Stigmergy: A Wide Area Service Discovery Protocol for Active Networks

Kalaiarul Dharmalingam and Martin Collier
Research Institute for Networks and Communications Engineering (RINCE),
Dublin City University, Republic of Ireland

Abstract

The touted goal that drives the field of Active Networks lies in the concept of dynamic service provisioning. In this, when an Active Node is presented with a request for a service, it will be able to either provide the service: (i) immediately – if the requested service is available locally; or (ii) after an initial delay – caused due to discovery, deployment and instantiation of the service dynamically. The delay to dynamically discover and deploy Active Services can affect the overall quality of service perceived by end applications. Hence, efficient service discovery mechanisms are essential for proper operation. In this paper we present a service discovery protocol, referred to as Stigmergy, which supports the discovery of Active Services in the network. The key feature of the Stigmergy protocol is that each Autonomous System in the network is treated as an independent two level caching structure in which the upper level, $L1$, contains pointers to active services that are present in the lower level, $L0$. This protocol, by self-organising network nodes that are under a common administrative control into virtual cache clusters, maximises the chances of discovering the required services within minimal latencies. Furthermore, the Stigmergy protocol is completely distributed and follows a best-effort cache co-operation model. Finally, this protocol avoids the need to configure and maintain independent caching frameworks for service discovery purposes.

I. Introduction

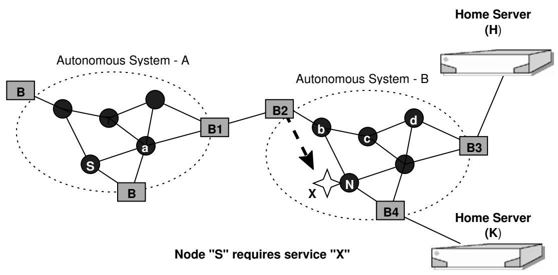
The current Internet suffers from slow network evolution, due to lengthy standardisation processes and compatibility concerns. One way to overcome the network evolution problem is to introduce programmability into network nodes, a feature already available in end user systems. Active Networks are targeted towards achieving this goal. In Active Networks, the traditional model of packet forwarding, *store-and-forward*, is replaced by *store-compute-forward*, thereby enabling packet processing support at intermediate nodes as they travel through the network. The key feature of Active Networks is the flexibility to dynamically deploy new services at network nodes in response to user demands.

Data packets in Active Networks carry the name of the Active Service which should process them at intermediate network nodes. When a Active Router is presented with a request for a service, the node will be able to either provide the service: (i) immediately – if the requested service is available locally; or (ii) after an initial delay – caused due to discovery, deployment and instantiation of the service locally. Setting up services on-the-fly involves the following costs: (i) increased memory requirements – the packets that arrive during service discovery phase will have to be buffered until the service is instantiated locally; and (ii) increased initial delay experienced by those packets that arrive when the service is not available locally. Thus, the delay to dynamically discover and deploy Active Services can affect the overall quality of service perceived by end applications. Hence, efficient protocols are required for providing service discovery support. Note a detailed analytical study that formulates the dynamics of an Active Networking system providing dynamic service provisioning can be found in our earlier work [1].

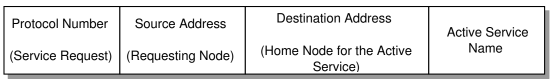
Some of the approaches that are currently followed by existing Active Networking systems for service discovery support are: (i) loading from the source [2]; or (ii) loading from a code server [3]. The "load from source" approach does not scale to a large user population. This is because each individual end application will have to download and host service codes locally before usage. This operation is performed even if the intermediate nodes host the required service, thus introducing redundant traffic within the network. Alternatively, the use of code servers to host Active Services has been proposed. In this model, code servers are arranged in a hierarchy, similar to DNS nodes in the current Internet. When a network node requires a new service, it performs a search through this hierarchy to locate and download the desired module. The major problems that arises when working with this approach are: (i) code servers will have to be configured to co-operate with neighbouring repositories in the hierarchy; (ii) the request does not always follow the shortest path route to the server hosting the required service; and (iii) those nodes that are higher in the hierarchy become points of implosion under heavy loading conditions. We believe that a decentralised solution is more appropriate, taking into account the ever expanding size of networks.

