Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

AW2 Report

Christian Vogt

DHT Design Comparison and Applicability for WebRTC-based Systems

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Christian Vogt

DHT Design Comparison and Applicability for WebRTC-based Systems

AW2 Report eingereicht im Rahmen der Veranstaltung MINF-AW2

im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Eingereicht am: 28. Juli 2013

Christian Vogt

Title of the paper

DHT Design Comparison and Applicability for WebRTC-based Systems

Keywords

DHT, WebRTC, Web-based DHT, Performance, Churn

Abstract

This document gives an inside on performance related characteristics a DHT imposes. Popular design decisions ruled by other DHT designers and their effect on performance are investigated and mapped to our current work centering around a DHT implementation running in the users browser that we call a web-based DHT.

Contents

1 Introduction

The Web of today is a huge system of interconnected nodes providing a multitude of data to its' users. Efficiently distributing and structuring the content still remains a challenge, though. As a new approach, we are currently investigating a concept for a peer-to-peer (P2P) based content sharing facility that centers around the users' data instead of the servers'. To conveniently serve data, this approach uses web browsers as its application platform which enables a broad deployment across different devices and operating systems.

Web browsers are currently implementing the WebRTC technology which, for the first time, allows for direct P2P-communication between them. While this technology forms the basis of the transport layer, using a distributed hash table (DHT) as a routing layer allows the system to scale in a logarithmic manor. The three main criteria that mark a DHT implementation are Security, Scalability and Performance. Each aspect has been extensively researched over several years. While security and scalability are important factors, this paper focuses on performance. To drill down on the specifics, important related work has been chosen that summarizes performance from both, the implementers and the designers point of view. The insights of this work raised the need for an emulation component for measurement purposes to verify the implications imposed by design decisions.

This paper is structured as follows. Chapter 2 introduces the core technologies used, namely WebRTC and distributed hash tables (DHT). Chapter 3 talks about the core design decisions and their implications on performance. The insights of this work are used in chapter **??** to determine the overlapping design issues for a web-based DHT.

> "The Internet is the world's largest library. It's just that all the books are on the floor." — John Allen Paulos

2 Background

For completeness, this section describes fundamental components that are mandatory to understand the remainder of this paper. These components are the WebRTC protocol suite and DHTs as described in section 2.1 and 2.2.

2.1 WebRTC

WebRTC is a web based approach that allows an implementation (which typically is a web browser) to communicate with another implementation directly over a peer to peer connection [1]. In order to achieve this goal, a set of protocols is being drafted in the IETF Working Group *Rtcweb* and a corresponding API is defined by the W3C to allow web browsers to leverage the frameworks benefits.

A WebRTC communication instance is called a *PeerConnection* and consists of two peers at either ends. To initiate and preserve such a connection, a signaling mechanism has to be used which is left to the application as much as possible. This is justified by the unique nature of web browsers that cannot easily preserve states over website reloads and the large variety of applications using different protocols such as SIP or XMPP/Jingle for signaling. The pursued approach to give the application control over the signaling plane is called *JSEP* and specified in [12]. JSEP uses the Session Description Protocol (SDP) media descriptions that have to be sent to the other peer over an arbitrary, bidirectional channel.

Apart from session management, WebRTC covers audio and video as well as data transport over a PeerConnection. Non-media data transport aspects of WebRTC are specified in [6] that introduces the *DataChannel* that, at its core, uses a SCTP over DTLS over UDP protocol stack which will be further elaborated on in section 4. As of today, Mozilla Firefox and Google Chrome implement the API and other vendors are expected to follow when the API stabilizes.

Even though the WebRTC framework provides peer to peer connections, it does not deal with important aspects such as routing between the peers or storage capabilities that form a P2P Overlay. Still, a subset of WebRTC, namely the DataChannel, can be used as the substrate of such an Overlay.

2.2 Distributed Hash Tables

Two classes of P2P Overlays exist: *Structured* and *Unstructured*. Unstructured overlays organize participating peers in a random graph and use mechanisms such as flooding or random walks to query for content. Because content is nondeterministically distributed over peers, each traversed peer has to perform expensive local operations which makes this approach inefficient for content that is rarely replicated among peers.

Structured Overlays, on the other hand, provide the benefit of deterministic information placement, i.e. content is stored not at random locations but at specific peers to make queries more efficient. This is achieved by incorporating Distributed Hash Tables (DHT) which form the basis of such a structured overlay and support the scalable storage and retrieval of key-value pairs in large scale systems by introducing a key-based routing layer (KBR). The KBR hides away the complexity and provides a simple `value=get(key)` and `put(key, value)` interface to the overlay application.

