Emergent Consensus in Decentralised Systems Using Collaborative Reinforcement Learning

Jim Dowling, Raymond Cunningham, Anthony Harrington,
Eoin Curran, and Vinny Cahill

Distributed Systems Group, Trinity College, Dublin, Ireland

Abstract. This paper describes the application of a decentralised coordination algorithm, called Collaborative Reinforcement Learning (CRL), to two different distributed system problems. CRL enables the establishment of consensus between independent agents to support the optimisation of system-wide properties in distributed systems where there is no support for global state. Consensus between interacting agents on local environmental or system properties is established through localised advertisement of policy information by agents and the use of advertisements by agents to update their local, partial view of the system.

As CRL assumes homogeneity in advertisement evaluation by agents, advertisements that improve the system optimisation problem tend to be propagated quickly through the system, enabling the system to collectively adapt its behaviour to a changing environment. In this paper, we describe the application of CRL to two different distributed system problems, a routing protocol for ad-hoc networks called SAMPLE and a next generation urban traffic control system called UTC-CRL. We evaluate CRL experimentally in SAMPLE by comparing its system routing performance in the presence of changing environmental conditions, such as congestion and link unreliability, with existing ad-hoc routing protocols. Through SAMPLE's ability to establish consensus between routing agents on stable routes, even in the presence of changing levels of congestion in a network, it demonstrates improved performance and self-management properties. In applying CRL to the UTC scenario, we hope to validate experimentally the appropriateness of CRL to another system optimisation problem.

1 Introduction

In the future, many distributed systems will consist of interacting, autonomous components that organise, regulate and optimise themselves without human intervention. However, with increasing system size and complexity our ability to build self-managed distributed systems using existing programming languages, top-down design techniques and management infrastructures is reaching its limits [1], as solutions they produce require too much global knowledge.

Self-managing distributed computer systems, on a scale comparable with biological autonomic systems, require a decentralised, bottom-up approach to their

O. Babaoglu et al. (Eds.): SELF-STAR 2004, LNCS 3460, pp. 63–80, 2005.

construction. Self-managing decentralised systems can be modelled as collections of self-managing agents using decentralised coordination to enable a weak form of consensus to emerge between a group of agents partial views of the system [2, 3]. Consensus between the agents' partial views of the system can be used as a basis for system optimisation, coordinating the execution of self-management actions and the collective adaptation of agents to a changing and uncertain environment. The benefits of such an approach include improved scalability, the possibility of establishing self-management properties, self-optimisiation, the lack of centralised points of failure or attack, as well as possible system evolution by evolving the coordination models of the agents.

The construction of self-managing distributed systems using decentralised coordination models presents a number of challenges. These include designing a suitable representation for an agent's local view of the system, the provision of self-management actions for agents that allow them to adapt to changes in their local environment, and the design of feedback models that update the agent's local view of the world.

This paper discusses collaborative reinforcement learning (CRL), as a technique for building decentralised coordination models that addresses these challenges. CRL extends reinforcement learning (RL) with positive and negative feedback models to update an agent's policy, i.e., its local model of how to interact with the system, and enable a certain degree of consensus to emerge between agent policies. We introduce two application areas for CRL, a routing protocol for Mobile Ad Hoc Networks (MANETs) called SAMPLE and an Urban Traffic Control (UTC) system called UTC-CRL. Both of these systems are designed to leverage consensus between agent policies to implement self-managing system properties as system optimisation behaviours. The goal of CRL is to enable agents to produce collective behaviour that establishes and maintains the desired system-wide self-management properties. However, in open, dynamic distributed systems, the environment is non-stationary and CRL enables agents to continually update their consensus on more optimal policies in order to be able to maintain the system-wide self-management properties in a changing environment. In the evaluation of SAMPLE we show how the protocol enables routing agents to continually maintain consensus on any stable routes in the network to improve system routing performance.

This paper is structured as follows. Section 2 introduces decentralised coordination and is followed in section 3 by a description of the CRL model. Section 4 presents SAMPLE as both a CRL system and an on-demand routing protocol for ad hoc networks. We compare simulation results for SAMPLE with two widely used on-demand MANET routing protocols in different scenarios and explain the differing abilities of the protocols to adapt and optimise to a view of the MANET environment. Section 5 introduces our approach to applying CRL to the problem of Urban Traffic Control. The final section provides some conclusions to the work presented in this paper.

2 Self-Managing Systems Using Decentralised Coordination

Coordination is the process of building programs by gluing together active pieces [4], where active pieces can be processes, autonomous objects, agents or applications. Coordination is the logic that binds independent activities together into a collective activity. Both centralised and decentralised coordination models have been developed to describe the "glue" that connects computational activities. A multi-agent system built using a centralised coordination model is one where the behaviour of the agents in the system is controlled either by an active manager agent or by a predetermined design or plan followed by the agents in the system [5]. A system built using a decentralised coordination model is a self-organising multi-agent system [5], whose system-wide structure or behaviour is established and maintained solely by the interactions of its agents that execute using only a partial view of the system.

Coordination models are necessary for the construction of self-managing distributed systems as they organise the self-management and self-adaptive behaviour of agents towards system goals. A lack of coordination among agents in a distributed system can lead to interference between the different agents' self-management behaviour, conflicts over shared resources, suboptimal system performance and hysteresis effects [6]. For example, a distributed system that is composed of self-managing agents, where agents optimise their behaviour towards agent goals is not necessarily optimised at the system-level, as there is the possibility that conflicting greedy decisions taken by agents may result in sub-optimal resource utilisation or performance at the system-level. In order to optimally adapt a system to a changing environment the agents must respond to changes in a coordinated manner, but in decentralised environments, the coordination mechanism cannot be based on centralised or consensus-based techniques.

