Bringing Name-based Publishing to the Web

Christian Vogt
HAW Hamburg
Departement of Computer Science
Hamburg University of Applied Sciences, Hamburg, Germany
christian.vogt@haw-hamburg.de

ABSTRACT

Name-based publishing decouples content from location by omitting host information in the identifiers and building name resolution into the underlying network itself. This concept allows for highly scalable and efficient dissemination of content in widely distributed systems and is one of the main building blocks in the vision of Information-centric Networking (ICN). This paper illustrates the motives behind naming content, compares different approaches and outlines implications on software architectures. While ICN remains a vision of the future, this paper additionally illuminates our approach to name-based publishing that is usable as of today. Brower-based Open Publishing (BOPlish) paves the way to the aforementioned communication concept by reducing the scale to designated Interest Groups instead of targeting the Internet as a whole.

Keywords

Naming Content, Information-centric Networking, Internet Architecture, Content Dissemination

1. INTRODUCTION

In recent years, our understanding off user-generated content has changed. Instead of residing on an individual's personal computer, people want to share their media and documents with other users and access them anywhere. The Internet allows for such an always-on access model and an increasing number of users move their content to cloud services on the Web. Such services include Dropbox, Flickr and Youtube, each having its distinct use case. While providing an inexpensive and easy way to handle user-generated content, these platforms suffer from privacy issues and lack interoperability with each other. Moreover, transferring content to such a platform also means handing trust and control over the content to the service provider.

Content on the Web is mostly addressed by HTTP URLs. Such URL always references a location denoted in the host part of the URL. This host-based approach reflects the current one-to-one Internet architecture but makes these URLs far from ideal for one-to-many communication. Cloud services use Content Delivery Networks (CDN) to distribute the load on their servers and be able to reduce RTDs and increase bandwidth when accessed from around the globe. As the host-based HTTP URLs do not support multiple locations of content, CDN providers redirect requests using modified DNS servers or HTTP redirects.

A novel approach to handling user-generated content is pursued by Information Centric Networking (ICN). ICNs core idea is to evolve the network architecture to a data-centric design and interpreting data as a first class citizen of the architecture. Different ICN proposals have been introduced that try to accomplish the task of creating a data-centric architecture. ICN uses names to identify content but does not include a location in the name. Instead, the network itself can resolve the name of the content to a beneficial location. By caching the requested content on the way from the source to the receiver, subsequent requests can be answered with low RTDs and high bandwidth. In this paper, we focus on the naming, name resolution and data routing aspects of the different proposals to gain knowledge that can be reused in our own approach.

Both approaches, using CDN or ICN for name-based publishing are problematic. CDNs are expensive and proprietary. Application providers have to depend on a centrally controlled, closed infrastructure as a vital component of their system. ICN, on the other hands, is transparently built into the network layer but is still in its infancy. It has unsolved problems [11] and its broad deployment is far off.

This paper gives an overview about current and future name-based publishing techniques (Section 2) and our own, novel approach that is called Browser-based Open Publishing (BOPlish) in Section 3. In Section 4, the resulting architecture is evaluated and the paper is concluded with an outlook.

2. PROBLEM STATEMENT AND RELATED WORK

To showcase the problem of host-dependent content, consider a simple web site. When a browser is pointed to a URL such as `http://www.nytimes.com`, it first resolves the DNS name to an IP address of a single host (one of its DNS A records; `170.149.168.130` in this case). Afterwards, it downloads the HTML code from that source by using a HTTP GET request. The HTML may include references to content such as pictures (see Figure 1) which are subsequently downloaded from the initial host address.

```
<img src="/images/image.png" />
```

Figure 1: HTML snippet that mandates the browser to request additional content from the current domain's webserver

Now, maybe the web site is owned by an international service provider and gets requested from all around the world. Until now, we are requesting content from a single source, which in our case of `nytimes.com` is registered in NYC, New York. This is problematic from many points of view: a) The web site owner needs a beefy machine and/or internal load balancing to handle all the content requests b) Client response times from a geographical far place will be very high due to long routes c) Unnecessary load is put on the Internet's core routers when multiple clients request the same content which gets repeatedly routed all over the world.

An issue resides in the address that is used to reference resources on the Web. HTTP uses a URI scheme that denotes a host name (that resolves to an ip address via DNS) of a single location. Thus, by using these URLs we introduce a binding between the name of the content and its location. As a result, content that is available at multiple locations has to have multiple distinct URLs. Moreover, if the location changes, accessing the content becomes impossible until the DNS A records are changed to the new location. Such a change takes hours or even days to be propagated through the DNS infrastructure.

