The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'06)

UPNP SERVICE DISCOVERY FOR HETEROGENEOUS NETWORKS

José M. Sánchez Santana, Marina Petrova, Petri Mähönen
RWTH Aachen University, Department of Wireless Networks
Aachen, Germany

ABSTRACT

It is desirable that future networks make the use of services and other resources simple for everyone. In that sense, service discovery comes to fill a gap in the networking field, since it eliminates most of the configuration and maintenance tasks. The Universal Plug and Play (UPnP) architecture does so by dividing the communication session in six different phases, and by providing the user with a friendly interface to the services. UPnP was mainly designed with wired or fixed wireless infrastructure networks in mind, and thus is not the optimum solution for the future mobile ad hoc networks (MANETs), where devices are usually battery operated and mobile. In this paper, we present an extension for the UPnP architecture which is designed specifically for wireless ad hoc networks. We made a performance analysis, and show that it has very promising performance characteristics. These extensions were carefully designed to be minimal ones, in order to make it compatible with the existing UPnP framework. The paper shows that UPnP is a promising alternative as service discovery protocol for heterogeneous networks.

I. INTRODUCTION

The Universal Plug and Play (UPnP) [1] protocol suite defines a set of standards that allows all kinds of networking devices, wired or wireless, to interact seamlessly with each other. Every device can either offer capabilities (in the form of services or resources), or request other capabilities offered at any point of the network in a simple and standardized way. Users benefit from this fact, since configuration and maintenance tasks are removed from the system. In the literature, these protocol suites are known as *service discovery* architectures. Of paramount importance here are the *service discovery protocols* that perform the low-level tasks involved in the service advertisement and requesting phases. In addition to UPnP, we find a wide range of examples, such as SLPv2 [2], Jini [3] or Rendevouz [4].

Most service discovery protocols are designed to run smoothly on wired or fixed wireless infrastructure networks, since they rely on a central repository that caches information about all resources present in the network. However, when mobile ad hoc or mesh networks are deployed (e.g. for entertainment or industrial monitoring purposes), a central repository is not anymore the most efficient solution [5], and one has to rely on a more robust, but also more resource-hungry, peer-to-peer service discovery protocol.

Some authors have already proposed solutions for service discovery in MANET scenarios [6, 7, 8, 9]. None of these solutions are commercially available. In this paper, we will show the feasibility of a simple extension to the commercially popular UPnP protocol suite that achieves improved performance in wireless ad hoc networks, while still being backward compatible in both wired and fixed wireless infrastructure domains. We modify the service discovery protocol to dynamically adapt it to the type of underlying network infrastructure.

The paper is structured as follows. Section II introduces previous work on service discovery for MANETs, as well as on UPnP. Section III serves to understand the rest of the paper as it introduces UPnP and its discovery protocol. The protocol extension itself is detailed in section IV. In section V, we present simulation results of our protocol, and show how it outperforms the standard version. Finally, conclusions and pointers for future work are given in section VI.

II. RELATED WORK

We are interested in the design of a peer-to-peer service discovery protocol that adapts efficiently to the needs of resource constrained mobile ad hoc networks. Current research in this field follows two main strategies: *service-aware* routing protocols, or *intelligent forwarding* of service information at the application layer. These two approaches define the two main families of service discovery protocols for MANETs.

An example of the first family is the protocol proposed by Perkins *et al.* [6] that piggybacks service discovery information on AODV routing control packets. This reduces the amount of messages, and eliminates the gap between service discovery and routing.

Several examples can be found that belong to the second protocol family. In IBM's DEAPspace [7] each node maintains a cache to store all services offered in the network, the same way the distance vector routing protocols maintain a cache to store routes to all nodes. In Konark [9], nodes perform *intelligent forwarding* of service requests, since only the difference between a node's offered services and the services required by the message is forwarded. Finally, in Group Based Service Discovery (GSD) [8] service requests are *selectively forwarded* based on group information.

