Scalable DDS Discovery Protocols Based on Bloom Filters

Javier Sanchez-Monedero, Javier Povedano-Molina &
Juan M. Lopez-Soler

University of Granada

- DDS needs discovery protocols: procedure to put in contact publishers and subscribers

- Discovery is a common problem in distributed networking

- Discovery can compromise the scalability of DDS

- The state of the art in discovery is important to review

New functional relationships and network topologies to consider: centralized, pure and hierarchical peer-to-peer

Optimization techniques: locality, cache and Bloom filters

- **Main goal**: to improve the scalability of DDS discovery protocols by using bloom filter technology and researches in peer-to-peer (P2P)

2.The State of the Art in Discovery

- Reference Discovery procedures

  - *Chord: a scalable peer-to-peer lookup protocol for internet applications*.

  - *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems*

  - *OSDA: Open Service Discovery Architecture* and *Distributed Search in Semantic Web Service Discovery* thesis.

  - *Directory Facilitator and Service Discovery Agent (DFSDA)*.

  - *Kademlia: A Peer-to-Peer Information System Based on the XOR Metric*.

  - LDAP directories.

  - *Peer-to-Peer* networks like ed2k or Gnutella

- Adopted terminology

  - Client ~ Node ~ Agent

  - Server ~ SuperNode ~ DirectoryFacilitator

- ## Discovery procedure issues
  - Functional relationship between the entities of the network
    - Client-Server
    - Centralized peer-to-peer (P2P)
    - Pure P2P
    - Hierarchical P2P
  - Network entities distribution
    - Unstructured topology (ed2k)
    - Ring in Distributed Hash Table (DHT) implementations (Chord, Pastry…)
    - Trees (LDAP Directories)
    - Hierarchical combination (OSDA, DFSDA)
  - Database and information representation and storing
    - DHT
    - Bloom filters
  - Learning, cache and local versus global discovery

- Pure P2P are theoretically scalable, but not in practice. For instances

    - *Chord* does not scale in scenarios with 2000 nodes

    - *Pastry* can turn out to be non-functional given the incurred management traffic overhead.
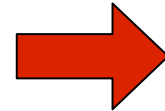
        Bjurefors, F. Larzon, L.A. Gold, R.: "*Performance of Pastry in a Heterogeneous System*". Proceedings of the Fourth International Conference on Peer-to-Peer Computing, 2004.

- Some hierarchical combination uses DHTs to maintain a structure for global discovery
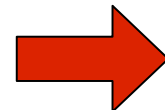
2.The State of the Art in Discovery

- State of the art main conclusions:

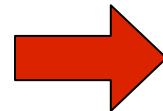|		
Pure P2P does not scale well in practice	→	Hierarchical P2P
Network entities distribution	→	Hierarchical combination (OSDA)
To reduce network traffic and storage in Database and information representation	→	Bloom filters

Contents

3.1. Introduction to Bloom filters

- High Level Ideas

    - Because DDS entities lists can be long and unwieldy to manage

    - We move from: "*Give me the list of what you have*"

    - to the paradigm of: "*Give me information so I can figure out what you have*"

    - Bloom filters allow to achieve the new paradigm

- Given a set S = {x1,x2,x3,...xn} the problem is to answer queries of the form:

    Is element y in the set S?

- Bloom filter (BF) technology provides answers in:

    - "Constant" time (time to hash).

    - Small amount of space.

    - But with some probability of being wrong (false positives).

3.1. Introduction to Bloom filters

Start with an m bit array, filled with 0s.

V | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Hash each item x_j in S k times. If $H_i(x_j) = p$, set $V[p] = 1$.

V | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

To check if y is in S, check V at $H_i(y)$. All k values must be 1.

V | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

Possible to have a false positive; all k values are 1, but y is not in S.

V | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

Source: Michael Mitzenmacher, *"Codes, Bloom Filters, and Overlay Networks"*

Element a

Bit Vector v

$H_1(a)=P_1$

$H_2(a)=P_2$

$H_3(a)=P_3$

$H_4(a)=P_4$

m bits
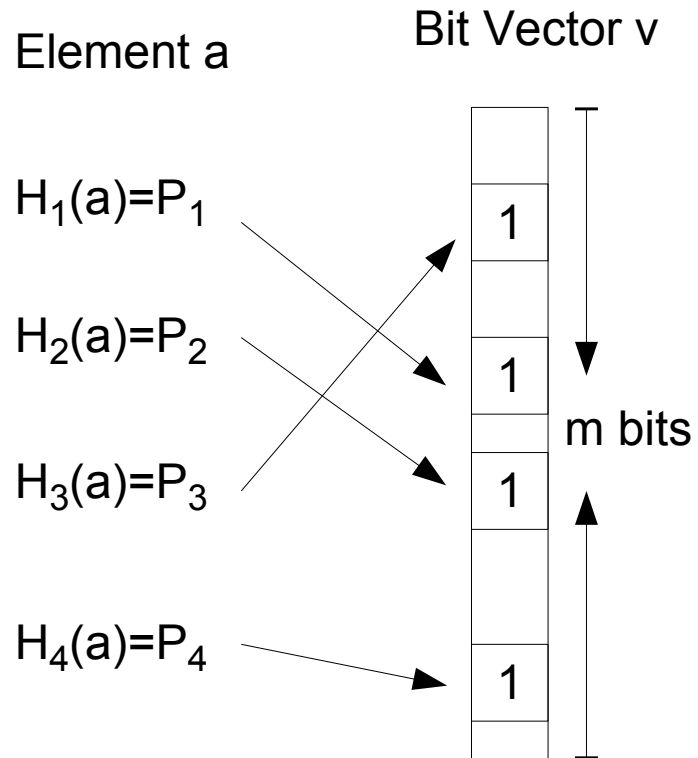
- Given a
  - Vector v of size m
  - k hash functions, $H_i(x)$
  - *A set S with n elements*

