Application-Layer Anycasting

By Samarat Bhattacharjee *et al.*

Presented by Matt Miller
September 30, 2002

Motivation

- Given that multiple replicas of a service are available, how do we connect to the “best” one for a particular client?
- Anycast has been defined as a service and a framework specified for the IP layer. How can we specify an anycast framework at the application layer?

Key Contributions

- Presents arguments why anycast should not be implemented at the network layer
- Provides an application layer framework for implementing anycast
- Enumerates possible filters and metrics that could be used and how they could be supported
- Adapts server pushing for updating state information that trades off accuracy for control overhead

Limitations of Network Layer Anycast

- Address space issues in IPv4
  - Use existing addresses and make identification difficult
  - Use a separate set of addresses and risk inefficient routing
- Requires router support to avoid delivering to multiple hosts

Limitations of Network Layer Anycast

- Most protocols would like all data for a connection delivered to one IP address once a service is found
- “Best” only refers to shortest hop count. At the application layer, many other metrics (possibly user-defined) may be applied.

Service Location

- How to find a service
  - Multicast to find it
  - Use name server architectures
  - Caching a resource location where it is frequently accessed
- How to find the “best” service
  - Gather information from servers and efficiently search through it
  - Servers periodically push their local state

Replicated Services

- Replicated services are equivalent in content and/or functionality from an application perspective
- Compute servers are machines which are capable of running a particular computation
  - Server statistics such as CPU load may be an important criteria

Anycast Domain Names

- Anycast Domain Names (ADN) identify an anycast group of potentially dynamic IP addresses
- The group could also be specified as domain names or aliases instead of IP addresses

Anycast Name Resolution

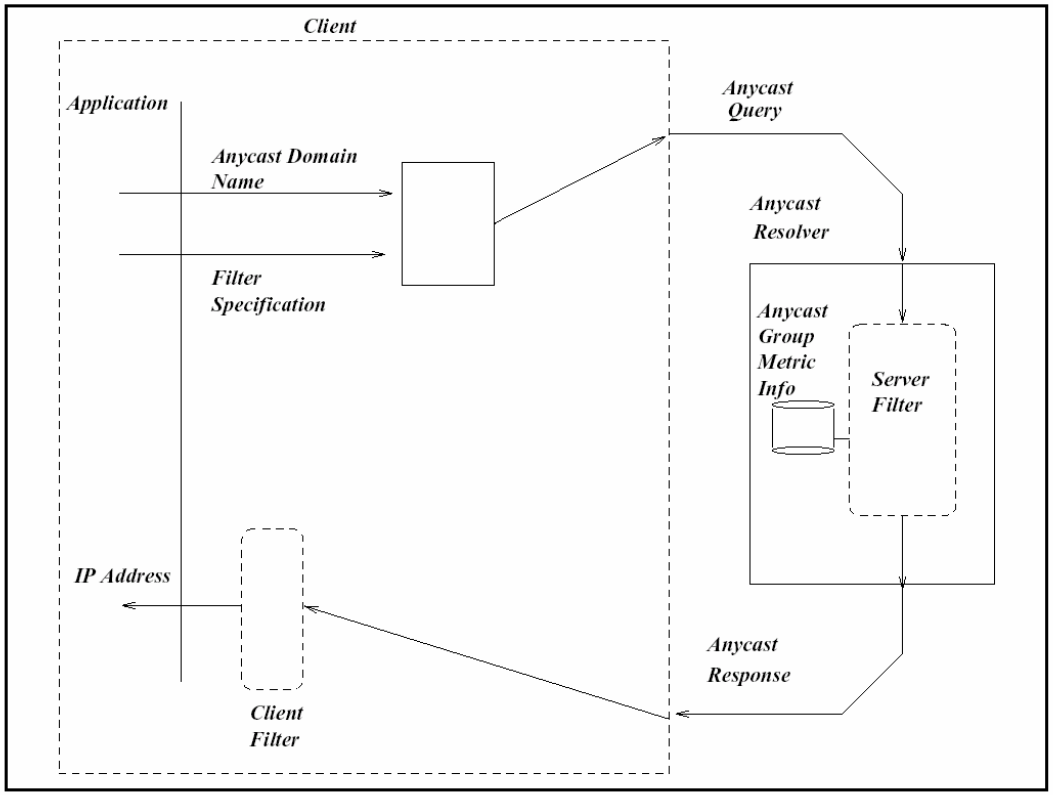

FIGURE 1: Anycast Name Resolution Query/Response Cycle

Anycast Name Resolution (2)

