Technical Report:	***CSM-394***

PhD in Open Constraint Satisfaction

Technical Report 3:

The Scenario One Strategies

By

Timothy Gosling

Contents

1.1 Introduction

At present various electronic market places, auctions and negotiation systems exist, in the near future full electronic supply chains will be possible and indeed desirable to improve efficiency [4].
This situation, however, presents a problem. While humans are good at negotiation and situation analysis there are less able to handle large volumes of information and numbers of transactions. What is need is a computer-based strategy for handling these situations. The strategy does not need to be the perfect negotiation, although it must be competent, but it must be able to deal with more negotiations more rapidly than a human operator could. A core objective of this work is to develop strategies that are able to make a profit in a situation where customers are continually requesting bundles of products and may need to be negotiated with, suppliers must be negotiated with and there is a limit to both the communication capacity available and the amount of information about the market place.

The simple supply chain model (SSCM) was developed to allow the description of simple market scenarios with the aim of developing strategies to tackle such scenarios [5]. The SSCM however, while simple, is not trivial to tackle. As a result various scenarios have been developed to further restrict the use of the SSCM and so provide an incremental approach to tackling the problem [6]. This document provides details of a strategy to tackle the first of the scenarios described, Scenario One.

The remainder of this document is broken down into four sub sections. The first provides a brief overview of Scenario One. The second provides the strategy used by Customers under this scenario. The third provides details of the Supplier strategy. The final section provides details of the more complex Middleman scenario.

1.2 Strategies

1.2.1 The Scenario

The scenario being dealt with at this stage operates within the model originally proposed [5] however; several constraints have been placed upon the model to make the initial problem considerably more tractable.

The first of these constraints is that there is only one supplier for each type of product in the model and that each should only make use of fairly simple strategies in their attempt to sell goods to middlemen. In addition it is assumed that middlemen know what product each of the suppliers sell[1]

The second constraint is that each customer in the system only knows about one middleman – this forces them to use this one agent and prevents them from needing to make any decision about which agent to use, or prefer. Again each customer is to use an extremely simple strategy in attempting to obtain packages of goods.

1.2.2 Customers Strategy

Customers in this instance are considered very naïve, given their preferences they will generate preferred travel days and pass this information on to the middleman along with the entertainment they would like and the maximum price they are willing to pay. The expiry time for this message will be set to the day before the latest the customer can make arrangements by, this will be whatever the outbound flight day is minus the customers *complete* parameter. As middlemen initially don't know about customers it is the customers responsibility to contact the middleman with this information – at this time it is suggested customers will do this as soon as possible thus providing middlemen with as much information as possible early on – this approach should be changed in future forcing middlemen to deal with new customer requests on top of existing ones.

When a customer receives an offer from the middleman it will accept the offer providing the flight dates are within its required range and duration, that accommodation is available, that the entertainment it required will be available and that the price suggested is no higher than the customers budget. Assuming the offer is made within the response time laid down by the customer the offer will be immediately accepted if it fulfils these criteria. If it fails to fulfil the criteria the original offer the customer made will be made again as a counter offer if there is time left to deal, otherwise a reject message will be sent.

The customer mechanism described here is very simple and does not include an ability to bargain over the price of goods or indeed any other aspect of the travel package and as such is subject to change in future scenarios.

1.2.3 Supplier Strategy

The supplier strategy is again quite simple. Suppliers wait to receive request from Middlemen. When a request is received it is checked against the available stocks of the Supplier. If there is no stock available to fulfil any part of the request a reject message is sent. If the request can be completed in full and the offered price is acceptable, the requested goods will first be reserved before an accept message is generated. If the request cannot be completed in full, or the offered price is unacceptable then a counter offer is generated. The counter offer will full fill the request as far as possible and have the minimum counter offer price the supplier is willing to accept at that time.

[1] The model does not explicitly provide a mechanism for the discovery of what products each supplier sells however, each middleman could discover it using the existing communications system. Each middleman would start off by offering very low prices for each type of product to each supplier. Since the suppliers will attempt to negotiate for a better price on the products that they sell their product range is revealed. The middleman need only reject the suppliers counter-offer to continue with operation but with the knowledge of product supply.