The probability of a false positive can be expressed as

$$f = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-kn/m}\right)^k$$

Some Bloom filters additional issues

- Bloom filters work better when they are not full

- Two design parameters must be tuned

    - m the size of the filter

    - k the number of hash functions

- It is needed to deal with false positives

- Note that to delete an item implies re-building the entire filter

- Depending on m and the size of a key, the updates should be done in different ways:

    - To send the entire filter

    - To send just the updated information change

Contents

1. **Introduction**

2. **The State of the Art in Discovery**

3. **Bloom filters**

   3.1. Introduction to Bloom filters

   3.2. BF practical uses

4. **SDP-Bloom**

5. **Hierarchical P2P in DDS Discovery**

6. **Conclusions**

- The first BF use was in Spell Check (~70s):
  - The filter represents a list of valid words
- Summary Cache (Cache Digest in Squid)
  - Squid is a high-performance proxy caching server for web clients, supporting FTP, gopher, and HTTP data objects.
  - A Cache Digest is a summary of the contents of an Internet Object Caching Server. It contains, in a compact (i.e. compressed) format, an indication of whether or not particular URLs are in the cache.
- OSDA (*Open Service Discovery Architecture*)
  - Similar use to Cache Digest
  - Used to reduce network traffic in local and global discovery

Noura Limam, Joanna Ziembicki, Reaz Ahmed, Youssef Iraqi, Dennis Tianshu Li, Raouf Boutaba, and Fernando Cuervo. *Osda: Open service discovery architecture forefficient cross-domain service provisioning*. Comput. Commun., 30(3):546–563, 2007.

Contents

- DDS SDP (*Simple Discovery Protocol*)

- DDS uses DDS publication for discovery purposes (different ports are used)

- Two consecutive process:

  – First: Participant discovery protocol

  – Secondly: Endpoint discovery protocol

- Use of special (Built-in) *Topics* and *DataReader/DataWriter* to advertise participants/publications