Increasingly, researchers are investigating decentralised coordination approaches to establish and maintain system properties [7, 8, 9, 10]. Decentralised control is based on defining local coordination or control models for components that only have partial views of the system, support only localised interaction and have no global knowledge. Agents typically store locally a partial, estimated model of the system and interaction protocols defined between neighbouring agents enables them to collectively improve the accuracy of their local, estimated models [11, 12]. This can often result in convergence between the estimated models of neighbouring agents on a common view of the system or environment [13]. Agents that have converged models can coordinate their behaviour using their local models to perform collective adaptive behaviour that can establish and maintain system-wide properties. These system properties emerge from the local interaction between neighbouring agents and with no explicit representation of system properties on the level of the individual agent [7, 8].

Decentralised coordination techniques have been developed that are based on cooperation [10, 11] and competition [14] between agents. Both approaches are

typically evaluated by how they optimise some system property, such as a self-managing property of the system. There is also the possibility that the system may attempt to optimise more than one system property. Multiple objective functions can be used to describe optimisation problems where there is more than one competing objective function.

Some problems associated with decentralised models include the uncertain outcome of control actions on agents, as their effect may not be observable until some unknowable time in the future. Also, optimal decentralised control is known to be computationally intractable [9], although systems can be developed where system properties are near-optimal [15, 16, 13, 12], which is often adequate for certain classes of applications.

3 Collaborative Reinforcement Learning

CRL is a decentralised approach to establishing and maintaining system-wide properties in distributed systems. CRL is an extension to Reinforcement Learning (RL) [17] for decentralised multi-agent systems. CRL does not make use of system-wide knowledge and individual agents only know about and interact with their neighbouring agents.

CRL can be used to implement decentralised coordination models based on cooperation and information sharing between agents using coordination actions and various feedback models, respectively. The feedback models include a negative feedback model that decays an agent's local view of its neighbourhood and a collaborative feedback model that allows agents to exchange the effectiveness of actions they have learned with one another. In a system of homogeneous agents that have common system optimisation goals and where agents concurrently search for more optimal actions in different states using Reinforcement Learning (RL), collaborative feedback enables agents to share more optimal policies [17], increasing the probability of neighbouring agents taking the same or related actions. This process can produce positive feedback in action selection probability for a group of agents. Positive feedback is a mechanism that reinforces changes in system structure or behaviour in the same direction as the initial change and can cause the emergence of system behaviour or structure [3, 18]. In CRL, the positive feedback process continues until negative feedback, produced either by constraints in the system or our decay model, causes agent behaviour to adapt so that agents in the system converge on stable policies. In effect, agents can establish consensus with their neighbours on more optimal self-management actions to take given a particular system state.

Given a certain degree of consensus between agents on their policies, the goal of system optimisation is to have the agents' policies converge on values that produce collective behaviour that meets the system optimisation criteria [19]. However, in open dynamic distributed systems, the system's environment is non-stationary and we require agents than can collectively adapt their behaviour to the changing environment to continue to meet the system optimisation criteria. We believe that the adaptability of system behaviour to changes in its environ-

ment is as important an evaluation criterion for complex adaptive distributed systems as the more traditional criterion for static environments of convergence and stabilisation on optimal system behaviour.

3.1 Reinforcement Learning

In RL, an agent associates actions with system states in a trial-and-error manner and the outcome of an action is observed as a reinforcement that, in turn, causes an update to the agent's *action-value policy* using a reinforcement learning strategy [17]. The goal of reinforcement learning is to maximise the total *reward* (reinforcements) an agent receives over a time horizon by selecting optimal *actions*. Agents may take actions that give a poor payoff in the short-term in the anticipation of higher payoff in the longer term. In general, actions may be any decisions that an agent wants to learn how to make, while states can be anything that may be useful in making those decisions.

RL problems are usually modelled as Markov decision processes (MDPs) [17, 20]. A MDP consists of a set of states, $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, a set of actions, $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$, a reinforcement function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and a state transition distribution function: $P : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$, where $\Pi(\mathcal{S})$ is the set of probability distributions over the set $\mathcal{S}$.

3.2 Coordination in CRL

CRL system optimisation problems are decomposed into a set of discrete optimisation problems (DOPs) [7] that are solved by collaborating RL agents. The solution to each DOP is initiated at some starting agent in the network and terminated at some (potentially remote) agent in the network. Each agent uses its own policy to decide probabilistically on which action to take to attempt to solve a DOP. In CRL the set of available actions that an agent can execute include *DOP action*s, $\mathcal{A}_{p_i}$, that try to solve the DOP locally, *delegation action*s, $\mathcal{A}_{d_i}$, that delegate the solution of the DOP to a neighbour and a *discovery action* that any agent can execute in any state to attempt to find new neighbours.

The goal of CRL is to coordinate the solution to the set of DOPs among a group of agents using delegation and discovery actions. This is achieved by ensuring that an agent is more likely to delegate a DOP to a neighbour when it either cannot solve the problem locally or when the *estimated cost* of solving it locally is higher than the estimated cost of a neighbour solving it.