Once a counter-offer is generated the products referred to in the counter offer will be reserved and essentially removed from availability. If an accept message is generated by either the Supplier in response to a Middleman offer or counter offer or by a Middleman in response to a Supplier counter offer the reserved goods will be permanently removed from availability. If either the Supplier or Middleman generates a reject message in relation to the negotiation the reserved goods will be returned to availability for use in future negotiations.

The acceptable price of an offer or counter offer is determined by a negotiation strategy. This strategy determines an acceptable per unit price given the length of time available for negotiations and the length of time so far expended. Since this strategy generates a unit price, for multiple items the value supplied would have to be multiplied accordingly. Various different mechanisms could be used to generate this per unit price however the one suggested is described in Determining Successful Negotiation Strategies: An Evolutionary Approach [2]. In this document a number of negotiation 'tactics' are described. It is suggested that suppliers follow the Boulware/Conceder strategy that is detailed – by tuning the control parameter it would thus be possible to make Suppliers more or less easy to deal with from the Middleman's perspective.

1.2.4 Middleman Strategy

1.2.4.1 Introduction

The strategy proposed here for use by the middleman is considerably different to that original proposed for tackling the un-revised model. One of the key elements of this change is in the use of customer groups. Customers are grouped by some similarity measure (see later) and as such negotiations for their required goods can be amalgamated. Another key element is the more effective handling of messages both too and from the agents. The strategy proposed here deals with all the negotiations simultaneously with the highest priority (again see later) new or reply messages being sent as soon as possible. The combination of these two elements allows agents using this strategy to make better use of available bandwidth and time in completing their task with respect to the previous strategy proposed.

The basic mechanism the agents employ can be described thus:

```
   REPEAT
1    Process Incoming Messages
2    Process Groups
3    IF there is currently available bandwidth
4      Send highest priority message from groups
   UNTIL stop
```

This mechanism assumes that additional bandwidth is added intermittently to its allocation in order that it might continue negotiations. The negotiation processes themselves are handled under the Process Groups part of the mechanism. The advantage with this general structure is that new messages are continually processed and the groups updated even if no outbound communication bandwidth currently remains.

Much of the rest of the discussion about agent strategy will now focus on describing in greater detail the elements shown above however it is worth mentioning now some of the information elements that are considered present within each agent. Note that additional pseudo code reinforces the discussion within the rest of the document – much of this code refers to the information discusses in the following section. Additional it is worth noting that the code makes use of Java style reference to sub-components of objects in some cases.

1.2.4.2 Agent Information

The agent should maintain various information for it's own use and for the use in later analysis of the agent's performance.

First and foremost the agent maintains information about each negotiation it is involved with. This information consists of the complete bidding history as well as the relevant message id numbers for

identification of incoming related messages and which agent the negotiation is with. Information about negotiation that have been completed or timed out is retained in a reduced form. This form includes which agent the negotiation was with, what the upper and lower offers were from both sides, the number of units under consideration and the final out-come of the negotiation (accepted, rejected or timed-out by either party).

$$Negotiations = (CNs, SNs)$$

$$CNs = \{CN, CN, \mathrm{K}\}$$
$$CN = (CNID, CustomerID, WaitingFor\,\mathrm{Re}\,ply, CNBiddingHistory, \exp ires)$$
$$CNBiddingHistory = \{CNOffer, CNOffer, \mathrm{K}\}$$
$$CNOffer = (UsThem, time, Accept\,\mathrm{Re}\,jectOffer, CRID, value)$$

$$CRs = \{CR, CR, \mathrm{K}\}$$
$$CR = (CRID, PackageC, \exp ires, flagged\_non\_profit, flagged\_ignore)$$

