Taking advantage of topology info in worldwide content networks

adolfomaria.rosasgomez@telefonica.com

March 2014

1 INTRODUCTION

A worldwide content network is nothing easy to build and run. Even though today the basics of CDNs have been established and many different CDNs are out there serving bytes from many thousands of happy publishers to millions of happy end users, very few commercial CDNs can claim to be 'worldwide CDNs'. I can only think of one, maybe two.

There are more private-purpose content networks with worldwide reach than pure-play CDNs or even commercial CDNs with worldwide reach. Google, Netflix, Microsoft, Yahoo… all of them run worldwide networks with private infrastructure for content delivery. Some Telco: Comcast, AT&T, KPN, Telefonica… have built their own internal/commercial CDNs. Some of these Telco CDNs have a wide coverage but very few if any can claim to be global. If we classify a CDN as 'global' in case it works as a single content network capable of moving any content to any PoP, capable of consolidated reporting about the whole network, running content routing for the whole network and having PoPs in at least three continents … then only one of the above mentioned Telco CDNs qualifies while all the private CDNs mentioned qualify.

The special category of networks that I'm talking about, have problems of their own. Size matters and running content routing for a network with PoPs in three continents poses some interesting problems about content instantiation, request routing, requester geo-location and the influence of internet and CDN topology.

Does the CDN owner have control of packet routing through its network? The answer is 'NO' in most of the cases. Most overlay owners are NOT network owners. The network owners are the Telcos. Creating an overlay on top of internet is not the same thing that creating an internet backbone. The raw network offers a single service to the content network: connection point to point, but it never promises to route packets through any specific path or to conserve a bounded latency and of course forget about having a regular traffic shaping. No. Most overlays can only ensure communication end to end, period. If we think twice about this statement, then, which is the value of the overlay? Internet is an IP network, packet communication end to end is granted if you know the addresses of both endpoints so… wouldn't you expect overlays to give you good things beyond ensuring end to end connection? Yes every one of us should expect more than connectivity from a content network.

I will explore in this article technology and contributions from CDN space to worldwide-reach network management. How to handle worldwide content routing? How to ensure good Quality of Experience? Which are the challenges? Which are the solutions? Are current CDNs using the best technology? Are current CDNs active innovators? Do they offer significant value on top of internet?

Especially important is to realize that Internet is a patchwork of weaved networks. The owners of the different patches are Telcos and they have plenty of information about their property. Are Telcos leveraging topology info? Are the other stakeholders able to benefit from Telco's topology knowledge?

2 CREATING VALUE THROUGH OVERLAYS AND CONTENT NETWORKS

So, how can a content overlay add value to networking? (… compared to the raw Internet).

There are several ways:

-providing content discovery, a content directory (list of references) and a mapping to content instances, faster than internet DNS: affecting response latency. This 'responsiveness' is key in content services today.

-providing storage that is closer, bigger, more reliable than publisher site: affecting quality of experience (QoE) for consumer's delivery and scaling up the number of consumers without hampering that QoE.

-providing security against Denial of Service, content tampering, content robbery: affecting QoE for publishers and consumers thus supporting content business, making it possible, making it profitable.

-changing packet routing to optimize content routing: affecting average end to end latency in backbones and affecting backbone and link reliability. This is something that only router owners can do, so not every overlay is in the position to add this value. It is complex. It is risky. It may affect other services. But it may be extremely cost effective if done properly.

Most CDNs, no matter if they were built for private exploitation or open commercialization, just rely on internet naming system (DNS) as a means to content discovery. This leads to fundamental shortcomings in performance as DNS is a collaborative distributed system in which intermediate information is 'cached' for various TTL (Time To Live) in various places so end-to-end there are several players affecting latency and they do not work coordinated with a goal in mind. They do not treat content differently from any other named object. And most important they do not share the same policies. All in all we should be grateful that DNS works at all, being an impressive feat that we can retrieve any object from internet in a 'reasonable time'. But if you design your own content network, no matter whether you put it on top of internet or you build a huge private network, please start thinking of a different, better way to discover content.

Alternate approaches to content discovery.

This lesson has been understood by p2p networks. They probably didn't do the discovery for performance reasons. Most probably they didn't want to use open URIs or URLs as their activities were bordering law. So they created private names, new URIs, and created directory services. These directory services work as fuzzy naming systems; you query the directory with 'keys' and the directory responds with a list of URIs. In P2P networks these URIs are massively replicated throughout the peers, so anyone that has stored a fragment of a title referred by a certain URI is a candidate to provide bytes from that content to others. P2P networks have their own algorithms to seek peers that have instances of a given URI. Please notice that in this 'P2P world' a single content is represented by a URI, but that URI, that 'name', does not represent a single instance of the file. The URI represents ALL instances. So the P2P network has an additional task that is to keep track of available instances of a URI in the neighborhood of a requester in near real time. There are a lot of interesting hints and clues for future networking in P2P, but let's not think about all these fantastic ideas now and just focus on the directory service. There are even distributed directories like the network named Chord or the network named Kademlia.