- The Discovery process can be tuned with specific *QoS policies*

- Discovered participants/publications are stored in a local database

- The discovery process is started from a list of known host
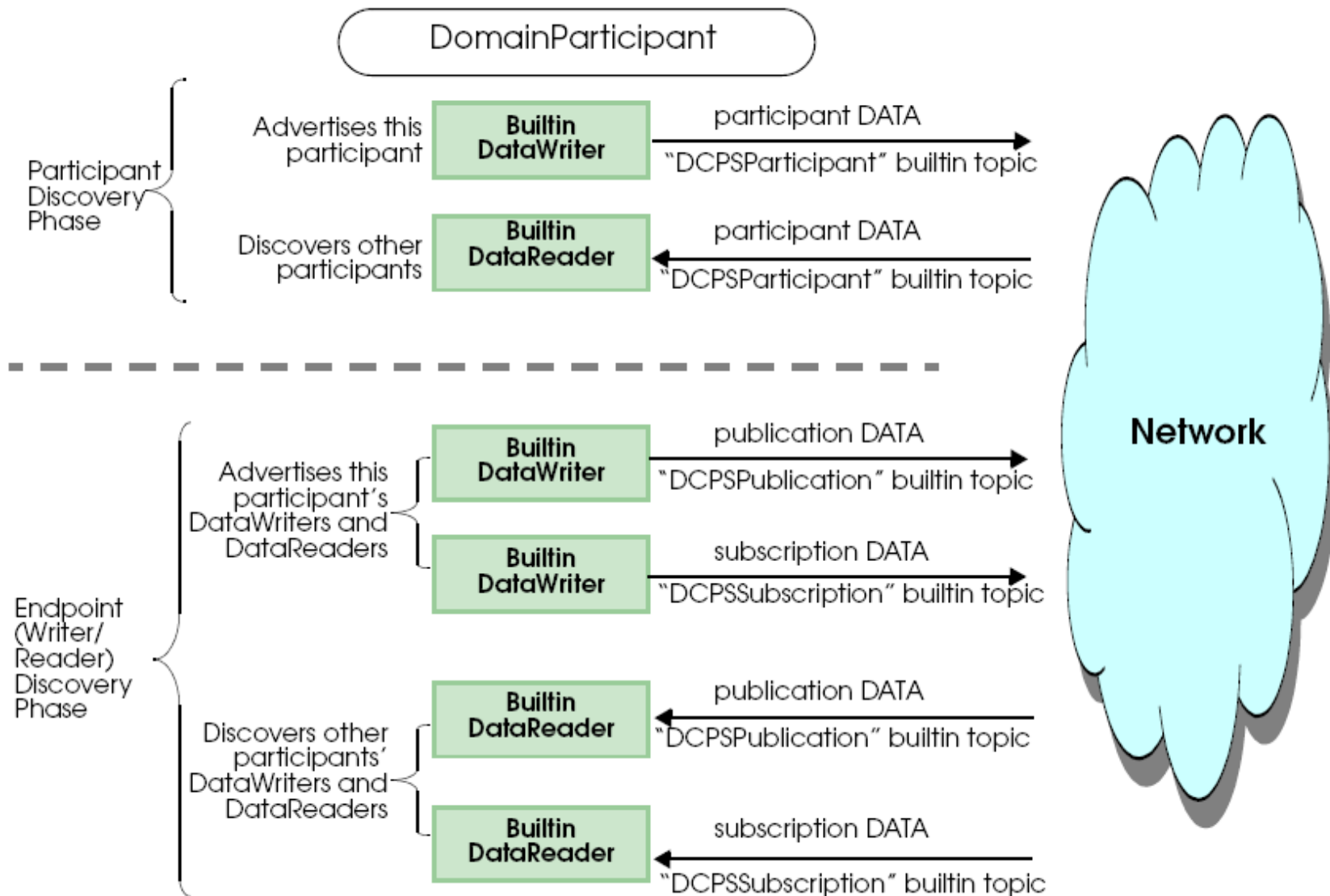
Figure: The Discovery Process phases. Source: DDS-RTPS Interoperability Wire Protocol" document ptc/2006-08-02
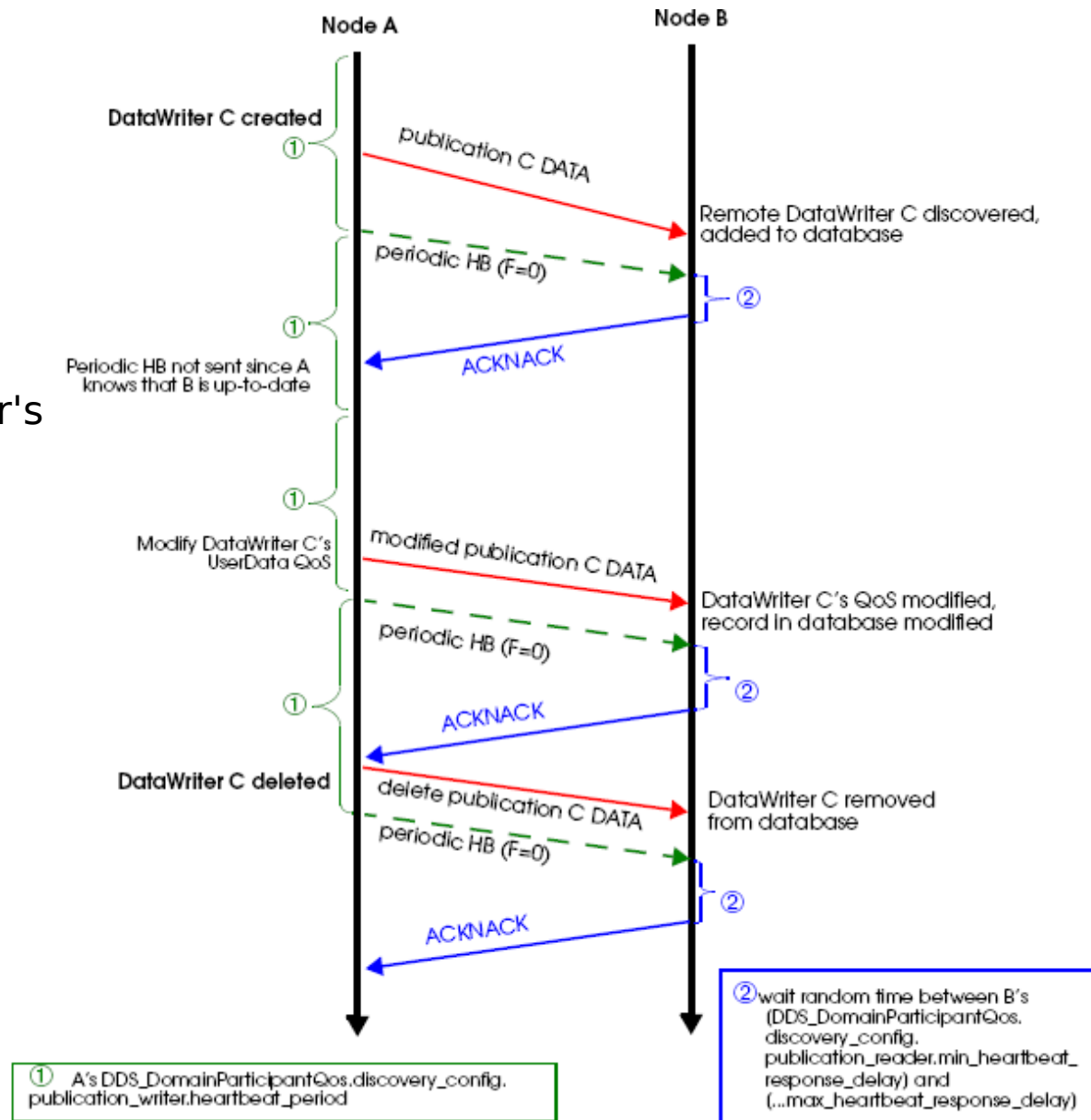
- The Participant Discovery Protocol (PDP) (1st discovery stage)

  - PDP is restricted to discover participants in the same *domain*

  - PDP is based on best-effort communications

  - PDP uses `DCPSParticipant` built-in topic to exchange information about participants

  - Participant DATA are sent to known peers when a `DomainParticipant` is created/deleted. The remote participant stores the information in an internal database.

  - When new Participants DATA are received, the own Participants DATA are sent

- **Endpoint Discovery Protocol (EDP) (2nd Discovery Stage)**

  - When a new participant is discovered, its EndPoints must be matched

  - **Built-In Topics:** `DCPSPublication, DCPSSubscription`

  - Two pairs of Built-In Endpoints for:

