

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



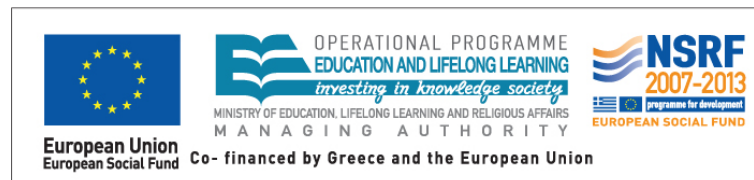
**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Information-Centric Networks

Section # 3.3: DNS Issues

Instructor: George Xylomenos

Department: Informatics



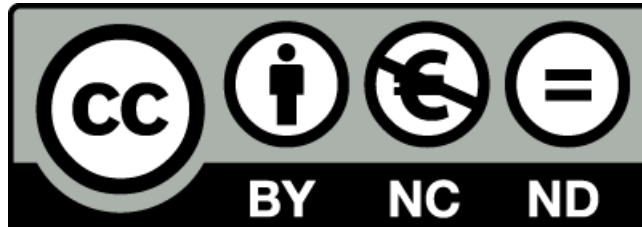
Funding

- These educational materials have been developed as part of the instructors educational tasks.
- The **“Athens University of Economics and Business Open Courses”** project only funded the reformatting of these educational materials.
- The project is being implemented as part of the Operational Program “Instruction and Lifelong Learning” and is co-financed by the European Union (European Social Fund) and national funds.



Licencing

- These educational materials are subject to a Creative Commons License.



Week 3 / Paper 3

- The design and implementation of a next generation name service for the Internet
 - Venugopalan Ramasubramanian and Emin Gun Sirer
 - ACM SIGCOMM 2004
- Main point
 - DNS is slow, vulnerable and not dynamic
 - CoDoNS is a DHT based alternative
 - It can work with or without DNS
 - PlanetLab tests show that it works very well

Introduction

- Susceptibility to DoS attacks
 - Limited server redundancy
 - 80% of domains only have two authoritative servers
 - 32% of the servers have a single connection to the Internet
 - The root servers are not that many
 - 20% of DNS servers suffer from severe security flaws
- Name-address translation is slow
 - Up to 30% of web transactions require >1 sec for DNS
 - Caching does not work very well due to the skewed tree
 - CDNs require very small TTL values
- Caching prevents dynamic mapping
 - Changes to DNS may take a long time to propagate
 - Services cannot be relocated quickly

Design goals

- A DNS replacement should have the following properties
 - Higher performance than DNS
 - Resilience to attacks
 - Fast (but secure) update propagation
- Cooperative Domain Name System (CoDoNS)
 - Uses a DHT for self-organization, scalability and resilience
 - Adds a proactive caching layer to replicate mappings
 - Wire-protocol compatible with DNS
 - Clients simply direct their queries to CoDoNS servers
 - Names not added to CoDoNS are translated by DNS
 - Decouples namespace management from physical delegation
 - Name records are self-validating
 - You can use many namespace operators for the same names

Problems with legacy DNS

- Study of DNS delegation chains
 - Based on two web directories and 500 most popular domains
- Failure resiliency – bottlenecks
 - Delegation bottlenecks
 - How many servers need to be compromised to control a domain?
 - 78.63% of domains rely on two servers
 - Over 90% rely on three or less nameservers
 - Physical bottlenecks
 - How many gateways need to be compromised to control a domain?
 - 33% of domains are bottlenecked at a single gateway
 - Redundant name servers are typically in the same area
 - Even Microsoft used to have all its servers on the same area

Problems with legacy DNS

- Failure resilience – implementation errors
 - 2% of servers have the serious tsig bug
 - Can be used to control the server
 - 19% of servers have the negcache problem
 - Can be used for DoS attacks
- Performance – latency
 - 1-2 seconds are quite common
 - Largely due to the long tail of name popularity distribution
 - Caching cannot help rarely accessed names
 - CDNs have made things worse
 - The use of low TTLs reduces caching efficiency
 - But it is required to perform server selection

Problems with legacy DNS

- Performance – misconfigurations
 - 14% of domains return inconsistent responses
 - Due to delegation errors and timeouts
- Performance – load imbalance
 - The higher levels are necessarily loaded
 - The 16 root nameservers are became 60
 - Performance and reliability issues
- Update propagation
 - The TTL has to balance caching and update propagation
 - Low TTL means limited caching
 - High TTL means slow update propagation
 - 40% of domains use TTLs of one day or more