$$SNs = \{SN, SN, \mathrm{K}\}$$
$$SN = (SNID, SupplierID, WaitingFor\,\mathrm{Re}\,ply, SNBiddingHistory, \exp ires, funds, needfunds)$$
$$SNBiddingHistory = \{SNOffer, SNOffer, \mathrm{K}\}$$
$$SNOffer = (UsThem, time, Accept\,\mathrm{Re}\,jectOffer, DateAmount, value)$$

$$OldNegotiations = (OCNs, OSNs)$$

$$OCNs = \{OCN, OCN, \mathrm{K}\}$$
$$OCN = (CNID, CustomerID, UCO, LCO, UAO, LAO, outcome, PackageC, value)$$

$$OSNs = \{OSN, OSN, \mathrm{K}\}$$
$$OSN = (SNID, SupplierID, USO, LSO, UAO, LAO, outcome, units, value)$$

Identifier	***Meaning***
CNID	ID number for the customer negotiation
CustomerID	Customer identifier
WaitingForReply	Boolean – true if waiting for a customer response, false otherwise
expires	An expiry time
UsThem	Boolean – true if offer from customer/supplier, false otherwise
AcceptRejectOffer	State variable – accept equates to an accept message, reject to a reject message and offer to a counter offer specified by value
value	The value of an offer
CRID	Customer Requirement Identifier
PackageC	As specified in SSCM document Customer to Middleman communications
flagged_non_profit	The requirement is flagged as being unprofitable during negotiations
flagged_ignore	The requirement is flagged to ignore the non-profit warning flag
SNID	Supplier Negotiation Identifier
SupplierID	Identifier of the supplier
DateAmount	DateAmount as specified in the SSCM document Supplier to Middleman

	communications
funds	Current funds available for negotiation
needfunds	Boolean – true indicates flagging for requiring extra funds for negotiation
UCO	Upper customer offer – the highest package price the customer suggested
LCO	Lower customer offer – the lowest package price the customer suggested
UAO	Upper agent offer – the highest package/unit price the agent suggested
LAO	Lower agent offer – the lowest package/unit price the agent suggested
USO	Upper supplier offer – the highest unit price the supplier suggested
LSO	Lower supplier offer – the lowest unit price the supplier suggested
outcome	Accepted, Rejected or settled at the value specified
units	Number of units involved in supplier negotiation (details condensed from DateAmount information.

A list of known supplier and customer agents is maintained each maintains links to its related active and completed negotiations.

$$KnownAgents = (KnownCustomers, KnownSuppliers)$$
$$KnownCustomers = \{KnownCustomer, KnownCustomer, \mathrm{K}\}$$
$$KnownCustomer = (CustomerID)$$
$$KnownSuppliers = \{KnownSupplier, KnownSupplier, \mathrm{K}\}$$
$$KnownSupplier = (SupplierID, p)$$

Identifier	***Meaning***
p	Product selected from the set of products defined in the SSCM document that the supplier supplies

A list of products with best, worst and median prices per unit is maintained. All known products initially have these values flagged as unknowns.

$$\mathrm{Pr}\,oductInformation = \{PI, PI, \mathrm{K}\}$$
$$PI = (p, known, highest, lowest, median)$$

Identifier	***Meaning***
known	If price information is available about the product
highest	The highest price paid per unit for this product
lowest	The lowest price paid per unit for this product
median	The median price paid per unit for this product

An incoming message list is also maintained for messages received at any time.

Details of the group entity are supplied below.

1.2.4.3 Processing incoming messages

The process incoming messages part of the agents' negotiations handling mechanism essentially acts to clear all received incoming messages and redirect them to the relevant negotiations. The process is further responsible for the creation of new groups and adding customers to them.

Incoming messages may include a reply-to identifier that specifies which negotiation it is connected with. A message including a reply-to identifier is valid if that reply-to identifier matches the identifier of the last message sent to sending supplier/customer and that negotiation has not already ended either through acceptance/rejection or time-out. If a replying message is invalid it is simply discarded.