    - Advertising local Endpoints

    - Discovering remote Endpoints

  - Uses ACK-NACK mechanism for reliability

  - Use of piggybacked Heartbeat to determine liveliness of endpoints

Source: RTI DDS User's
    Manual
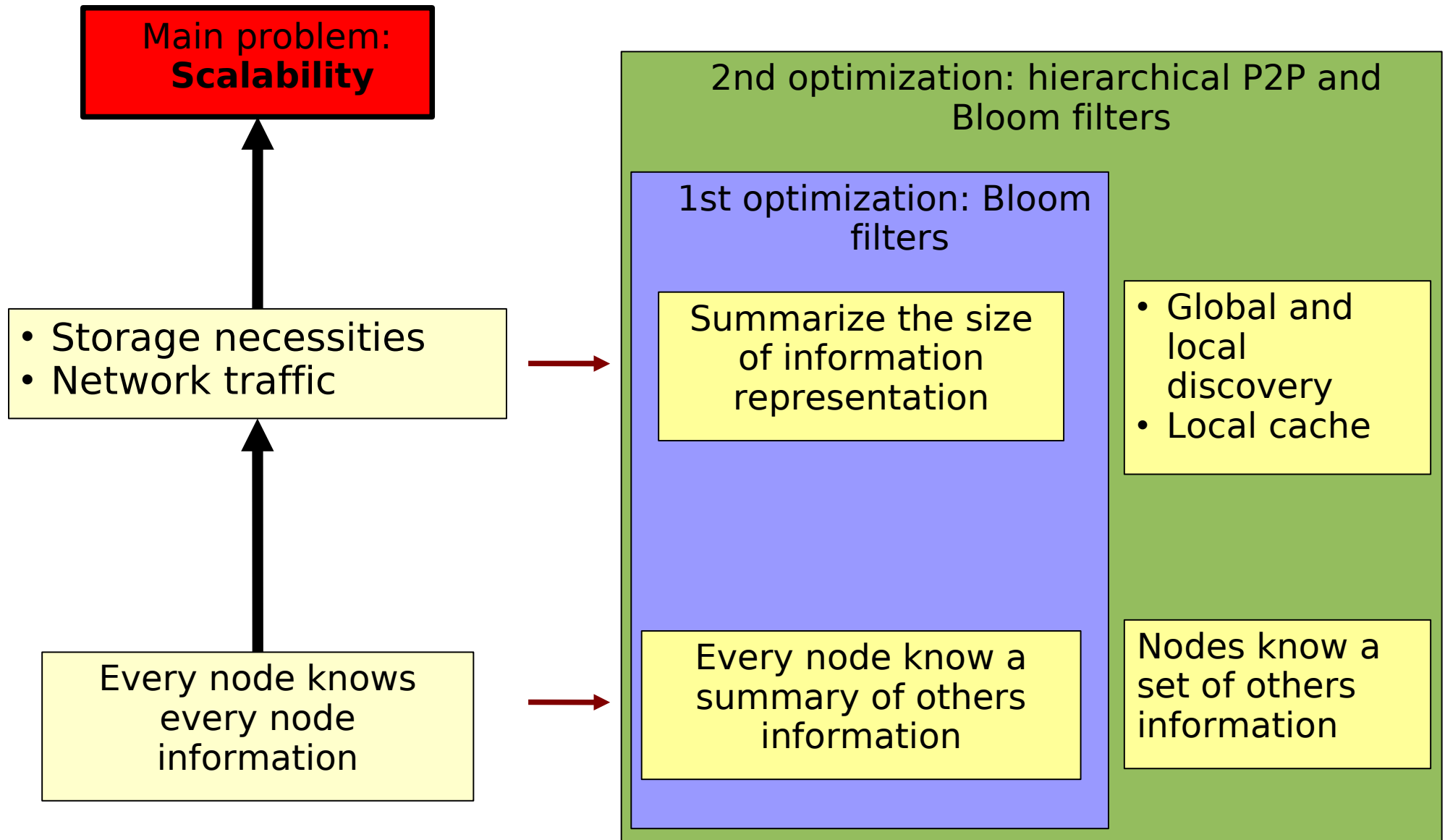
Main problem: Scalability

- Storage necessities
- Network traffic

Every node knows every node information

2nd optimization: hierarchical P2P and Bloom filters

1st optimization: Bloom filters

Summarize the size of information representation

Every node know a summary of others information

- Global and local discovery
- Local cache

Nodes know a set of others information

Contents

- In DDS, mostly for those scenarios with a big number of Endpoints in each Participant, we identify two problems to be solved:

  - Memory requirements in nodes.

  - Network traffic.

- We call SDP-Bloom our new SDP variant, which is based on Bloom filter technology

- Caused by interchanging and storing the complete remote list of Endpoints: "give me all information you have"

- In SDP-Bloom each Participant will send its own Bloom filter which encodes its Endpoint set: "give me the information to know what you have"

- In SPD-Bloom each Participant stores information of all entities but with significantly smaller size.

- Similarly, network traffic is also reduced.

- However, new problems must be solved:
  - False positives
  - EndPoint Updates
  - CPU cost for building the BF

- When BFs should be sent to the other Participants?
  - First approach: to include the filter in Participant DATA messages of the PDP, and to use EDP to deal with updates and false positives
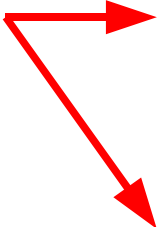 - Second approach: to sent the BF in the EDP (seems to be closer to the DDS standard)

Contents

4.3.SDP-Bloom algorithm description

1) Just before enabling the new Participant, a BF is created for representing the Endpoints set.

2) Each time a new local Endpoint is created and enabled, it is added to the filter.

3) The Participant starts discovering remote Participants by using Simple Participant Discovery Protocol.

4) The Participant asserts all new discovered Participants and applies the Endpoint Discovery Protocol.

5) For each remote Participant, it sends its BFs instead of the complete set of local Endpoints. In the same way, it will receive the remote Participant filters and will proceed to match its Endpoints with each filter.

- Design decisions: the *m/n* ratio in our Bloom filter

  - The larger the m/n ratio, the lower error rate

  - While the lower the m/n ratio, the lower BW and memory requirements

  - The relation *n* (Endpoints/Participant) is the number of keys to include in the filter.

Error Rate	N (Keys)	M (Required Size)	BF (Bytes)
0.01	20	211	27
0.01	100	1053	132
0.01	500	5262	658
0.001	20	409	52
0.001	100	2043	256
0.001	500	10215	1277

  - Note the Error Rate (*f*) is the probability of obtaining a false positive Endpoint in a given Participant.

4.4.SDP-Bloom design decisions

- Design decisions: information updates

  - Participants discovery still working like SDP (soft-state)

  - Endpoints are updated publishing changes on-demand:

    - Initially, we consider that the Endpoints/Participant relation is so small that is "cheaper" to send the new filter each time.

