more recent work considered the application of CAR to sensor networks with mixed mobility (so called SCAR[36]). In this scenario, sources of information (which can be fixed or mobile) deliver information to sinks (which can also be fixed or mobile). As connectivity is intermittent, information is buffered at intermediate nodes which are more likely to be able to forward the data to a sink. Note that data can be delivered to any sink. To reduce the delivery delay, messages are replicated to multiple neigbouring nodes before forwarding along the utility gradient towards the sinks, in a similar manner to Spray and Focus.[34]

Three measures of context information are used to decide on routing. The first, is the change rate of connectivity (CRC) which essentially is a metric of the change in the local connectivity graph, a measure of relative mobility through the network. The degree of colocation with the sink is proposed as the second measure of utility. Lastly, the remaining battery level. The battery level is the proportion of remaining energy, relative to the initial battery level. Clearly, this protocol does not handle heterogeneity in initial battery level. By using the ASH framework, the energy level can be ranked on the interval [0;1], regardless of the distribution of energy in the absolute sense.

The absolute level of energy is ranked using ASH, and then the energy ranking is used to construct the utility parameter. Thus, the introduction of the simple ASH protocol as an underlay vastly increases the scope of SCAR,

allowing it to be used in networks with widely heterogeneous distributions in energy. It should also be noted that ASH can also be used to rank the calculated utility, as SCAR also has a threshold value, ζ, which is used to control replication.

1.8.3. *A simple cross layer protocol*

In this example we demonstrate how network information can be used at all levels of the traditional network stack, collapsing the strict segmentation, leading to a simpler implementation. We rank both energy and connectivity separately and use the ranked data throughout all the levels of the network stack.

1.8.3.1. *Medium Access*

Nodes with low ranks both in connectivity and energy are not active in network tasks such as routing and replication. They essentially act as leaf nodes, only injecting packets into the network. As a leaf node is not required to route other node's packets, there is no cause for it to ever attempt to listen for other node's data transmissions (it does still need to observe 'Hello' packets in order to maintain its correct rank). In addition, due to its scarce resources, it should not have to compete equally with higher ranked nodes for access to the medium. Furthermore, as packets can only ascend the hierarchy, there is no point in a low ranked node listening to a high ranked node's transmission. We present a simple slotted MAC scheme to demonstrate how ranking can result in a sensible preferential access to the shared medium. This is shown in Fig. 1.29 which shows the behaviour of each node in its assigned slot. This is only one possible arrangement of slots. A more realistic approach might be to have wider or more slots for higher ranked nodes as they spend more time on the medium as they will have a greater amount of data to send. The scheme does not have to be slotted (relaxing the requirement of synchronization), but can also be made into a random access protocol, where the probability of listening to the medium is based on the node's level.

1.8.3.2. *Routing and Replication*

We now show one possible method to control data delivery in a network ranked by ASH. Low ranked nodes perform direct delivery to higher ranked nodes. High ranked nodes share information amongst themselves, epi-

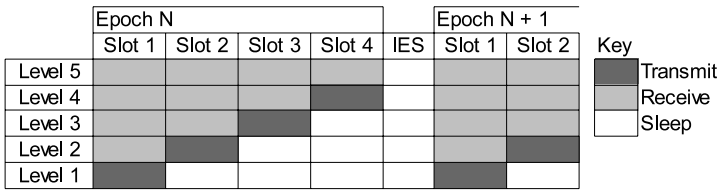


Fig. 1.29. Slotted MAC organisation example for a 5 level network. Based on their level, nodes choose what action to take in each slot. Note that the lowest level nodes spend the majority of their time in low power sleep mode.

demic style. Thus, replication and delivery is controlled according to rank. Clearly, duplicating a message across low level nodes does not achieve any useful redundancy, as these nodes are not active in disseminating information and are likely to be severely resource constrained compared to their higher ranked peers. Thus, messages are replicated with a probability that increases with node level. Hence, delivery amongst low level nodes will resemble direct routing (with low traffic overhead, but high latency) and delivery between high level nodes will resemble epidemic routing (with high traffic overhead and low latency).[37]