2.1 CDN Approaches

Content Delivery Networks (CDN) aim to provide high volume content to the network edges by using widely distributed infrastructure in order to optimize performance and/or save bandwidth. The content is replicated to a big number of servers (so called *surrogates*) that are spread out to reach network proximity to a large number of clients. CDNs are typically operated by large companies (e.g. Akamai [7]) that only provide their service for a considerable amount of money. In order for a CDN to work, the requesting client has to be redirected to a nearby surrogate. As of today, two approaches are widely used to do so, namely DNS- and HTTP-based redirection.

2.1.1 DNS-based Redirection

Every valid fully qualified domain name (FQDN) on the Internet can be mapped to a authoritative DNS server. With DNS-based redirections, DNS is configured to delegate name resolution to a server managed by the CDN provider. The provider then uses policies and/or metrics like geolocation and delay measurements to determine a nearby surrogate. This approach is easy to implement since only the DNS configuration is changed. On the other hand, the approach has a site-wide scope, meaning that the web site owner cannot pick the content that is to be served by the CDN. A common use case is to only serve big, static content (like images and videos) over the CDN and serve dynamic content directly. Thus, such a per object resolution is not possible using DNS-based redirections.

2.1.2 HTTP-based Redirection

Another mechanism that is widely used by CDN providers as of today is HTTP-based redirection. The authoritative DNS server of the domain that is to be served through the CDN is not changed, but the HTML code used to reference external resources. Considering the example from above, the delivered content is rewritten as shown in fig 2.

```
<img src="http://cdn.org/xyz/images/image.png" />
```

Figure 2: HTML snippet that mandates the browser to request additional content from an external domain `cdn.org`

The host name `cdn.org` now refers to the CDN provider which, like with DNS-based redirection, uses custom authoritative DNS servers to redirect the `cdn.org` requests to a nearby surrogate. This approach allows for finer granularity of the request routing mechanism as the website owner can control which resources are served by the CDN. On the other hand, it requires a tedious amount of work to change all resource URLs of the domain and map them to the respective CDN URL.

2.1.3 Summary of CDN approaches

We have seen two different mechanisms for content distribution that have a common goal: Circumvent the location-dependence of HTTP URLs. As of today, CDNs are widely deployed and make up an increasing portion of the Internet's traffic. Cisco even expects that CDNs will carry over half of the Internet's traffic in 2017[1].

Because classic CDNs require a huge amount of widely distributed infrastructure, they are typically owned by big companies that sell their content delivery service to customers that can afford it. A few recent approaches to so-called hybrid CDNs try to use the requesting clients of the CDN as surrogate servers in a P2P manor to reduce the need in self-owned infrastructure. This in turn can lead to reduced costs for the CDN clients. Most hybrid CDNs make use of HTTP-based redirection as explained above (e.g. [1]).

2.2 ICN approaches

Todays Internet architecture is founded on a host-centric approach which was designed for one-to-one communication. However, as the rise of CDNs indicates, todays Internet services mirror one-to-many communication for which the architecture has not been built for [4]. ICN is a revolutionary vision to build a clean slate architecture that better fits group communication claims and treats information (instead of location) as the main building block [5]. This is achieved by allowing content receivers to ask for information, not for location. The underlying network layer is capable of directing the request to a location completely hidden from the requesting client. As a result, the location of data becomes irrelevant, making it simple to introduce caches distributed throughout the network. Many architectures have been introduced, the most prominent being DONA [6] and NDN [5]. The approaches have a common goal but differ largely in their data naming schemes and the name resolution of these identifiers. Moreover, the naming scheme also dictates how the data routing works [3]. We will now take a closer look at these differences. For a deeper exploration of the different ICN architectures, [12] gives an in-depth summary.

2.2.1 Naming

By using HTTP, we can address content that resides on a remote host. In ICN, there is no distinct host we can connect to. Therefore, the identifiers change to not denote a

[1]according to Cisco's VNI forecast

host anymore (like in HTTP) but to name the content itself. Two rivaling approaches exist on how to do that, either use hierarchical or flat (e.g. a hash) names (Figure 3).

```
ndn://alice/images/image.png
dona://134(...)0f6:dfe(...)164
```

Figure 3: NDN hierarchical identifier and DONA flat identifier

Hierarchical names have the benefit that they can easily be aggregated. When considering that the routers use the names of the content to decide on the next hop, it is preferable to reduce routing table sizes by aggregating names. NDN uses such hierarchical names and even requires excessive aggregation to allow for reasonable sized routing tables. Aggregation could e.g. be performed at the ISP level (with ISPs assigning prefixes to their customers) but this reintroduces a binding to location. The existence of the location-identity binding is the main argument for flat names (as used in DONA), which allow for a complete decoupling of location and identity but cannot easily be aggregated. Coping with a huge amount of unaggregatable identifiers requires either huge routing tables or external infrastructure. Finding a scheme that allows for both, effective aggregation and location-independence of the system without bloating routing tables is still subject to research activities [3].

