Green Resource Allocation Algorithms for Publish/Subscribe Systems

Distributed Systems Course Paper Presentation

Artem Tsikiridis

December 14, 2015

cs.grad.aueb

Outline

Background

Green/Energy Saving Algorithms

- Green → sustainable energy practices
- Green IT initiatives are a priority

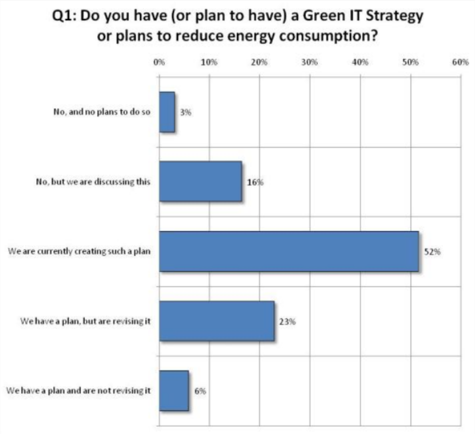


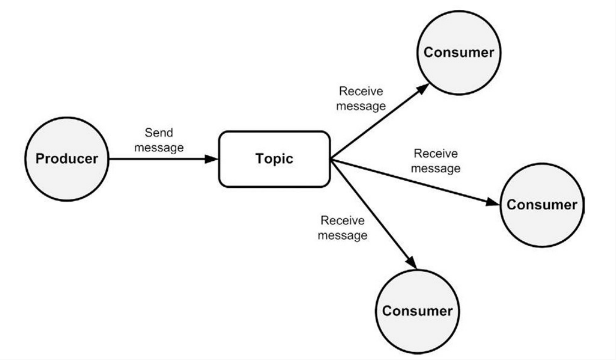
Figure 1: source: Symantec Green IT Report, 2009

Efficient use of resources leads to **lower** IT operational costs.

Message Broker Pattern

- Module translating messages from messaging protocol of the sender to messaging protocol of the receiver
- Widely used in computer networks
- Minimizes mutual awareness of applications $\rightarrow$ effective decoupling
- Apache ActiveMQ, Apache Kafka, Celery

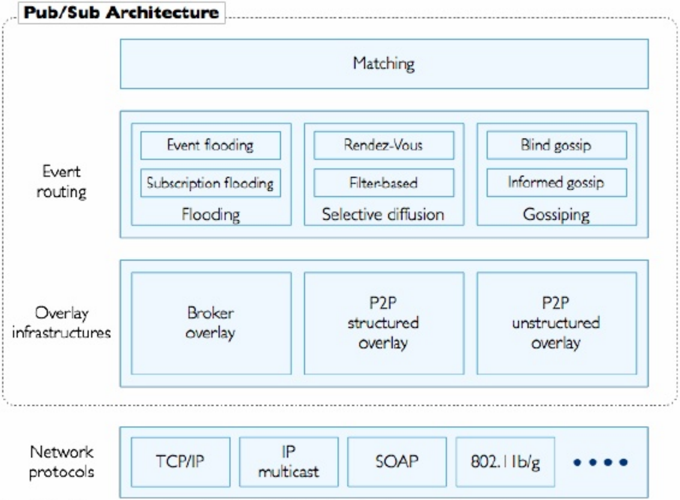
Publish/Subscribe Systems



- Publishers (senders) put messages to classes
- Subscribers (receivers) express interest in classes of messages
- Broker-based systems: publishers post messages to message broker, subscribers register subscriptions with that broker
- Used by Green IT implementors, provides network scalability

Generic Publish/Subscribe Systems

■ A generic architecture of a publish/subscribe system:



From "Distributed Event Routing in Publish/Subscribe Communication Systems: a survey" R.Baldoni, L. Querzoni, S. Takoma, A. Virgillito midlab tech.rep. 2007, to appear (springer)

Middleware Laboratory
MIDLAB

Overview/Prerequisites

Overview

Main Goal: **minimize** number of brokers, while **maximizing** resource utilization of allocated brokers

- To minimize number of brokers $\rightarrow$ minimize number of messages
- By minimizing number of brokers $\rightarrow$ minimize size of network
- As a result, publication hop count is improved (less complexity to place a publication)

How? Design and implement a 3-phase scheme to **reconfigure** the publish/subscribe system.

Phases of 3-phase scheme

- Phase One: Gathering of performance and workload information from the network using bit-vectors.
- Phase Two: Allocation of subscriptions to brokers using the info from Phase One
- Phase Three: Recursively construct broker overlay with already allocated subscriptions
- After 3-phase scheme: Strategically relocate publishers to new broker overlay

Solving an optimization problem which is proven to be $\mathcal{NP}$-complete.