CoDoNS: Beehive

- Beehive is a proactive replication framework
 - Allows $O(1)$ lookups on prefix matching DHTs
 - Can be used on Pastry and Tapestry
 - These DHTs route objects by matching prefix digits
 - Normally this requires $O(\log N)$ steps
 - Beehive proactively caches objects on the path to a node
 - Replication at level n means that an object is within n steps
 - This means that it is cached at all nodes with n matching digits
 - The trick is to use popularity to decide on caching
 - You need to know the popularity ranking of objects
 - Then you set a goal for the average hops for a match
 - A formula provides the level of replication for each object
 - Replication is predictable, unlike caching
 - You can update replicas because you know where they are

CoDoNS: Beehive

- How do you know how popular an object is?
 - Combination of local measurements and aggregation
 - Each node locally tracks object access frequencies
 - Periodically each node aggregates values from descendants
 - Recursively, all data reach the home node of the object
 - The home node pushes the estimate to replicating nodes
 - Each node calculates the replication level for each object
 - A replication protocol is used to insert/update/remove replicas
 - Each node only talks to nodes one level away from itself
 - Each node only needs to track nodes one hop away
- Response to flash crowds and attacks
 - The access frequencies change rapidly
 - The replication level is increased automatically

CoDoNS: architecture

- Namespace management <> name resolution
 - Each institution contributes some nodes to CoDoNS
 - These nodes self-organize into a DHT
 - Nameowners purchase name certificates from operators
 - Names are inserted into CoDoNS with these certificates
 - The home node for a name is calculated by hashing
 - The home node holds a permanent record of the data
 - It also manages the replication protocol
 - Each object is replicated close to the home node for failover
 - The namespace does not have to be hierarchical
- What happens with names outside CoDoNS?
 - Their “home node” fetches the data from DNS
 - It is also responsible to monitor the DNS for changes

CoDoNS: implementation

- CoDoNS is layered on top of Pastry and Beehive
 - Each query is routed via Pastry to the home node
 - Either a cache or the home node responds
 - Beehive proactively replicates popular objects
 - Data can also be entered in local CoDoNS servers
 - This avoids asking for it from a faraway home node
- There is no other caching in CoDoNS!
 - Only replication and locally entered data
 - Replicated data does not time out
 - It is only proactively modified
 - Each node knows with whom to replicate data
 - CoDoNS can update data very quickly

Issues and implications

- CoDoNS is based on DNSSEC
 - DNS records are digitally signed by operators
 - Public keys and certificates are stored in DNS
 - Each node can verify the authenticity of signed records
 - CoDoNS also caches the certificates
 - Only signed data can be inserted into CoDoNS
 - All servers check verify data signatures
 - CoDoNS uses certifying resolvers for DNS data
 - Multiple resolvers are used to ensure authenticity
- CDNs are supported in a special way
 - CoDoNS cannot be used to do the “stupid DNS tricks”
 - Redirection records are used to select servers instead
 - They are replicated like any other record

Evaluation

- Based on a PlanetLab deployment
 - 75 nodes were used, getting data from DNS
 - Queries were issued after the Pastry DHT stabilized
 - Comparison with regular DNS
- Lookup performance
 - Initially CoDoNS is slower than DNS
 - It needs to fetch data from DNS first
 - Eventually it is much faster than DNS
 - Records are cached after being fetched from the DNS
- Flash-crowd effect
 - Modeled as a large scale change (reversal) of object popularity
 - During the switch CoDoNS slows down
 - A bit later it starts outperforming DNS again

Evaluation

- Load balance
 - Initially the home nodes of popular objects are overloaded
 - Eventually load is spread evenly due to proactive caching
 - Even with flash crowds, CoDoNS adapts quickly
- Update propagation
 - 98% of replicas are updated within one second
 - This allows DNS to handle dynamic objects
- How much does this cost?
 - Nodes need to store 10% of total records for this performance
 - Roughly 13 MB per node
 - Low bandwidth usage for all network activities
 - 12.2 KB/s on average

Counterpoint

- What about the comparative study of DNS with DHTs?
 - That paper (supplementary reading) does not endorse DHTs
 - It shows that DNS works better for some things
 - Plus, it can be modified to work as good as DHTs in others
- BUT, it does not compare DNS with CoDoNS
 - It uses a simple DHT (Chord) without proactive caching
 - DHTs without modifications do indeed have many problems
 - They cannot compete with DNS for hierarchical namespaces
 - The core idea in CoDoNS is not the DHT but Beehive
 - Proactive replication is heavily used
 - Otherwise lookups are not particularly fast
 - Replication improves the common case
 - It is important to understand what is compared in each case!

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

End of Section # 3.3

Course: Information-Centric Networks, **Section # 3.3: DNS Issues**

Instructor: George Xylomenos, **Department:** Informatics