Messages that do not include a reply to identifier essentially mark the beginning of a new negotiation thread. Messages from suppliers will at this stage be ignored although in future this is unlikely to be

the case. Messages from customers indicate a new set of requirements to be fulfilled by the middleman. In the present scenario messages of this type are all received at the beginning of the simulation. Having determined that the customer has a requirement, that requirement is evaluated and if considered feasible must then be added to a group for consideration and handling. If a suitable group isn't available one is created or the requirement is added to the failure group. A permanent failure group is available for any requirements that cannot be met.

This now leaves the question of customer requirement evaluation and groups to address. A detailed discussion of groups is left to the following section however; requirement evaluation is dealt with here.

Customer requirement evaluation is a relatively simple process. For each product the customer requires the median product cost is subtracted from the customers budget. Any product that has no available median is ignored and the median of median values is subtracted. If this guess value is also unavailable then a pre-set amount is subtracted. Any requirements' with a remaining budget below a set point above zero are immediately added to the failure group. The remaining requirements are added to another group for handling.

At this stage group allocation is very simple as all customer requirement information should be available virtually immediately to the middlemen. Allocation is based upon the time to completion of the requirement in question and is dealt with in more detail below.

The following pseudo code is intended to clarify the above:

```
    Process Incoming Messages
      DO
1       msg = Get next message from the queue
2       IF msg from known Supplier
3         IF msg in response to active negotiation thread
4           Read msg and adjust the negotiation thread accordingly
          END IF
        END IF
3       IF msg from a Customer
6         IF msg in response to active negotiation thread
7           Read msg and adjust the negotiation thread accordingly
          ELSE
8           Read msg to determine contents
9           IF msg is new requirement
10            req = Generate New Requirement (msg)
11            Requirement Evaluation and Placement (req)
            END IF
          END IF
12        Discard msg
13    WHILE There are messages left to process
    END OF Process Incoming Messages
```

It is now necessary to provide information on the Requirement Evaluation and Placement element of the above (11). While other elements could also be defined their operation is straightforward for example determining the relevance of a message to a thread is simply a matter of looking at the related message Ids of the thread and as such is a lookup, updating threads accordingly amounts to amending the message information to the thread so the current negotiation state is known.

```
    Requirement Evaluation and Placement
1     val = Evaluate Requirement (req)
2     IF val >= minimum_profitability
3       Add To Group (req)
      ELSE
4       Add To Failure Group (req)
    END OF Requirement Evaluation and Placement
```

The evaluation process is as follows:

```
    Evaluate Requirement (req)
1     current_profit = req.customer_budget
2     FOR all products in req
3       IF median exists for product
4         current_profit = current_profit – product.median
        ELSE
5         IF global product price median exists
6           current_profit = current_profit – global_median
          ELSE
7           Current_profit = current_profit – default_cost
          END IF
        END IF
      END FOR
      Return current_profit
    END OF Evaluate Requirement
```

The 'Add To Group' sub-routine still needs to be defined, this is done in the following section.

1.2.4.4 Groups Processing

Groups provide for the collection of customer requirements that can be handled together for the purpose of negotiating with suppliers. A middleman may support several groups simultaneously along with an additional failure (or unsatisfiable) group used exclusively for rejecting customer offers.

The groups consist of the customer requirements (linked to their related negotiations with the customer), the negotiations being undertaken/to be undertaken and a customer priority (calculated in the same manner as above – i.e. largest profit is highest priority). The first of these attaches the needs of a customer to the group and the ability to reply to that customer. The second of these attaches the communication with the suppliers with regards to these needs to the group. The final element specifies which customers' needs are most important if they can be satisfied. This final element allows conflicts between customers within the group to be resolved if (for instance) there is insufficient amounts of one product for all customers to be satisfied. In addition groups maintain a simple state, active and inactive.

Allocation of customer requirements to groups can now be dealt with.