Another aspect of the debate between flat and hierarchical names is the decision between human-readableness and self-certification. While DONA can use a cryptographic hash of the content as its identifier and thus offer implicit content certification, NDN has to use an external trust mechanism.

2.2.2 Name Resolution and Data Routing

In order to find a specific piece of content, ICN uses name resolution to find a location just like CDNs use DNS to find a nearby surrogate. When a location of the content is found, it has to be transferred to the requesting entity. Therefore, data routing is used to find a path over which the actual content is transferred. Depending on the ICN implementation, data routing and name resolution can be coupled (e.g. NDN) or decoupled (e.g. DONA). In a coupled approach, the data routing follows the reverse path of the path found by the name resolution. In a decoupled approach, the data routing is independent of any previously found paths.

Coupling the data routing means to either a) store routing states in the intermediate hops traveled by the name resolution query or b) integrate this information into the content query packets on the way. Decoupled approaches allow for more flexibility as control and data flow can be separated.

2.3 Providing a Path to Name-based Publishing

This section described two contrasting approaches, namely CDN and ICN, that share the same goal: Mitigating the fact that the Internet evolved from a host-centric to a content-centric platform. To cope with the new requirements, CDNs introduce a widely deployed, proprietary infrastructure and operate on the application layer. ICN visions an architecture that centers around the content itself making efficient group communication possible by routing on names instead of locators. As of today, CDNs are widely deployed while ICN remains a concept of the future with many open issues.

This results in a chicken-and-egg problem as CDNs already seem to offer what is promised by ICN: An efficient publishing infrastructure. CDNs, though, are neither openly available nor do they allow for user-generated content. Moreover, because CDNs operate on the application layer and are mostly proprietary, sharing content between web applications or even different CDNs requires service-specific APIs. In a ICN-world, sharing content between services is implicitly achieved because the network itself allows for the required interoperability.

In the next chapter, BOPlish is introduced. BOPlish allows clients to join an overlay network over which content can be distributed. The content is named similar to the ICN approaches – without a location property in the identifiers. With the classical CDN approaches, users do not directly publish content. Instead, only the service providers are allowed to access the CDN-specific APIs. As a result, users do not have any direct control over their content. In BOPlish, we try to mitigate that fact by giving control back into the users hand. As such, BOPlish focuses on user-generated data meaning that every consumer can also be a publisher (so called *prosumer*). BOPlish tries to counter the the currently observed shift to only use a few, very large service providers such as Google, Facebook etc. by offering an open, decentralized publishing infrastructure.

Just like CDNs, BOPlish operates on the application layer which enables wide deployment as of today. Instead of operating on an Internet-wide scale like ICN does, we limit the scope of the content publication to interest groups of specific content.

3. BOTTOM-UP APPROACH TO NAME-BASED PUBLISHING

The BOPlish content sharing facility consists of a JavaScript library that can be included in todays web applications either by running directly in the browser or on a server using a JavaScript runtime environment like Node.js. A potential user navigates to a web page and automatically joins the content publishing infrastructure. After the user has joined the overlay network, he can request content or publish content himself. It is important to note that the overlay can be spanned across web sites, as such a user that joined from `example.org` can communicate with a user from `example.com`. This allows for a freely, domain independent content distribution which is not tied to specific services. BOPlish uses WebRTC as its transport mechanism, allowing for point-to-point connections between the client's browsers. In this section, we will go through the different aspects of the architecture by focussing on the content naming scheme, the name resolution facility and the data routing as introduced by ICN in Section 2.2. The BOPlish architecture described in this section is subject to ongoing work and shall provide an overview about the similarities to other approaches (i.e. CDN and ICN).