3.3 Partial Views of System

In dynamic distributed systems, agents typically have a changing number of neighbouring agents that can be used to help solve a given DOP. To model an agent's dynamic set of neighbours, CRL allows agents to execute discovery actions and then establish *causally-connected states* with any newly discovered neighbour. Causally-connected states represent the contractual agreement between neighbouring agents to support the delegation of DOPs from one to the other. Causally-connected states map an *internal state* on one agent to an *external state* on at least one neighbouring agent. An internal state on one agent

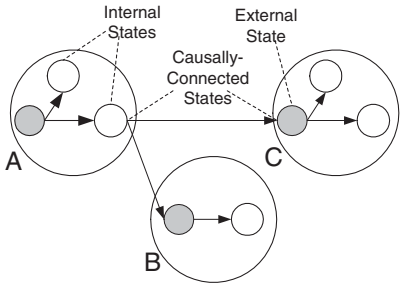


Fig. 1. Causally-Connected States between the MDPs in Agents A, B, C

can be causally-connected to external states on many different neighbouring agents, see Figure 1. An agent's set of causally connected neighbours represents its *partial-view* of the system.

3.4 Feedback Models

There are various feedback models in CRL, including a collaborative feedback model that allows agents to exchange the effectiveness of actions they have learned with one another and a negative feedback model that decays an agent's local view of its neighbourhood.

Collaborative Feedback as Advertisement. In CRL, neighbours are informed of changes to $V_j(s)$ of a causally-connected external state, s, at agent n_j using an *advertisement* . Each agent maintains a local view of its neighbours by storing V values for causally-connected states to neighbours in a local cache, $Cache_i$. The cache consists of a table of Q-values, for all delegation actions, a_d, at agent n_i, and the last advertised $V_j(s)$ for successful transition to the causally connected state. A $Cache_i$ entry is a pair $(Q_i(s, a_j), r_j)$, where r_j is the cached $V_j(s)$ value. When the agent n_i receives a $V_j(s)$ advertisement from neighbouring agent n_j for a causally-connected state s, it updates r_j in $(Q_i(s, a_j), r_j)$. Examples of implementation strategies for V advertisement in distributed systems include periodic broadcast/multicast, conditional broadcast/multicast, piggybacking advertisement in transmission/acknowledgement packets and event-based notification.

Decay as Negative Feedback. In certain decentralised systems an agent's set of neighbours changes dynamically and to overcome problems related to unreachable neighbours, an agent's cached V_j values become stale using a *decay* model [7]. In CRL, we decay cached V_j information in the absence of new advertisements of V_j values by a neighbour as well as after every recalculation of Q_i values. The absence of V_j advertisements is one aspect of negative feedback and allows the removal of stale cache entries. The rate of decay is configurable, with higher rates more appropriate for more dynamic network topologies.

3.5 Distributed Model-Based Reinforcement Learning

CRL enables system optimisation in multi-agent RL systems, and states in the multi-agent system may be distributed over different agents in the network. As a result, state transitions can be either to a local state on the current agent or to a remote state on a neighbouring agent. In distributed systems, when estimating the cost of the state transition to a remote state on a neighbouring agent we also have to take into consideration the network *connection cost* to the neighbouring agent. For this reason, we use a *distributed model-based reinforcement learning algorithm* that includes both the estimated optimal value function for the next state at agent n_i, $V_j(s')$, and the connection cost, $D_i(s'|s,a) \in \mathbb{R}$ where $a \in \mathcal{A}_{d_i}$, to the next state when computing the estimated optimal state-action policy as $Q_i(s,a)$ at agent n_i (see equation (1)).

In the learning algorithm of CRL, our reward model consists of two parts. Firstly, a *MDP termination cost*, $R(s,a) \in \mathbb{R}$, provides agents with evaluative feedback on either the performance of a local solution to the DOP or the performance of a neighbour solving the delegated DOP. Secondly, a connection cost model, $D_i(s'|s,a)$, provides the estimated network cost of the attempted delegation of the DOP from a local agent to a neighbouring agent. The connection cost for a transition to a state on a neighbouring agent should reflect the underlying network cost of delegating the DOP to a neighbouring agent, while the connection cost for a transition to a local state after a delegation action should reflect the cost of the failed delegation of the DOP. The connection cost model requires that the environment supplies agents with information about the cost of distributed systems connections as rewards.

The update rule used in the learning algorithm or CRL is:

$$Q_i(s,a) = R(s,a) + \sum_{s' \in S_i} P_i(s'|s,a).$$

$$(D_i(s'|s,a) + Decay\,(V_j(s'))) \tag{1}$$

where $a \in \mathcal{A}_{d_i}$. If $a \notin \mathcal{A}_{d_i}$, this defaults to the standard model-based reinforcement learning algorithm [20] with no connection costs or decay function. $R(s,a)$ is the MDP termination cost, $P(s'|s,a)$ is the state transition model that computes the probability of the action a resulting in a state transition to state s', $D_i(s'|s,a)$ is the estimated connection cost and $V_j(s')$ is $r_j \in Cache_i$ if $a \in \mathcal{A}_d$, and $V_i(s')$ otherwise.