The probability of a level k node sending a message to a level j node is given by

$$p_{send} = \begin{matrix} 1 : j > k \\ p_t : j = k \\ 0 : j < k \end{matrix} \tag{1.18}$$

where p_t is the horizontal (i.e. across the same levels in the hierarchy) transmission probability. Once the message has been sent to a higher level node, the node can either keep the message or delete it. The probability of a level k node keeping a sent message for future replication is given by

$$p_{replicate} = r^{L-k+1} \tag{1.19}$$

where r is a parameter that controls the degree of the replication. This equation results in a probability of keeping a message for future replication that increases with level in the hierarchy. The value of r affects the expected number of copies present in the network.

To prevent thrashing due to rank promotion and demotion due to attribute changes, packets are not allowed to descend in rankings, even if

the host's ranking has dropped. This prevents issues with packets forming loops in the hierarchy, leading to rapid network exhaustion.

1.8.3.3. *Application*

Information about the node's local variables, such as energy and connectivity can also be used at the application layer. A node with a low ranking in energy is likely to expire sooner than once with a high energy ranking. Thus, based on the energy ranking, the application running in the node can shut down or throttle back energy consuming tasks. For example in a GPS based system, the GPS receiver consumes a large amount of power. To conserve energy, the sampling rate of the system can be reduced, leading to a greater node lifetime at the cost of location resolution. The application can also use lossy compression algorithms to reduce the volume of data that needs to be sent over the radio medium. The amount of compression can be controlled by the rank in the hierarchy.

1.9. Related Work

The primary contribution of this work is the presentation of methods that can be used to rank heterogeneous attributes in a network into a relative framework. As such it is complementary to many routing protocols as it can act as a network underlay, enhancing their performance.

The idea of using hierarchies in networks is not a new one, but to the best of our knowledge this is the first work that has considered the general situation of how to dynamically cast any measurable resource into a network wide ranking system. Many networks use hierarchies that are imposed at design time. These are akin to the static hierarchies discussed in Section 1.2 where there are gross differences that result in a pre-defined social structure. In the Data Mule structure, there are three classes of nodes - low energy stationary nodes, high capability mobile nodes and base-stations.[22]

Some network hierarchies are dynamic. For example in the Low Energy Adaptive Clustering Hierarchy (LEACH), a node is selected from a small group to act as a 'cluster-head' which is responsible for relaying the combined information of the local group.[38] As nodes age, the role of cluster-head is rotated amongst the group. This two-level hierarchy is an example of a despotic or totalitarian structure. Other cluster based schemes have improved on the performance of LEACH, yet still retained the two-tier structure.[39]

The work that is most closely related to ours is the role assignment algorithms of Römer et al,[40].[41] In this scheme, nodes decide on their role within the network based on information acquired from their local neighbourhood, by populating a cache of node properties and running node assignment rules based on their cache. Their algorithms were explicitly designed for stationary networks, as a change in the neighbourhood would result in a cache update. They introduced high level compiler directives that are used to decide on the most suitable role for a node (such as cluster-head or ordinary node), whilst satisfying requirements such as coverage. Their algorithms require the specification of time-out factors and explicit update methods - ASH is lightweight in comparison to the role assignment algorithms as it piggybacks onto existing network control packets providing a transparent evaluation of local role. To control flooding, the role assignment algorithms use a limited hop neighbourhood, whereas ASH discovers network wide information in order to assess rank.

The ranking of nodes can be viewed as sorting them into order according to their attributes. We use a scaled domain though to represent the maximum value of the attribute as 1 and the minimum as 0, so this method is not entirely a simple sorting procedure. Some recent work has been performed on the theory of Random Sorting Networks.[42] Random Sorting Networks are networks for sorting information that are created at random. This is similar to our pairwise ASH scheme presented in Section 1.4 which can also be viewed as a randomized version of a Bubble Sort.

We use a very simple agent based approach. Far more sophisticated ant based routing and discovery methods have been presented in the literature which use stateful agents and pheremone trails,[43],[44].[45] However, it was our intention to keep ASH formation and maintenance as simple and lightweight as possible. This can be seen in our simulation results, where the error dropped to acceptable levels with a cache size of 3 entries and an agent repetition rate of 1 entry per 'Hello' message.

