Mobile Host Location Tracking through DNS

Andreas Pappas[1,2], Stephen Hailes[1], Raffaele Giaffreda[2]
[1]University College London, [2]BTexact Technologies

Abstract: Mobile IP is widely regarded as the only viable solution to many of the challenges of mobile networking. However, we contend that it actually seeks to address two separate and separable concerns: mobile host location tracking and routing to the mobile host, the first of which is better addressed using dynamic DNS. Existing approaches to mo bility have largely dismissed dynamic DNS as a viable mechanism for location tracking in view of its supposed inefficiency. In this paper, we contend that this view is mistaken and that recent developments in dynamic DNS force a re -evaluation of this viewp oint. We further argue that dynamic DNS is compatible with the end-to-end philosophy that has successfully driven Internet development since its inception whereas the mobile IP clearly is not. In order to justify our claims, the performance of dynamic DNS with respect to dynamic update propagation speed has been tested and results indicate that it can indeed be used in a macro-mobility handling scheme.

1. Introduction

In conventional static systems, nodes rarely change IP address and so, given appropriate allo cation mechanisms, routing can be accomplished by reference to the address. Given the fundamental importance of this name resolution process, the major concerns of any system mapping between names and addresses is reliability and rapid response as opposed to timeliness of updates. The Domain Name System (DNS) was designed to provide just such a service, and whilst it has always theoretically had greater power than the provision of white pages services, it has been implemented with these constraints in mind.

On the other hand, the nature of mobility means that mobile hosts necessarily change point of attachment; a fact that would cause conventional IP sessions to break. Consequently, any universal mobility handling mechanism must deal with two separate concems: mobile host location tracking and routing to the mobile host. The current mechanism of choice is Mobile IP [3] (and Mobile IPv6), which attempts to handle both of these aspects of mobility; DNS is retained as in the static case as a simple name lookup mechanism. However this split is not the only one possible; it has long been realised that it would be possible to cause mobile nodes to update their DNS entries to reflect their current points of attachment, removing the need for home agents. The major concerns with such an approach have to do with transparency and efficiency. These two concerns are interlinked; thus, in the case of transparency, existing applications continue to work as expected in a mobile environment, given triangular routing, even if not as efficiently as one might hope for. If one rewrites existing applications to make best use of dynamic DNS, then transparency for legacy applications has been sacrificed (but the end-to-end principle has been preserved [1]).

The major reason for discarding the latter approach has simply been one of efficiency of implementation: DNS implementations have demonstrated relatively poor performance of DNS with respect to updates, largely because this has been of secondary concern to date for reasons described above. Another limiting factor has been the inefficiency of caching when dealing with dynamic information, although it has been argued that the increase in DNS traffic resulting from non-caching will not have a significant effect on global traffic [4]. However, developments in dynamic DNS mean that these problems are starting to be addressed. Thus, several mobility solutions that utilise DNS have been proposed recently [1, 6]. However, even in these solutions DNS is not regarded as fast enough to deal with fast updates [6].

The aim of this paper is to re -evaluate the suitability of the DNS as a location tracking mechanism for a mobility solution in the light of recent developments. The hypothesis is that the dynamic functionality introduced in DNS operation is adequate to support a mobile host location handling mechanism which, when combined with a seamless hand-over mechanism, can provide similar or better performance than Mobile IP whilst overcoming the problems related to Mobile IP. Additionally, dynamic DNS can support a wide range of other services apart from host mobility, such as load balancing, content routing, service discovery, personal mobility, profile storage etc. These features are already emerging in the Internet and we expect them to become increasingly common in the future.

This research demonstrates the ability of dynamic DNS to handle dynamic updates at rates that are desirable in a macro-mobility scenario. This is the result of empirical testing of DNS during dynamic operation, which demonstrates the ability of DNS to handle the load of high rate dynamic updates in a mobile environment. The issues associated with both DNS and Mobile IP with respect to mobility are discussed in section 2. The testing procedure and results are provided in sections 3 and 4 respectively, while section 5 presents possibilities for improvements and future work. Section 6 concludes this paper.

