A Specification-based Anycast Scheme for Scalable Resource Discovery in Distributed Systems

Javad Zarrin
Instituto de Telecomunicações
3810-193 Aveiro, Portugal
Email: javad@av.it.pt

Rui L. Aguiar
Universidade de Aveiro
3810-193 Aveiro, Portugal
Email: ruilaa@ua.pt

João Paulo Barraca
Universidade de Aveiro
3810-193 Aveiro, Portugal
Email: jpbarraca@ua.pt

***Abstract*—Anycast is a powerful paradigm for managing and locating resources in large scale distributed computing systems. This paper presents a novel specification-based anycasting scheme for resource discovery in such environments. The effectiveness of our proposal is demonstrated through simulation results, in which we observed a remarkable performance enhancement in different aspects (such as discovery latency, discovery cost, discovery load, etc.) over similar non-anycast based discovery methods.**

I. Introduction

Anycast can be considered as a powerful paradigm for resource discovery in large scale distributed systems. It enables communication between a source node and the nearest (or the best) member of an anycast group. The proximity metric (or the metric for being the best) can be defined in terms of hop count, delay or the minimum amount of load [1]. However, at present, there are challenging issues that prevent easy adoption and deterministic deployment of anycasting in general and particularly for reliable and scalable resource discovery application. These limitations include the issues such as followings:

CH#1: Lack of support for session-oriented communications which makes anycasting inappropriate specfically for stateful applications [2].

CH#2: Lack of support for globally scalable anycast routing in the current network routers, due to the fact that routing paths to any-cast groups cannot be aggregated [3], [4].

CH#3: Dynamic nature of anycast creates a significant overhead for updating and maintenance of the anycast groups specifically for discovering resources in large scale dynamic distributed computing environments (i.e., including joining, leaving or failing nodes/resources) [3], [4].

The main contribution of this paper is to propose a novel specification-based anycasting scheme which deals with the aforementioned challenges for discovering resources in large scale computing environments. The rest of this paper is structured as follows: Section II proposes a system model for deploying anycast for a hierarchical attribute-based resource discovery. It also provides an overview of the proposed anycasting scheme. Section III presents the details of our algorithm. Section IV summarizes the evaluation results and Section V provides our conclusion.

II. Specification-based Anycasting

In this paper, we are interested in proposing a scalable resource discovery approach based on anycasting to discover processing resources (e.g. a core) in large scale many-core-enabled computing environments which are saturated with huge number of heterogeneous distributed resources, connected using heterogeneous interconnection and network access. For doing this, in the first step, we describe resources (i.e. computing resources) in the system based on their specific attributes (i.e. computation and communication characteristics) in a multi-layer hierarchy.

The depth of the hierarchy (i.e. number of layers in resource description model) and the definition of each layer might range from very high level (e.g. super clusters, clusters) to very low level (e.g. processing core, ALUs) depending on the architecture designing aspects. In this paper, we assume that our description model contains three levels (layers) of the hierarchy: Core Layer or Inter-Core Level (i.e. layer$_{ln}$ which has attributes such as Data-PU channels, number of Instruction-PU channels, vector length, core clock rate, etc,.), Die Layer or Inter-Chip Level (i.e. layer$_{an}$ which has attributes such as which has attributes such as cache coherence, instruction set architecture (ISA), micro architecture, interconnection network, etc,.) and Node Layer or Inter-Board Level (i.e. layer$_{sn}$ window size, total number of cores, memory size, Die count, etc,.).

The model is conducted by gathering and combining the individual attributes (ranging from more abstract information in the higher layers to more detailed characteristics in the lower layers) in each layer, augmented with information aggregation and summarization techniques, in order to create the layer-stamps . In fact, all specifications (i.e. attributes) of each layer as well as their values are represented by a single layer-stamp.

In the next step, according to the aforementioned resource description model, we divide the distributed resources in the network in a set of distributed hierarchies using a self-organized and self-configured virtual backbone (i.e. overlay) on top of the underlying physical network. Along this line, the unstructured resources in the system are organized within virtual nodes due to their homogeneity and proximity parameters (i.e. their similarities and locations). Subsequently, the individual $vnode$s start to negotiate with each other in a multi-round distributed fashion to seek agreement on the contribution (i.e. module-

role or $vnode$ type) of each party in the overlay hierarchy. As negotiations evolve, each $vnode$ shapes its own system-view by improving and consolidating its own knowledge on the entire system. The resulted overlay contains three different types of virtual-nodes: leaf-nodes (LNs), aggregate-nodes (ANs) and super-nodes (SNs) which take position in layer$_{ln}$, layer$_{an}$ and layer$_{sn}$ of the hierarchy respectively. Each LN maintains its own layer-stamp (i.e. leaf-stamp) which demonstrates all the common characteristics of resources (i.e. processors) represented by the leaf-node. Similarly, ANs and SNs maintain aggregate-stamps and super-node-stamps on behalf of their LN members and AN members accordingly.