- Works like DNS server
- A service and domain name are specified
- The domain name is resolved by hierarchtically querying ADN servers until an authoritative response or cached entry is found
- The ADN maintains a list of IP addresses for a service and associated metrics

Anycast Name Resolution (3)

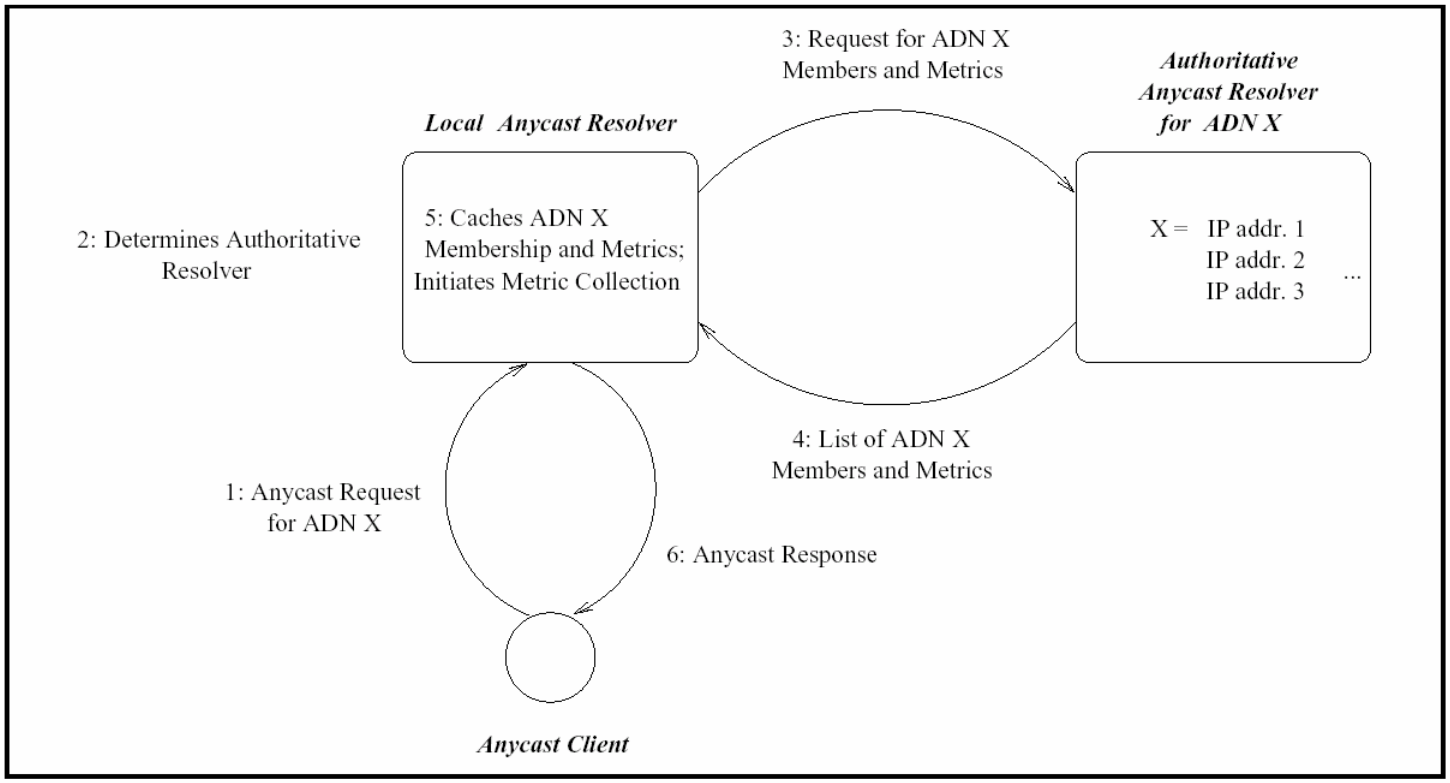

FIGURE 2: Anycast Resolver Architecture

Filtering

- The local ADN resolver can filter addresses given by authoritative entity
- The client must handle multiple or no addresses being returned by the resolver
- Three proposed filters
  - Content-independent
  - Metric-based
  - Policy-based