2. Mobility, Mobile IP and DNS

The notion of mobility has evolved to include a wide variety of services, not just host mobility. A somewhat facile, but illustrative, maxim is that mobility is the ability to communicate with anyone, anywhere, anyhow, anytime. In order to achieve this, we require much more functionality than current mobility solutions provide.

The de facto Internet mobility standard is Mobile IP. As described above, the main design constraint leading to the adoption of Mobile IP was a perceived need to support existing applications transparently, and, consequently, to maintain a static IP address to avoid session breakage. Mobile IP overcomes this problem by maintaining a static home IP address that is used to hide the variable point of attachment IP address from higher layers, thus maintaining the session. However, this practice does introduce several problems. These arise from the fact that Mobile IP maps IP addresses to IP addresses and performs encapsulation in an overlay network, a practice that has caused significant problems for multicast, QoS, security, routing, hand-off latency and latency arising from tunnelling/de-tunnelling for thin clients [5]. There are further questions over scalability as the number of mobile hosts increases significantly and those hosts roam between different networks both geographically and in terms of layering.

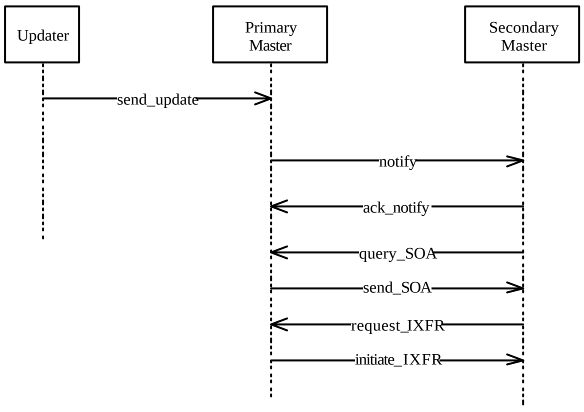

Figure 1 DNS update message sequence

On the other hand, DNS has always been considered a static database. Before the introduction of dynamic DNS, it was necessary to make all changes to DNS records manually by editing the zone files where records are stored. The advent of dynamic DNS meant that it became possible to support automatic updates of DNS records. Now any entity (host, DHCP server etc.) may be configured to update certain records based on pre-defined policies. The basic primitives of dynamic DNS are update, notify and incremental zone transfer, along with mechanisms for securing these operations [7, 8]. Update allows any entity to update (add, delete or modify) records in a DNS zone. Notify allows instant notification of secondary DNS servers that a zone has changed, so that they do not have to wait until their data expire. Incremental zone transfer (IXFR) allows the selective transfer of zone data that have changed in the primary master's zone file. This reduces the amount of data transferred, which is particularly useful for large zone files. The process of updating the authoritative servers for a zone involves extensive messaging between the secondary and primary name servers, as illustrated in Figure 1. When an entity issues an update, the update is forwarded to the primary master server, even if it is first sent to another server. The primary master is the only one which can commit an update. When the primary receives the update it checks to see if the prerequisites are met and then commits the update. It then issues a notification message to all registered secondary name servers in its zone file. The secondaries must acknowledge this message. Then the secondaries query the primary master for the current start of authority (SOA) record. If this indicates that the primary contains updated information, they request an incremental zone transfer, which is then initiated by the primary master. These enhancements allow for much more flexibility in DNS operation as well as new areas of operation, i.e. in mobility solutions as a location tracking mechanism. They result in significant performance increase from traditional DNS.

For support of seamless mobility using this approach, a session must not be bound to an IP address - port number 4-tuple, since these change when hosts move. Regrettably, most current applications have been built around the assumption that IP addresses and ports do not change during a session – an assumption that was reasonable in the era of the static Internet. However, we believe that the end-to-end principle, which implicitly requires changes to be made at the end hosts, has repeatedly proven itself as the guardian of evolvability through openness [13]. The reliance on an overlay network is a logical step in facilitating the development of new technology but is an indication of the immaturity of that technology. As the technology matures and becomes more widely used, we believe that its natural home lies in the endpoints in common with similar technologies. In this particular case, we believe that our proposed approach will allow simple, end-to-end, DNS-based mobility (i.e. for nomadic users), allow session mobility, service mobility, user mobility and role mobility. It will allow routing of individual data streams to any address so that a session can be transferred to another device (address) quite easily. Indeed, there are already solutions that support this behaviour [1, 2]. With session mobility in place, host mobility is much more simple and is no longer a routing issue (as in Mobile IP), but merely a matter of acquiring a new address. Seamless hand-offs can also be dealt with at higher layers. In general, DNS-supported mobility does not result in routing anomalies such as those introduced by Mobile IP; every host is treated in the same way by the network. This avoids problems with firewalls and with IPSEC. It also enhances scalability.