Groups at this time are divided along time lines. Each group (except the failure group) is responsible for handling requirements that need to be dealt with within a particular time frame. The time frames may over lap in which case allocation is based on least membership. A requirement should be a member of a group if its completion time (or rather assumed completion time based on negotiation time-out with the customer) falls within the group's specified range of handling times, the group is inactive and has equal or less members than all other suitable groups. The handling times are described as a start time and length (all groups are currently fixed length in time). If no group is available and the requirements start time is valid (i.e. exists after group activation time from the current time) a new group will be created with a start time of the requirements completion time). This mechanism should ensure that all requirements that could be dealt with are. The 'Add To Group' sub-routine mentioned above can now be defined:

```
    Add To Group (req)
1     IF req.completion_time > now + activate_time
2       Find Group (req)
      ELSE
3       Add To Failure Group (req)
      END IF
    END OF Add To Group
```

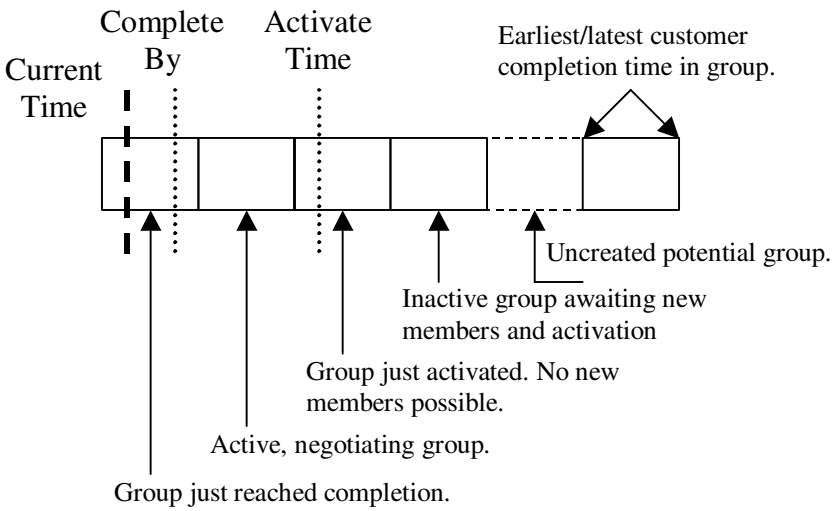
The 'Find Group' routine must now be elaborated on:

```
    Find Group (req)
```

```
1     number = 1000 (infinity)
2     grp = NULL
3     FOR all groups
4       IF grp.active = false AND
5          grp.starttime <= req.completion_time AND
6          grp.endtime >= req.completion_time AND
7          req.completion_time < number
8          Add Requirement To Group (grp, req)
9          number = req.completion_time
        ELSE
10         grp = Create Group (req.completion_time)
11         Add Requirement To Group (grp, req)
        END IF
      END FOR
    END OF Find Group
```

The sub-routines 'Add Requirement To Group' and 'Create Group' are relatively self-explanatory and elaborated on in the main text.

The group activation time has already been mentioned and will now be explained. Newly created groups are deemed inactive until a certain set time before the current time. Groups that haven't reached this time are dormant and allow the agent to concentrate or more immediate concerns. The time between the group activation time and the complete by time is intended to allow a group that amount of time to perform negotiations before any of its customer requirements hit their deadlines. New customer requirements can only be added to inactive groups to prevent interference with the negotiation processes already underway. If an inactive group is unavailable for adding a requirement to then either a new group must be formed or the requirements added to the failure group. Any new requirement with a completion time falling within the activation period will automatically be added to the failure group. An (idealised – groups may overlap) illustration of group membership and activation is shown below:



The group-processing task itself is responsible for all the negotiations being undertaken and thus all the communication between a middleman and other agents. On each iteration each group first re-evaluates it's customers' priorities using the mechanism described above (largest profit equals highest priority). If the group is active then each of the negotiations with suppliers has its priority re-evaluated and the highest priority negotiation generates a response message that is added to the current outgoing message list and should be sent if possible.

The group-processing task is comparatively simple, at least while in the inactive state. When a group is initially created and requirements added it is inactive. Customer priorities are updated intermittently and any customers that fall below the set profit margin are removed and added to the failure group.

