ISSN 2047-3338

# Using Swarm Intelligence to Optimize Caching Techniques for Ad Hoc Network

Appavoo Paramasiven
Computer Science and Engineering Department, University *of Mauritius,*
*Réduit, Mauritius*
p.appavoo@uom.ac.mu

***Abstract*— Caching has been used to improve the performance of network-dependent computer system. A number of cache replacement schemes along with invalidation reports schemes have been devised for MANETs. Apart from improving the response time of request in MANET, caching also increases the network lifetime. In this paper, it is shown how the ant colony optimization is applied to caching system so as to further improve the performance and lifetime of the ad hoc network.**

***Index Terms*— Ad Hoc, Network, Cache, Swarm Intelligence, Ant Colony Optimization and Geocasting**

## I. INTRODUCTION

AD hoc networks have become ubiquitous while giving rise to new set of application models that are coined different names. The ever-challenging issues of such networks are limited-power, limited-bandwidth and high latency. The latter is noticeable in interactive applications especially when the communicating parties are quite far apart in this multi-hop network. One method brought forward to mitigate these characteristics, which are the norms rather than the exceptions, of ad hoc network is caching.

## II. RELATED WORK

A number of cache replacement schemes (CRS) and cache consistency management methods were developed, each optimized for a certain context of use. The replacement schemes in [1], [2] and [3] can broadly be classified based on the way the existing objects are swapped: (1) least recently used, (2) least recently used with the minimum size, (3) largest goes out first, and (4) custom-based devised formulae. The most widely used technique in managing cache consistency is through the dissemination of invalidation reports as in [4], [5], [6], [7] and [8] which contain updates on the state of the objects at the server. The scheme of invalidation reports sent by the sources of the respective objects can be classified as stateless-based (synchronous and asynchronous) and stateful-based as in [9]. In the synchronous stateless method, the server sends invalidation reports (IRs) periodically while in the asynchronous mode, the server sends IRs only when the objects are updated. The main drawbacks with these two methods is that the first one tends to use up more of the scarce resources like power to transmit at a higher frequency to maintain cache consistency while the second allow disconnected nodes to only reconnect with a flushed cache. The stateful approach allows the server to use callback methods to update cached copies at the client nodes. The problem of disconnected nodes crop up once again. It is also useful to point out that frequent disconnection is a characteristic of ad hoc network. Therefore, frequent flushing of cache is not uncommon.

SAMPCAN [10] revolutionized the caching replacement method by downsampling a cached object to create space for a newly accessed object. The downsampled version despite being of a lower quality still meets other requests. SAMPCAN allows for two types of objects, namely (1) XML, for textual data or information and, (2) images.

## III. SWARM INTELLIGENCE: ANT COLONY OPTIMIZATION (ACO)

Swarm intelligence (SI) is a class of algorithms emerged from the nature-inspired techniques. The idea that came out from SI is the ability to coordinate without communication [11], whereby while agents were individually performing local tasks, they were also contributing to a global function. One such discovered methods that have been applied to routing algorithms in [12] and [13] is the Ant colony Optimization (ACO), which is based on the way ants find the optimal way from their nest to a food source. While in nature, ants lay down pheromone to indicate the directions, in MANET, nodes mainly record the quality of the route to the destination, using parameters like number of hops, route stability, etc.

## IV. PROPOSED CACHING SYSTEM USING ACO

### *A. Overview*

The proposed cached system uses the concept of using pheromone trails to retrieve up-to-date data objects from nearby caching nodes instead of having the same request being fulfilled by a far-away server or data source. This reduces latency in having access to the desired data and is at the same time energy-saving. The latter is especially critical to the server and its neighboring nodes that are usually constantly participating in the forwarding of the requests and replies.