3. Testing

Our test-bed consisted of 5 Intel PIII 650 MHz machines with 128 Mb of physical memory, 384 Mb of virtual memory and 10 Gb HDD. They were connected on a virtual LAN (VLAN) through a switch. The network operated at 100 Mbps and delays were of the order of 1ms. All machines ran BIND v 9.2.0 DNS on FreeBSD v 4.2.

In order to simulate a real DNS structure and load, approximately 1.2 million A records were loaded in 20 zone files. A load of approximately 3,000 qps (queries per second) was delivered to each server in order to simulate real query load conditions. Two DNS architectures were tested: a flat one in which all secondary servers had a single (primary) master and a "deep" architecture in which each server (apart from the primary) had only one master and one slave. Load conditions (queries) were created using QUERYPERF, a tool included in the BIND 9.2.0 distribution. Updates were generated and forwarded using scripts and NSUPDATE, which is the tool used for dynamic updates. Updates were carried out either by secure update (using TSIG) [7] or by checking the IP address of the updater (allow-update by IP address option).

The metric employed to assess system performance was the time required from the instant the update was received by the primary master (PM) until the point at which an incremental zone transfer (IXFR) was initiated by the PM, as shown in Figure 1. No attempt was made to remove delay added by the network latency, since it forms a very small proportion of the delays measured, given our network configuration. Whilst it is undoubtedly the case that such latency will become increasingly important in determining overall performance in larger scale deployments, this is true for any type of communication between distant entities.

4. Results

Results indicate that response to messages by the servers is immediate. From the instant the PM receives an update message, it commits it (to the journal file) in approximately 4ms, irrespective of the query load on the server. The impact of a query load on the server can only be seen in the number of queries and updates it can handle, and not on the propagation speed of updates. Additionally, the size of the zone files does not affect performance; neither does the number of secondary servers. Therefore the flat architecture, as expected, is much faster than the 'deep' one. With no queries forwarded to a name server, it can achieve a rate of approximately 1,000 ups (update per second). However, when presented with a query load of 3,000 qps, the update rate drops to around 300 ups. These rates refer to group updates, each group consisting of 40 updates. If updates are sent individually, the update rate drops dramatically, as can be seen in Figure 2. This is the result of increased read-write operations on the journal file as well as increased messaging between updater and PM because each update is sent in a single message and processed individually. Therefore, there is a significant gain in performance if updates are grouped based on zone and RR type before being processed by the PM.

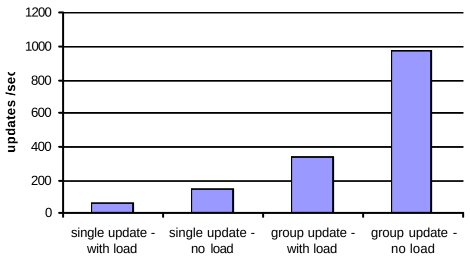

Figure 2 Update rate

Figure 3 shows the distribution of intervals at which an IXFR is initiated from the primary master to a slave server, under a query load and when idle respectively. This is approximately the time required for an update to propagate along the flat architecture. The average transfer interval is 529ms when idle and 607ms under the load. These values strongly depend on network delays and are expected to increase in wide-area conditions. It is obvious that the load does not significantly affect the update propagation time but, as stated above, it does affect the rate of updates that the server can handle. These figures, nevertheless, depend largely on the processing power and memory of the server and would improve if current technology (faster CPU and RAM) was used.