Such a DHT maps peers to a large uniform random identifier space. Content is then mapped to the same identifier space by consistently assigning keys to each data object (e.g. by using a hash function like SHA-1). As a final step, the content is assigned to a unique peer in the system which is chosen by the overlay protocol in use (e.g. Chord [11], Pastry [10], CAN [8]). Every peer has to maintain only a small routing table consisting of selected identifiers and corresponding connections to each peer. When a traversing lookup arrives, the protocol chooses a nearby peer out of its routing table to send the request to. This is possible because of the consistent mapping of data and peers to the same identifier space and thus maintaining a mathematical relation between each other.

DHT-based systems can make certain assumptions about the number of overlay hops it takes to reach the peer that holds the content, typically $\mathcal{O}(\log N)$ where N is the number of participating peers ([7] give further details about different overlays). Structured overlays are typically based on similar designs like ring- (e.g. Chord or Pastry) or geometry-based approaches (e.g. CAN). Besides the usage as a basis for structured overlays, DHTs also find a use in other systems such as the Apache Cassandra NoSQL-Database[1], the Amazon SimpleDB Key-Value store[2] or the GlusterFS distributed filesystem[3].

In section 3, popular design decisions and tradeoffs of DHT-based systems will be further elaborated on while evaluating differences and spotting possible issues with the introduced WebRTC-based implementation.

[1] http://cassandra.apache.org/
[2] http://aws.amazon.com/de/simpledb/
[3] http://www.gluster.org/

3 DHT Design Decisions and Implications on Performance

In this section, design decisions that have been rendered for existing DHTs and their implication on performance will be discussed. To do so, a typical DHT architecture is introduced first. Afterwards, the most prominent design decisions and their performance implications are discussed. Then, performance metrics and measurement conductions that are specific to P2P systems are introduced. The section ends with a summary that concludes the different aspects.

3.1 Abstract Architecture

Figure 3.1 shows the simplified abstract architecture of the web-based P2P system this work contributes to. It is based on the API proposal by Dabek et al. [5] containing a key-based-routing-layer. This approach allows for interchangeable DHT implementations and a loosely coupled, application-layer independent development process. Moreover, the similarities to other implementations allow us to easily adapt the design decisions further elaborated on in section 3.2.

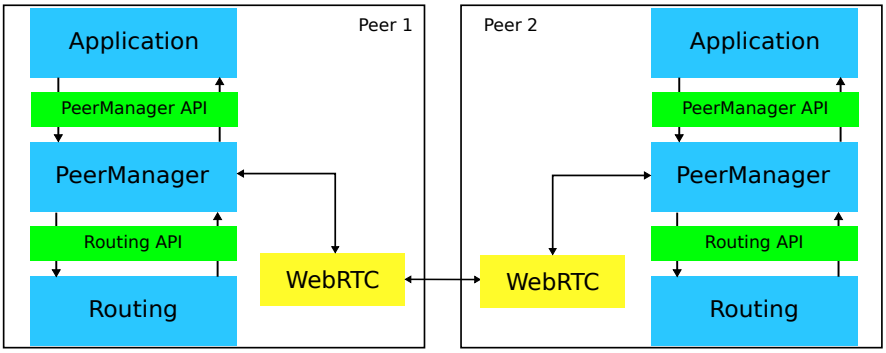

Figure 3.1: Abstract architecture based on the common API by Dabek et al. [5]

3.2 Design Decisions and Performance Characteristics

DHTs use diverse techniques such as different routing mechanisms, proximity neighbor selection, erasure coding, replication and server selection to accommodate for varying application's requirements. Dabek et al. [4] summarize the effects on performance (namely latency and throughput) from prominent DHTs by using both a simulator with an Internet latency model and a custom implementation of a DHT named *DHash++* that combines multiple techniques to optimize latency and throughput capacities. This section will present key results in further detail by explaining the design decisions and how they affect performance.

Data Layout. DHTs might not store actual data but use a separate layer of indirection, i.e. the values might only consist of a link to the actual data (such as an URL). Generally speaking, integrating lookup and storage functionality in the DHT benefits latency because the data can be piggybacked on the fetch response instead of triggering a new request to the indirection layer.

DHTs that store data have to decide on the size of the data units. A data unit might refer to a disk sector-like fragment of a file, a whole file or even an entire file system image. In general, large unit sizes lead to a inferior number of lookups while splitting large files into smaller units distributes the load over more peers.