II. Globally Unique Labels for Active Services

In our model, the server function of hosting Active Services is distributed across: (i) the intermediate network nodes; and (ii) the "home nodes" located at network edges. By "home node" of an Active Service, we refer to a node that is responsible for permanently hosting the Active Service. In order to identify each Active Service uniquely in the network, we propose to follow the URI naming scheme [4]. In this, names for individual Active Services are derived by concatenating: (i) the address of the home node of that Active Service; and (ii) its service specific label name. These attributes are set by the home node and do not change for the lifetime of an Active Service. For example $< 192.168.254.1/packet - duplication\ -\ service >$ denotes a label of an Active Service. Here, "192.168.254.1" represents the address of the home node and its service specific label is "packet-duplication-service". This Uniform Resource Identifier (URI) based scheme coupled with network level routing support is used to build the Stigmergy protocol.



(a) An Example of Service Discovery in ABC Framework



Protocol Number (Service Request)	Source Address (Requesting Node)	Destination Address (Home Node for the Active Service)	Active Service Name

(b) Format of a Request Packet in Stigmergy

Fig. 1

III. Stigmergy Protocol Concept

The key feature of the Stigmergy protocol is that each Autonomous System in the network is treated as an independent two level caching structure in which the upper level, $L1$, contains pointers to Active Services that are present in the lower level, $L0$. For example in Fig. 1(a), information about various Active Services that present at Active Nodes within the Autonomous System A, are stored at level $L1$, which can subsequently be used by other nodes in the network for service discovery. However, many issues have to be addressed to realise such a system. They are:

(a) Which node(s) in an Autonomous System should be elected to function as $L1$ point(s) ?
(b) How to construct the $L1$ level representation for an Autonomous System?
(c) What would be the criteria used to decide whether the Active Router which requires a service should initiate a discovery process, (i.e. contact distributed $L1$ levels in the network) or contact the home node of the Active Service directly?
(d) What search strategy should be adopted to route service requests through various $L1$ levels within the network?

(a) Border Routing Nodes as Aggregation Points: The border router nodes of an Autonomous System are the most suitable points to act as $L1$ locations for the following reasons. A routing path for packet in the Internet comprises segments that span different Autonomous Systems. Individual Autonomous Systems contain both interior and exterior routing nodes (border nodes). The former route packets within the domain, while the latter perform inter-domain routing, i.e. between neighbouring Autonomous System (Fig. 1(a)). Thus, when a packet needs to traverse an Autonomous System, it must be routed through one of its border nodes. Thus, by storing pointers at border routing nodes of an Autonomous System, we can create an exact map of the services available within it.

(b) Self-organising Network Caches: The list of Active Services present at caches of each individual Active Node can be announced to the border router nodes of the domain[1]. The border routing nodes can record these announcements into the "$L1$ Table". The information recorded into the $L1$ Table should contain the name of the Active Service and the address of the node on which it currently resides. Thus, the model follows a self-organising technique to build the information for the $L1$ level.

(c & d) The Combined Model: The decision as to whether a node requiring a service must search the network caches or contact the home node directly is not clear cut. One approach is described below. The node which requires the service sends a request to the home node. Should the request pass through a border node which contains a reference to the service, the border node suppresses the request and deals with it itself. Otherwise the request will reach the home node for processing.