3.6 Emergent Consensus

We advocate an experimental approach to the validation of the emergence of consensus in CRL systems. Due to the lack of a global view of the system, individual agents cannot validate the existence of system properties and their existence cannot be validated analytically. We can only validate their existence through external observation of the system, e.g., through experimentation. The process through which agents can achieve consensus in CRL systems is decentralised coordination using coordination actions and feedback. They enable con-

sensus to emerge between groups of agents on more optimal actions to execute actions given shared system states. The convergence of agent policies in CRL is a feedback process in which a change in the optimal policy of any RL agent, or a change in the system's environment as well as the passing of time causes an update to the optimal policy of one or more RL agents. In CRL, changes in an agent's environment can provide feedback into the agent's state transition model and connection cost model, while changes in an agent's optimal policy provides collaborative feedback to the cached V values of its neighbouring agents using advertisement. Time also provides negative feedback to an agent's cached V values and allows converged policies to be 'forgotten', enabling policies to converge again on different behaviours.

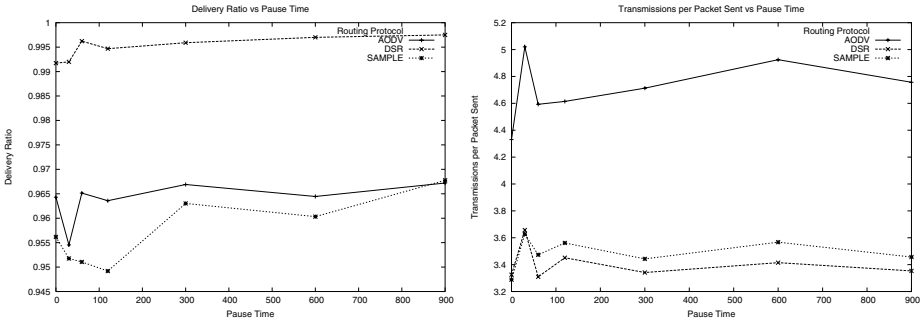
As a result of the different feedback models in CRL, agents can utilise more information about the state of the system and their neighbouring agents to enable groups of agents to converge on similar policies. In particular, collaborative feedback enables agents to share policy information with their neighbours and achieve consensus on more optimal actions to execute in given system states. This consensus between neighbouring agents on the actions to execute, assuming the agents perceive the system to be in a similar state, can be the basis for collective behaviours such as system self-management behaviour.

4 SAMPLE and CRL

The CRL model was used to build a MANET routing protocol called SAMPLE [12]. SAMPLE is a probabilistic on-demand ad hoc routing protocol that contains system-wide self-managing properties, such as the adaptation of network traffic patterns around areas of congestion and wireless interference and the exploitation of stable routes in the environment.

Ad hoc routing is a challenging problem as it exhibits properties such as the lack of global network state at any particular node in the network and frequently changing network topology due to node mobility. This ensures that system properties of the protocol only emerge from local routing decisions based on local information and that routing agent's behaviour has to frequently adapt to a changing environment.

Two major assumptions of the two most popular MANET routing protocols, Ad-Hoc On-Demand Distance Vector routing (AODV) [21] and Dynamic Source Routing (DSR) [22], is that the network has a random topology and that all radio links function perfectly. Both protocols make these static assumption about the MANET environment in order to avoid route maintenance problems typically encountered by proactive routing protocols in MANETs. AODV and DSR use on-demand routing where routes to destinations are only discovered when needed using flooding [21, 22] and these routes are discarded when a number of transmission failures over a network link occur, even though that transmission failure may be due to a temporary phenomena such as congestion or wireless interference.



(a) Delivery Ratio (b) Transmissions Per Packets Sent

Fig. 2. Perfect Network Links in a Random Topology. Performance with Varying Load. 64 byte packets

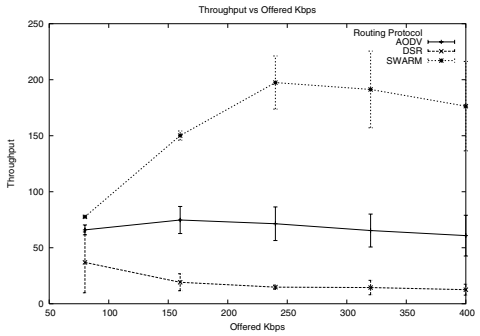
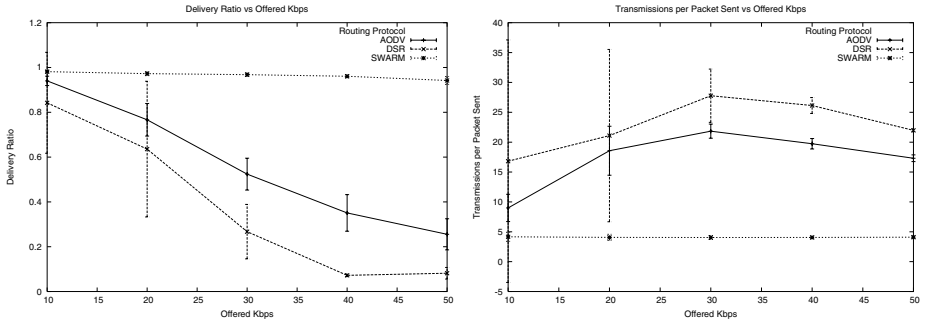


Fig. 3. At a varying load, 512 byte packets, SAMPLE delivers a data throughput of up to 200Kbps. This throughput approaches the theoretical limit of the throughput achievable in a multi-hop 802.11 mobile ad hoc network scenario

In our evaluation of SAMPLE, in sections 4.1 and 4.4, we show how CRL enables routing agents in SAMPLE to establish consensus on the location of stable routes in a network with different quality network links and use this consensus to optimise system routing performance. In SAMPLE we attempt to optimise multiple, often conflicting system routing performance criteria, including maximising overall network throughput, maximising the ratio of delivered packets to undelivered packets and minimising the number of transmissions required per packet sent.