Resource discovery in LN and AN layers is performed by using a DHT-based approach and a probability-based mechanism which is presented in our previous work [5]. But for resource discovery in super-node level (i.e. layer$_{sn}$) we propose the shortest path, routing-based anycast method which uses the top level layer stamps (i.e. node's specifications in SN layer) to create any-cast groups while nodes in this layer are able to dynamically adjust their own interest any-cast group based on the required resource conditions provided by incoming discovery requests from the lower layers. We define SN nodes as the resource providers (RP$_{sn}$s) which provide Super-node Query-Management Services (SQMS) to both entities in their local cell (i.e. a group of vnodes which are sharing a common SQMS-ID) and the other SQMS providers in the system.

CH#1 denotes that the stateful communications cannot be directly supported by the native network-level anycasting since there is no guarantee that subsequent packets from the same session arrive in the same anycast server. The reason is that the routing path between certain nodes (e.g. the source and destination node) may vary with the potential changes of the network configuration and consequently the anycast packets may arrive in different members of the anycast group. Our resource discovery approach does not suffer from the issue mentioned in CH#1, since it does not rely on stateful fully anycast-based communication. Instead, a discovery requester (i.e. a SN) initially sends a "Forward" request using anycast to find the closest SN in the anycast group. Once an anycast member receives a "Forward" request it can explicitly communicate to the original requester using the sender's unicast address. Hence, the requester would be able to perform the rest of the transaction using traditional unicast operations.

In order to deal with CH#2, we propose two different approaches for anycasting which are as follows:

Network-level Approach- We extend the functionality of the routers to make a distinction to distinguish between regular unicast routing request and anycast routing request. However this modification is bound to SN routers (i.e. the routers that directly communicate with SN nodes). By doing this, it would be necessary to upgrade the regular routers. Both type of routers also operate as regular routers for non-anycast (unicast) requests. Using this approach, upon receiving a discovery request (from the requesters in the lower layers of the current hierarchy or from other RP$_{sn}$ in the same layer) in RP$_{sn}$, the receiver would be able to forward the request to the closest RP$_{sn}$ by extracting the anycast address of the destination from c_{sn} (i.e. the description of the query conditions in layer$_{sn}$ which is the representative of the specifications of the desired resources in SN-layer) if the c_{sn} requirements are failed to be met in the current SN. The discovery request would be transferred to the lower layers of the current hierarchy whenever the c_{sn} requirements are fully obtained by SN node in top of the hierarchy.

Application-level Approach- Each SN creates its own anycast address by hashing the layer-stamp of the SN node which is the representative of all the specifications of resources in layer$_{sn}$. Using this approach, the anycast address of each SN is a function of the resource specifications in the super-node's layer. The anycast address of a node may change when the node's specifications (e.g. dynamic resource attributes) are modified. SN nodes with uniform specifications advertise the same anycast addresses. With respect to CH#3, the creation and maintenance of the anycast groups are significantly lightweight, since the anycast groups can automatically be created and they can also dynamically be changed without creating any necessity for communication among the group members or group registration. SN nodes advertise their anycast address as well as unicast address to other SN nodes in their neighborhood. The neighborhood can be specified based on proximity metrics such as number of hops or delay. Whenever a RP$_{sn}$ is decided to anycast a discovery request to other potential SNs in the network, it passes the request to its local anycast-resolver which is responsible to determine the unicast address of the best possible destination. Subsequently, the discovery request would be forwarded to the destination by using its unicast address and the regular routing. The anycast-resolver operates as followings: The anycast address of the potential destination for a given discovery request, is extracted from c_{sn} mentioned in the request. Anycast-resolver checks the list of registered SNs in the current node and selects the ones that their anycast address are similar to the anycast address of the given request. In the next step, the unicast address of a SN in the list with minimum number of hops (i.e. minimum delay) is returned to RP$_{sn}$ as the final destination of the request. If the desired anycast destination is not found among the registered SNs, the discovery request is forwarded to the closest SN in the list. If the list is empty, the query is terminated and the proper update (i.e. reply) message would be sent to the original requester. If c_{sn} conditions are not existed for the given discovery request (i.e. when the creation of the anycast address is not feasible), the discovery request is forwarded to the closest SN in the list.