1.10. Conclusions and future work

1.10.1. *Future directions*

This work has discussed methods of forming a social hierarchy amongst nodes in various situations. However, there is scope for further exploration into some of the areas which have yet to be addressed. One such avenue is security and protecting the nodes from malicious attacks. For example,

a rogue node could falsely advertise maximum rank and attribute and so act as an attractor for packets. In this way, it could remove and use information from the network, preventing it from reaching the base-station. In conventional networks, this is equivalent to a node masquerading as a base-station. We are currently exploring methods of protecting the network from attacks of this sort.

All the methods for forming the Adaptive Social Hierarchies form a linear dominance hierarchy. In this type of hierarchy, only ordinality is relevant, not the degree of difference between nodes. In a heavily resource partitioned network, there will be nodes with a large attribute value and nodes with low attribute values, but nothing in between. Such would be the case with small battery powered sensors connected into a mains network, with no intermediate nodes in terms of resource size. In this case, two nodes which are close to each other in rank value could have large difference in resource value. This could place an unfair load on a falsely ranked node. This suggests that a non-linear hierarchy should be formed in this instance. This could be done by nodes exchanging a histogram reflecting the distribution of resource values across the network. We are currently investigating how exactly to undertake this task, whilst keeping network overhead low and being able to rapidly react to network changes.

In Section 1.6 we discussed how to use simple random agents to disseminate rank/attribute information across the network, to recreate the effect of randomized meetings. The agents used are extremely simple and essentially stateless. There is a large body of work on agent based (also known as ant inspired) algorithms for exploring networks and routing data. We are planning on extending the agent based approach in a more refined manner, which will hopefully lead to faster convergence and lower error thresholds.

Another area which needs to be explored is how to factor in the rate of change of rank, both into the rank determination process and also the routing protocol. Thus nodes can be ranked according to their rank reliability or stability. Nodes would thus avoid using intermediates which display large rank variance. We are also looking at using some of the ideas presented in CAR,[31] in particular the prediction process to result in a more stable and useful system.

1.10.2. *Conclusion*

We have presented a novel biologically inspired method of forming a hierarchy based on differences between individual nodes. This hierarchy adapts

to changes in node resources and provides the ability for nodes to determine their role in the network relative to their peers. Three different approaches to forming the hierarchy have been presented. The first method assumes that nodes engage in pairwise meetings and from this sort or exchange their ranks according to the relative order of their attributes. We showed that this leads to poor adaption to node insertion and removal, and to this end refined the protocol by introducing a reinforcement factor which spreads the ranks out evenly across the ranking space.

In the next method, we removed the assumption that nodes undertake pairwise exchange of information, as this imposes constraints on the MAC layer. Instead, we investigated how a single transmitter can broadcast its rank/attribute information to a number of receiving peers. Based on this newly acquired information, they update their ranks using a win/loss ratio method as used by researchers in the biological literature to measure dominance. This was found to have slow convergence and adaption to changes. To remedy this, we used the switching idea from pairwise ASH to result in rapid adaption to resource variation.

Both of the methods are designed with mobile networks in mind. In stationary networks, due to a limited discovery horizon, the ranks do not converge to their correct values. To this end, we presented our third method, Agent ASH, which replicates the effect of randomized meetings by spawning random agents which carry rank/attribute information to non-neighbouring nodes, resulting in correct ASH convergence.

The effect of real world mobility was shown to result in more rapid convergence. Agent ASH works well for both purely stationary and mixed mobility networks, whereas the other two methods work best in mobile networks. To avoid the formation of loops, levels were introduced, along with hysteresis to ensure that messages only permeate upwards through the hierarchy.

Lastly we examined some possible applications for the Adaptive Social Hierarchy approach, showing how it can be used as a network underlay to enhance the performance of existing protocols by providing resource abstraction and also as a powerful cross layer management and routing protocol.

