Some thoughts about CDNs immediate future and Internet evolution.

Adolfomaria.rosasgomez@telefonica.com

Madrid, February 2014.

1. INTRODUCTION

A CDN (Content delivery Network) is a Network overlaid on top of internet. Why bother to put another network on top of internet? Answer is easy: the Internet as of today does not work well for doing certain things, for instance content services for today's content types. Any CDN that ever existed was just intended to improve the behaviour of the underlying network in some very specific cases: 'some services' (content services for example), for 'some users' (those who pay, or at least those whom someone pays for). CDNs do not want nor can improve Internet as a whole.

Internet is just yet another IP network combined with some basic services, for instance: 'object names' translation into 'network addresses' (network names): DNS. Internet's 'service model' is multi-tenant, collaborative, non-managed, and 'open' opposite to private networks (single owner), joined to standards that may vary one from another, non-collaborative (though they may peer and do business at some points) and managed. It is now accepted that the 'service model' of Internet, is not optimal for some things: secure transactions, real time communications and uninterrupted access to really big objects (coherent sustained flows)…

The service model in a network of the likes of Internet , so little managed, so little centralized, with so many 'open' contributions, today can grant very few things to the end-to-end user, and the more the network grows and the more the network interconnects with itself the less good properties it has end to end. It is a paradox. It relates to complex systems size. The basic mechanisms that are good for a size X network with a connection degree C may not be good for another network 10^6X in size and/or 100C in connection. Solutions to internet growth and stability must never compromise its good properties: openness, de-centralisation, multi-tenancy …. This growth& stability problem is important enough to have several groups working on it: Future Internet Architecture Groups. These Groups exist in UE, USA and Asia.

Internet basic tools for service building are: a packet service that is non-connection-oriented (UDP) and a packet service that is connection-oriented (TCP) and on top of this last one a service that is text-query-oriented and stateless (HTTP) (sessions last for just one transaction).A name translation service from object names to network names helps a lot to write services for Internet and also allows these applications to keep running no matter the network addresses are changing.

For most services/applications Internet is a 'HTTP network'. The spread of NAT and firewalls makes UDP inaccessible to most internet consumers, and when it comes to TCP, only port 80 is always open and even more only TCP flows marked with HTTP headers are allowed through many filters. These constraints make today's internet a limited place for building services. If you want to reach the maximum possible number of consumers you have to build your service as an HTTP service.

2. OBJECTS, OBJECT NAMES AND CONTENT

A decent 'network' must be flexible and easy to use. That flexibility includes the ability to find your counterpart when you want to communicate. In the voice network (POTS) we create point to point connections. We need to know the other endpoint address (phone number) and there is no service inside POTS to discover endpoint addresses not even a translation service.

In Internet it was clear from the very beginning that we needed names that were more meaningful than network addresses. To make the network more palatable to humans Internet has been complemented with mechanisms that support 'meaningful names'. The 'meaning' of these names was designed to be one very concrete: "one name-one network termination" … and the semantics that will apply to these names were borrowed from set-theory through the concept of 'domain' (a set of names) with strict inclusion. Pairs name-address are modelled making 'name' to have such an structure that represents a hierarchy of domains. In case a domain includes some other domain that is clearly expressed by means of a chain of 'qualifiers'. A 'qualifier' is a string of characters. The way to name a subdomain is to add one more qualifier to the string and so on and so forth. If two domains do not have any inclusion relationship then they are forcefully disjoint.

This naming system was originally intended just to identify machines (network terminals) but it can be ,and has been, easily extended to identify resources inside machines by adding subdomains. This extension is a powerful tool that offers flexibility to place objects in the vast space of the network applying 'meaningful names'. It gives us the ability to name machines, files, files that contain other files (folders), and so on… . These are all the 'objects' that we can place in internet for the sake of building services/applications. It is important to realise that only the names that identify machines get translated to network entities (IP addresses). Names that refer to files or 'resources' cannot map to IP network entities and thus, it is the responsibility of the service/application to 'complete' the meaning of the name.

To implement this semantics on top of Internet they built a 'names translator' that ended up being called 'name server'. Internet feature is called: Domain Name Service (DNS). A name server is an entity that you can query to resolve a 'name' into an IP address. Each name server only 'maps' objects placed in a limited portion of the network. The owner of this area has the responsibility of maintaining the names of objects associated to proper network addresses. DNS just gives us part of the meaning of a name. The part that can be mapped onto the network. The full meaning of an object name is rooted deeply in the service/application in which that object exists. To implement a naming system that is compatible to DNS domain semantics we can for instance use the syntax described in RFC2369. There we are given the concept of URI: Uniform resource Identifier. This concept is compatible and encloses previous concepts as URL: Uniform Resource Locator and URN: Uniform Resource Name.