P2P organization of a content network maybe too aggressive compared to the classical client-server approach. It is difficult to run a business based on fully open P2P cooperation, as there must be responsibility and maintenance of network nodes and everyone wants to know that there is someone who receives payment and has the responsibility to run the nodes. Responsibility cannot be distributed if the network must be commercially exploited. Today there exist P2P commercial content networks (Octoshape) but they are P2P only at machine level as they run central policies and they own all the SW of the peers and have the full responsibility of network behavior. Remember: responsibility cannot be distributed.

Content instantiation and routing strategies

So let's say that we have a classical approach to content networking, closer to client-server than to P2P. Let's say that our network is huge in number of nodes and in locations (a truly global network as defined in our introduction to this article). How do we map requests to content instances? There are strategies that we can follow.

-Forward-replicate (pre-position) all content titles in all PoPs. This sounds expensive doesn't it? Yes it is too expensive. Today CDN PoPs are meant to be close to the requester (consumer), for this reason there are many PoPs and in a truly global network PoPs may form an extremely distributed set with some PoPs far apart from others. In this schema putting all titles from your customers in all places (PoPs) is plainly impossible. Either you have very few customers, or they have very few titles or both things at the same time which doesn't look like good business for any CDN. The reasonable approach to this strategy is creating a few, distributed 'BIG Repositories' and mapping the PoPs requests to them. In this approach a request will be mapped to a local PoP, in case of miss the PoP will go to the closest repository which is BIG and, if you have a policy of forward replication of new titles to Repositories, there are high chances that the title is just there. In the (rare) case of a miss you can forward your query to the very source of the title: your customer.

-Do not forward-replicate ANY title. Just do plain caching. Let your PoPs and caches at the edge collect queries, run a pure location-based request-routing mapping each query to the closest cache/PoP and let the cache go to title source (customer) and thus get 'impregnated' with that title for some time (until eviction is needed to use space for another title).

The above mentioned two strategies have important differences that go beyond content discovery latency. These differences affect the way you build your request routing. It is now an A-priori to decide if you will be proactive specializing some PoP/cache in some content and thus route every request , even far ones to the place content is, or on the contrary let the content map to the caches following demand. It is an important choice, it makes all the difference between long-tail and mega-hit distribution.

Dynamic cache specialization: URL hashing

Some authors are confident that the most dynamic approach can work: use URL hashes to specialize caches in some URLs (and thus some content titles) but let caches be filled with content dynamically as demand of titles evolves regionally. This strategy 'sounds reasonable'. It could be easily implemented by assigning a content router to some region and configure it to apply a hash to every title URL to be distributed in that region in a way that the whole URL space is distributed evenly over cache/PoP space. I've said that 'sounds reasonable' but…what about the real world performance of such strategy? Does it really work? Does it adapt

well to real world demand? Does it adapt well to real world backbone topology? Unfortunately the responses are No, No and No.

In the real world content demand distribution over the title space is not flat. Some contents are much more popular than others. If you prepare your hashes for a flat, even distribution over all caches you'll be wasting some precious resources in 'long tail' titles while at the same time your 'mega-hits' are not given enough machines to live in. So the hash function must be tweaked to map popular titles to a bigger spot than long tail titles. You can start to see the difficulty: we do not know popularity A-priori, and popularity changes over time…so we would need a parameterized hash that changes the size of hash classes dynamically. This represents another efficiency problem as we cannot let the hash classes move from one machine to another wildly… we need to capitalize on content being cached in the same machine for some time. We have to balance popularity-change-rate against cache-efficiency so we can adapt reasonably well to the change in popularity of a title without destroying cache efficiency by remapping the hash class too often.