The UPnP architecture is also a subject of further study. The UPnP Forum, a consortium of over 700 vendors, has set up working committees that continue the standardization process. In addition to this, a number of authors have also proposed changes to the architecture. However, to the best of our knowledge, no results have been published so far that address the necessity to modify UPnP to adapt it to the needs of MANETs.

III. BACKGROUND

The UPnP architecture comprises a set of protocols for enabling networking entities of all types to share one another's resources. Every stage of a device lifetime is covered: addressing, advertisement and discovery of services, control of

1-4244-0330-8/06/$20.00

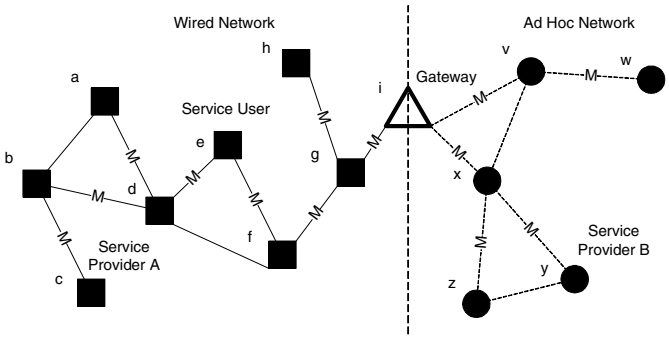

Figure 1: Sample heterogeneous network where UPnP could be deployed. Solid lines correspond to wired links, and dashed to wireless links.

those services, and shutdown. Two entities are defined by the standard: *controlled devices*, providers of services, and *control points*, users of those services. Any physical or logical UPnP device can be composed of any number of them. The UPnP solution is built on top of the Internet Protocol (IP) stack, and thus is network and transport layer dependent. There are a number of reasons as to why this architecture has become popular during the last years. On one side, this solution leverages a number of popular Internet protocols and description languages that are widely adopted by the growing IP community. On the other side, it is supported by a strong industry consortium.

The Simple Service Discovery Protocol (SSDP) is the discovery protocol included in the UPnP architecture. Due to its peer-to-peer structure, all nodes have the same functionality. At the same time, the lack of a central point of failure (e.g. directory agent or similar) makes it a good candidate for service discovery in mobile ad hoc networks.

A global peer-to-peer structure requires some type of information flooding or sharing with all (or a set of) nodes in the network, which can be costly in terms of network resources. The approach taken in this case is to rely on *multicasting*. Therefore, all SSDP nodes in the network need to belong to the same multicast group, so that service requests and advertisements are received by all nodes generating the minimum network load possible.

Four message types are defined in SSDP: service advertisement (*alive*), service query (*discover*), service shutdown (*bye-bye*) and service response (*response*). The first three are sent to a predefined multicast group address, and thus are received by all nodes with SSDP capabilities. The last one is sent as a response to a query, and thus it is unicasted towards the originator of the query.

With the help of Fig. 1, we can understand how SSDP works. When a service provider (a peer that hosts a service) boots up, and after network addresses have been correctly assigned, it actively advertises its capabilities to all other SSDP nodes. In this case, service providers A and B (nodes c and y) send alive messages to nodes up to n hops away in the multicast tree structure (n is implementation dependent). Links belonging to the multicast tree are denoted by the letter M on top of it. In SSDP, no difference is made between nodes in the ad hoc part and nodes in the wired part of the network.

When a node needs to find a service (we name that node a service user), it issues a discover message to all nodes up to n hops away in the multicast tree. Assuming n is high enough, if node e issues a discovery message for service B, then it is first received by d and f. In turn, node d forwards it to a and b, and b forwards it to c. On the other side of the network, f forwards the message to g and so on. If we start from the ad hoc section of the network, the gateway (node i) forwards the discover message to nodes v and x, v forwards it to w, and x forwards it to z and y, where a match is found and a response is sent to node e. In the wireless part of the network, tree multicast message retransmissions have taken place.

Alive and response messages carry information about new services, and thus provoke an update of the local caches at the nodes that receive them. However, after a certain amount of time called *lifetime*, this information expires and must be purged from the caches. The lifetime field in alive and response messages gives such information, and must be set by the providers of services when an announcement (alive or response) is sent.

