Efficient In-band Service Announcement Through IPv6 Address Encoding

Christoph P. Mayer, Christian Hübsch, Martin Röhricht
Institute of Telematics – Karlsruhe Institute of Technology (KIT) – Germany

Abstract—**The announcement and discovery of network services represents one of the most important functionalities in today's networks. In order to find and use services provided by remote systems, service management and dedicated service discovery protocols are used. These mechanisms introduce, however, additional overhead, especially in resource-constrained environments such as sensor networks. In this paper, we propose an in-band service announcement mechanism through IPv6 address encoding. In this scheme, IPv6's large address space is used by a system to encode its service information into the host part of its IPv6 address. This information can then be extracted and efficiently used by other systems, allowing for service announcement without the need for additional protocols.**

I. Introduction

In distributed networks without a central management entity, discovery of services provided by other systems is a common problem. Often, dedicated service announcement and discovery protocols, such as the *Service Location Protocol* [1], are deployed. However, these protocols induce communication overhead and additional complexity. This may contradict specific deployment goals, e. g. in energy-constrained sensor networks. Hence, it would be beneficial to provide a service announcement and discovery mechanism without having to rely on a dedicated protocol.

IPv6's large address size [2] holds great potential to encode information about services provided by a system. If systems encode their local services into their IPv6 address directly, other systems can extract network-wide services "on-the-fly" from the IPv6 network traffic. This, for example, enables proactive caching of information about services in the network. In this paper we propose different mechanisms to encode service information into the host part of the IPv6 address. We then exemplarily show how this information can be used for in-band service announcement and discovery without incurring additional overhead.

II. Encoding Schemes

A. Bitwise Encoding

If the overall number[1] of possible services is ≤ 62, the host part of the IPv6 address can be used to directly encode provided services of a system. Every service in the globally defined service list $S = \{s_1, \dots, s_{62}\}$ is assigned a dedicated bit, defined as 2^{i-1} for service s_i. A system encodes the set of services $\{s_x, s_y, \dots, s_z\}$ it wants to announce as its IPv6 address host part by concatenating $2^{x-1}|2^{y-1}|\dots|2^{z-1}$ through a bitwise OR operator. Figure 1 shows an example of the bitwise encoding for two arbitrary services that each map towards one specific bit. As the service list S is globally defined, every system can extract the set of services announced by a system through the host part of its IPv6 address.

[1]Considering "universal/local" and "individual/global" bits [2].

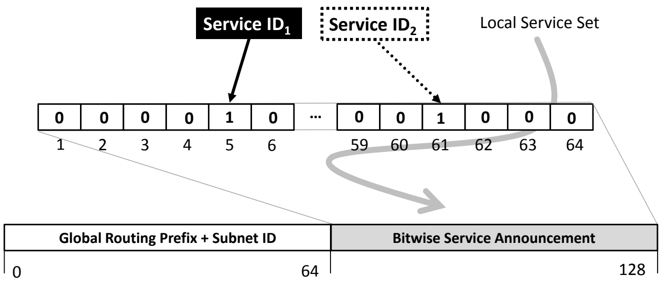

Figure 1. Bitwise service encoding of two network services.

In case two systems on the local network announce the same set of services, they may end up with identical IPv6 addresses. By using the IPv6 duplicate address detection mechanism the systems check whether the IPv6 address is already in use. In this case they can randomly encode an additional service they actually do not provide. Requests for such a service will eventually be denied, resulting in small additional overhead in the network but still providing the benefits of a simplified service announcement scheme.

B. Bloom Filter-based Encoding

While typically the set of services provided by a single system is rather small, the overall pool of possible services can be large. Bitwise encoding is, however, inherently very limited with respect to the pool size. In this section we describe how *Bloom Filters* [3] can be used to encode service information for a potentially larger pool size, such as IANA's "Protocol and Service Names Registry".

A Bloom Filter is a probabilistic data structure that supports element insertion and query. Given a bit array $B = \{b_1, \dots, b_m\}$ of size $|B| = m$, the Bloom Filter uses k hash functions $H = \{h_1, \dots, h_k\}$ with $h_i(\cdot) \mapsto [1, m]$ to insert an element s_j from a universe $S = \{s_1, \dots, s_n\}$ by setting $\forall h_i \in H : B[h_i(s_j)] = 1$. An element s_j is queried from the Bloom Filter by testing whether all required bits $B[h_i(s_j)]$ of all hash functions $h_i \in H$ are set to 1. Bloom Filters allow storage of elements in a very compact form, and irrespective of the original size of an element $|s_i|$. Their probabilistic nature results in *false positives*, but *no false negatives*. I. e., if an element was inserted, it is successfully queried. However, an element may be successfully queried, although it was never inserted. This false positive probability can be approximated as $p_{\text{false}} \approx (1 - e^{-kn/m})^k$. Given m

and n, the optimal number of hash functions is $k = \frac{m}{n} \ln 2$. With p_{false}, the number of bits needed in the Bloom Filter for encoding an element s_i is $\ln(1/p_{\text{false}})/\ln^2(2)$.