4.1 SAMPLE Experiments

We have implemented the SAMPLE routing protocol in the NS-2 network simulator [23]. We compare the performance of the SAMPLE routing protocol to that



(a) Delivery Ratio (b) Transmissions Per Packet Sent

Fig. 4. Stable Route Topology. Performance with Varying Load. 64 byte packets

of AODV and DSR, in two different scenarios. The first scenario is a random network scenario that is designed to test how SAMPLE compares with AODV and DSR when the MANET environment reflects their static design assumptions. The second scenario is a metropolitan area MANET, where a subset of the links in the network are stable, that is designed to test the ability of the protocols to adapt their routing behaviour to achieve consensus on the location of the stable routes in the network. We also introduce congestion into both scenarios to investigate the effectiveness of the protocols in adapting their routing behaviour to a changing MANET environment. Our goal here is to compare the performance of the SAMPLE routing protocol in different, changing environments with on-demand protocols that make static assumptions about the MANET environment using the system optimisation criteria identified in Section 4.

The SAMPLE routing protocol combines routing information with data packets, and as a result the metric of *number of routing packets* is not a valid one for comparison with AODV and DSR. For this reason, we use the number of transmissions (unicast or broadcast) that each protocol makes per application packet sent during the simulation run as a metric to compare the protocols. This metric represents the cost to the network of routing each data packet.

4.2 Perfect Network Links in a Random Topology

The comparative performance of the protocols in a scenario that mimics the random topology used in [24] is first evaluated. A simulation arena of 1500m x 300m is used, with the transmission power of the radio interfaces set to 250m. The random way-point mobility model is used, with a maximum speed of 20 m/s and varying pause times. Constant bit rate traffic of 64 byte packets, 4 packets per second, with 10 flows between random pairs of nodes is used.

We compare the SAMPLE routing protocol to AODV and DSR in an idealised experiment with no packet loss added to the simulation. Figure 2 shows the packet delivery ratio and transmissions-per-packet metrics as they vary with

the level of mobility in the network. In this scenario, both AODV and DSR have near-optimal packet delivery ratios, while SAMPLE has between 1% and 4% worse packet delivery ratios, and all protocols have a near-minimal cost (in terms of network transmissions). As there is no packet loss and all links are of perfect quality, AODV's favouring of routes with the shortest hop-count shows the best performance.

4.3 Achieving Consensus on Stable Links in a Metropolitan Area MANET

We have also evaluated the performance of SAMPLE against that of AODV and DSR in a network scenario based on a metropolitan ad hoc network. In this scenario, there are a subset of nodes in the network that are not mobile. The network scenario is motivated by the recent appearance of ad hoc networks designed to supply Internet access to mobile nodes.

In this scenario, we anticipate that certain nodes in the network will be immobile for extended periods of time, resulting in stable links between them, and that the traffic patterns in the network will be concentrated on the subset of nodes that have Internet connectivity. In the experiments presented here we use 3 server nodes. Each client sends constant-bit-rate traffic to one of the servers at a rate of 4 packets per second. The number of client nodes in the network is varied in order to create congestion in the network. There are 33 fixed nodes in these simulations, and 50 mobile nodes. The fixed nodes in the simulation provide stable links in the network which the routing protocols could exploit. SAMPLE's configurable parameters have been tuned to provide improved performance for this scenario, see [12] for more details.

Figure 4 shows the variation in performance of the three routing protocols as the number of clients in the network is increased. The packet size sent by clients was kept fixed at 64 bytes, sent 3 times a second.

As the number of clients in the network is increased, the offered throughput to the routing protocols is increased. This in turn increases the level of packet loss and the amount of contention that the Media Access Control (MAC) protocol must deal with. This increased congestion increases the number of failed MAC unicasts in the network. Figure 4 shows that increased packet loss results in lower throughput and packet delivery ratios in AODV and DSR, but that SAMPLE is able to maintain high throughput and packet delivery ratios with high levels of packet loss.

In [25] it was demonstrated that for multi-hop 802.11 networks, the achievable throughput is significantly less than the transmission rate of the radio interfaces. The maximum achievable data throughput in an 802.11 ad hoc network is approximately 0.25Mbps (which [25] achieved using 1500 byte packets). Figure 3 shows that SAMPLE manages to approach this limit in this scenario. This shows experimentally that using CRL, SAMPLE can meet the system optimisation criteria of maximising network throughput in the metropolitan area MANET scenario.

4.4 Emergent Consensus and System Optimisation in SAMPLE

In the metropolitan area MANET environment, there are a subset of the links in the network that are stable. Collaborative feedback adapts routing agent behaviour to favour paths with stable links, producing the emergent stable routes in the network. AODV and DSR do not allow nodes to differentiate between multiple available links in this manner, due to their static assumptions about network links. In SAMPLE positive feedback in link selection by routing agents means that the routing behaviour of agents converges on the stable network links in the environment. SAMPLE maintains a near-optimal delivery ratio and using a minimal number of transmissions even at high offered throughput for 802.11 MANETs. An important lesson from SAMPLE is the need for experimentation, as the emergence of more optimal routing properties is sensitive to tunable parameters in CRL and to the update rule of the learning algorithm used in SAMPLE.

