SCALABLE SIGNALING UNDERLAY FOR OVERLAY NETWORKS

Yangcheng Huang, Saleem N. Bhatti
Y.Huang@cs.ucl.ac.uk, S.Bhatti@cs.ucl.ac.uk
Networks Research Group, Department of Computer Science
University College London, Gower Street, London WC1E 6BT, UK

Abstract - This paper presents the design of a scalable decentralized signaling underlay infrastructure, which features a DHT based management information storage and query-based state lookup mechanism. The signaling underlay is aimed to apply a decentralized "peer-to-peer" style searching and discovering engine into the management and control plane of the overlay network, including grid networks and p2p applications, to facilitate deployment of QoS service.

1. INTRODUCTION

Over the years, there has been an increasing desire that the Internet be used to carry data flows with specific QoS constraints. However, this requires all network elements on a path to cooperate to provide such a "better-than best-effort" service. The networks would require a tremendous amount of state information to provision, maintain, validate, and bill for these new services, across heterogeneous networks and devices with different management capabilities. Such a requirement demands more research effort on network management and control issues, especially on signaling between elements, to control and distribute network state.

This paper presents the design of a scalable decentralized signaling underlay infrastructure, which features distributed hash table (DHT) based management information storage and query-based state information lookup mechanism. This proposed signaling infrastructure applies a decentralized peer-to-peer (p2p) style searching and discovering engine into the management and control plane of the overlay network, including grid networks and p2p applications, to facilitate deployment of QoS services.

The paper is arranged as follows. In section 2, we discuss the design principles of the signaling underlay. Then in section 3 we present details of the proposed infrastructure and related mechanisms. In section 4, we give some implementation details, and in section 5, we show some preliminary experimental results. We introduce related work in section 6, and draw conclusions and list future work in section 7.

2. SCALABLE SIGNALING UNDERLAY

In this section, we present motivating factors for scalable signaling functionality, and from them extract requirements and design principles for a scalable signaling underlay.

2.1 Problems with overlay networks

In recent years, peer-to-peer overlay networks have received much attention, especially in decentralized data-sharing and discovery ([2] [4] [8] [9]). The network overlay abstraction provides flexible and extensible application-level management techniques that can be easily and incrementally deployed despite the underlying network. However, there are several issues to be addressed.

Despite performing suitably in scalability and deployment, overlay solutions make data-sharing and state management more complex, resulting in additional complications in developing such systems. The complexity of managing a network increases dramatically as the number of services and the number and complexity of devices in the network increases. Also, although an overlay network may facilitate end-to-end management and control, it still needs efficient skills to manage underlying network resources.

2.2 Design Principles

Generally, the signaling underlay should be applicable in a very wide range of scenarios and at the same time be simple in implementation and lightweight in resource consumption. This section will analyze such requirements in detail.

The design principles include:

- *Decentralized architecture.* Traditionally, the common framework based on a centralized approach is straightforward but not proved to scale well; although a distributed approach with a hierarchical architecture appears flexible, conceptually, it scales only in number of nodes, not at the network level. That is, inside network domains, a distributed framework may work well but with the cost of increased complexity in measurement and control. However, a distributed approach is not satisfactory at the domain level, lacking efficient cross-domain network measurement and management support. For a cross-domain signaling underlay, a decentralized architecture seems the sensible choice.
- *Modularity and isolation.* To work in multiple scenarios, the system must be designed modularly. With such a layered, modular design (Figure 1), the system can adapt itself to different scenarios, with the

least modification of kernel signaling function design, and be more flexible in resource consumption, self overload and effects to its host environments. In addition, the system should achieve *isolation* in two aspects. The signaling messages should be separated from data, to be independent of detailed applications and work in different scenarios; it should also be separated from control information and protocols; that is, signaling elements are not expected to do resource allocation as controllers or agents; their functionality should focus on cooperation and coordination between network entities.