IV. Extensions to Universal Plug and Play (UPnP)

A. Motivation

There are a number of reasons that motivate the extension of SSDP, and the UPnP architecture altogether, when the targeted networks are heterogeneous, possibly consisting of wired, fixed wireless infrastructure and mobile ad hoc sections.

In UPnP, for a service user to find all possible resources present in a heterogeneous network, a multicast routing protocol needs to run on all nodes, wired or wireless. Although multicasting helps in keeping the number of forwarded packets to a minimum, ad hoc multicast routing protocols are not yet mature, and it is a question if they will ever be a part of common mobile ad hoc networks.

Through set of simulations, not detailed here due to space limitations, we show how a typical peer-to-peer service discovery protocol (in this case SSDP) performs in MANETs. Several multicast routing strategies are taken, ranging from simple *flooding* to a more complex *multicast tree based* protocol. It turns out that, when a multicast tree based routing protocol is used, tree partitioning occurs, and packet delivery rates decrease below that of simple blind flooding.

Finally, the need to constantly maintain a tree does not make multicast tree based routing an appealing approach to be used in networks where resources are scarce, such as MANETs. A constant network load also implies a higher energy consumption, thus reducing the lifetime of battery-operated systems.

In short, the extensions to the UPnP architecture presented in this section are motivated by the lack of a standard multicast routing protocol for mobile ad hoc networks, the poor performance of multicast routing compared to much simpler solutions, and the constant network utilization of multicast, which implies a lower life for mobile devices. However, we must underline that assessments are only valid for a typical peer-to-peer service discovery protocol, such as SSDP. Other applica-

tion types may indeed profit from the use of multicasting.

B. Wireless SSDP

For our new design, we choose to remove multicasting in the ad hoc part of the network, and rely only on the unique broadcast property of the wireless channel. Of course, *broadcast storms* [10] are an issue here, and need to be taken into consideration. In the rest of this section, we explain how to avoid the problem in order to obtain an scalable and efficient discovery protocol.

Our approach uses the concept of *intelligent forwarding* of service requests at the application layer (multicast routing is not required anymore), together with a caching of service information on all nodes. This is done in the ad hoc section of the network, while in the wired or fixed wireless infrastructure sections the protocol behavior remains unchanged. In the following, we will refer to the new implementation of SSDP as wireless SSDP (wSSDP).

The general idea of wSSDP is quite simple. Like in SSDP, *alive* messages are sent by nodes that host services to advertise its presence; *byebye* messages indicate that a certain service will shutdown shortly; and *discover* messages are used to query a particular type of service. A *response* message is sent back to the originator of a request if a service match happens. Certain wSSDP messages (alive, discover and byebye) are sent to the *broadcast address* (and not to the UPnP multicast address) of the ad hoc network. All nodes in range will receive and process them at the application layer, and thus bypass the routing layer. There it is decided whether a particular message should be forwarded or not. This is done based on the information stored at the *local cache* (e.g. which services are available at the neighboring nodes).

Basic message structure remains the same for backward compatibility with older versions of SSDP. Only the following piece of XML code is added to the HTTP body of alive, byebye and discover messages:

```
<?xml version="1.0"?>
<wupnp>
 <bcastID>value</bcastID>
 <hopCount>value</hopCount>
 <maxHopCount>value</maxHopCount>
</wupnp>
```

Current SSDP implementations will discard this body and thus will not be affected by this change.

The *bcastID* tag is used to uniquely identify a broadcast message, and so to ensure that it is not forwarded more than once by the same node. It is incremented by one every time a node sends a broadcast message. The *hopCount* tag indicates the number of times that a message has been forwarded. Finally, the *maxHopCount* tag limits the maximum number of forwards for a certain message.

The local cache found in every node stores the following information about services: *ServiceType*, *ServiceId*, *IsLocal*, *Location*, *Expiration* and *BeatCount*.