    - Deleting imply to rebuild the filter.

      - Counting BF supports deletion, but it probably use too much space

      - At the moment we consider to use original Bloom filters

- Design decisions: managing false positives

  - The matching process consist in test if a Topic bellows to a filter.

  - For each desired Endpoint which matches with a filter we can assert the desired remote Endpoint and try to connect with it.

  - Normally, we do not have problems, but if it fails we can:

    - Obtain the complete remote Endpoint list for this Participant, as Endpoint Discovery Protocol does.

    - Ask the remote Participant about this Endpoint.

    - Deduce that we got a false positive and do no assert the Endpoint.

start

Build or update the BF

Send BF to the discovered Participants

Match local Endpoint with remote BF

Try to Subscribe

yes

no (fail)

Assert remote Endpoint

Deduce false positive

Wait a refresh time

Local changes

No

No

No

New remote information
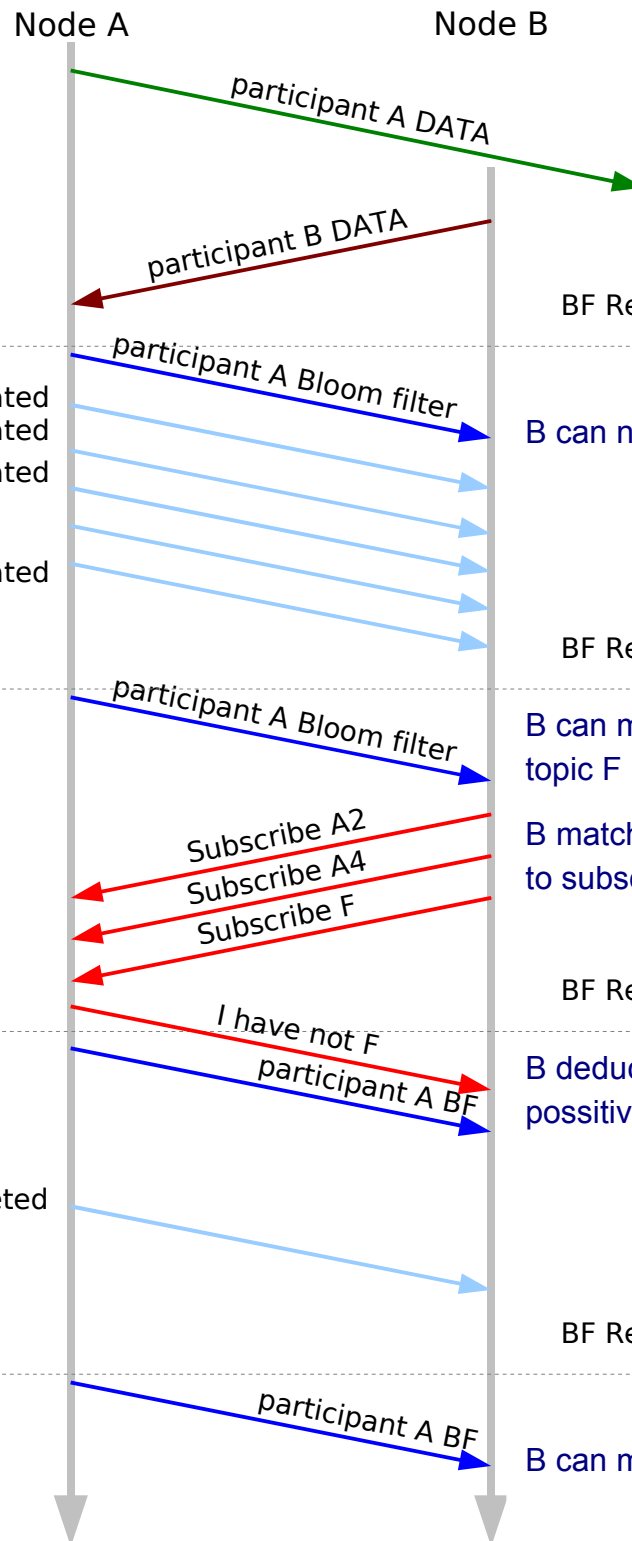
End?

SDP-Bloom

- Participant A whish to inform about its Endpoints
- Participant B whish to mach A2, A4 and F
- We omit PDP information

Node A   Node B

participant A DATA →

← participant B DATA

BF Refresh period

participant A Bloom filter →   **B can not match any topic**

DataWriter A1 created
DataWriter A2 created
DataWriter A3 created
...
DataWriter An created

BF Refresh period

participant A Bloom filter →   **B can match A1-N topics, topic F is a false positive**

Subscribe A2
Subscribe A4
Subscribe F   **B matchs A2, A4, F and tries to subscribe A2,A4,F**

BF Refresh period

I have not F →
participant A BF →   **B deduces it was a false possitive and annotates it**

DataWriter A3 deleted

BF Refresh period

participant A BF →   **B can match A1,A2,A4-N topics**

Storage

	SDP	SDP-Bloom
Node B		
Node C		

4.6.SDP-Bloom Scalability Analysis

Scenario name	Participants (P)	Topics (T)	Endpoints (E)
Small System	100	400	2000
Medium System	1000	1000	20000
Large System	10000	9100	200000