A. *Working of the Stigmergy Protocol*

The Stigmergy protocol for performing service discovery is presented below. An Active Node (node S in Fig. 1(a)) on facing a miss for an Active Service generates a request to the home node (node H) of the required service (say X). Recall that the address of the home node for an Active Service can be extracted from its name (section II). The format of the request packet is shown in Fig. 1(b). The request packet is routed along the shortest path route to the home node. In Fig. 1(a), the shortest path route from S to H is, S-a-B1-B2-b-c-d-B3-H. When the packet reaches a border router[2] (e.g. either B1 or B2), it performs the following operations.

First, it extracts the name of required Active Service from the request packet. Next, it checks whether there is a corresponding entry for the particular service in its $L1$ Table. If the result is true (i.e. a hit), the border router signals the node holding the service to send a copy of the service to the node requiring the service code (i.e. to the source address of the packet). However, when there is no entry in the $L1$ Table (i.e. a miss), the border node simply forwards the packet on the shortest routing path towards the packet's destination address, i.e. the home node of the Active Service. In the case of B1 as in Fig. 1(a), it does not contain an entry for the service. Hence, it simply forwards the packet towards H. However, when the packet reaches B2, it identifies that the required service is present at node N. Hence, B2 signals N to deliver the requested service to node S. Thus, the border node functions as a packet deflector service within the network for service discovery packets.

[1]It is assumed that each node in a domain is aware of the address list of its border routers
[2]Its assumed that Stigmergy packets are recognised by border routing nodes based on a unique protocol number, while intermediate network nodes route them as regular data packets.

B. Discussion

Minimising State at Border Routers: In the above approach, each border router of an Autonomous System stores pointers for all the services present within it. This state information at border routers can be reduced by exploiting the phenomenon of route aggregation present in the Internet [5]. For example considering the Autonomous System B in Fig. 1(a), the shortest route paths from N to H and K are through $B3$ and $B4$ respectively. In this case, routing requests for H, would always traverse $B3$. Based on this fact, $B3$ can restrict itself to caching only those entries of services which belong to H, while $B4$ can restrict itself to entries of K. Recall that names of Active Services include the address of the home node. This feature can be exploited to achieve an optimal storage model for the Stigmergy protocol.
Status of Routing Nodes: Routers may go out of service abruptly without prior announcements due to node or link failures. Hence, border routers should avoid assigning discovery requests to those nodes that have announced participation in the $L1$ cache, but are later unavailable for service. For this purpose, the $L1$ Table must also record the status of the Active Nodes that are participating. The periodic routing updates sent by Interior Gateway Protocols (e.g. RIP, OSPF protocols) can be used for this purpose.

C. Key Benefits of the Stigmergy Protocol

Large Virtual Caches: This protocol, by self-organising network nodes that are under a common administrative control into virtual cache clusters, maximises the chances of discovering the required service locally.
Zero Cache Cooperation: The Stigmergy protocol is completely distributed and follows a best-effort cache co-operation model. By this we mean that Active Nodes can join/leave a virtual cache group depending on a best-effort basis. Furthermore, this architecture avoids the need to configure and maintain independent caching framework for service discovery purposes.
Locate Unpopular Services with Minimal Delay: The search requests always follow the shortest path to the home node of the Active Service. This feature is of immense value to unpopular services which are not widely deployed in the network.

IV. Evaluating Deployment Latencies of Active Services

We use simulations to study the service deployment latencies in wide area networks, such as the Internet. The service deployment delay consists of: (i) the delay to obtain the necessary bytecode of the service; and (ii) the delay to instantiate the service locally. However, the time to download the required bytecodes constitutes the significant element of the total service deployment time. We used the Network Simulator (ns) package [6] for performing the simulations. Our analysis was performed using the core-stub network topologies generated by the GT-ITM [7] package. The topology used in our simulations is shown in Fig. 2(a). The network has an average node degree 3.6, with effective path bandwidth of 10Mbps and link delay of 30ms.