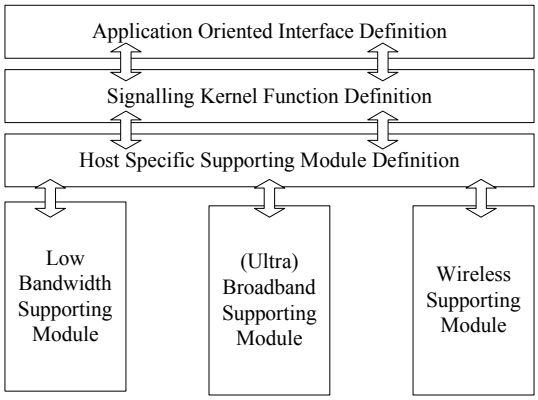


Figure 1. Modular system design.

- *Data integrity.* The signaling system aims to manage changing network state information, so it is necessary to maintain state integrity in signaling by updating explicitly, including deleting state information that is obsolete, refreshing state information in case of state changes (caused by failure or nodes leaving and joining). Such an updating process should be as quick as possible, especially for mobile nodes, in order not to interrupt services.
- *Security.* There are several security issues that need to be addressed.
  (1) Transparency in signaling. Network (domain) topology and structure information should be hidden from end nodes (users) and from other network domains. During the forwarding process, a signaling protocol should keep route information transparent to unrelated nodes.
  (2) Authentication. There should be authentication methods to identify signaling peers with each other, with ID and key management mechanisms, such as shared secret or public key methods.

  Note that, we do not consider specifically security issues in this paper.
- *Scalability.* The system should scale to Internet scale deployment, and in resource consumption, with minimal effects under self-overload to its host environment.
- *Resilience.* The signaling system should be resilient in case of internal failure and external failure. Here, internal failure is caused by node changes (node joining and leaving) or malfunction behaviour, whilst external failure is from host environments and network connections. Currently, most peer-to-peer systems only address internal failure in resilience and management issues [2, 4 and 9], while this research aims to achieve both. In addition, the signaling elements should respond to changes and refresh relevant state information quickly to localize update effects and reduce updating traffic overhead.

Given the design principles above, we now present our design of a signaling underlay.

3. ARCHITECTURE

There are two main components in the system: Distributed Hash Table (DHT) and query engine (Figure 2).

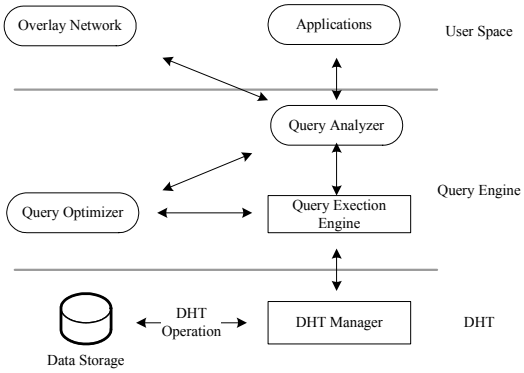


Figure 2. Architecture of Signaling Underlay.

An instance of each DHT and query engine component is run on each participating signaling point.

3.1 Research Scope

This research focuses on the control of the underlay network management to provide "better-than best-effort" network service for the overlay network. Generally, three issues are of concern: network measurement (state collection), signaling and state control. This paper only discusses the signaling part. Resource control (resource allocation and re-allocation) and network measurement methods are not considered here.

We assume that service providers are ready to implement such a signaling infrastructure, since different administrative domains may apply different security policy and deny the signaling flow by default.

3.2 Definitions

Signaling Point (SP) is defined as any network element that runs the signaling process or any network element that is involved in the signalling.

Signaling Forwarding Point (SFP) is defined as the signaling entity whose role is to receive signaling messages

from another entity and then forward it to the next entity, according to a node selection algorithm.

Signaling Initiating Point (SIP) is defined as the signaling entity that initiates the signaling; an end-system or a router.

Signaling Sink Point (SSP) is the entity where the signaling process terminates.

Signaling Path is the path along which the signaling messages transit; signaling packets can be delivered along a data path, or with its own routing policy.

Signaling Underlay is the collection of SPs, SFPs, SIPs, SSPs and the network paths that together comprise our signalling system. The signaling entities transmit signaling message through underlying network. Logically these entities form a virtual peer-to-peer-style signaling layer, possibly with different topology from lower network layer, and across different administrative domains (AS) and technology domains (wireless or wired network).