5 Urban Traffic Control and CRL

We believe that Urban Traffic Control is an appropriate application domain for the CRL technique. The UTC and MANET application domains exhibit some similar characteristics. Road traffic-signal controllers can be modelled as autonomous agents. Routing network traffic along a link is analogous to running a green signal-control phase for this street and the system goal is to maximise overall road network throughput by altering signal timings to minimise congestion and vehicle delay.

The suburban road network provides the links between agent controllers. The vehicle density or flow-rate provides a measure of the attractiveness or general fitness of this link. If the quality of service or throughput of the link degrades suddenly due to traffic incidents, sudden increase in traffic volumes or traffic signal operation then the link will become congested and it will be sub-optimal to route vehicles along this link.

The signal-control agent obtains information about its local environment from its sensor infrastructure. The overall UTC system objective may be to optimise vehicle throughput the network however this policy must be implemented by autonomous agents that only possess timely and verifiable information about their local environment. The key challenge of trying to implement a global optimisation strategy based on partial knowledge of a dynamic system is therefore common to both the UTC and MANET application domains.

5.1 Optimisation in Urban Traffic Control

Advances in sensor infrastructure resulting in online vehicle and traffic flow detection have enabled the advent of adaptive traffic control systems capable of online generation and implementation of signal timing parameters[26]. Adaptive control systems such as SCOOT[37] and SCAT [38] are widely deployed throughout the world.

However the adaptive traffic control systems that are currently deployed are hampered by the lack of an explicit coordination model for junction or agent collaboration and are typically reliant on prespecified models of the environment that require domain expertise to construct. These models are typically used as an aid to sensor data interpretation and strategy evaluation and may often be too generic to adequately reflect highly dynamic local conditions[28].

These systems have a limited rate of adaptivity and are designed to respond to gradual rather than rapid changes in traffic conditions. They employ centralised data processing and control algorithms that do not reflect the localised nature of traffic disturbances and flow fluctuations. Yagar and Dion[36] state that: "smooth traffic and uncomplicated networks provide good opportunities for progression, in which case a centralised quasi real-time model, such as SCOOT is appropriate as a model of central control. However, when faced with significant demand fluctuation or interference..., a distributed real-time model is required".

Current research on next generation UTC systems is focusing on applying novel control strategies and AI techniques to the domain. Hoar et. al [27], have investigated the application of swarm intelligence to cooperative vehicle control and isolated signal setting optimisation. Their approach is based on the stigmergic model of environment mediated communication to enable inter-vehicle and vehicle-signal controller cooperation. The fitness function evaluated seeks to minimise the average waiting time of all vehicles in the network. Initial tests indicate promising results and increased adaptivity to high rates of change in traffic flow.

Abdulhai et. al [28], have applied reinforcement learning techniques to UTC control. Using an unsupervised learning technique they have demonstrated that an AI approach can at least match the performance of expert-knowledge based pre-timed signal plans without the requirement of a network model or traffic engineering expertise. The system attempts to minimise the total delay incurred by vehicles in all queues on a junction approach. The evaluation results relate to the operation of a single intersection controller and does not substantially address distributed signal coordination between agent junctions.

The authors cited above have applied novel techniques to the UTC domain however they are essentially adopting centralised approaches to the problem. Their evaluation scenarios are not representative of a real-sized UTC network. Such approaches are not scalable and Lo and Chow[39] highlight the difficulty faced by attempting to optimise a real-sized UTC problem in a centralised fashion. They apply a genetic algorithm based control strategy with possible solution sets encoded as binary strings. For a network with 3 agent junctions operating with a constrained number of control variables the sample solution space expands to 2^{70}. The addition of agent junctions capable of a normal set of control actions causes the solution space to grow exponentially and results in an intractably large solution space.

We believe that adopting a decentralised approach to UTC optimisation is the only feasible method of tackling the problem on a large scale. We believe this decentralisation is best attained by modelling individual junctions as agents and

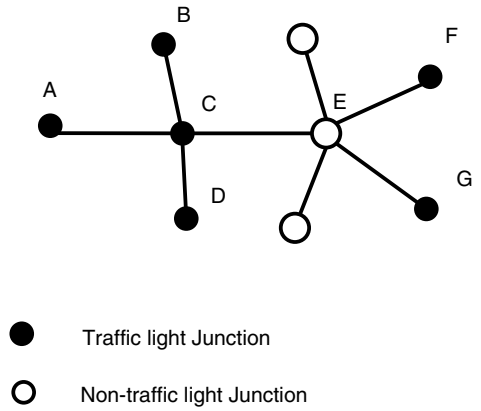


Fig. 5. Sample Layout of a set of Junctions

allowing consensus to emerge between cooperating agents on a suitable system control strategy.

5.2 UTC-CRL

The similarities between the UTC and MANET environments coupled with difficulties exposed by current UTC strategies has prompted the application of the CRL model to the UTC domain. A large scale UTC system may be modelled in a decentralised manner using CRL. Using the UTC-CRL approach, traffic lights would act as agents, choosing a particular action to take, such as changing from one phase to another and receiving a reward from the environment, for taking this action. UTC-CRL allows traffic light agents to share their local view of the congestion at their junction with neighbouring agents to enable convergence towards a shared view of the congestion in the surrounding area.