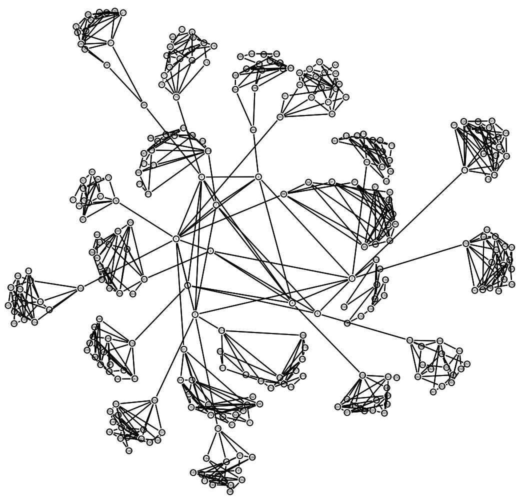
1) Stigmergy Protocol Vs. Code Server Model: In the simulation setup, three nodes are selected to represent the Active Node requesting service (client), the node containing the required service (cache) and the home node of the Active Service. The distance between the client and the home node is configured to be 14 hops. The location of the cache is selected at a midway point between the client and the home node, i.e. 7 hops in this case. The server and cache nodes are configured to host various services of sizes in the range 500Bytes to 32KBytes. TCP is used for communication purposes. In the first set of experiments, we measure the service deployment delay without caching. The client node is configured to contact the home node for the bytecode file necessary to deploy a service. Fig. 2(b) shows the results of this analysis. In the next set of experiments, the client uses the Stigmergy protocol to discover the cache node, and requests the code from it. Fig. 2(b) shows the results of this analysis. Note this delay is sum of the discovery and code fetching delays.

2) Stigmergy Protocol Vs. Non-caching Scheme: Here we analyse the performance of the Stigmergy protocol with various ratios of caching level support available in the network. For this purpose, we measured the variation of downloading delay incurred by the Stigmergy protocol as a function of cache distance from the client node; and then, using equation 1 we calculated the normalised delay of the cache-less approach, i.e. where the network nodes do not participate in distribution of Active Services

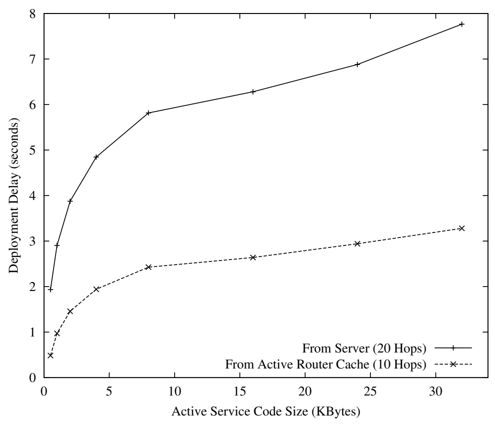
$$Normalised \quad Overhead \quad of \quad Cache-less \quad = \frac{T_{Standard} - T_{Cache}}{T_{Cache}} \tag{1}$$

where, $T_{Standard}$ denotes the time to obtain the service from the home node, while T_{cache} denotes the time to download from the cache. The simulation setup for the measurements involved a server (the home node), client and a cache node. The server was configured to be 15 hops away from the client. For every measurement we varied the cache location in relation to the client from a distance of 2 to 14 hop counts, in increments of 2. We also measured the downloading delays for various client locations and server positions. The measurements were performed for code sizes of 500Bytes and 2.5KBytes. Fig. 2(c) shows the results from this analysis. Next, using equation 1, we evaluated the overhead incurred by the cache-less approach relative to the Stigmergy protocol. The results are presented in Fig. 2(d).