In summary, ASH provides a framework for nodes to discover their role within diverse networks, by allowing resource abstraction. This leads to simpler routing and management protocols, that are removed from the imposition of absolute values. This work is an example of the application of a common method of self-organisation in nature to network management

and control.

1.11. Bibliography

References

1. A. Markham and A. Wilkinson. Ecolocate: A heterogeneous wireless network system for wildlife tracking. In *International Joint Conferences on Computer, Information and Systems Sciences and Engineering (CISSE'07)* (December, 2007).
2. E. O. Wilson, *Sociobiology: The New Synthesis*. (Harvard University Press, 1975).
3. K. Tsuchida, T. Saigo, N. Nagata, S. Tsujita, K. Takeuchi, and S. Miyano, Queen-worker conflicts over male production and sex allocation in a primitively eusocial wasp, *International Journal of Organic Evolution*. **57**(10), 2356–73 (Oct, 2003).
4. T. Monnin and C. Peeters, Dominance hierarchy and reproductive conflicts among subordinates in a monogynous queenless ant, *Behavioral Ecology*. **10** (3), 323–332, (1999).
5. B. Holldobler and E. Wilson, *The Ants*. (Cambridge: Harvard University Press, 1990).
6. T. Schjelderup-Ebbe, Bietrage zur sozialpsychologie des haushuhns, *Zeitschrift fur Psychologie*. **88**, 225–252, (1922).
7. I. Chase, Behavioral sequences during dominance hierarchy formation in chickens, *Science*. **216**(4544), 439–440, (1982).
8. A. Guhl, *Social Hierarchy and Dominance*, chapter Social behavior of the domestic fowl, pp. 156–201. Dowden, Hutchinson, and Ross, (1975).
9. B. Forkman and M. J. Haskell, The maintenance of stable dominance hierarchies and the pattern of aggression: Support for the suppression hypothesis, *Ethology*. **110**(9), 737–744, (2004).
10. F. A. Issa, D. J. Adamson, and D. H. Edwards, Dominance hierarchy formation in juvenile crayfish *procambarus clarkii*, *Journal of Experimental Biology*. **202**(24), 3497–3506, (1999).
11. R. C. Brace and J. Pavey, Size-dependent dominance hierarchy in the anemone actinia equina, *Nature*. **273**, 752–753, (1978).
12. S. Cote, Determining social rank in ungulates: a comparison of aggressive interactions recorded at a bait site and under natural conditions, *Ethology*. **106**, 945–955, (2000).
13. D. Greenberg-Cohen, P. Alkon, and Y. Yom-Tov, A linear dominance hierarchy in female nubian ibex, *Ethology*. **98**(3), 210–220, (1994).
14. S. Altmann, A field study of the sociobiology of rhesus monkeys, *Annals of the New York Academy of Sciences*. **102**, 338–435, (1962).
15. W. Jackson, Can individual differences in history of dominance explain the development of linear dominance hierarchies?, *Ethology*. **79**, 71–77, (1988).
16. I. D. Chase, C. Tovey, and D. S.-M. amd M. Manfredonia, Individual dif-