The *ServiceType* identifies the class of service, and *ServiceId* uniquely identifies a particular instance of a service. Further, *IsLocal* indicates whether the service is offered locally or remotely. If the service is remote, then *Location* shows how one should contact a particular service, and *Expiration* contains the lifetime of a service, that is, the amount of time that it remains in the local cache before it is purged. Otherwise if the service is local, then *Location* is left unused. Finally, *BeatCount* indicates how many consecutive alive messages have been received from that service. It serves as an *stability parameter* that allows us to make a more intelligent decision on whether to forward a packet through a certain node or not. It only makes sense in the case of remote services.

Upon reception of an alive or byebye message, a node performs the following operations:

```
if (bcastID not known) {
  Update local cache
  hopCount++
    if (hopCount < maxHopCount) {
      Rebroadcast message
    }
}
```

In all other cases the message is dropped. Updating the local cache means adding (or deleting) service type, id, location and expiration information (contained within the body of the HTTP message header), as well as incrementing the beat count for that particular service if needed. The beat count is incremented if two consecutive alive messages are received from the same service provider, otherwise the beat count is reset to zero. This can be deduced from the value of the *bcastID* tag.

The next piece of pseudocode shows the steps taken by a node when a discover message is received:

```
if (bcastID not known) {
  hopCount++
  if (hopCount < maxHopCount) {
    if (SearchTarget not specified) {
      Send_response(local_services)
      Rebroadcast_message
    } else {
      Search_local_cache(SearchTarget)
      if (SearchTarget found locally) {
        Send_response
      } else if (SearchTargets found  == 1
      AND BeatCount > BeatCountTHR) {
        Unicast_request_to_Location
      } else if (SearchTargets found > 1) {
        Rebroadcast_message
      }
    }
  }
}
```

As it is seen, after checking for valid *bcastID* and *hopCount*, a discovery message is rebroadcasted in the following cases:

- A search target is not specified, and thus any service will produce a match.
- More than one service matches the search target.

In turn, a discovery message is unicasted to the service location in the case that only one service matches the search target, and its beat count parameter is higher than a predefined beat count threshold (BeatCountTHR).

This forwarding scheme can be easily understood by going back to Fig. 1. The beat count threshold is disabled in this case (i.e. set to zero), and the *maxHopCount* of alive messages is set to one, so only nodes x and z will initially know about service provider B (node y). If a discover message with search target B is issued at node i (the gateway), nodes v and x receive it. Since only x knows about service B, then it unicasts the message to node y, and a service match occurs. Node v drops the message. Thus, only two messages will need to be transmitted. In this case, using wSSDP saves us at least one message retransmission when compared to the same case shown in section III for SSDP. Besides, here no multicast tree management traffic is generated, so the network remains silent when no wSSDP messages are issued.

V. Simulations

In order to compare the performance of the original SSDP and our proposal, we implemented SSDP and wSSDP in the network simulator ns-2 [11].

In our simulation scenario, the physical interfaces are of type IEEE 802.11b. A set of ns-2 extensions [12] that provide a more accurate propagation model, stochastic error models and a new approach for measuring interference are used.

The topology consists of 20 wireless nodes (only mobile ad hoc section is simulated) moving in a square scenario where its size is modified from $25x25m$ to $200x200m$. Nodes follow the Random Waypoint (RWP) mobility model with a minimum speed of $0.5m/s$, a maximum speed of $2.5m/s$ and a pause time of $5s$. Simulation time is chosen so that the results show a low standard deviation.

As unicast routing protocol we use well-known AODV [13]. In the case of SSDP simulations we also need a multicast routing protocol. Therefore we use multicast AODV (MAODV) [14], which is a natural extension of AODV supporting *core based* bidirectional multicast tree creation and maintenance. Both network diameters of AODV and MAODV are set to a high value (32 and 10 hops respectively), so that they do not influence the results.

For SSDP and wSSDP, message size was fixed to an estimate of 250 bytes. We also set the time interval at which alive messages are sent to $50s$, and the lifetime of alive and response messages is set to $200s$.