Content-Independent Filter

- Random selection of a member
- Return all members of the group
- Return a subset of *n* members of the group

Metric-Based Filter

- Select the best member according to a single metric
- Select the best member according to a function of multiple metrics
- Select the best by sequentially applying filters

Policy-Based Filters

- Vague description, not based on performance measurements
- Generally, it would be a boolean function which determines whether an address meets a policy criteria or not

Filter Issues

- How can clients tell resolvers what filter to run
  - Use well-known identifiers
  - Allow clients to give procedural description
- How is it implemented
  - Create a new function with filters
  - Specialized domain names (Metric-Qualified ADN)
    - Backwards compatible
    - E.g. **`ServerLoad.wwwnews%cc.gatech.edu.any`**

Metric-Qualified ADN Implementation

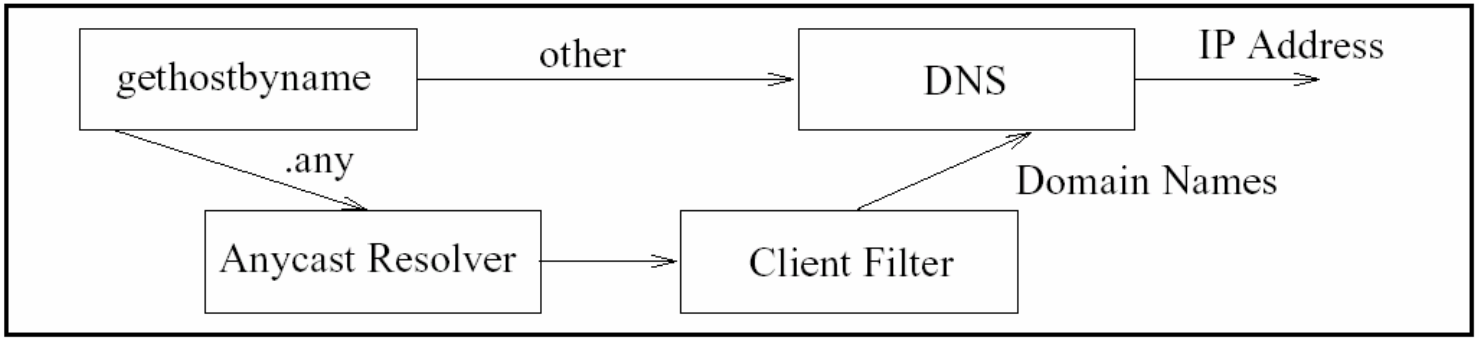

FIGURE 3: Implementation using Metric-Qualified ADNs

Metrics

- Metrics are relative rather than absolute
- Goal is to get reasonable accuracy without excessive network or server load
- Possible metrics
  - Latency
  - Throughput
  - Server Load

Metric Collection

- Remote Server Performance Probing
  - Proxies periodically query replicated servers to determine how potential clients would perform
- Server Push
  - Servers send data when changes occur
  - Could be multicast to all interested anycast resolvers

Metric Collection (2)

- Probe Locally-Maintained Server Performance
  - Probe request reads static data from the server which is periodically updated
- User Experience
  - Users give their preference of servers that have performed well in the past
  - No burden on server, but could be very inaccurate
  - Accuracy may be increased if clients share experiences

Metric Collection (3)

- Example of server push
  - If a particular metric has changed by more than a certain threshold in a time interval, push the data.
  - Otherwise, decrement the threshold by a specified amount. When it reaches zero, push the data.
  - Demonstrates the tradeoff in accuracy and control overhead

Metric Collection (4)

	Net Load	Server Mod	Server Load	Exercises Net Path	Accuracy
Probing	$2PT_p$	No	Moderate	Yes	Moderate
Server Push	T_s	Yes	Low	No*	High
Reading Server Log	$2PT_p$	Yes	Low	No*	High
User Experience	None	No	None	Yes	Low/Varies

(* See note in text)

Table 2: Comparison of Metric Collection Techniques

Conclusions

- Shows application-layer anycast is feasible and provides basic framework
- Gives clients more control in selecting servers and is easily extendible
- Opens issues
  - How to specify policy filters
  - How to provide client-to-server metrics in a scalable way
  - Stability in service location