Another arising design question of data layout is the location of the data unit. If a particular data unit is likely to be read by peers in a specific geographical location, it makes sense to condense data units instead of spreading them widely. On the other hand, this makes the system more vulnerable to network outages and leads to an uneven balanced load distribution in the system compared to a system with randomly placed data units.

Iterative vs. Recursive Routing. A number of hops is required to reach a key's corresponding peer. The requests can be routed over theses hops in an iterative or recursive manner as shown in Figure 3.2. Apparently, recursive routing reduces the latency by half because intermediate hops forward the request directly instead of answering with the address of the next hop.

The downside of recursive routing is a more difficult failure detection. When an intermediate peer fails to route the request, the sender cannot immediately take notice and thus cannot retry or reroute as fast as by using iterative routing. Dabek et al. suggest a combination by providing a fallback mechanism from recursive to iterative routing after numerous failed attempts.

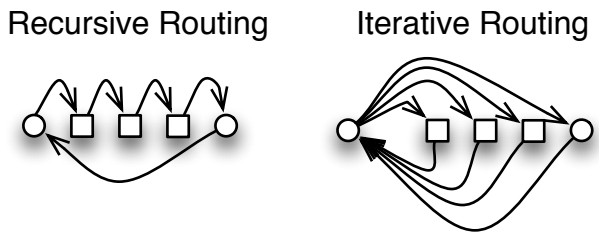

Figure 3.2: Recursive and iterative routing in a DHT with round shapes for sender (left), receiver (right) and rectangular for intermediate peers that route the request

Proximity Neighbor Selection. If taken into account that a short hop count in the overlay can result in significantly longer paths in the underlay, proximity neighbor selection (PNS) enables a DHT peer to influence the choice of peers in its peer table. A search algorithm has to be used to find nearby peers whereas the crucial part of this algorithm is a efficient latency prediction between peers that only consumes minor peer resources (such as Vivaldi [3]).

By using PNS, Dabek et al. achieve an approximated 1.5 times the average round trip time compared to the underlying network regardless of the system's peer count.

Transport and Congestion Control. Typical DHT implementations use either a TCP- or a UDP-based approach that is compatible with todays broadly deployed IP-stack. The usage of custom protocols on top of UDP are able to increase performance by using a congestion control that is specifically suited for P2P communication purposes. The web-based DHT introduced in this paper, though, depends on the WebRTC DataChannel as its transport mechanism that is based on a DTLS over SCTP over UDP approach. At the time of writing, the DataChannel specification is not finalized and is expected to change in the near future, therefore performance related assumptions have to be pushed to future work.

3.3 Churn Impact

Rhea et al. [9] reveal that deployed DHT systems, contrary to simulation-based results, suffer from major performance losses or even break down when exposed to high churn rates. To encounter this issue, they present *Bamboo*, a churn-resilient DHT implementation. This section describes the key design decisions that led to increased resistance against churn and their implications on performance.

Fast and accurate failure detection. In section 3.2 iterative and recursive routing have been confronted and the results show that the latency benefits from recursive routing come with a price of more complicated failure detection. Using recursive routing also brings the advantage of a smaller number of connections because there is no need to establish additional connections to the intermediate hosts. This in turn makes failure detection easier because each peer only has to monitor the peers in its routing table making recursive routing a good choice for systems with a high rate of churn. To accommodate the more complicated failure detection, static resilience can be used to give intermediate peers the opportunity to choose an alternative path for the request. Thus, the initiating peer does not have to be informed about peer failures in the overlay path.

Reactive vs. proactive failure recovery. To detect these malfunctioning links between peers, a reactive or proactive strategy can be used. Proactive recovery reacts to failed links by immediately searching for a new peer imposing load on the system. If the increased load overburdens the network, a positive feedback loop is formed and the network can collapse. Therefore, periodic failure recovery with fixed intervals has an advantage although it cannot bring the system back to a consistent state as fast and imposes management traffic even if no requests are pending.

3.4 Measuring Performance

In contrast to client/server systems, even simple measurements of P2P systems such as link stress can be difficult to perform as no central node exists but measurements have to take a great quantity of links into account. Simulation (such as p2psim[1]) and Emulation (such as ModelNet [2]) approaches are possible encounters for complex measurement scenarios. While a simulator mimics the behavior of a complete system, an emulation can replace the original implementation or parts of it and behave just like the real one (including external communication). Therefore, simulation can evaluate and compare DHT algorithms such as the base Pastry or Chord ones but have to be implemented on top of the simulation framework, thus the code is mostly not reusable for a real implementation. Emulation, on the other hand, allows a real implementation to be tested by providing an environment with realistic latency and throughput characteristics.