3 LEVERAGING NETWORK INFORMATION

Influence of current network technology in the quality of experience

In the real world backbone topology and access network topology in metropolitan areas could create you serious problems. In any typical big city you'll see something like a BRAS (Broadband Access server) per 50000-150000 subscribers. About 10-15 DSLAMs per BRAS, which is 5000 to 10000 subscribers per DSLAM. BRASes are joined together through a metro LAN, most likely using n-10Gbps links: n=1,n=2,n=3. At some point this metro LAN reaches a transit datacenter in which a BGP router links that region to other regions (cities/metro areas) probably using n-10Gbps links or 40-Gbps links (still not common) and running IGP. The transit site may or may not be connected to other ISPs/international trunks through EGP. Today DSLAMs are being slowly phased out and replaced by optical multiplexers. One side being IP over optical in a big trunk, the other side a PON that replaces the DSLAM subscriber area. This change improves dramatically latency from end user to IP multiplexer (the biggest drawback of DSL technology) but it does not increase much the IP switch bandwidth or density of access lines, as these numbers impact directly on the cost of the switch backplane electronics and that part is not greatly influenced by optical technology.

Network technology latency watermark

The deployment of fiber (PON) is enhancing the QoE of content services dramatically by reducing latency. As we have noted, latency to receive the requested content is key in the experience and using previous technology (DSL) there was a practical limit in transmission due to multilevel modulation and interleaving in DSL. If you have access to worldwide data about CDN latency (some companies provide these data to compare CDNs performance all over the world) you can track the progress of PON deployment worldwide. In a country with most lines DSL a CDN cannot perform in average better than 50ms for query latency. In a country with a majority of fiber accesses the best CDN can go down to 10-20ms. This is what I call the 'network technology latency watermark' of a country's network. Of course these 50ms or 20ms are average query-response times. The average is calculated for many queries; most of them with cached DNS resolutions (depending on TTL values), so impact of DNS resolution on these averages is low. The highest impact comes from the 'network technology latency watermark' of the infrastructure. Round trip time (RTT)

on DSL is 50-60 ms. RTT on fiber is 5 ms. RTT in any place in the world today is a mix… but what if you know where is fiber deployed and where not? You could plan ahead your content distribution using this knowledge.

As a more general statement, you can take content-routing choices based on ALL your knowledge about the network. These choices may then follow a fairly complex set of rules involving business variables, time dependent conditions and last but not least low-level network status. Packet routers cannot apply these rules. You need to have applications imposing these choices in real time on your content network.

Topology knowledge: representation and application

The classic traffic engineering is based on a probabilistic model of links and switches (a large set of interconnected queues). If you were to instantiate copies of content in some parts of the network and forward queries to concrete machines you will need a proper way to represent ALL the knowledge that you can possibly gather about your own network and maybe other networks outside of yours. This knowledge representation should be appropriate to write routing actions based on it and flexible to match business criteria. These actions should be at the end decomposed into packet routing actions so the content routing agent will ideally interface to the network routers giving them precise instructions about how to route the traffic for a specific consumer (query) and a specific instance of a title (a named copy). This interface content-router to network-router is just not there… It is not possible to communicate routing instructions to OSPF/ISIS routers. It is much more possible to communicate to BGP routers, to private enhanced DNSs, and recently through the use of OpenFlow it starts to happen that a content-router can control the forwarding engine of multiple network-routers that implement OpenFlow. But this last option even being very powerful is very recent and only a few large scale deployments exist today (Google is the most significant).

Recently several initiatives have appeared to tackle this topology handling problem. The most famous maybe P4P and ALTO. P4P was born in the P2P academic community to give network owners the opportunity to leverage topology plus performance knowledge about the network without compromising business advantage or revealing secrets or weakening security. ALTO is the IETF counterpart of P4P.

Knowing the exact topology and detailed link performance of any internet trunk may give opportunities to application owners and other Telcos to perform their own traffic engineering tuned to their benefit, so you can understand that Telcos are not willing to provide detailed info to anyone even if they offer to pay for it.

P4P and ALTOS are 'network information briefing techniques' that concentrate in the network information that is relevant for the design of overlays over wide area networks. The goal of the overlay is usually to maximize spread and quality of a service while minimizing cost. To pursue this goal it would be good to know 'where' the 'peers' or 'nodes' are in the world and 'how costly' it is to send a message from one node to another. Both techniques create a partition of the world (as we deal with internet notice that the world is just the space of all IP addresses). This partition creates a collection of classes or subsets of IP addresses. Each class joins IP addresses that are equal/close using a certain partition criterion. Each class is identified with a number: Partition ID (PID). If we select a proper partition criterion, for instance: two IPs lay in the same partition class if their access lines go to the same BRAS, (or to the same BGP router),(or lay in the same AS), etc… then we will break the whole IP space in a number of classes and the 'cost function' between any two IP addresses can be roughly approximated by a simplified cost function between their classes.

So a convenient way to resume a whole network is to provide a description of classes: a list of all IP addresses that lay in each class (or a rule to know class for every IP), and a cost function class-to-class.