As for the wSSDP specific parameters, the *maxHopCount* takes the values two and three. The beat count threshold is disabled.

Finally, nodes are divided into passive (10 nodes) and active (10 nodes); where the active take part in service discovery and the passive ones not. The active nodes are further divided into service users (5 nodes) and service providers (5 nodes). The number of different services is fixed to one, because this will not affect our simulations. The number of discover messages sent during simulation time is also fixed (200). We distribute them so that, in average, all service users issue the same number of discover messages, and all service types are demanded with equal probability.

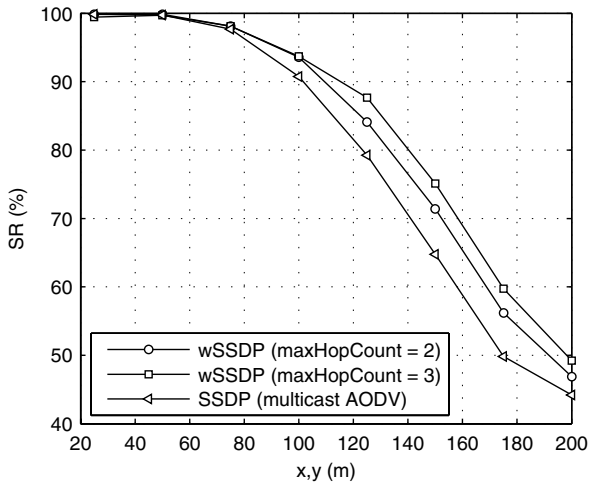

Figure 2: Success rate of wSSDP compared to SSDP.

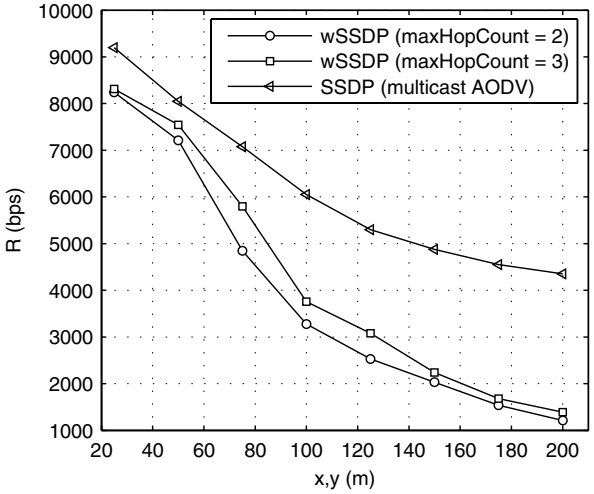

Figure 3: Average bandwidth consumption of wSSDP compared to SSDP.

In order to extract meaningful information from the data, we need to define a set of metrics. In first place, service availability is measured by the *success rate (SR)*, which gives an idea of the percentage of successful queries performed during simulation time. On the other hand, network load is measured by the *average bandwidth (R)*. This includes all packets generated at the application layer (SSDP and wSSDP) and at the routing layer (AODV and MAODV).

Figs. 2 and 3 compare the performance of SSDP and wSSDP by using the success rate and the average bandwidth utilization as metrics.

As a general trend, we see how success rate decreases as the network size increases. The reason for this is twofold. On one side, as the average inter-node distance increases, a higher number of packets are dropped due to errors in the wireless channel; on the other side, due to the nature of the underlying multicast routing protocol, tree partitioning occurs. The latter

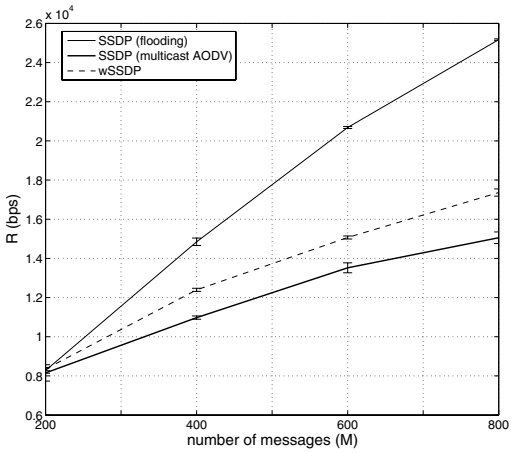

Figure 4: Average bandwidth consumption with increasing service discovery traffic.

is only applicable to SSDP, and thus the better performance of wSSDP.