To be able to compare different DHTs in scenario-based approaches – that allow for reproducible environments – is a crucial requirement to evaluate the variety of design decisions

[1] http://pdos.csail.mit.edu/p2psim/
[2] http://modelnet.ucsd.edu/

presented in chapter 3 and their implications on performance. The choice of emulation or simulation depends on the use cases as stated above but emulation seems to provide a more practical oriented approach.

Once a measurement environment has been set up, the next step is to find metrics that describe performance attributes of the system. Some of the metrics that are commonly used in 1:1 communication like *Link Stress*, *Node Stress* and *Packet duplicates* can be reused while other, more fine-grained metrics like *Delay Stretch* and *Lookup Sucess Rate* have to be chosen to characterize the special nature of DHTs.

Delay Stretch. By using a overlay schema like chord, intermediate routing steps are introduced that lengthen the path between two peers. Delay Stretch is defined as the ratio between the resulting path and direct communication between these peers using a distance metric like round-trip delay:

$$d := \frac{\sum dist_y^x}{dist_v^u}, \text{mit } x, y : \text{intermediate peers on the routing path from } u \text{ to } v.$$

The delay stretch gives an assumption on the network efficiency, being optimal at a delay stretch of 1 while an increase results in deteriorated efficiency. To obtain the delay stretch, a sufficient number of peers have to participate. According to Castro et al. [2] high link and node stress on any of the intermediate peers may distort the delay stretch measurement.

Lookup Success Rate. In a DHT, lookups are likely to fail because an intermediate peer can at any time exit the system or be overloaded. The lookup success rate in proportion to time shows how the implementation reacts to certain scenarios like outtakes or higher than usual rates of churn. Obtaining the lookup success rate depends very much on the implementation and if routing is done in a recursive or iterative manor. Iterative routing makes it easy to determine lookup failure while the situation is more complex when using recursive routing as described above.

3.5 Summary

By combining multiple techniques, Dabek et al. achieve high cumulative performance improvements. By using recursive instead of iterative routing the latency has been significantly reduced and proximity neighbor selection achieves 1.5 times the average round trip delay compared to the underlying network. Further latency improvement can be achieved by using

specialized transport protocols. There are drawbacks though, that have not been taken into account. The authors presume that malicious nodes do not exist in the system. These would have a negative effect on performance when taken actions against, eventually making the read and write algorithms more expensive by introducing some kind of authentication facility.

Another factor that has not been examined by Dabek et al. is churn, the rate of peers joining and leaving the system. As Rhea et al. [9] point out, churn has a crucial impact on performance and many DHT implementations suffer from high rates as described in section 3.3. The authors introduced effective countermeasures by using static resilience and reactive failure recovery.

In order to identify potential performance related issues early in the development process, emulation or simulation can be used to obtain metrics that characterize the system as good as possible. The test results can than be used to iteratively improve the system.

4 Web-based DHT

The motivation driving this paper is the development of a DHT that uses WebRTC DataChannel as its communication underlay, namely a *web-based DHT*. To drill down on the specific characteristics such a web-based DHT imposes, a definition is given first of. Further on, this chapter concludes how the insights of chapter 3 can help in designing the DHT. The remainder of the chapter presents our plans for future work.

4.1 Implications for a Web-based DHT

While common DHTs are built to communicate over the global Internet, a web-based DHT can be seen as a subset of these that, in contrast, solely use the clients web browser as its application platform. As the DHT runs in the browser, there is no need for the client to install additional software and the DHT can become an transparent component of a website enabling a broad deployment. Though, there are drawbacks introduced by the web environment that have to be taken into account.

In contrast to a full fledged application that fulfills a specific task, browser windows (i.e. web sites) are opened/closed in a more frequent manner. Still depending heavily on the use case, this can result in low session times leading to higher churn rates. The Bamboo-DHT introduced

effective countermeasures, namely *static resilience* and *proactive failure recovery* which come at a price of increased management overhead respectively greater routing complexity.

Moreover, instead of using a simple TCP socket, further complexity is introduced by the WebRTC transport layer. How this layer affects the implementations' performance can hardly be estimated because important aspects of the specification regarding congestion control and prioritization mechanisms using the underlying SCTP layer are not finalized. This will be an important subject to our future research activities.

As elaborated on in this paper, implementing a DHT that performs good under any use case is not feasible. Tradeoffs have to be made in order to counteract increasing churn rates, different data layouts or geographical properties. The lessons learned from the design decisions analysis show that using an emulator as soon as possible in the design process to verify the performance criteria is a crucial requirement. Therefore, such an emulator that supports headless peer emulation (meaning that no actual browser window has to be running) and allows for conducting measurements based on the introduced metrics is currently under development.