For the naming system to be sound and useful it is necessary that an authority exists to assign names, to manage the 'namespace'.. Bearing in mind that translation process is hierarchical and

can be delegated; many interesting intermediation cases are possible that involve cooperation among service owners and between service and network owners. In HTTP the naming system uses URLs. These URLs are names that help us in finding a 'resource' inside a machine inside the Internet. In this framework that HTTP provides, the resources are files.

What is 'Content'?

It is not possible to give a non-restrictive definition of 'content' that covers all possible content types for all possible viewpoints. We should agree that 'content' is a piece of information. A file/stream is the technological object that implements 'content' in the framework of HTTP+DNS.

3. THE CONTENT DISTRIBUTION PROBLEM

We face the problem of optimising the following task: find & recover some content from internet..

Observation 1: current names do not have a helpful meaning. URLs (HTTP+DNS framework) are 'toponymic' names. They give us an address for a content name or machine name. There is nothing in the name that refers to the geographic placement of the content. The name is not 'topographic' (as it would be for instance in case it contains UTM coordinates). The name is not 'topologic' (it gives no clue about how to get to the content, about the route). In brief: Internet names, URLs, do not have a meaningful structure that could help in optimising the task (find & recover).

Observation 2: current translations don't have context. DNS (the current implementation) does not recover information about query originator, nor any other context for the query. DNS does not worry about WHO asks for a name translation or WHEN or WHERE… as it is designed for a semantic association 1:1, one name one network address, and thus, why worry? We could properly say that the DNS, as is today, does not have 'context'. Current DNS is kind of a dictionary.

Observation 3: there is a diversity of content distribution problems. The content distribution problem is not, usually, a transmission 1 to 1; it is usually 1 to many. Usually there is for one content 'C' at any given time 'T' the amount of 'N' consumers with N>>1 most of the times. The keys to quality are delay and integrity (time coherence is a result of delay). Audio-visual content can be consumed in batch or in stream. A 'live' content can only be consumed as a stream. It is very important that latency (time shift T=t1-t0 between an event that happens at t0 and the time t1 at which that event is perceived by consumer) is as low as possible. A pre-recorded content is consumed 'on demand' (VoD for instance).

It is important to notice that there are different 'content distribution problems' for live and recorded and also different for files and for streams.

A live transmission gives to all the consumers simultaneously the same exact experience (Broadcast/multicast), but it cannot benefit from networks with storage, as store-and-forward

techniques increase delay. It is impossible also to pre-position the content in many places in the network to avoid long distance transmission as the content does not exist before consumption time.

An on-demand service cannot be a shared experience.. If it is a stream, there is a different stream per consumer. Nevertheless an on demand transmission may benefit from store and forward networks. It is possible to pre-position the same title in many places across the network to avoid long distance transmission. This technique at the same time impacts on the 'naming problem': how will the network know which is the best copy for a given consumer?

We soon realise that the content distribution problem is affected by (at least): geographic position of content, geographic position of consumer and network topology

4. CURRENT CDNS: OPTIMISING INTERNET FOR CONTENT

-to distribute a live content the best network is a broadcast network with low latency: classical radio & TV broadcasting, satellite are optimal options. It is not possible to do 'better' with a switched, routed network as IP networks are. The point is: IP networks just do NOT do well with one-to-many services. It takes incredible effort from a switched network to let a broadcast/multicast flow compared to a truly shared medium like radio.)

-to distribute on demand content the best network is a network with intermediate storage. In those networks a single content must be transformed into M 'instances' that will be stored in many places through the network. For the content title 'C', the function 'F' that assigns a concrete instance 'Cn' to a concrete query 'Ric' is the key to optimising Content delivery. This function 'F' is commonly referred as 'request mapping' or 'request routing'.

Internet + HTTP servers + DNS have both storage and naming. (Neither of HTTP or DNS is a must.)

There is no 'normalised' storage service in internet, but a bunch of interconnected caches. Most of the caches work together as CDNs. A CDN, for a price, can grant that 99% consumers of your content will get it properly (low delay + integrity). It makes sense to build CDNs on top of HTTP+DNS. In fact most CDNs today build 'request routing' as an extension of DNS.

A network with intermediate storage should use the following info to find & retrieve content:

-content name (Identity of content)

-geographic position of requester

-geographic position of all existing copies of that content

-network topology (including dynamic status of network)

-business variables (cost associated to retrieval, requester Identity, quality,…)

Nowadays there are services (some paid) that give us the geographic position of an IP address : MaxMind, Hostip.info, IPinfoDB,… . Many CDNs leverage these services for request routing.