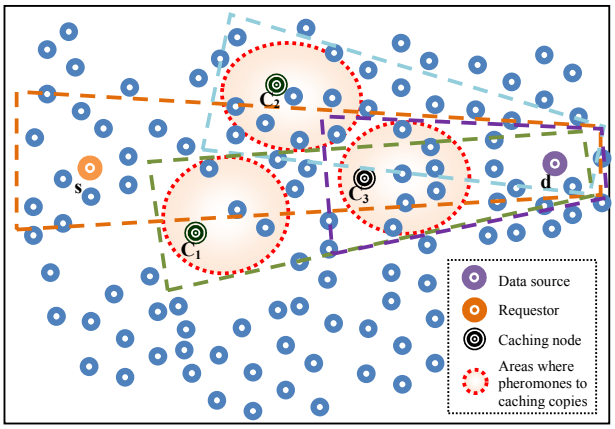


Fig. 1. Caching with ACO

Certain nodes that participated in the forwarding of the object keep certain information, the pheromone. Request for similar object follows the pheromone trails to reach closer up-to-date cached copy. In the event that the caching node is requesting other objects, the pheromone trails evaporate as neighboring nodes update their lists of caching objects available. While the caching node uses $CRS_a$ to update objects in their respective cache stores, certain neighboring nodes uses $CRS_a$ to update their list of neighbors' cached objects.

This allows upcoming requests for similar objects to be directed to a closer reliable source. Pheromone trails are used to reach consistent cached copies of what is required. The former also evaporates (1) as the cache storage is being updated and/or (2) as the nodes experience with weak connectivity as ongoing activities of caching are missed.

Fig. 1 gives an overview of how the proposed cached system operates. $C_1$, $C_2$ and $C_3$ requested for $object_x$ at some point in time, if SAMPCAN [10] is used for example, they cached a copy of the object for their own future access. Nodes (within a certain range to the requestor) that forwarded the reply and those who heard the reply kept the following information: $object_x$, requestor, GPS coordinates of requestor, replier, GPS coordinates of the replier and timestamp, which is basically the pheromone, in a close cached-object list (CCOL).

If requestor is requesting different objects more frequently, existing cached objects are downsampled or erased completely. At the same time, neighboring nodes kept the information for a time duration after which it is removed from the CCOL, that is the pheromone evaporates.

The isosceles trapezoid shape of the forwarding zone (FZ), to be supported by the underlying routing protocol, ensures that neighboring nodes that are well aware of the activities of the requestor so that they can update their CCOLs. The SENCAST [14] routing protocol can be slightly modified, as per the algorithm defined for the FZ computation in the next section, so that nodes have the ability of determining whether they are forwarders.

Given that after some times, S issues an object discovery for $object_x$ ($ODO_x$), it receives up 4 replies, i.e. from $C_1$, $C_2$, $C_3$ and D, containing the following information: $object_x$, replier, GPS coordinates of the replier, timestamp and $object_x$_status. Note that the object_status increments with each update performed on that particular object at the source. In certain cases, S has the ability to determine if cached copies at $C_1$, $C_2$ or $C_3$ are up-to-date by comparing their respective object_statuses with that received from D. S chooses to retrieve the closest and up-to-date copy of the object by sending a request for $object_x$ ($RFO_x$) by providing the GPS coordinates of the closest replier and the replier itself (the IP address).

Cache consistency is ensured by the source. D sends invalidation reports (IRs) in an asynchronous stateless mode so that caching node may update objects' copies or discard same as required. IRs are sent to the location of requestors of the updated objects. This can be achieved using any geocasting protocol.

### *B. Algorithm*

#### *1) Client/Caching node (requestor)*

Client nodes requesting for an object keep a copy of the requested object till either IRs discards the object or the object is swapped or downsampled to create space for newly requested objects.

Thread 1:

Send $ODO_x$:
While not $ODO_x$ timeout:
    Wait
If no reply received at all:
        *(to be taken care by the routing protocol…)*
    Expand FZ
    Increase ODO timeout
        Send $ODO_x$
Else:
    If no reply is received from the data source and cache consistency required is high:
            *(to be taken care by the routing protocol…)*
        Expand FZ
        Increase $ODO_x$ timeout
        Send $ODO_x$
    Else:
        Discard caching nodes with outdated copies of $object_x$
        Calculate closest caching node
        Send $RFO_x$ to the closest caching node