5. Future work - Suggestions for improvement

In this section, we propose several enhancements to the operation of DNS. These could significantly improve performance, particularly in poor network conditions. The amount of messaging between the primary and secondary masters following a DNS update can be prohibitive for fast updates in poor network conditions. This can be avoided if some messages are omitted or combined. Thus, for example, the NOTIFY and SOA response messages sent by the primary master could be combined by sending the new SOA in the notification message, thus avoiding one round-trip time. To avoid potential denial-of-service attacks in this case [9], it would be necessary for the notification message to be authenticated by the primary master. Furthermore, an IXFR could explicitly be initiated by a primary master (without a notification) in order to avoid 83% of total network delays between primary and secondary masters. However, this approach might compromise consistency of zone data, i.e. in case a secondary master was down and had an old version of the zone file. This situation could be avoided if the secondary master only accepted IXFRs that were consistent with its current SOA record, i.e. the primary master must assume that all secondary masters have a current (and similar) zone file. Otherwise a fallback mechanism to default behaviour would be triggered. Another potential enhancement would be to override the server hierarchy and implement a peer-to-peer policy with respect to dynamic behaviour. In other words, any name server would be able to receive and commit updates and initiate transfers to other authoritative name servers. This approach might also have consistency implications if consistency were not guaranteed, but would be much faster than the traditional behaviour. In order to minimise the load on name servers, mobility domains could be created and isolated from static domains, thus minimising queries on mobility domains and updates on static domains. Furthermore, it would allow enhancements to dynamic DNS behaviour without disrupting legacy DNS systems. Mobility domains could be further split into sub-domains that would allow update grouping, which significantly boosts performance.

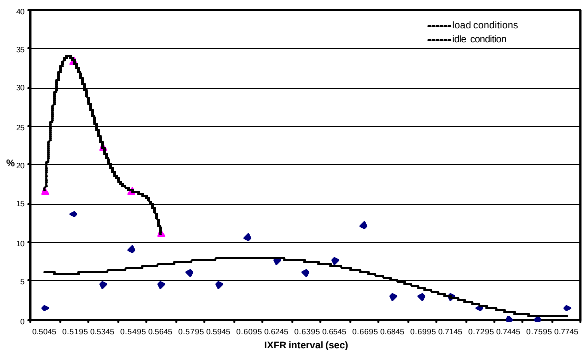

Figure 3 Distribution of update intervals

6. Conclusion

Despite the popular belief that DNS is slow and not well suited to providing a mobility solution, we argue from an architectural viewpoint and demonstrate practically that this is no longer the case. In short, this research has demonstrated that current implementations of DNS are suitable for use in mobility solutions as well as any other services (i.e. load balancing) or applications that require DNS updates at rates as high as once per second. We believe that this rate could be further increased if the modifications that we suggest are made to DNS.

7. References

[1] A. Snoeren and H. Balakrishnan. An End-to-End Approach to Host Mobility. *6th IEEE/ACM International Conference on Mobile Computing and Networking (Mobicom '00)*. Boston, MA, August 2000.
[2] V. Zandy and B. Miller. Reliable Network Connections. *(To appear in) Mobicom 2002*, 2001.
[3] C. Perkins. IP Mobility Support for IPv4. *RFC3220*. January 2002.
[4] J. Jung, E. Sit, H. Balakrishnan and R. Morris. DNS performance and the effectiveness of caching. *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2001.
[5] G. Karagiannis, Mobile IP State of the Art Report. *Ericsson Open Report.* June 1999.
[6] Y. Chen and T. Boult, Dynamic Home Agent Reassignment in Mobile IP, *IEEE Wireless Communications and Networking Conference 2002*, Orlando, FL, USA. March 2002.
[7] B. Wellington. Secure Domain Name System (DNS) Dynamic Update. *RFC3007*. November 2000.
[8] P. Vixie, S. Thomson, Y. Rekhter and J. Bound. Secret Key Transaction Signatures for DNS (TSIG). RFC 2845. May 2000.
[9] P. Albitz and C. Liu. DNS and BIND, 4th Edition. *O'Reilly & Associates, Inc.* 2001
[10] P. Vixie. A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY).*RFC 1996*. January 1996
[11] P. Vixie, S. Thomson, Y. Rekhter and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). *RFC 2136*. April 1997.
[12] M. Ohta. Incremental Zone Transfer in DNS. *RFC 1995*. August 1996
[13] J. Saltzer, D. Reed and D. Clark.. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, November 1984.