It seems that there are solutions to geo-positioning, but still have a naming problem. A CDN must offer a 'standard face' to content requesters. As we have said content dealers usually host their content in HTTP servers and build URLs based on HTTP+DNS so CDNs are forced to build an interface to the HTTP+DNS world.. On the internal side, today the most relevant CDNs use non-standard mechanisms to interconnect their servers (IP spoofing, DNS extensions, Anycast,…)

5. POSSIBLE EVOLUTION OF INTERNET

-add context to object queries: identify requester position through DNS. Today some networks use several proprietary versions of 'enhanced DNS' (Google is one of them). The enhancement usually is implemented transporting the IP addr of the requester in the DNS request and preserving this info across DNS messages so it can be used for DNS resolution. We would prefer to use geo-position better than IP address. This geo position is available in terminals equipped with GPS, and can also be in static terminals if an admin provides positioning info when the terminal is started.

-add topological + topographical structure to names: enhance DNS+HTTP. A web server may know its geographic position and build object names based on UTM. An organization may handle domains named after UTM. This kind of solution is plausible due to the fact that servers' mobility is 'slow'. Servers do not need to change position frequently and their IP addresses could be 'named' in a topographic way. It is more complicated to include topological information in names. This complexity is addressed through successive name-resolution and routing processes that painstakingly give us back the IP addresses in a dynamic way that consumes the efforts of BGP and classical routing (ISIS, OSPF).

Nevertheless it is possible to give servers names that could be used collaboratively with the current routing systems. The AS number could be part of the name. It is even possible to increase 'topologic resolution' by introducing a sub-AS number. Currently Autonomous Systems (AS) are not subdivided topologically nor linked to any geography. These facts prevent us from using the AS number as a geo-locator. There are organisations spread over the whole world that have a single AS. Thus AS number is a political-ID, not a geo-ID nor a topology-ID. An organizational revolution could be to eradicate too spread AS and/or too complex AS. This goal could be achieved by breaking AS in smaller parts confined each one in a delimited geo-area and with a simple topology. Again we would need a sub-AS number. There are mechanisms today that could serve to create a rough implementation of geo-referenced AS, for instance BGP communities.

-request routing performed mainly by network terminals: /etc/hosts sync. The abovementioned improvements in the structure of names would allow web browsers (or any SW client that recovers content) to do their request routing locally. It could be done entirely in the local machine using a local database of structured names (similar to /etc/hosts) taking advantage of the

structure in the names to guess parts of the mapping not explicitly declared in the local DB. Taking the naming approach to the extreme (super structured names) the DB would not be necessary, just a set of rules to parse the structure of the name producing an IP address that identifies the optimal server in which the content that carried the structured name can be found. It is important to note that any practical implementation that we could imagine will require a DB. The more structured the names the smaller the DB.

6. POSSIBLE EVOLUTIONS OF CDNS

It makes sense to think of a CDN that has a proprietary SW client for content recovery that uses an efficient naming system that allows for the 'request routing' to be performed in the client, in the consumer machine not depending of (unpredictably slow) network services.

Such a CDN would host all content in their own servers naming objects in a sound way (probably with geographical and topological meaning) so each consumer with the proper plugin and a minimum local DB can access the best server in the very first transaction: resolution time is zero! This CDN would rewrite web pages of its customers replacing names by structured names that are meaningful to the request routing function. The most dynamic part of the intelligence that the plugin requires is a small pre-computed DB that is created centrally, periodically using all the relevant information to map servers to names. This DB is updated from the network periodically. The information included in this DB: updated topology info, business policies, updated lists of servers. It is important to realise that a new naming structure is key to make this approach practical. If names do not help the DB will end up being humungous.

Of course this is not so futuristic. Today we have a name cache in the web browser + /etc/hosts + cache in the DNS servers. It is a little subtle to notice that the best things of the new schema are: suppress the first query (and all the first queries after TTL expiration). Also there is no influence of TTLs, which are controlled by DNS owners out of cdn1, and there are no TTLs that maybe built in browsers….

This approach may succeed for these reasons:

1- Not all objects hosted in internet are important enough to be indexed in a CDN and dynamism of key routing information is so low that it is feasible to keep all terminals up to date with infrequent sync events.
2- Today computing capacity and storage capacity in terminals (even mobile) are enough to handle this task and the penalty paid in time is by far less than the best possible situation (with the best luck) using collaborative DNS.
3- It is possible, attending to geographic position of the client, to download only that part of the map of servers that the client needs to know. It suffices to recover the 'neighbouring' part of the map. In case of an uncommon chained failure of many neighbour servers, it is still possible to dynamically download a far portion of the map.