Participant messages (unicast)
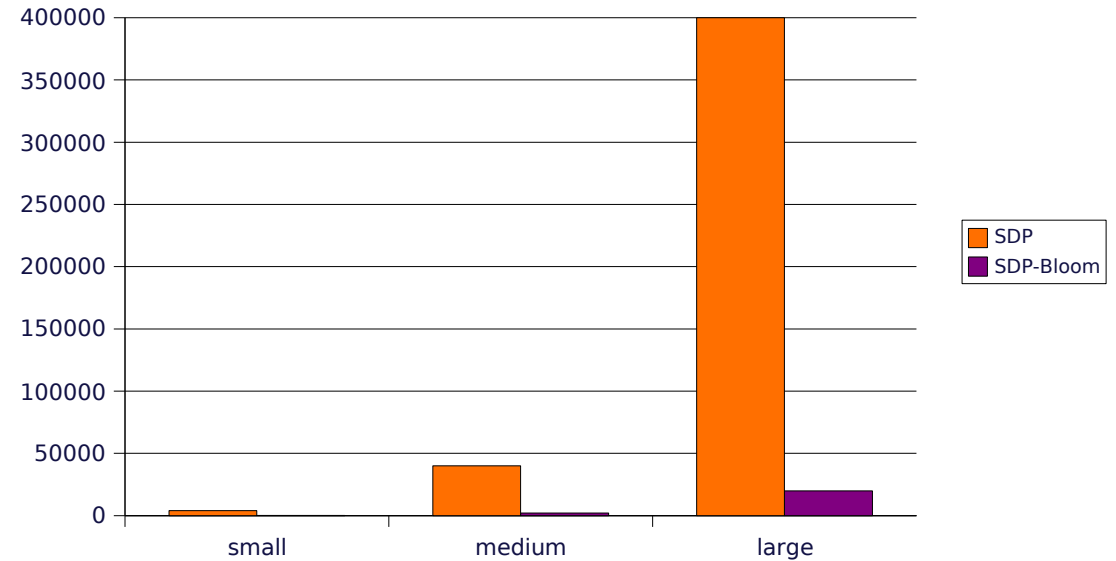
Participant memory consumed (KB)
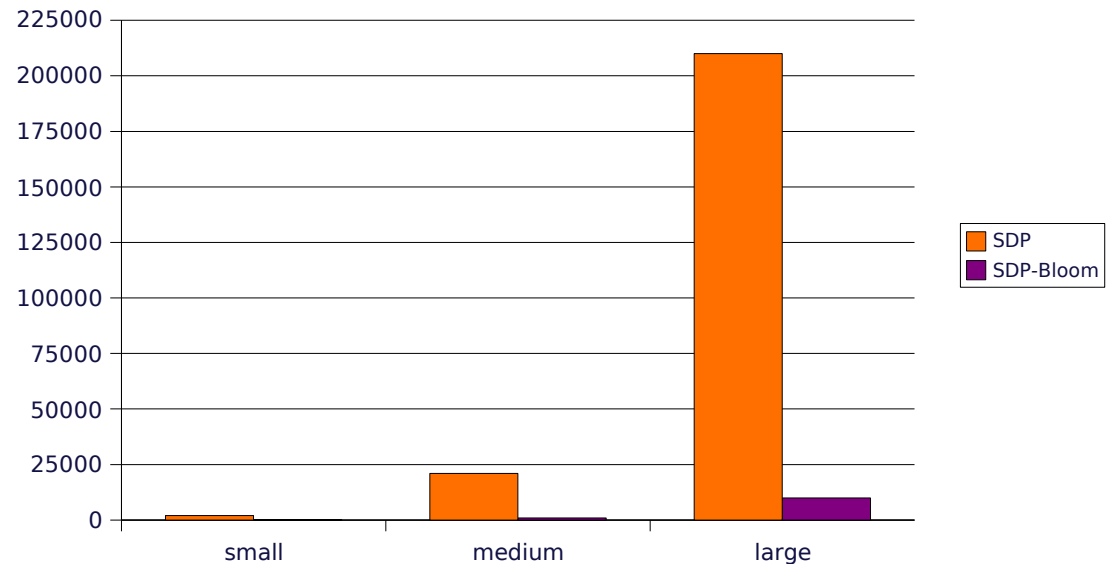
1. Introduction

2. The State of the Art in Discovery

3. Bloom filters

4. SDP-Bloom

5. Hierarchical P2P in DDS Discovery

6. Conclusions

Motivation and overview

- Pure P2P: DHTs and overlay P2P networks provide replication, fault tolerance and scalability up to 2000 nodes

- Hierarchical P2P

  - Global discovery. Global information stored in a DHT.
    - SuperNodes do global discovery according to Nodes petitions
  - Local discovery. Each SuperNode "adopts" a set of Nodes
    - The adoption depends of the key assigned in the DHT
    - SuperNodes and Nodes do local discovery in a Domain

- Scalability of Hierarchical P2P:

  - Participants with a short life time should not be included in DHT

  - Only a subset of nodes will be selected in the DHT, then the DHT works better in terms of the cost of maintaining the self structure.