The basic procedures are thus as follows:

```
    Process Groups
      Clear Current Outgoing Message List
1     FOR all groups (grp)
2       IF grp.active = false
3         IF grp.starttime >= now+ activate_time
4           Make Group Active (grp)
5           msg = Active Group Update (grp)
6           Add Message To Current Outgoing Set (msg)
          ELSE
7           Inactive Group Update (grp)
          END IF
        ELSE
8         Active Group Update (grp)
9         msg = Active Group Update (grp)
10        Add Message To Current Outgoing Set (msg)
        END IF
      END FOR
    END OF Process Groups
```

```
    Inactive Group Update (grp)
1     FOR all requirements (req)
2       val = Evaluate Requirement (req)
3       IF val < minimum_profitability
4         Remove Requirement From Group (grp ,req)
5         Add To Failure Group (req)
        END IF
      END FOR
    END OF Inactive Group Update
```

Once a group becomes active its membership is locked and requirements can only be removed not added. When initially made active a group calls upon the inactive procedure to remove bad requirements before generating group requirements from those remaining. The required negotiation threads are then created based upon the group requirements and an initial allocation of funds is provided.

```
    Make Group Active (grp)
1     grp.active = true
2     Inactive Group Update (grp)
2     grp.grpreq = Generate Group Requirement
3     FOR all products in the group requirement (prd)
4       Generate Negotiation Thread (prd)
      END FOR
5     funds = Set available fund to total of all requirements
6     Allocate Funds To Threads (funds)
    END OF Make Group Active
```

This leaves the Allocate Funds To Threads and Active Group Update procedure to define. The fund allocation mechanism is quite simple and reveals something of the manner in which negotiation threads behave within the system. Negotiations may be flagged as requiring further funds – in the initial case all negotiations will do this. Funds are allocated from a central ‘pot’ in direct proportion of their product value medians (or other indicators as discussed for evaluation) multiplied by the number of units required. On each iteration negotiations that manage to obtain a price lower than expected, return the excess to the ‘pot’ for reallocation if necessary. Any negotiation that appears to require additional funds may request funds from the pot as needed – although this is the responsibility of the individual negotiation strategy.

```
    Allocate Funds To Threads (funds)
1     FOR all flagged negotiation threads (nt)
```

```
2       p = Generate Relative Proportion Of Funds Required (nt)
3       Allocate funds*p Funds To Thread (nt)
2     END FOR
    END OF Allocate Funds To Threads
```

The Active Group Update mechanism is more complex. Once a group becomes active its membership is locked to new members although existing members may still be removed to the failure group. On each iteration an evaluation is carried out to determine if any requirements have become unprofitable. If so a waiting mechanism is employed and decision made about whether to retain the requirement.

This mechanism operates as follows. A short delay is tolerated if possible to allow counter offers relating to the customer requirement under question from suppliers to arrive. If accept messages are received the requirement will not be dropped if the cost of the accepted goods is greater than then likely un-profitability of the requirement. If some replies are not forth coming within the time the requirement will be retained for safety's sake. Otherwise the customer requirement will be added to the failure group and the remaining negotiations have its related required items removed (i.e. the group requirement will be updated). The precise delay strategy used is a fixed proportion of the total amount of time left to negotiate being waited for or until a fixed cut-off point close to the group start time is reached. If after this time all the replies are not forthcoming or the agent is within the fixed cut-off period to begin with the requirement will be retained for safety as mentioned.

Following this safety procedure the negotiation priorities are calculated and the highest priority negotiation thread is able to generate a message to be hopefully be sent. Exactly what messages are sent are related to the chosen negotiation strategy however initial offers must be made over several iterations to determine availability of products. If (in subsequent iterations) it is discovered a product required by some customer is unavailable or there is insufficient for a lower priority customer that customer's requirement is sent to the failure group, the groups requirements recalculated and an attempt made to essentially prevent purchase of other goods linked to that customer's requirement. If, as mentioned above, during the course of negotiations a customer's requirement becomes negatively profitable a decision must then also be taken about removing the requirement from the group.