III. Resource Discovery

The behavior of the RP$_{sn}$ is dependent on the type and status of the received discovery message and the characteristics of the given query. Whenever a RP$_{sn}$ receives a query, the query type retrieval is performed and accordingly different mechanisms and procedures are conducted to resolve or redirect the query. The "Forward" event is the most regular communication event within layer$_{sn}$. The "Forward" receiver, as the first step, assesses whether the layer stamp (i.e. the super-node-

stamp) will be qualified due to the c_{sn} query conditions or not. If it is qualified, a downward self-event is triggered in the current super-node where the SQMS provider itself as the event-receiver acts as a QMS provider which conducts the downward query in the lower layers. The SQMS provider, later will receive a response from the lower layers either in the form of upward (if query fails or remains uncompleted in the lower layers) or update messages (if query is resolved or completed). On occurrence of the update event, the RP$_{sn}$s redirect the incoming messages to the orginal requesters. Receiving upward event is exclusively dedicated to RP$_{sn}$s. In fact, on occurrence of an upward event, the RP$_{sn}$ will be ensured that the requirements of the correspondent query can not be met in the lower layers of the current SN's cell and it is needed to direct the query to other remote cells in the system.

Algorithm 1: Anycast based Forwarding in RP$_{sn}$

```
Input: sq_i= The received sub-query message
/* sq:sub-query, nb:neighbor                                   */
if sq_i.type==forward then
    if RP_sn is qualified due to sq_i.c_sn then
        send(sq_i, downward, RP_sn )
    else
        anycast-address=mapped-ac-address(sq_i.c_sn)
        anycast(sq_i,forward, anycast-address)
else if sq_i.type==upward then
    if sq_i.c_sn is existed then
        anycast-address=mapped-ac-address(RP_sn.layer-stamp_sn)
        anycast(sq_i,forward, anycast-address)
    else
        target=random(nb:(nb ∈ SN-Neighbors)∧(nb ∉ sq_i.visited-qms-ids) )
        send(forward, target )
else if sq_i.type==update then
    RP_sn Updates the original requester
```

As we see in Algorithm 1, if RP$_{sn}$ receives an upward query, it means that the receiver have already been qualified for the c_{sn} query conditions, but the query conditions in the lower layers have not been achieved. Accordingly, RP$_{sn}$ sends a forward message to an any-cast address which is extracted from c_{sn} of the given query. The forward message will automatically be redirected to the nearest SQMS provider in the system which has the same any-cast address. Using the proposed anycast scheme significantly reduces the search space for the given query. It automatically limits the search space to only the SQMS providers in the system that certainly would be able to fulfill the c_{sn} query conditions.

IV. Evaluation and Results

In this section, we evaluate the performance of our proposed 3-layered anycast based resource discovery model (ARD3) with respect to scalability (in terms of latency and number of messages per discovery request) and efficiency (in terms of discovery messages and discovery load per node). ARD3 is a 3-layered resource discovery model which employs DHT-based discovery (i.e. a discovery approach based on a variation of Chord DHT) in layer$_{ln}$, probability-based discovery in layer$_{an}$ and anycast-based discovery in layer$_{sn}$. We compare ARD3 to a 2-layered random-walk based resource discovery (DPRW2) which similar to ARD3 leverages DHT and probability based methods in layer$_{ln}$ and layer$_{an}$ accordingly. Despite ARD3, in DPRW2, the queries with c_{sn} requirements are guided in layer$_{an}$ using a random-walk based method instead of anycasting in a separated extra-layer (i.e. layer$_{sn}$).

To do our evaluation, using Omnet++/INET-Framework and OverSim simulation tools, we simulate a distributed dynamic computing environment containing various number of computing resources in which a constant number of resources (i.e. requesters) simultaneously issue the discovery requests to the system. The time interval between each pair of consecutive queries issued by a requester is defined by an exponential distribution. We also assume that each requester issues 10 consecutive resource requests to the system over the simulation time. The discovered resources will be reserved for each discovery request. The reserved resources for each process will be released after execution time period which is defined by a Weibull distribution. Table I presents the simulation parameters for our evaluation.