  - Local discovery allows using local cache in each group

The Distributed Hash Table in Pure P2P

- Each peer stores a subset of (key, value) pairs in the system

- Core operation: Find node responsible for a key

  - Map key to node

  - Efficiently route insert/lookup/delete request to this node

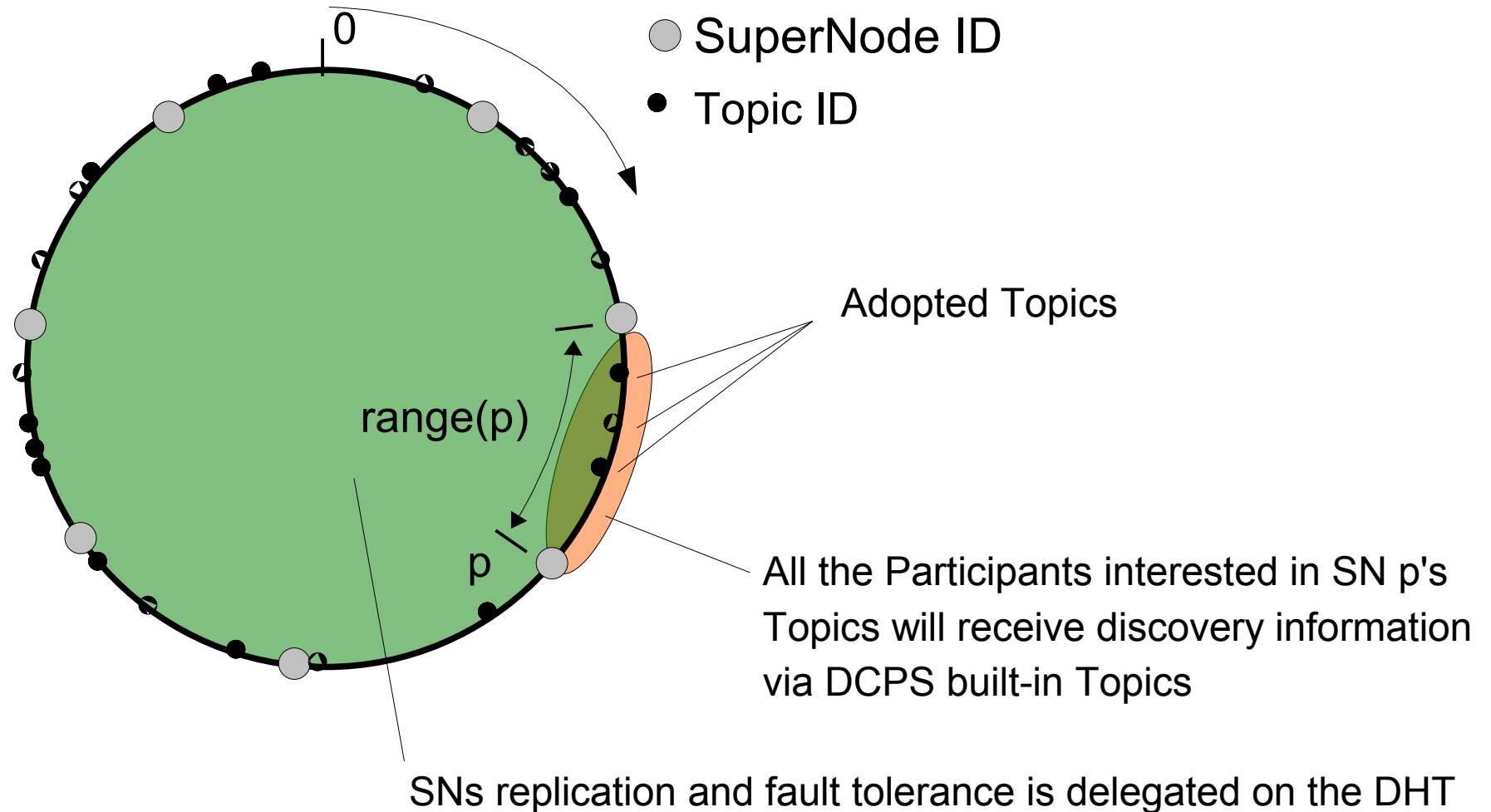- Overlay P2P networks: Key Based Routing (KBR) networks

peer ID

resource key

0

range(p)

p
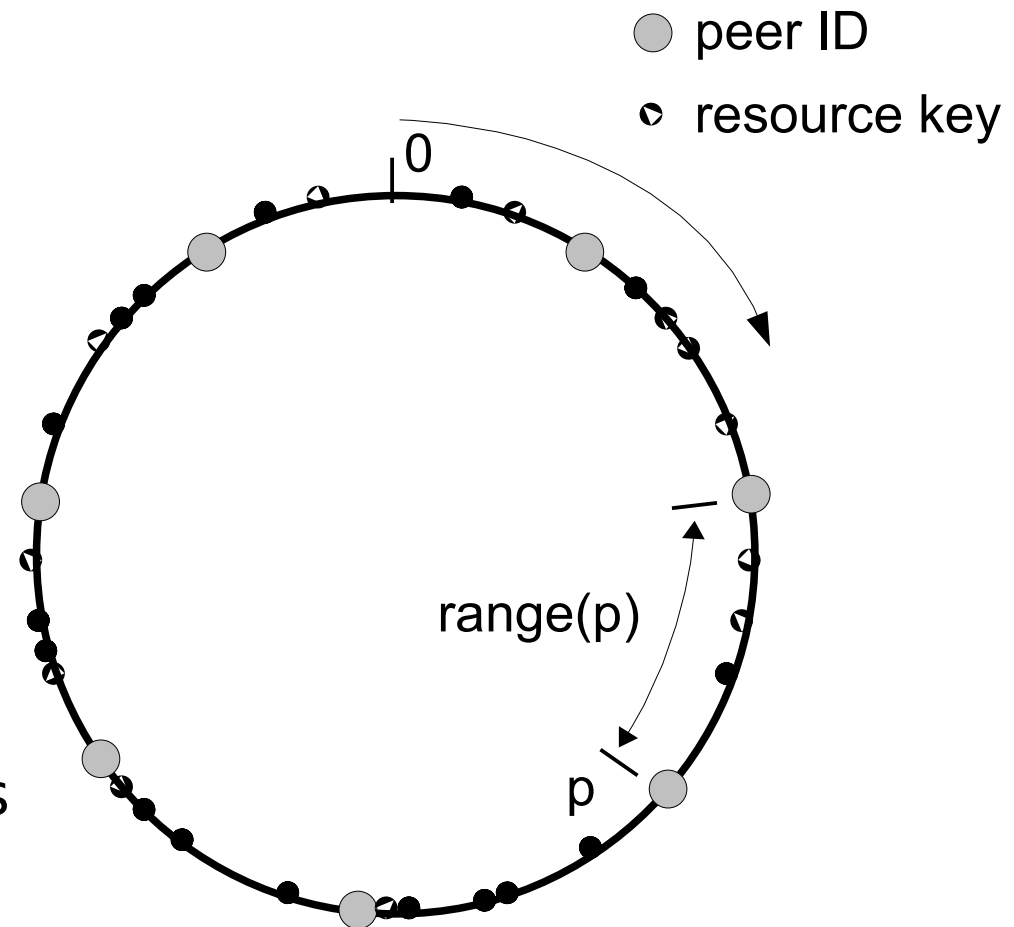
The Distributed Hash Table in Hierarchical P2P and DDS:

- The nodes in DHT will be the SuperNodes in discovery

- The resources Key will be a Topic

- The value will be the list of Participants which are interested in a Topic

- DHT (key,value) = (Topic, Participant list)

- A SuperNode *adopts* a set of Participants interested in the Topics which the SN manages.