Thread 2:
Listen for IRs
Update cached stores

#### *2) Intermediate node (forwarders)*

If packet received is a request (RFO/ODO):
        Check CCOL
    If a copy of $object_x$ is available nearby:
        Encapsulate packet
            Forward same to the caching node
    Else if node is within FZ (Method isForwarder() is called) :
            Request is forwarded to data source
Else if packet received is a reply for an object:
        Forward packet to the requestor
    If distance between requestor and current node < $threshold_a$:
        Check CCOL
        If requestor is in the list:

Use $CRS_a$ to update neighboring cached objects lists

Else:

Add an entry in CCOL

Else (packet is an IR):

Forward packet to its destination, i.e. the caching node

Check CCOL

If requestor is in the list:

Use $CRS_a$ to update neighboring cached objects lists

The optimum value for $threshold_a$ depends on the context of use.

*3) Source node (node maintaining the required object)*

Thread1:

Consolidate updates received

Send IRs to Requestors' of updated objects

Thread2:

Reply to RFO/ODO messages

*4) Caching node (any node caching objects)*

*Thread 1:*

If packet received is a request (RFO/ODO):

If a copy of $object_x$ exist:

Reply the requestor

*Thread 2:*

If packet received is IR:

If frequency of access to object > $threshold_b$:

$time_x$=[ sizeOfObject (bytes) / averageTransferRateBetweenHops (bytes/sec) ] * NoOfHopsToSource

wait 2 * $time_x$

send $ODO_x$

send $RFO_x$

The optimum value for $threshold_b$ depends on the context of use. Also, to a certain extent, the waiting time of 2*$time_x$ allows caching node closer to the source to retrieve the updated objects so that other caching nodes retrieve same to closer caching nodes instead of requesting same from the data source.

*5) $Node_p$ determining whether it is a forwarder or not*

The transformation from geodetic to a 2D local grid can be performed as shown in [15]. Using the location of the requestor (s) and the data source (d), *p* can derive the equation $Y = m(X) + C$. Then *p* finds out the (i) distance between itself and the perpendicular intersection, *q*, with the imaginary line joining s and d, and (ii) the FZ threshold, t, at q, given the threshold $t_1$ and $t_2$ at s and d respectively. Note that the threshold values are the two sides of the trapezoid FZ.

Fig. 2 illustrates.

isForwarder(position of $requestor_s$, location of data $source_d$, threshold $t_1$, threshold $t_2$):

$m = (s.Y - d.Y)/(s.X - d.X)$

$C = s.Y - m(s.X)$

$K = p.X + m(p.Y)$

If $(s.X - d.X) != 0$ and $m != 0$:

$q.X = (K/m - C)/(m + 1/m)$,

$q.Y = m * q.X + C$

Else If $(s.X - d.X) != 0$ and $m == 0$:

$q.X = p.X$,

$q.Y = C$

If $(s.X - d.X) == 0$:

$q.X = s.X$ or $q.X = d.X$

$q.Y = p.Y$

$t = t_2/2 + ( \sqrt{((s.X - q.X)^2 + (s.Y - q.Y)^2)} / ( \sqrt{((s.X - d.X)^2 + (s.Y - d.Y)^2)} ) * (t_1 - t_2)$

If $( \sqrt{((p.X - q.X)^2 + (p.Y - q.Y)^2)} ) < t$:

Return true, i.e. p forwards the packet

Else

Return false, i.e. p discards the packet

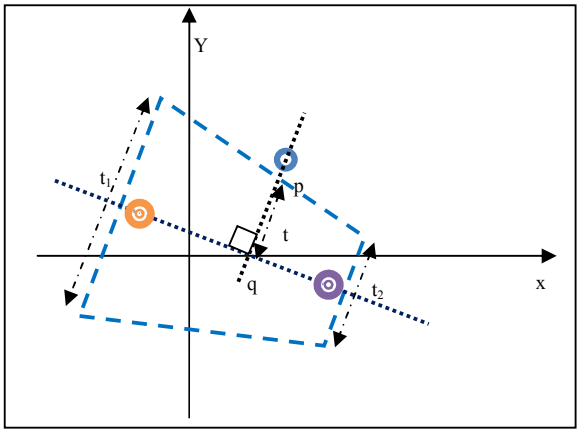


Fig. 2. FZ delimitation

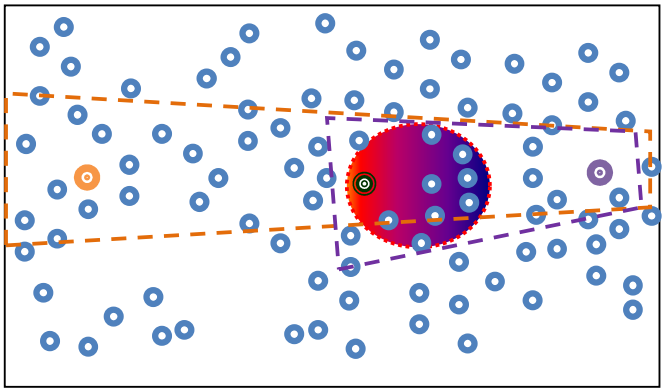

Fig. 3. Scenario of a caching node, for a requested object

## V. Discussions

### A. Caching node within the FZ of requestor

The caching node replies the requestor that a copy of the required object is available. Neighboring nodes with pheromone trails forwards the request to the same caching node. It can also be the case that several caching nodes and the source node of the object reply to an ODO message. Optimally, the requestor chooses to send an RFO to the closest up-to-date caching node. Fig. 3 depicts the scenario of a caching node, for a requested object, and neighboring nodes with pheromone trails lying in the FZ.

### B. Caching node outside FZ of requestor

#### 1). With pheromone trails inside the FZ

In some cases, caching nodes lying outside the FZ can also be closer to the requestor. The availability of such resources is made aware through pheromone trails of certain nodes covered by the FZ. Fig. 4 shows examples of successful routes of ODO messages using pheromone trails.

#### 2). Without pheromone trails inside the FZ

In the worst case scenario, neither pheromone trails to caching nodes lying outside the FZ exists nor is the existence of caching nodes inside the FZ. Then the server reply is the only respond that the requestor receives. Fig. 5 illustrates.

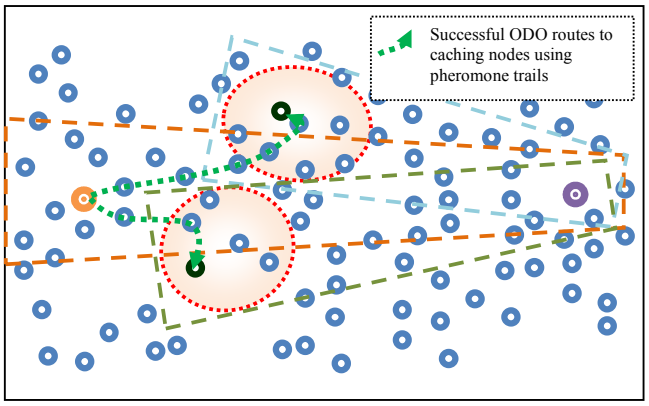


Fig. 4. With pheromone trails inside the FZ Fig 3 Caching node outside FZ of requestor: with pheromone trails

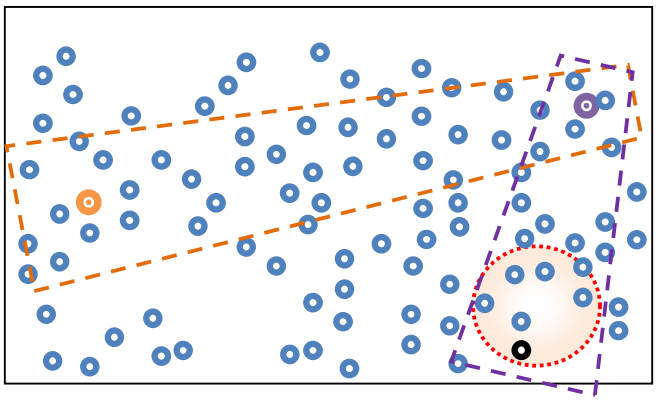

Fig. 5. Without pheromone trails inside the FZ

### C. Evaporation of pheromone trails

Ideally pheromone trails must lead to consistent cached copies of the requested object. The trapezoid form of the FZ allows most neighboring nodes to hear the concerned caching node activities. Depending on the different ODO messages, they update their respective CCOL. As such, objects that are swapped in the caching storage are reflected by neighboring nodes that collectively build up the trails to the caching node whenever required. Nodes experiencing weak connectivity flush their CCOL as this may lead to inconsistent cache objects or wasting more energy by forwarding ODO to non-existent objects.