3.3 DHT Based State Storage

Distributed Hash Tables (DHTs) have been proved to be an effective way for decentralized information storage, e.g. CAN [8]. In the following, we focus only on our DHT design.

In every signaling point (SP), there are one or more DHTs with the data structure shown in Table 1. In the KEY field, we define two types of data: *IP-pairs* and *IP-sub-network*. In the VALUE field of the DHT are (1) values of link characteristics, if the targeted link is in local domain; (2) another SPs' ID (we use an IP address for simplicity for now), which holds the desired data, if the targeted link is in another domain. Thus, in the DHTs, there are two types of items. One is *Data_Item*, which stores the desired data, such as link/path characteristics; the other is *Pointer_Item* which stores a pointer to (the location) of the desired data.

To illustrate, we explain this in the following two examples.

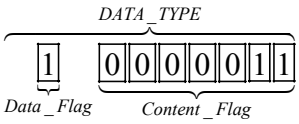
D (key, value) = ((128.16.64.6, 128.16.64.7), (60,100))

The above value means, the characteristic of the link (or path) from node (128.16.64.6) to node (128.16.64.7) is 60Mbit/s (bandwidth) with (maximum) delay 60 ms.

D (key, value) = ((202.16.1.0, 202.16.6.0), (202.16.1.1))

This means, the network state information can be found in host/server with the IP address *202.16.1.1*.

In order to differentiate these two types of values and also to label the content of value field, we specify a DATA_TYPE field as below. The first bit, *Data_Flag,* is 0 if the VALUE field stores a pointer to another SP. (POINTER_ITEM, IP address of other SP in this case); if data_flag is 1, the *Content_Flag* denotes what kind of content is stored in the VALUE field. For example, we can define 0000011 as *Bandwidth_Delay*, which means the value field stores bandwidth and delay information.



The structure of the DHT is as follows.

Node I		
DATA_TYPE	KEY	VALUE
10000011	(128.16.64.6, 128.16.64.7)	(60, 100)
10000011	(128.16.64.45, 128.16.64.46)	(54, 10)
…	…	…
0xxxxxxx	(10.1.24.0,10.1.24.0)	(10.1.24.238)
0xxxxxxx	(20.16.1.0,20.16.6.0)	(20.16.1.1)

Table.1 Data Structure used in DHT

In our future design, variable-length data structures will be defined to contain different types of signaling data.

3.4 State Query Mechanism

To discover network state information across domains, we propose a light-weight searching/discovery mechanism.

3.4.1 State Query

Since network state is stored and managed in a decentralized way, we need to send queries to the related signaling points (SPs) in the domain A if we want to know the network characteristics (state) of A. When a SP receives a query, it will look up its own data first with our *nearest-matching* algorithm a shown in Figure 3.

The algorithm has three possible results.
(1) The desired data (or pointer to the SP where the desired data is stored) is found by an exact match between the KEY field of the DHT with the network values (e.g. IP addresses);
(2) By nearest-match, the algorithm returns a pointer to the SP where the desired data can most likely be found (or the SP which is more likely to know where is the desired data); here we use, SPs with "similar" IP addresses, such as *10.13.1.21* and *10.13.4.2*, are mostly likely to "know" each other;
(3) When there is no match, it will select a next-hop SP from its neighbouring signaling points and forward the unresolved query to the next SP. Simply, flooding based (broadcasting or multicasting) is most straightforward, but too costly and only suited for "highly replicated items" [3]. Below we consider an optimized method to choose a next-hop SP for our use.

```
SP_DHT* SP_Entity:: SP_Query_Handler (SP_Query q)
{
  SP_Entity* sp=this;
  SP_DHT* dht_entry= sp → getDHTEntry(sp);
  SP_DHT* found=sp → lookup(q.key, dht_entry);
  If (found!=null) {
  // Find desired data by exact matching, then return the item
     return found;
  }
  // Otherwise, find the item of maximum matching
  SP_DHT* t;
  SP_DHT* p=dht_entry;
  // variable 'merit' is the merit of matching
  int merit=0;
  while(p!=null)
  {
    int m=sp.match(q.key,p → key);
    if (m>merit)
    {
      merit=m;
      t=p;
    }
  }
  if merit ==0, return null;
  return t;
}
```

Figure.3 Pseudo Code of Nearest-Match Algorithm

Note that, through changing the semantics of the "match" method of Figure 3, the algorithm can be applied in other circumstances and treated as a general algorithm.