Using CRL to tackle this problem has a number of advantages. Firstly, CRL gives an explicit framework to model the collaboration and coordination between agents. Secondly, as CRL is a distributed learning technique, traffic light agents collaborate to allow consensus to be established on the optimal phases to choose at a junction in an UTC environment that is uncertain and dynamic. This in turn allows the flows of traffic in an area to be optimised by the collaborating traffic light agents. Positive feedback is used to reinforce the selection of such optimal phases. Negative feedback discourages traffic light agents to act in a greedy fashion as this adversely impacts on congestion levels at neighbouring traffic light agents.

For example in figure 5, a traffic light agent at junction C uses information from its neighbouring traffic light agents, $\{A, B, D, F, G\}$ to optimise the flow of traffic in the area surrounding junction C. Similarly, junction C communicates with its neighbours to allow them to optimise the flow of traffic in their individual area. The traffic light agent would then communicate its view of the environment to the traffic light agents that are both upstream and downstream of it.

Indirect feedback is given by vehicles crossing the junction from one or more approaches to one or more exits. Secondly, traffic light agents advertise their view of their local environment to neighbouring agents. Instead of advertising the cost to a particular destination as in SAMPLE, a traffic light agent in UTC-CRL advertises its view of the level of congestion associated with a particular approach from an upstream junction or associated with a particular exit towards a downstream junction.

For example in figure 5, junction C would advertise to junction A its view of the congestion on the approach from junction A. Similarly, junction A would advertise to junction C its view of the congestion on the exit from junction A towards junction C. By incorporating each of these views into their calculations, junctions A and C can establish convergent views of the congestion level which can then in turn inform their collective decisions of which action is most appropriate.

The type of system optimisation achievable in UTC-CRL depends on the actions, states and rewards used. The actions available to a particular junction correspond to a subset of the possible phases at that junction. The set of possible states relate to the type of system optimisation desired. For example, one optimisation criteria may be to minimise average number of vehicles waiting at a junction. Given such a criteria, the states of the traffic light agent at that junction correspond to a tuple of values that represent the number of vehicles waiting on each approach to that junction. Another possible optimisation criteria would be to maximise the average velocity of vehicles in the system. This would require the average velocity of vehicles on each approach to be wirelessly communicated to the traffic light agent at the junction.

The reward model depends on the particular optimisation strategy being pursued. When trying to minimise the number of vehicles waiting at a junction, the reward received by an agent on executing an action should be related to the number of vehicles that traverse the junction as a result of the particular action being selected. Alternatively, when trying to maximise the average velocity, the reward received by executing an action should be related to the change in the velocities of the vehicles on the various approaches to the junction.

Finally, as CRL is a completely decentralised approach with agents having a similar number of states and actions, this should allow UTC-CRL to scale to a large number of traffic light junctions.

5.3 Evaluation Strategy

As highlighted in section 4.1, SAMPLE takes an experimental approach to validate the emergence of consensus between mobile hosts in a MANET. In a similar approach, UTC-CRL is taking an experimental approach to validate the appropriateness of CRL in a large scale UTC setting. In particular, an objective of the UTC-CRL experimental approach is to verify that consensus can emerge between collaborating traffic light agents and this emergence of consensus allows optimisation of traffic flows in this large scale setting. The envisioned setting for this work corresponds to the Dublin city area which consists of 248 traffic light

junctions, over 750 non-traffic light junctions and over 3000 links joining these junctions. Given this layout of junctions, models of traffic flows between different areas of the city are currently under development that correspond to currently available census data for the greater Dublin area [31]. We have constructed a model of the road network for Dublin city centre and a traffic model containing realistic vehicle volumes. We are currently evaluating MDPs to validate the appropriateness of CRL in this domain.

6 Conclusions

Decentralised computing systems should establish and maintain consensus on environmental or system properties with minimal external intervention in order to provide system-wide self-managing behaviour. The challenges of how to achieve consensus between agents' local, partial views of the system in order to establish such properties can be met using decentralised coordination techniques. CRL is one such coordination technique that enables consensus on optimal policies to emerge between interacting agents in a decentralised system. In this paper we described CRL as a decentralised coordination technique and discussed its application to a MANET routing protocol called SAMPLE and Urban Traffic Control problems. From our evaluation of SAMPLE as a CRL system, we show that CRL through feedback, advertisement and decay enables consensus to emerge on stable routes in a MANET and how this can be used to improve the system routing performance in MANETs with stable nodes.

References

1. Montresor, A., Meling, H., Babaoglu, O.: Towards self-organizing, self-repairing and resilient distributed systems. Future Directions in Distributed Computing. vol LNCS 2584 (2003).
2. Visscher, P.: How self-organization evolves. Nature. vol.421 799–800 (2003).
3. Camazine, S., Deneubourg, J., Franks, N., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press (2003).
4. Gelernter, D., Carriero, N.: Coordination languages and their significance. Commun. ACM vol.35 no.2 97–107 (1992).
5. Goldin, D. and Keil, K.: Toward domain-independent formalization of indirect interaction. 2nd Int'l workshop on Theory and Practice of Open Computational Systems (TAPOCS) (2004).
6. Efstratiou, C., Friday, A., Davies, N., Cheverst, K.: Utilising the event calculus for policy driven adaptation in mobile systems. Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (2002) IEEE Computer Society 13–24 June (2002).
7. Dorigo, M. Di Caro, G.: The ant colony optimization meta-heuristic. New Ideas in Optimization (1999).
8. Andrzejak, A., Graupner, S., Kotov, V., Trinks, H.: Adaptive control overlay for service management. Workshop on the Design of Self-Managing Systems. International Conference on Dependable Systems and Networks (2003).