Figure 4 shows an overview of the current BOPlish architecture further described in [10], [9]. It consists of three components: a) The Name Resolver API that uses a dis-

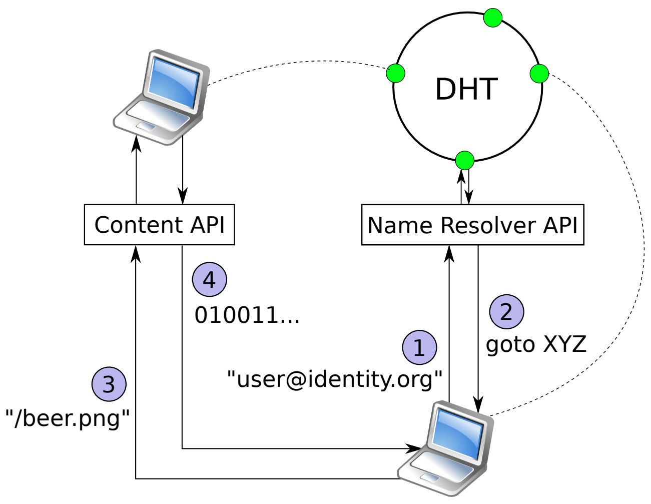

Figure 4: Nodes in BOPlish retrieve content by issuing a lookup of the content's user ID to the underlying DHT (1) which returns a pointer to the actual node that holds the content (2). This pointer is then used to open a WebRTC Data Channel to the peer, query for the content (3) and transfer it (4) [9]

tributed hash table (DHT) to allow for the resolution of location-independent identifiers; b) The Content API that can be queried by a remote host to access the content that is announced by the publisher; c) A bootstrap component that allows a new user to join the existing network (not shown in Figure 4).

3.1 Naming Content in BOPlish

In order to give names to content, we use the URI scheme shown in Figure 5 (a). The significant difference to HTTP URLs (or any other location-based URIs) described above is, that the namespace of the URI does not denote a location (like in URLs) but an location-less identity. Figure 5 (b) shows how BOPlish URIs could look like. `alice` denotes an identity that can be fed into the name resolution system. It is important to note that even if the content location changes, this URI will stay the same.

```
(a) bop:namespace:protocol
     [/protocol-specific[?parameters]]
(b) bop:alice:document/img/images.png?
     csum=sha1:1234abc...
```

Figure 5: BOPlish URI syntax (a) and example BOPlish URIs for file transfer (b)

Besides the location property, the content name denotes a file-system like path that corresponds to a specific piece of content. Content is not restricted to files, it is easily possible for a user to publish an API that allows for other use cases (e.g. chat, media streaming etc.). Moreover, the path may contain wildcards that allow for searches and aggregation of content. The query-part of the URL optionally denotes parameters which might be used to verify the integrity and authenticity of the received content.

3.2 Name Resolution and Data Routing in BOPlish

We identified name resolution as a technique of ICN approaches. Instead of operating on the network layer like ICN, we introduce an overlay network which is capable of unbinding the relation between location and content identifier. This mechanism is realized by using a DHT which uses a hash of the identifiers as key and returns a pointer to the node that holds the content as value. This indirection allows the system to handle names and locations separately which we identified as a requirement for a content-centric architecture above. Because only the name-location resolution depends on a DHT and we limit the scale to specific interest groups instead of the Internet as a whole, the DHT can be designed to be highly churn-resistant and redundant. This is a crucial requirement as DHT implementations tend to be fragile when peers join/leave the network in a high frequency [8].

Data Routing in the BOPlish architecture is decoupled from the name resolution. Instead of using the reverse path of the name resolution, a WebRTC connection between the content receiver and one or more of the publishers is opened. Coupling the data routing with the name resolution is also possible but routing the content through the DHT would impose unnecessary load, leading to poor performance regarding the name lookup. Moreover, depending on the DHT implementation, the overlay path can be disadvantageous because it is not aware of geographical and performance properties of the overlay hops. The reference to a location is obtained by using the return value of the DHT name resolution procedure. If the connection to the publisher fails, the content receiver can always re-query the DHT to find the updated location information. This allows for mobility of both, the content receiver and the publisher because the DHT entry can easily be updated without requiring a name change of the content's identifier.

3.3 Enabling Group Communication

When recalling the initial motivation of a user-centric content publishing facility, the described architecture lacks support for group communication means. Currently, content is transferred point-to-point from user to user. ICN introduces caches distributed throughout the network to allow for efficient group communication. Due to the flexible naming scheme and resolution, it would also be feasible in BOPlish to replicate and serve content from multiple peers. A prominent approach to such large-scale distributed applications is the publish/subscribe communication paradigm. [2] factor out the requirements of such a system: The decoupling of the communicating entities in time, space, and synchronization. In order to fulfill the requirements, the data structure used by the name resolver is extended as shown in Table 1.