The average bandwidth consumed decreases as the network size increases. Since the success rate graph shows a lower number of successful service queries, this obviously implies a lower number of responses, and thus the lower traffic generated. The constant multicast tree maintenance traffic generated by MAODV sets a lower limit for the bandwidth consumed by SSDP.

The more efficient design of wSSDP allows for a higher success rate while, at the same time, a lower bandwidth consumed.

Finally, Fig. 4 shows how the wSSDP approach tackles the scalability problem. Results are shown for the $25x25m$ scenario. As the number of service requests sent during the whole simulation time is increased, the average bandwidth consumed obviously increases. Here, it is shown how wSSDP scales only slightly worse than SSDP in combination with MAODV, but much better than SSDP when using blind flooding as the underlying message distribution scheme.

VI. Conclusions

In this paper, we have shown the feasibility of a solution that integrates service discovery in the wired, fixed wireless infrastructure and ad hoc domains. This has been done by extending UPnP, an already existing and popular service discovery architecture.

We have chosen UPnP because of the peer-to-peer architecture of its service discovery protocol, which is adequate for both wired and ad hoc networks; and because it is a commercially available solution with products already shipping in the market. Its modification was motivated by the need to remove multicasting altogether in the ad hoc part of a heterogeneous network. Multicast based solutions are not efficient, and imply a lower lifetime for battery operated systems.

We have shown by simulation how our solution outperforms the original one in service availability and bandwidth consumed. This has been achieved thanks to an intelligent retransmission of service queries at the application layer, bypassing the routing layer. The required extensions for standard UPnP are small, and we believe that the proposed method thus could further extend the usability of UPnP with minimal changes.

References

[1] UPnP device architecture version 1.0, June 2000.

[2] J. Veizades, E. Guttman, C. Perkins, and M. Day. Service Location Protocol, version 2. RFC2608, June 1999.

[3] Jini technology core platform specification, 2003.

[4] Bonjour protocol specifications. Available at http://developer.apple.com, 2005.

[5] P. E. Engelstad, Y. Zheng, R. Koodli, and C. E. Perkins. Service discovery architectures for on-demand ad hoc networks. *International Journal of Ad Hoc and Sensor Networks, Old City Publishing (OCP Science)*, 1(3), March 2005.

[6] C. Perkins and R. Koodli. Service discovery in on-demand ad hoc networks. Internet-Draft (expired), October 2002.

[7] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade. DEAPspace - transient ad-hoc networking of pervasive devices. *Computer Networks*, 35:411–428, 2001.

[8] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha. Toward distributed service discovery in pervasive computing environments. *IEEE Transactions on Mobile Computing*, 5(2):97–112, Feb. 2006.

[9] S. Helal, N. Desai, V. Verma, and C. Lee. Konark - a service discovery and delivery protocol for ad-hoc networks. In *Proc. of WCNC*, New Orleans, USA, March 2003.

[10] S-Y Ni, Y-C Tseng, Y-S Chen, and J-P Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. of MobiCom'99*, Seattle, WA, USA, August 1999.

[11] K. Fall and K. Varadhan. *The ns Manual*. December 2003. Available at *http://www.isi.edu/nsnam/ns/ns-documentation.html*.

[12] M. Wellens, M. Petrova, J. Riihijärvi, and P. Mähönen. Building a better wireless mousetrap: Need for more realism in simulations. In *WONS*, pages 150–157, 2005.

[13] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) routing. RFC3561, July 2003.

[14] C. Perkins and E. Belding-Royer. Multicast Ad hoc On-Demand Distance Vector (MAODV) routing. Internet-Draft (expired), July 2000.