- The **replication** and **fault tolerance** of SuperNodes is delegated to the DHT implementation

SuperNode ID

Topic ID

Adopted Topics

range(p)

p

All the Participants interested in SN p's Topics will receive discovery information via DCPS built-in Topics

SNs replication and fault tolerance is delegated on the DHT

Lookup (T,Participants) in the DHT is O (Log N), where N = number of SuperNodes

5.Hierarchical P2P in DDS Discovery

Discovery Process example:
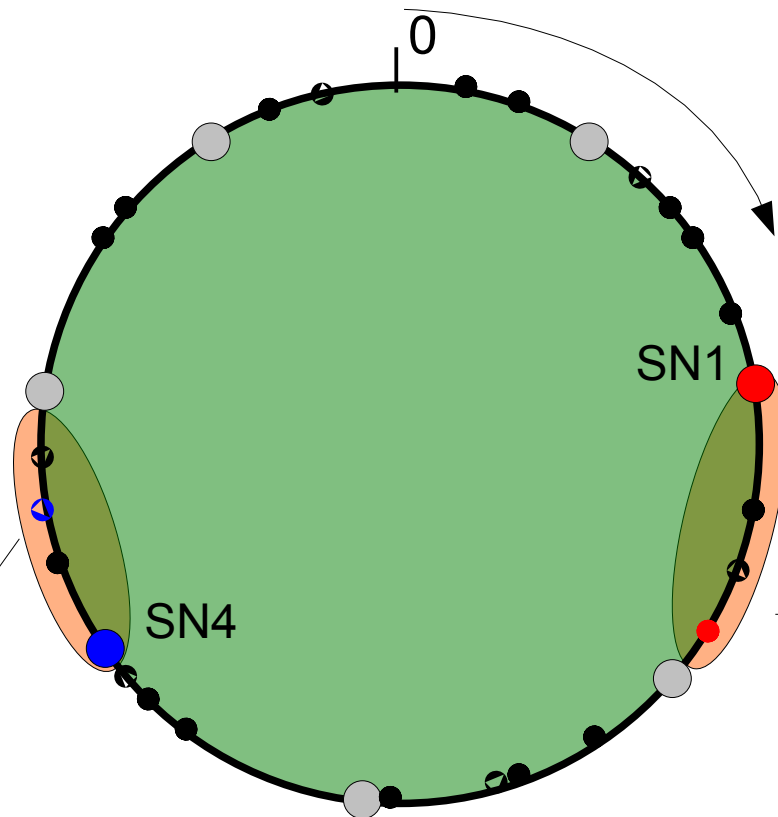
SuperNode SN1:
• Adopts TopicFoo and manages TopicFoo discovery issues

New Participant interested in TopicFoo:
• The hashed TopicFoo involve been adopted by SN1
• Creates `DCPS_TopicFoo` built-in endpoints for discovery relation with SN1
• SN1 add the Participant to the (TopicFoo,Participant list) and notifies existing Participants

New Participant interesting in TopicBar
• The hashed TopicBar involve been adopted by SN4
• Created `DCPS_TopicBar` built-in endpoints for discovery relation with SN4

1. **Introduction**

2. **The State of the Art in Discovery**

3. **Bloom filters**

4. **SDP-Bloom**

5. **Hierarchical P2P in DDS Discovery**

6. **Conclusions**

- In this work we propose to use Bloom filters in DDS discovery.

- While in SDP the consumed traffic depends on the number of topics, by adopting our solution this dependence is relaxed.

- The improvement will be better as the number of topics grows.

- BFs can reduce network traffic and storage requirements

- We have shown preliminarily that hierarchical P2P can be used in DDS discovery.

- We provides replication, better fault tolerance and balanced load.

  - Therefore, it promises even better scalability

Thank you very much

Bibliography

- Burton H. Bloom. *Space/time trade-offs in hash coding with allowable errors*. Commun. ACM, 13(7):422–426, 1970.

- Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. *Summary cache: a scalable wide-area web cache sharing protocol*. IEEE/ACM Trans. Netw., 8(3):281–293, 2000.

- Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. *Chord: a scalable peer-to-peer lookup protocol for internet applications*. IEEE/ACM Trans. Netw., 11(1):17–32, 2003.

- Antony Rowstron and Peter Druschel. *Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems*. Lecture Notes in Computer Science, 2218:329, 2001.

- Noura Limam, Joanna Ziembicki, Reaz Ahmed, Youssef Iraqi, Dennis Tianshu Li, Raouf Boutaba, and Fernando Cuervo. *Osda: Open service discovery architecture for efficient cross-domain service provisioning*. Comput. Commun., 30(3):546–563, 2007.

- Michael Mitzenmacher, *Codes, Bloom Filters, and Overlay Networks*. http://www-math.mit.edu/~steng/18.996/MIT_Talk.ppt

- Fredrik Bjurefors, Lars-Ake Larzon, and Richard Gold. *Performance of pastry in a heterogeneous system*. In P2P '04: Proceedings of the Fourth International Conference on Peer-to-Peer Computing (P2P'04), pages 278–279, Washington, DC, USA, 2004. IEEE Computer Society.

- Celeste Campo. *Directory facilitator and service discovery agent*, 2002.