ferences versus social dynamics in the formation of animal dominance hierarchies, *Proceedings of the National Academy of Sciences*. **99**(8), 5744–5749, (2002).
17. E. Bonabeau, G. Theraulaz, and J. Deneubourg, Dominance orders in animal societies: The self-organization hypothesis revisited, *Bulletin of Mathematical Biology*. **61**(4), (1999).
18. W. Hamilton, Geometry for the selfish herd, *Journal of Theoretical Biology*. **31**, 295–311, (1971).
19. H. Kruuk, Predators and anti-predator behaviour of the black-headed gull *Larus ridibundus*, *Behaviour Supplements 11*. **11**, 1–129, (1964).
20. C. Carbone, J. D. Toit, and I. Gordon, Feeding success in african wild dogs: Does kleptoparasitism by spotted hyenas influence hunting group size?, *The Journal of Animal Ecology*. **66**(3), 318–326, (1997).
21. S. Creel and N. Creel, Communal hunting and pack size in african wild dogs, *Lycaon pictus*, *Animal Behaviour*. **50**, 1325–1339, (1995).
22. D. Jea, A. Somasundara, and M.Srivastava, *Distributed Computing in Sensor Systems*, chapter Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks, pp. 244–257. Springer Berlin / Heidelberg, (2005).
23. M. Kendall, *Rank Correlation Methods*. (New York: Hafner Publishing Co, 1955).
24. C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *IEEE Mobile Computing Systems and Applications*, pp. 90–100, (1999).
25. D. Aldous and J. Fill, *Reversible markov chains and random walks on graphs (monograph in preparation.)*. `http://stat-www.berkeley.edu/users/aldous/RWG/book.html`.
26. C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. Chen, A survey of energy efficient network protocols for wireless networks, *Wireless Networks*. **7**(4), 343–358, (2001).
27. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, Wireless sensor networks: a survey, *Computer Networks Volume 38, Issue 4, 15 March 2002, Pages 393-422*. **38**(4), 393–422, (2002).
28. A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, pp. 28–32, New York, NY, USA, (2007). ACM.
29. C. Toh, Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks, *IEEE Communications Magazine*. **June**, (2001).
30. A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *Service Assurance with Partial and Intermittent Resources*. Springer Berlin/Heidelberg, (2003).
31. M. Musolesi, S. Hailes, and C. Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'05)*, pp. 183–189, (2005).

32. M. Musolesi and C. Mascolo. Evaluating context information predictability for autonomic communication. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pp. 495–499, Washington, DC, USA, (2006). IEEE Computer Society.
33. A. Markham and A. Wilkinson. The adaptive social hierarchy: A self organizing network based on naturally occurring structures. In *1st International Conference on Bio-Inspired mOdels of NEtwork, Information and Computing Systems (BIONETICS), Cavelese, Italy* (11-13 December, 2006).
34. T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, vol. 00, pp. 79–85, Los Alamitos, CA, USA, (2007). IEEE Computer Society.
35. T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 252–259, New York, NY, USA, (2005). ACM.
36. C. Mascolo and M. Musolesi. SCAR: context-aware adaptive routing in delay tolerant mobile sensor networks. In *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pp. 533–538, New York, NY, USA, (2006). ACM.
37. A. Markham and A. Wilkinson. A biomimetic ranking system for energy constrained mobile wireless sensor networks. In *Southern African Telecommunications, Networks and Applications Conference (SATNAC 2007)* (September, 2007).
38. W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 174–185, New York, NY, USA, (1999). ACM.
39. L. Ying and Y. Haibin. Energy adaptive cluster-head selection for wireless sensor networks. In *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, vol. 0, pp. 634–638, Los Alamitos, CA, USA, (2005). IEEE Computer Society.
40. C. Frank and K. Römer. Algorithms for generic role assignment in wireless sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp. 230–242, New York, NY, USA, (2005). ACM.
41. K. Römer, C. Frank, P. J. Marrón, and C. Becker. Generic role assignment for wireless sensor networks. In *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop*, p. 2, New York, NY, USA, (2004). ACM.
42. O. Angel, A. Holroyd, D. Romik, and B. Virag, Random sorting networks, *Advances in Mathematics*. **215**(10), 839–868, (2007).
43. P. B. Jeon and G. Kesidis. Pheromone-aided robust multipath and multipriority routing in wireless manets. In *PE-WASUN '05: Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc,*

sensor, and ubiquitous networks, pp. 106–113, New York, NY, USA, (2005). ACM.

44. F. Ducatelle, G. D. Caro, and L. M. Gambardella. Ant agents for hybrid multipath routing in mobile ad hoc networks. In *Second Annual Conference on Wireless On-demand Network Systems and Services (WONS'05)*, pp. 44–53, Los Alamitos, CA, USA, (2005). IEEE Computer Society. ISBN 0-7695-2290-0.
45. G. Di Caro, F. Ducatelle, and L.M.Gambardella, *Parallel Problem Solving from Nature - PPSN VIII*, chapter AntHocNet: An Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks, pp. 461–470. Springer Berlin/Heidelberg, (2004).

Index