In our proposal, the lower 62 bits of the IPv6 address are used to store the Bloom Filter, i. e. $m = 62$, as depicted in Figure 2. Given an optimal number of hash functions $k = \frac{m}{n} \ln 2$, the false positive probability for n services equals to $p_{\text{false}} = 2^{-62 \cdot \ln(2)/n}$. Obviously, only a relatively small number of services can be stored in the Bloom Filter with an acceptable false positive probability. However, we believe that encoding of 10 services—resulting in a false positive probability of $\approx 5\%$—can be considered enough for most scenarios. This results in an average of 6.2 bit required per element encoding. As rule of thumb, the Bloom Filter encoding only makes sense if the overall pool of services S is large, false positives are acceptable, and the original representation of a service s_i requires more bits than in a Bloom Filter approach, i. e. $|s_i| > \ln(1/p_{\text{false}})/\ln^2(2)$.

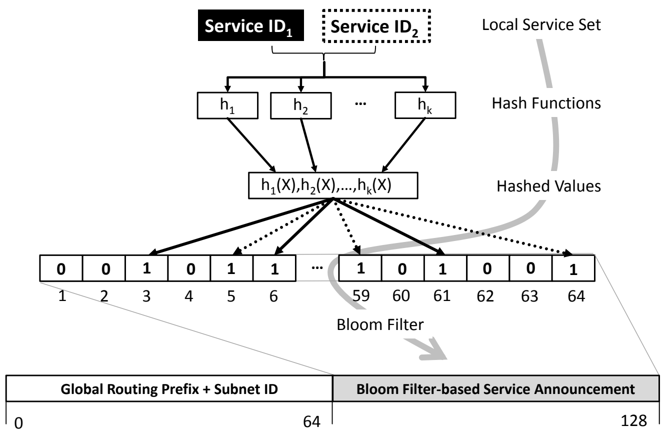

Figure 2. Bloom Filter-based encoding of two network services.

In contrast to the bitwise encoding scheme, IPv6 address collision can be prevented in advance using the Bloom Filter encoding mechanism. In order to prevent end systems from choosing the same IPv6 address, every end system encodes its own network card MAC address into the Bloom Filter. This represents a unique "service" that only the system itself announces. As MAC addresses are unique, this insertion results in a unique pattern in the IPv6 address with high probability.

III. Applicability and Use Cases

In this section, we give three exemplary use cases for the proposed service encoding scheme.

A. Pro-active Service Caching in LANs

For IPv6-enabled Local Area Networks (LANs), service encoding can be used supplementary to already deployed service announcement protocols. In this case a system can learn about available services in the network by simply collecting IPv6 addresses during operation. The gained information can then be cached pro-actively for existing service discovery mechanisms. In a LAN environment where—besides IPv6 addresses—also the MAC addresses are known, the MAC address can be used as additional information to only allow systems inside the LAN to decode services from an IPv6 address. When encoding services, a system uses its MAC address for seeding the Bloom Filter's hash functions. Decoding can then only be performed by systems that know the corresponding MAC address of the system. Such a scheme can further be extended by using secret keys as seeding for the hash functions. This way, only authorized systems are allowed to take part in the service discovery.

B. Sensor Network Service Discovery

Sensor networks are subject to strong resource constraints. In such environments dedicated service announcement and discovery protocols introduce a significant overhead. Using the presented service encoding scheme, sensors listening to the network traffic can collect IPv6 addresses on the fly and use the encoded service information to construct a local service repository. In face of a service request that reaches a sensor by accident—e. g. a Bloom Filter's false positive—sensors attach the IPv6 address to the request and forward it to a device they assume to provide this service, based on their local service repository. This iterative forwarding mitigates the drawback of false positives by coupling false positive probabilities and hence finding a valid service provider in a small number of steps.

C. Peer Sampling in Overlay Networks

In overlay networks "peer sampling" modules are often deployed to collect sets of other peers in the overlay and query information about those peers [4]. Depending on the metrics for subsequent peer selection, useful information can be encoded into the IPv6 addresses of peers. This allows early discarding of irrelevant peers from the sampling process and prevents unnecessary communication overhead with such peers.

IV. Summary and Conclusions

While service discovery and announcement are implemented at higher layers of the protocol stack, the large address space of IPv6 addresses allows to directly embed service information. In this paper we presented two schemes how such information can be encoded considering different scenarios and constraints. We illustrated the proposed mechanisms and their applicability by three different use cases. While the proposed approach does not require existing protocols to be adapted, it should be tuned to the specific application, e. g. to handle false positives in service announcements.

References

[1] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," RFC 2608 (Proposed Standard), Jun. 1999, updated by RFC 3224. [Online]. Available: http://www.ietf.org/rfc/rfc2608.txt

[2] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460 (Draft Standard), Dec. 1998, updated by RFC 5095. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[3] B. H. Bloom, "Space/time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[4] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-based Implementations," in *Proceedings of ACM/IFIP/USENIX International Conference on Middleware*, Toronto, Canada, Oct. 2004, pp. 79–98.