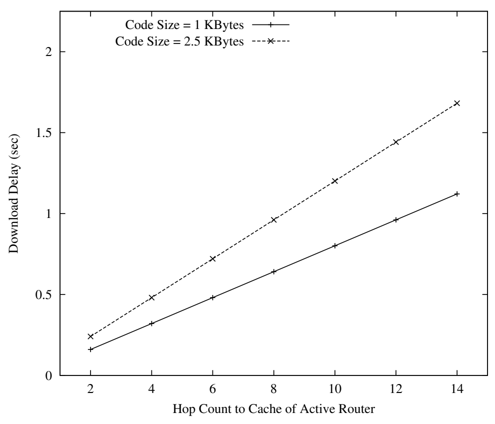
Discussion of Results: Fig. 2(b) shows the variation in downloading delay as a function of code size. By exploiting the caches present at Active Routers, the Stigmergy protocol achieves a near three fold reduction in downloading delay. The cache-less approach introduces considerably more delay than the Stigmergy protocol. This is shown in Fig. 2(d), where the delay associated with the cache-less approach is shown as a proportion of the corresponding delay for a network using the Stigmergy protocol. Excess delays imposed by active network systems on packets requesting services will adversely affect end applications. Furthermore, in such systems, packets will be dropped if the required set of services are not deployed



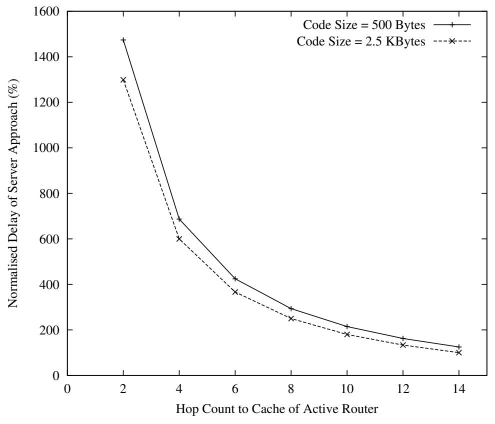
(a) Network Topology used for Analysing Stigmergy



(b) Service Deployment Delay in the Reactive Model



(c) Service Deployment Time for various Cache Locations within the Network



(d) Normalised Delay of the Cache-less Approach for Service Deployment

Fig. 2
PERFORMANCE ANALYSIS OF THE STIGMERGY PROTOCOL

within a predefined time bound. This will in turn increase the number of failed connections within the network. Hence, the complexity of the Stigmergy protocol, in comparison to the cache-less scheme, is justified by the superior delay performance of the network when it is used.

V. CONCLUSIONS

The delay to dynamically discover and deploy Active Services can affect the overall quality of service of an Active Networking system. We proposed a service discovery protocol, referred to as Stigmergy, which supports the discovery of active services in an Active Network. This protocol, by self-organising network nodes that are under a common administrative control into virtual cache clusters, maximises the chances of discovering the required services within minimal latencies. The Stigmergy protocol is completely distributed and follows a best-effort cache co-operation model. Furthermore, this protocol avoids the need to configure and maintain independent caching frameworks for service discovery purposes.

REFERENCES

[1] Kalaiarul Dharmalingam. Network Support for Multimedia Applications using the Netlets Architecture. *Ph.D Thesis, Dublin City University*, February, 2004.
[2] D. Wetherall John Guttag and David L. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. *In Proceedings of IEEE International Conference on Open Architectures and Network Programming (OPENARCH)*, 1998.
[3] D. Decasper and B. Plattner. DAN - Distributed Code Caching for Active Networks. *In Proceedings of IEEE INFOCOM*, pages 609–616, April 1998.
[4] Sollins K and Masinter L. Functional requirements for uniform resource names. *RFC 1737, http://www.cis.ohiostate.edu/htbin/rfc/rfc1737.html*, December 1994.
[5] S.V.Fuller, T.Li et al. Classless Inter-Domain Routng (CIDR): An Address Assignment and Aggregation. *RFC 1519*, 1993.
[6] Network Simulator, http://www.isi.edu/nsnam/ns/.
[7] Ellen W. Zegura et. al. How to model an internetwork. *In Proceedings of IEEE INFOCOM*, March 1996.