3.4.2 Optimized Query Forwarding

Every SP keeps track of its immediate neighbouring SPs. In our query-forwarding algorithm, we select one neighbour as a next-hop forwarder. As in Pastry [9], we adopt an ID-based mechanism for nodes.

Each signaling point has a unique identifier or nodeId. When the look-up process fails to find related data, an SP will select the neighbour with the nodeId that is "closest" to the key. Here, in our case, the nodeId is the IP address of the signaling point. That is, we select the neighbour whose IP address shares the longest prefix with the key. We acknowledge the limitations of using IP address as a nodeId but this is an interim solution. Such a solution becomes inadequate in certain scenarios such as multi-homed hosts, routers (multiple interfaces and so IP addresses) and hosts behind NAT (Network Address Translation) boxes. However, to focus on our signaling mechanism, we still adopt this solution for now, for the experiments in this study.

From experimental results of Pastry [9], such a mechanism is "highly effective" with O (log N) forwarding steps and O (log N) routing table size.

4. IMPLEMENTATION

In this section we present some implementation details in our prototype signaling system.

4.1 Hybrid Architecture

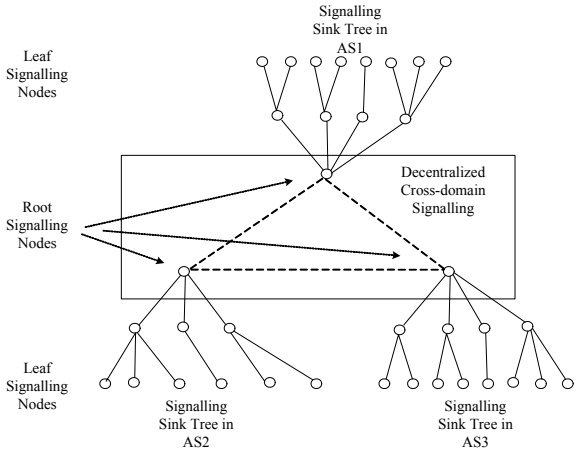


Figure.4 Hybrid Architecture

For our prototype a hybrid architecture is adopted (Figure 4). Inside administrative domains, we adopt a hierarchical design and construct a signaling sink tree in each domain; for signaling between domains, we choose a decentralized architecture and signaling only occurs between root signaling nodes of the signaling sink tree in each domain.

4.2 Signaling API

Next, we briefly outline the APIs, simplified for clarity.

nodeId= SP_Init() starts the local nodes as signaling points, initializes relevant states and return the nodeId. During this process, the SP will acknowledge its neighbours.

SP_LookUp (Key, dht_Entry) searches the DHT by the nearest-match algorithm to find desired data.

SP_Forward (query) called when no match is found, and a neighbour's nodeId is numerically closest to the KEY.

5. PRELIMINARY EXPERIMENTAL RESULTS

To evaluate the proposed signaling system, we implement a prototype in Java with simple topology as shown in Figure 5.

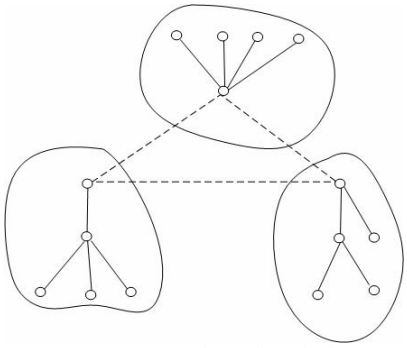


Figure.5 Topology of test bed

Each node runs Red hat 8 and we emulate three administrative domains. In each domain, the bandwidth of each link is stored in the DHT of each node, and is updated periodically using Iperf [7].

To simulate network changes, we use tools in Linux to change the link's bandwidth, and to allocation IP addresses to emulate multi-domains.