4.2 Conclusion

This paper introduced the idea of a web-based DHT. The most important design decisions made by DHT designers and their effect on performance have been extensively studied. Each design decision introduced a use-case related tradeoff that has to be taken into account. Moreover, it has been shown how DHT implementations can counteract the decreased performance that a high churn rate imposes. Finally, emulation or simulation can help to spot potential bottlenecks by obtaining DHT specific, performance-related metrics.

This work laid the foundation for being able to plan a web-based DHT. As performance is a limiting factor of todays DHT implementations, this work leaves an uncertainness if a web-based DHT that has further limiting boundaries such as the WebRTC transport mechanism will perform sufficiently well. On the other hand this depends very much on the use case, respectively the performance requirements. The next step will be a DHT implementation that evolves by using metrics obtained by the emulator while testing the introduced design approaches.

Bibliography

[1] Alvestrand, Harald: Overview: Real Time Protocols for Brower-based Applications / IETF. June 2011 (01). – Internet-Draft – work in progress

[2] Castro, Miguel ; Druschel, Peter ; Hu, Y. C. ; Rowstron, Antony: Exploiting network proximity in distributed hash tables. In: *in International Workshop on Future Directions in Distributed Computing (FuDiCo)*, 2002, S. 52–55

[3] Dabek, Frank ; Cox, Russ ; Kaashoek, Frans ; Morris, Robert: Vivaldi: a decentralized network coordinate system. In: *SIGCOMM Comput. Commun. Rev.* 34 (2004), August, Nr. 4, S. 15–26. – URL http://doi.acm.org/10.1145/1030194.1015471. – ISSN 0146-4833

[4] Dabek, Frank ; Li, Jinyang ; Sit, Emil ; Robertson, James ; Kaashoek, M. F. ; Morris, Robert: Designing a DHT for low latency and high throughput. In: *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*. Berkeley, CA, USA : USENIX Association, 2004 (NSDI'04), S. 7–7. – URL http://dl.acm.org/citation.cfm?id=1251175.1251182

[5] Dabek, Frank ; Zhao, Ben Y. ; Druschel, Peter ; Kubiatowicz, John ; Stoica, Ion: Towards a Common API for Structured Peer-to-Peer Overlays. In: Kaashoek, M. F. (Hrsg.) ; Stoica, Ion (Hrsg.): *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22, 2003, Revised Papers* Bd. 2735. Berlin Heidelberg : Springer–Verlag, 2003, S. 33–44

[6] Jesup, Randell ; Loreto, Salvatore ; Tuexen, Michael: RTCWeb Data Channels / IETF. February 2013 (04). – Internet-Draft – work in progress

[7] Lua, Eng K. ; Crowcroft, J. ; Pias, M. ; Sharma, R. ; Lim, S.: A survey and comparison of peer-to-peer overlay network schemes. In: *Commun. Surveys Tuts.* 7 (2005), April, Nr. 2, S. 72–93. – URL http://dx.doi.org/10.1109/COMST.2005.1610546. – ISSN 1553-877X

[8] Ratnasamy, Sylvia ; Francis, Paul ; Handley, Mark ; Karp, Richard ; Shenker, Scott: A scalable content-addressable network. In: *SIGCOMM Comput. Commun. Rev.* 31 (2001), August, Nr. 4, S. 161–172. – URL http://doi.acm.org/10.1145/964723.383072. – ISSN 0146-4833

[9] Rhea, Sean ; Geels, Dennis ; Roscoe, Timothy ; Kubiatowicz, John: Handling churn in a DHT. In: *Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA : USENIX Association, 2004 (ATEC '04), S. 10–10

[10] Rowstron, Antony I. T. ; Druschel, Peter: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*. London, UK, UK : Springer-Verlag, 2001 (Middleware '01), S. 329–350. – URL http://dl.acm.org/citation.cfm?id=646591.697650. – ISBN 3-540-42800-3

[11] Stoica, Ion ; Morris, Robert ; Karger, David ; Kaashoek, M. F. ; Balakrishnan, Hari: Chord: A scalable peer-to-peer lookup service for internet applications. In: *SIGCOMM Comput. Commun. Rev.* 31 (2001), August, Nr. 4, S. 149–160. – URL http://doi.acm.org/10.1145/964723.383071. – ISSN 0146-4833

[12] Uberti, Justin ; Jennings, Cullen: Javascript Session Establishment Protocol / IETF. February 2013 (03). – Internet-Draft – work in progress