## VI. Conclusion & Future works

The ACO can significantly increase the network lifetime by forwarding request to closer caching node than to distant data source. Using different scenarios it has been demonstrated how ACO is used to improve the overall functioning of the ad hoc network by decreasing latency. The pheromone evolves with respect to the availability of an object at a caching node. When an object is replaced in the cache store, the pheromone trails naturally evaporates following RDO messages forwarded/heard. There is no cost associated with the maintenance of the pheromone, nodes update their respective CCOLs based on the activities of its surrounding and depending on its level of connectivity to the network. Simulation results can also determine the optimum values for $threshold_a$ and $threshold_b$. The effect of mobility also has to be incorporated in the caching system as the former affects the pheromone trails.

## References


[1] Abrams M., Standridge C., Abdulla G., Williams S. and Fox E., "Caching Proxies: Limitations and Potentials," Proc. Fourth Int'l World Wide Web Conf., Boston, 1995.

[2] Aggarwal C., Wolf L. and Yu P., "Caching on the World Wide Web", IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 1, 1999

[3] Williams S., Abrams M., Standridge C. R., Abdulla G., and Fox E. A., "Removal Policies in Network Caches for World Wide Web Documents," Proc. ACM SIGCOMM, pp. 293-304, 1996.

[4] J. Jing, A. Elmagarmid, A.S. Helal, and R. Alonso, "Bit-sequences: an adaptive cache invalidation method in mobile client/server environments", Mobile Networks and Applications, vol 2, no 2, 115–127, 1997.

[5] J. Cai, K.L. Tan, "Energy efficient selective cache invalidation", Wireless Networks, vol 5, issue 6, p 489–502, 1999.

[6] G. Cao, "A scalable low-latency cache invalidation strategy for mobile environments", IEEE Transactions on Knowledge and Data Engineering, Vol 15, No. 5, p1251–1265, 2003.

[7] A. Madhukar, T. Özyer, R. Alhajj, "Dynamic cache invalidation scheme for wireless mobile environments",Wireless Networks, vol 15, no 6, p 727–740, 2009.

[8] Z.Wang, M. Kumar, S. Das, H. Shen, "Dynamic cache consistency schemes for wireless cellular networks", IEEE Transactions on Wireless Communications, vol 5, issue 2, 366–376, 2006.

[9] K. Tan, J. Cai, B. Ooi, "An evaluation of cache invalidation strategies in wireless environments", IEEE Transactions on Parallel and Distributed Systems, vol 12, no 8, 789–807, 2001.

[10] P. Appavoo, "SAMPCAN: A Novel Caching Technique for Client-Server Interaction Model In Large Ad Hoc Networks Using Resampling Methods", International Journal of Wireless & Mobile Networks, vol 3, issue 2, 165-180, 2011.

[11] S. Franklin, "Coordination without communication", URL: http://www.msci.memphis.edu/~franklin/coord.html, accessed on 19th July 2011.

[12] J. Wang, E. Osagie, P. Thulasiraman, Ruppa K. Thulasiram, "HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network", Ad Hoc Networks, vol 7, 690 – 705, 2009.

[13] M. Unes, U. Sorges, I. Bouazizi , "ARA – The Ant-Colony Based Routing Algorithm for MANETs", ICPPW '02 Proceedings of the 2002 International Conference on Parallel Processing Workshops, 2002.

[14] P. Appavoo and K. Khedo, *SENCAST: A Scalable Protocol for Unicasting and Multicasting in a Large Ad hoc Emergency Network*, International Journal of Computer Science and Network Security, Vol.8 No.2, p 154 – 165. February 2008

[15] C. G. Carlson and D. E. Clay, "The Earth Model – Calculating Field Size and Distances between Points using GPS Coordinates", Site-Specific Management Guidelines series-11, Potash & Phosphate Institute (PPI), 1999.