Armed with these two objects : IP to PID mapping and PID to PID square cost matrix, a content router agent may take sophisticated routing decisions that later can be turned into policies for BGP or DNS rules or OpenFlow directives.

4 REQUEST ROUTING

We need to decide, for a given region (a subset of IP addresses from the whole IP space), which query (a pair IP-URL) we map to which cache/streamer (IP). This decision making process happens at the regional request router. Using today's technology the content router can only be a server working at application level (OSI level 7), as far as there are no content oriented network mechanisms integrated in any available network technology. With the evolution of HTTP and OpenFlow we may find changes to this in the coming years.

(IPreq+URLtit) → IPca is our 'request routing function'. What parameters do we have to build this function?

-IPreq geo position (UTM)

-IPreq topology: AS number, BGP router ID, topology partition class (PID). Partitions are a good way of briefing all possible knowledge about topology.

-URLtit, do we know which caches currently have a copy of URLtit? Let's assume that we know exactly that. We may not have a list of URLs mapped to each IPca (each cache) but we can have for example a hash. The simplest form is: if we have N caches we can use the less significant m bits of URL, where 2^m=<N<2^(m+1). In case N=2^m we are done. In case 2^m<N we need to 'wrap the remainder up'. This means that the cache ID= less significant m bits, but if the resulting ID is bigger than the highest cache ID we obtain a new ID= ID-2^(m-1). This is simple but unfair for some caches that receive double requests than others. We can do better with a proper hash that maps any URL to N caches. But bear in mind that it is difficult to build good hashes (hashes with good class separation) without using congruence in an evident manner. Be careful with uneven hashes.

-Cost info: after partitioning: class to class distance. Usually this info can be represented by a matrix. In the P4P or ALTOS they do not specify any way to represent this distance. A convenient extension to the partitioning proposed there may be:

> .have a full-IP-space partition for each region. Extend the information of a region partition by adding the necessary classes out of the region to represent distance to other regions until we have a full-IP-space partition.
>
> .have a full-meshed cost matrix for each region: a square cost matrix class-to-class in every region.

It is very straightforward to program a content router with sophisticated rules once we have the partition ready for a region. What is really hard is to prepare for the first time and later maintain up to date the partition.

How to build a partition from classical routing info + other info

The classical routing in a worldwide network comprises BGP routes + local routes. Due to the current state of technology it can be usually appropriate to set a content router per BGP router as a maximum. It is usually not needed to handle sub-BGP areas with several content routers, and we could even use a single content router for a whole AS with several BGP routers that speak EGP. When partitioning the area covered by a BGP router we can use the routing table of the BGP router to identify prefixes (sub-networks) reachable from that router and we can combine this routing table with RADIUS services to map the IP addresses reachable by that router to one or more BRASes or DSLAMs (or equivalent multiplexer in PON). Usually BRASes speak BGP to other BRASes so many times we can just break down the BRAS area to DSLAM areas and that is not bad as a starting partition. Of course dynamic IP assignment can complicate things, but as far as the IP pool is the same for the same DSLAM/multiplexer it doesn't matter much that the IP addresses move from one access line to another in the same DSLAM/multiplexer.

We can refine partition over time by handling RADIUS info frequently to manage changes in the pool.

We can also run off-line processes that correlate the geo-positioning info that comes from any good global service or available terminal GPS to the partitioning done from RADIUS and BGP tables.

Also by checking frequently the BGP updates we can correct the cost matrix in case a link is broken or weakened. That effect can be detected and reflected in the cost matrix.

5 CONCLUSIONS

Maybe P2P academics and IETF never achieved their goals of making Telcos cooperate to overlays by providing topology insights through standard interfaces: P4P/ALTOS.

Anyway partitioning has demonstrated to be a good abstraction to put together a lot of useful information about the network : classical routing info both local and AS to AS, performance of routes summarized at class level, economics of routes in case some class to class happens through a peering.

Partitioning is practical today just using BGP info but it is hard to bootstrap and to maintain. We need a lot of computing power to mine down the BGP tables and RADIUS pools, and cannot refresh partitions and cost matrices too often, but it can be done fast enough for today's applications.

In the immediate future, as more wide area networks embrace OpenFlow we could get rid of BGP tables processing and RADIUS pools analysis. An OpenFlow compliant network can have central routing planners or regional routing planners and the routing info maybe exposed through standard interfaces and will be homogeneous for all routing devices in the network. It is just interesting to be here waiting for the first OpenFlow CDN implementation. Probably it has happened now and it is a private CDN, maybe one of the big ones that I mentioned that are non-commercial. In case it has not happened yet it will happen soon. In the meantime we can only guess how good is the implementation of the request routing function in our current CDNs. Would it be based on partitioning?