TABLE I: Simulation Parameters

Parameter	Values
Physical network size	5500-55000 resources
Interconnect topology	Mesh/Torus
Network topology	Random
Interconnect channel datarate	50Gbps
Network channel	100 Mbps
Desired Resources for each Request	3x20
Homogeneity rate of desired resources	33%
Frequency of Target Resources (FTR)	1650
Process Duration by sec	Weibull(λ=3.58,k=2.40)
Querying Interval by ms	Exponential(β=4000)
Consecutive Query Runs per Requesters	10
Rate of Requesters	1%

Figure 1 presents the evaluation results to compare the performance of the proposed anycast-based resource discovery (ARD3) to DPRW2 in different aspects. From Figures 1-a1 and 1-a2 we can see that the average message cost (measured in terms of the number of discovery messages propagated during the search) of ARD3 is slightly lower than that of DPRW2 while the average discovery latency per request is much lower than DPRW2. This shows that ARD3 provides better performance and scalability particularly in terms of discovery latency when varying the number of resources in the system.

This is because (i) ARD3 efficiently divides the exploring space to the anycast groups in a way that queries with c_{sn} requirements are only propagated among the SQMS providers that their specifications in layer$_{sn}$ essentially fulfill the c_{sn} conditions of the given query. This leads to the reduction in the message cost for ARD3 in comparison to DPRW2. DPRW2 provides an efficient probability mechanism to guide queries to the potential matched resources. However, the probability mechanism in DPRW2 is inefficient for queries with c_{sn} requirements, since DPRW2 employs random-walk method to direct queries (with c_{sn} requirements) in the system.

(ii) ARD3 controls the discovery procedure in a more intelligent manner, which consequently saves much unnecessary message overhead. On the other hand, due to the anycast nature of ARD3, SQMS providers are able to effectively guide the given queries to the closest qualified SQMS provider in the system. For ARD3, this significantly reduces the discovery latency for the queries in the system. But for DPRW2, since the selection of the next resource providers (particularly for the