Key	**Value**	
	Publishers	**Subscribers**
$h(alice)$	$\mapsto [CurrentPubs]$	$\mapsto [(CurrentSubs., RequestedURI)]$

Table 1: Extended name resolution data structure

As the name resolution is realized as a DHT and therefore distributed among the participating peers, it can be viewed as a persistent storage entity. Even when no publisher is available, subscribers can issue an interest in the content by adding themselves to the appropriate DHT entry. Conversely, the subscriber can get notified by the name resolution service when a publisher is available. Thus, a *time decoupling* is feasible. *Space decoupling* in BOPlish is possible by letting the name resolution service select the appropriate publisher for a requesting subscriber. As such, producers and consumers both have a limited view upon the current state instead of requiring full knowledge of each other. Due to the characteristics of the underlying WebRTC Data Channel implementation in JavaScript, *synchronization decoupling* is implicitly achieved as all incoming/outgoing messages are processed asynchronously.

Extending the data scheme that is used by the name resolver results in a flexible approach that fulfills the basic requirements for a pub/sub system introduced by [2]. On the other hand, load is burdened upon DHT because updates to the hash table's entries have to be made for each participating subscriber.

4. CONCLUSIONS AND OUTLOOK

This paper showed different approaches to publishing content on the Web, namely CDN, ICN and BOPlish. CDNs are widely used as of today but suffer from their centralized architecture, allowing a few large companies to be in charge of a huge portion of the Internets traffic. ICN-like architectures are far away on the horizon to evolve not only the Web but the whole Internet from a host- to a content-centric network infrastructure. BOPlish provides a path to ICN-like name-based publishing by introducing an overlay which does not depend on third-party infrastructure like CDNs does. Instead of requiring deep, network-layer changes to the infrastructure like ICN, BOPlish can be used as of today. By utilizing the Web platform, current web applications can easily be extended by a name-based content sharing facility that also allows to distribute content among users of multiple websites. This allows for a new type of open interoperability between web applications that is not feasible with today's infrastructure.

BOPlish focusses on user-generated content. The vision of such an architecture differs from todays use of the Internet. Instead of handing over the content to the service provider (or the CDN provider), users keep control over the content shared within a BOPlish interest group. The crucial difference to todays services is, that the service does not need to hold on to the content itself but accesses it on demand. While this allows for the introduced service interoperability, downsides can be identified, too. The user (instead of the service provider) has to make sure that a copy of the content stays available in the BOPlish network. As the BOPlish library is not limited to run in the browser, it could potentially also be run on NAS-like devices that provide the necessary availability for the specific kind of content.

We have already made good progress on the implementation details and implemented demo applications that utilize the core BOPlish functionality. Still, open issues remain that have not been dealt with. Besides the common P2P security topics, the concrete DHT implementation is an important issue for the name resolution performance and therefore the whole system. As DHTs tend to be fragile in regards to performance, it has to be carefully designed. Measurements of a larger scale have to be conducted to detect possible shortcomings of the BOPlish architecture.

5. REFERENCES

[1] M. El Dick, E. Pacitti, and B. Kemme. Flower-CDN: a hybrid P2P overlay for efficient query processing in CDN. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 427–438, New York, NY, USA, 2009. ACM.

[2] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Comput. Surv.*, 35(2):114–131, June 2003.

[3] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in Content-oriented Architectures. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ICN '11, pages 1–6, New York, NY, USA, 2011. ACM.

[4] M. Handley. Why the Internet Only Just Works. *BT Technology Journal*, 24(3):119–129, July 2006.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, and M. F. Plass. Networking Named Content. In *Proc. of the 5th Int. Conf. on emerging Networking EXperiments and Technologies (ACM CoNEXT'09)*, pages 1–12, New York, NY, USA, Dec. 2009. ACM.

[6] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A Data-Oriented (and beyond) Network Architecture. *SIGCOMM Computer Communications Review*, 37(4):181–192, 2007.

[7] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai Network: A Platform for High-performance Internet Applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, Aug. 2010.

[8] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[9] C. Vogt, M. J. Werner, and T. C. Schmidt. Content-centric User Networks: WebRTC as a Path to Name-based Publishing. In *21st IEEE Intern. Conf. on Network Protocols (ICNP 2013), PhD Forum*, Piscataway, NJ, USA, Oct. 2013. IEEEPress.

[10] C. Vogt, M. J. Werner, and T. C. Schmidt. Leveraging WebRTC for P2P Content Distribution in Web Browsers. In *21st IEEE Intern. Conf. on Network Protocols (ICNP 2013), Demo Session*, Piscataway, NJ, USA, Oct. 2013. IEEEPress. ICNP Best Demo Award.

[11] M. Wählisch, T. C. Schmidt, and M. Vahlenkamp. Backscatter from the Data Plane – Threats to Stability and Security in Information-Centric Network Infrastructure. *Computer Networks*, 57(16):3192–3206, Nov. 2013.

[12] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos. A Survey of Information-Centric Networking Research. *Communications Surveys Tutorials, IEEE*, PP(99):1–26, 2013.