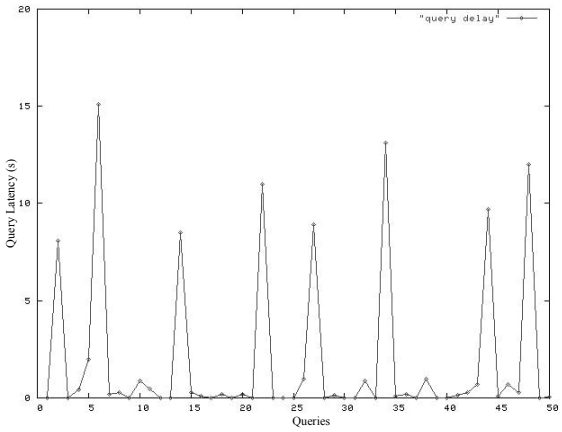


Figure.6 Preliminary Experimental Results on Query latency

We initialize a query from every leaf node, and query the bandwidth status in other domains. The targeted link, whose state is queried, is chosen at random.

The preliminary experimental results on query latency are shown in Figure 6. From the figure we can see that query results are returned within 10 seconds, most within 1 second.

6. RELATED WORK

PIER [2] is a peer-to-peer information searching and query engine based on database technology. It is intended to extend database query processing facilities into Internet-like distributed environments.

PASTRY [9] is an application level P2P routing and node location overlay. It provides an overlay routing/forwarding mechanism with a key-based approach to organize nodes and forward messages between them. It is designed to be resilient and adaptive to node failure and attacks, but not to low level failure such as network link failure.

Xen [6] is a system-level active services platform, providing support for virtualization, along with mechanisms for discovery of the location/existence of xenoservers. It only manages Xen nodes' state information.

X-Bone [5] is designed for automated deployment, management, coordination, and monitoring of IP overlay networks to reduce configuration effort and increase network component sharing. It features security solutions and deployment of multiple concurrent overlays. X-Bone uses multicast to simplify resource discovery.

Resilient Overlay Network (RON) [4] is an application-layer policy routing architecture. The nodes of RON monitor the network path status to detect (and recover from) path failure. RON uses round trip time (RTT) to estimate one-way loss probability and stores path performance in a separate performance database. RON uses sampled information to select routing paths with higher throughput and low latency.

7. CONCLUSIONS AND FUTURE WORK

This paper presents the design of a scalable decentralized signaling underlay, which is aimed to apply peer-to-peer style searching into the management and control plane of the overlay network. Preliminary experimental result shows that the query latency is satisfactory and traffic overhead does have a significant effect on other network flows.

However, there are some limitations that need to be resolved in future work.

First, the experimental environments emulated so far are too simple; we need to install it in more nodes to examine the traffic overhead and query latency in large-scale scenarios.

Second, methods, such as signaling aggregation and data caching, can be used to optimize query and improve the efficiency.

Third, state-updating policy should be defined to limit the transmission of updating message and reduce updating traffic.

Fourth, security issues need to be considered and implemented.

8. REFERENCES

[1] Matthew Harren, Joseph M. Hellerstein, Ryan Huebsch, Boon Thau Loo, Scott Shenker, and Ion Stoica, *Complex Queries in DHT-based Peer-to-Peer Networks*, IPTPS, March 2002.Springer-Verlag, LNCS .

[2] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, Ion Stoica. *Querying the Internet with PIER*, VLDB, 2003.

[3] Boon Thau Loo, Ryan Huebsch, Ion Stoica and Joseph M. Hellerstein. *The Case for a Hybrid P2P Search Infrastructure*, 3rd International Workshop on Peer-to-Peer Systems (IPTPS '04), San Diego, CA, Feb 2004.

[4] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris, *Resilient Overlay Networks*, Proc. 18th ACM SOSP, Banff, Canada, October 2001.

[5] Joe Touch, *Dynamic Internet Overlay Deployment and Management Using the X-Bone*, Computer Networks, July 2001, pp. 117-135.

[6] Paul Barham, et al., *Xen and the art of virtualization*, Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003, P164 - P177.

[7] NLANR (The National Laboratory for Applied Network Research)'s Iperf project web page: http://dast.nlanr.net/Projects/Iperf/.

[8] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker, *A Scalable Content-Addressable Network*, ACM SIGCOMM, 2001.

[9] Antony Rowstron, Peter Druschel, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*, Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany, Nov 2001.