9. De Wolf, T. and Holvoet, T.: Towards autonomic computing: agent-based modelling, dynamical systems analysis, and decentralised control. Proceedings of IEEE International Conference on Industrial Informatics 470–479 (2003).
10. Boutilier, C., Das, R., Kephart, J., Tesauro, G., Walsh, W.: Cooperative negotiation in autonomic systems using incremental utility elicitation. Uncertainty in Artificial Intelligence (2003).
11. Khare, R., Taylor, R.N.: Extending the representational state transfer (rest) architectural style for decentralized systems. Proceedings of the International Conference on Software Engineering (ICSE) (2004).
12. Curran, E., Dowling, J.: Sample: An on-demand probabilistic routing protocol for ad-hoc networks. Technical Report Department of Computer Science Trinity College Dublin (2004).
13. Jelasity, M., Montresor A., Babaoglu, O.: A modular paradigm for building self-organizing peer-to-peer applications. Proceedings of ESOP03 International Workshop on Engineering Self-Organising Applications (2003).
14. Panagiotis, T., Demosthenis, T., Mackie-Mason, J.-K.: A market-based approach to optimal resource allocation in integrated-services connection-oriented networks. Operations Research vol.50 4 July-August 2002.
15. Littman, M., Boyan, J.: A distributed reinforcement learning scheme for network routing. Technical Report CS-93-165 (1993).
16. Di Caro, G., Dorigo, M.: AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research vol 9 317–365 (1998).
17. Sutton, R., Barto, A.: Reinforcement Learning. MIT Press (1998).
18. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: from natural to artificial systems. New York Oxford University Press (1999).
19. Crites, R., Barto, A.: Elevator group control using multiple reinforcement learning agents. Machine Learning volume 33 2-3 235–262 (1998).
20. Kaelbling, L., Littman, M., Moore, A. Reinforcement learning: A survey. Journal of Artificial Intelligence Research vol 4 237–285 (1996).
21. Perkins, C.: Ad Hoc On Demand Distance Vector (AODV) Routing. IETF Internet Draft November (1997).
22. Johnson, D., Maltz, D., Broch, J.: DSR: The dynamic source routing protocol for multihop wireless ad hoc networks. Ad Hoc Networking 139–172 Addison-Wesley (2001).
23. NS-2 network simulator. Information Sciences Institute (2003).
24. Broch, J., Maltz, D., Johnson, D., Hu, J., Jetcheva, J.: A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. Mobile Computing and Networking 85–97 (1998).
25. Li, J., Blake, C., De Couto, D., Lee, H., Morris, R.: Capacity of ad hoc wireless networks. Proceedings of the 7th International Conference on Mobile Computing and Networking 61–69 (2001).
26. Klein, L.: Sensor Technologies and Data Requirements for ITS. Artech House (2001).
27. Hoar, R., Penner, J., Jacob, C.: Evolutionary Swarm Traffic: If Ant Roads had Traffic Lights. Proceedings of the IEEE Conference on Evolutionary Computation Honolulu Hawaii 1910–1916 (2002).
28. Abdulhai, B., Pringle, R., Karakoulas, G.: Reinforcement Learning for True Adaptive Traffic Signal Control. Transportation Engineering vol.129 May (2003).
29. Findler, N.: Harmonization for Omnidirectional Progression in Urban Traffic Control. Computer-Aided Civil and Infrastructure Engineering vol 14 Honolulu Hawaii 369–377 (1999).

30. Pendrith, M.: Distributed Reinforcement Learning for a Traffic Engineering Application. Proceedings of the Fourth Internation Conference on Autonomous Agents Barcelona Spain (2000).
31. Dublin Transportation Office: DTO Strategy Update - Full Report - Platform For Change. Available on: `http://www.dto.ie/strategy.htm` (2001).
32. Guestrin, C., Lagoudakis, M., Parr, R.: Coordinated reinforcement learning. Proceedings of The Nineteenth International Conference on Machine Learning 227–234 (2002).
33. Schneider, J., Wong, W., Moore, A. and Riedmiller, M.: Distributed value functions. Proceedings of the Sixteenth International Conference on Machine Learning 371–378 Morgan Kaufmann Publishers 1999.
34. Stone, P.: TPOT-RL applied to network routing. Proceedings of the Seventeenth International Conference on Machine Learning (2000).
35. Mariano, C, Morales, E.: A new distributed reinforcement learning algorithm for multiple objective optimization problems. Advances in Artificial Intelligence, International Joint Conference 7th Ibero-American Conference on AI 15th Brazilian Symposium on AI (2000).
36. Yagar, S. Dion, F.: Distributed Approach to Real-Time Control of Complex Signalized Networks. Transportation Research Record Vol.1 1–8 (1996).
37. Hunt, P. Robertson, R. Winton, R. Bretherton, R.: SCOOT- A Traffic Responsive Method of Coordinating Signals. Road Research Laboratory, TRRL Report 1014 (1981).
38. Sims, A.: The Sydney Coordinated Adaptive Traffic System. Proceedings of the ASCE Engineering Foundations Conference on Research Priorities in Computer Control of Urban Traffic Systems (1979).
39. Lo, H., Chow, A.: Control Strategies for Oversaturated Traffic. Transportation Engineering vol.130 July (2004).