The remaining negotiation process for products is itself controlled by a strategy selected from one of several tested in previous work [1] or [2]. Any of these strategies could be applied however (as described above) initial offers made by the middleman will always be zero cost offers in order to simply obtain product availability information from the supplier. The strategies under consideration essentially take over at the point a non-reject reply is received from the supplier. The strategies are in no way currently to consider issues beyond the unit cost of products although in future this is likely a useful thing to do. At this time it is considered that the entire set of tactics suggested in [2] should be used by the middlemen in order to give them a greater degree of flexibility.

Once negotiations have reached a successful conclusion customers will be informed with accept messages and the group can be discarded. Any customer requirements that could not be fulfilled will be reported with reject messages from the failure group. The failure group attempts to inform customers of the middleman's inability to fulfil their requirements but ultimately concedes communication time to current negotiations if necessary. This leads into the prioritisation mechanism.

```
    Active Group Update (grp)
1     RemoveCustomersDueToInsufficientProducts
2     IF NoNonIgnoreFlaggedCustomerRequirements
3       wait_time = 0

4     IF wait_time > 0                                          Primary
5       IF NegotiationsInSafeState                              negative
6         UpdateGroupRequirements                               profitability
7         UpdateNegotiationRequirements                         handling
8         RemoveFlaggedNonIgnoredCustomerRequirements           component
9         wait_time = 0
        ELSE
10        wait_time--
11        return (NULL)
        END IF
      ELSE
12      SetIgnoreOnFlaggedCustomerRequirements
      END IF

13    IF CheckAndSetRequirementValidity                         Primary
14      ResetMessageSendList                                    negotiation
15      GenerateMessagesFromNegotiations                        component
16      msg = SelectHighestPriorityMessage
17      return (msg)
      END IF
18      return (NULL)
    END OF Active Group Update
```

An element of some importance is the prioritisation of outbound communications from the agent. The following mechanism has been designed to allow more important messages to be communicated quickly while still allowing less important messages to be sent.

Prioritisation of groups and negotiations is comparatively simple. Within groups each negotiation item is associated with a value directly proportionate to its time-out time for reply. These values (for simplicity) are ranged between 1 and 0 linearly with 1 being assigned to negotiations that require a response today and 0 being assigned to messages that require a response group activation time from today. Negotiations that have not started are supplied with the value 1 for their initial message. The highest priority message is thus selected for the group. If a conflict occurs a series of resolution strategies may be applied. Firstly the type of message is reviewed, an accept message has higher priority than a counter-offer message which in turn has higher priority than a new offer message which is higher priority than a reject message. If a tie-break still exists the type of product under consideration is considered. Flights have higher priority than hotels that in turn have higher priority than entertainments. Responses to customers are lowest priority, however all groups (except the failure group) will only be relaying accept messages to customers and nothing else. If a conflict still remains one message is chosen at random. Having determined the highest priority messages per group the message to be sent is selected by multiplying the value initially obtained by a value between 1 and 0 determined linearly from each groups' completion time. The completion time is essentially the earliest possible time a customer requirement within the group must be satisfied by. If the requirement must be satisfied by today a value of 1 is used ranging to 0 on a day group activation days from now. If this fails to select a message, groups in their customer response phase have priority followed by those closest to completion and lastly the failure group.

With the various mechanisms described in place it should be possible for a middleman agent to perform at least reasonably under the current model and scenario and it is hoped that the strategy will act as a useful basis for future work.

1.3 References

[1] Simple Constrained Bargaining Game
Edward Tsang, Tim Gosling
[2] Determining Successful Negotiation Strategies: An Evolutionary Approach
Noyda Matos, Carles Sierra, Nick Jennings
[3] Population Based Incremental Learning – A Method for Integrating Genetic Search Based Function Optimisation and Competitive Learning
Shumeet Baluja
[4] On Agent-Mediated Electronic Commerce
Minghua He, et al, 2003
[5] Technical Report 1: The Simple Supply Chain Model (SSCM)
Tim Gosling and Edward Tsang
[6] Technical Report 2: SSCM Scenarios
Tim Gosling