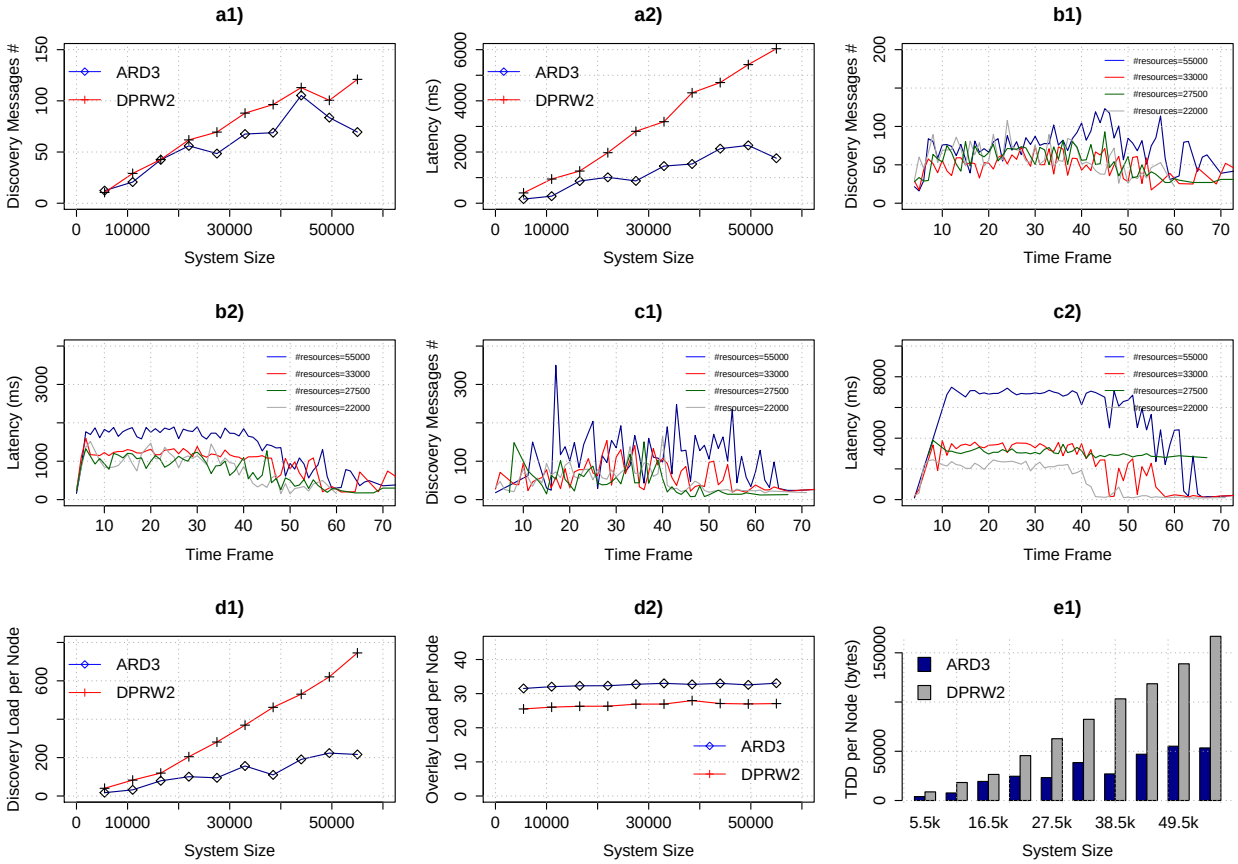

Fig. 1: Comparison between ARD3 and DPRW2: a1&a2) Average number of required discovery messages and discovery latency per discovery request for different system size, b1&b2) Mean number of discovery messages and discovery latency per request per time-frame (1000 ms) over time in different system size for ARD3, c1&c2) Mean number of discovery messages and discovery latency per request per time-frame (1000 ms) over time in different system size for DPRW2, d1&d2) Average discovery load (number of transmitted discovery messages) and average overlay load (number of transmitted messages to create the overlay) per node during simulation time (60000-80000 ms) for various system size, e1) Average transited discovery data per node during simulation time (60000-80000 ms) for various system size.

given queries with c_{sn}) is random, it leads to larger amount of latency in the results.

Figures 1-b1, 1-b2, 1-c1 and 1-c2 show the dynamic behavior of ARD3 and DPRW2 in terms of discovery message cost and latency per request per time-frame over time for various system size. As we can see in these figures and due to the aforementioned reasons, ARD3 provides better performance and scalability than DPRW2 over simulation time. Similarly, in Figure 1-d1 and Figure 1-e1, ARD3 shows better scalability and performance in terms of discovery load (average number of transmitted discovery message) per node and average transmitted discovery data per node during simulation time. However, both ARD3 and DPRW2 provide steady and almost similar behavior in terms of overlay load per node (i.e. average number of transmitted overlay message per node during simulation time). This happens because both approaches employ the same multistage hierarchical overlay algorithm [6] to implement 2-layered and 3-layered architecture for DPRW2 and ARD3 accordingly.

V. Conclusion

This paper presented a specification-based anycasting scheme for resource discovery in large scale distributed computing systems. The evaluation results prove that our proposed approach is widely scalable and it provides better performance and efficiency in comparison to the similar non-anycast based resource discovery methods.

References

[1] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," RFC, United States, 1993. RFC 1546.

[2] S. Bhattacharjee, M. Ammar, E. Zegura, V. Shah, and Z. Fei, "Application-layer anycasting," in INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE, vol. 3, pp. 1388–1396 vol.3, Apr 1997.

[3] T. Stevens, T. Wauters, C. Develder, F. De Turck, B. Dhoedt, and P. Demeester, "Analysis of an anycast based overlay system for scalable service discovery and execution," Comput. Netw., vol. 54, pp. 97–111, Jan. 2010.

[4] F. Hao, E. W. Zegura, and M. H. Ammar, "Qos routing for anycast communications: motivation and an architecture for diffserv networks," Communications Magazine, IEEE, vol. 40, no. 6, pp. 48–56, 2002.

[5] J. Zarrin, R. L. Aguiar, and J. P. Barraca, "Dynamic, scalable and flexible resource discovery for large-dimension many-core systems," Future Generation Computer Systems, vol. 53, pp. 119 – 129, 2015.

[6] J. Zarrin, R. Aguiar, and J. Barraca, "A self-organizing and self-configuration algorithm for resource management in service-oriented systems," in Computers and Communication (ISCC), 2014 IEEE Symposium on, pp. 1–7, June 2014.