Required components for all phases

- **CROC**: Coordinator for Reconfiguring the Overlay and Clients
  - External pub/sub client
  - Connects to any broker to collect info
  - Executes phases 2 and 3
  - Orchestrates reconfiguration
- **CBC**: CROC Back-end Component
  - Integrated into each broker
  - Responds to commands sent by CROC (information requests etc.)

Phase One

Phase One: Information Gathering

- Information gathering protocol implemented using publish/subscribe.
- CROC connects to first broker → sends *Broker Information Request* message
- Broker broadcasts it to all neighbors
- Broker replies to CROC with a *Broker Information Answer* when:
  - No neighbors to forward the BIR
  - Has recieved all BIA from other neighbors

Phase One: BIA

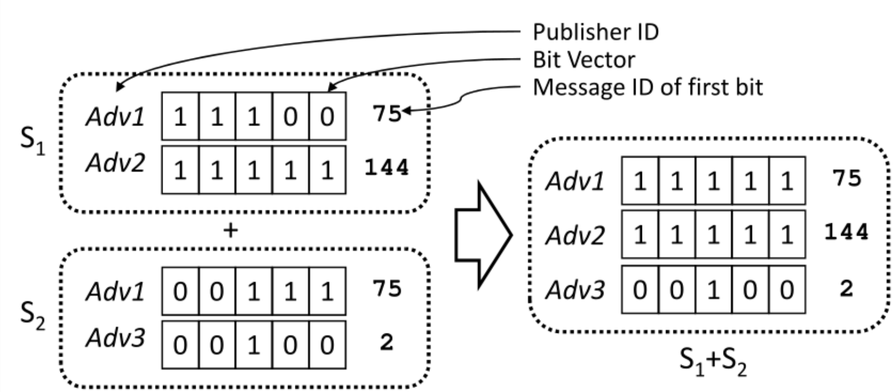
A Broker Information Answer (BIA) consists of:

- **URL**: Useful to reassign subscribers in phases 2 and 3
- **Matching delay function**: Linear function enabling CROC to predict input load of broker
- **Total Output Bandwidth**: CROC uses it to predict output load of the broker
- **Local subscriptions and profiles**: CROC may relocate subscriptions based on that
- **Local publishers and profiles**: CROC uses this info to predict load imposed by each subscription

Once CROC gets all BIA it executes Phase 2 (reassignment of subscriptions to brokers) and Phase 3(reconfiguration of broker overlay).

Subscription Profile: Captures publications sinked by subscription.

- Allows CROC to accurately estimate load requirements.
- Generated by broker's CBC



Phase Two

Phase Two: Subscription allocation algorithms

Three proposed algorithms:

- Fastest Broker First (FSF)
- Bin Packing
- Clustering with Resource Awareness and Minimization (CRAM)

subscription pool: All subscriptions reported in BIA in Phase 1.

broker pool: All brokers that sent a BIA message back to CROC

Algorithmic Input: subscription pool, broker pool

Algorithmic Output: set of non-connected brokers (possibly) with allocated subscriptions

Phase Two: Fastest Broker First

Data: subscription pool, broker pool
Result: set of non-connected brokers (possibly) with subscriptions
Sort (desc) brokers by total available bandwidth
while *Subscription pool not empty* **do**
    s = Randomly pick a subscription
    Remove s from pool
    Assign s to first broker in list that **"can handle it"**
    if *s cannot be allocated to any broker* **then**
        return broker pool
    end
end
return broker pool

- **broker "can handle" subscription**: Remaining output bandwidth $>$ 0 and incoming publication rate $<=$ max matching rate (BIA Matching delay)
- **Complexity**: O(S), S number of subscriptions

Phase Two: Bin Packing

Data: subscription pool, broker pool
Result: set of non-connected brokers (possibly) with subscriptions
Sort (desc) brokers by total available bandwidth
Sort (desc) subscriptions by bandwidth requirement
while *Subscription pool not empty* **do**
    s = Pick first subscription
    Remove s from pool
    Assign s to first broker in list that **"can handle it"**
    if *s cannot be allocated to any broker* **then**
        return broker pool
    end
end
return broker pool

- **Complexity**: O(Slog(S)), S number of subscriptions
- FBF and Bin Packing → *sorting* algorithms

Phase Two: CRAM Metrics

- CRAM is significantly different!
- Relies on the following subscription **closeness** metrics:
  - **INTERSECT**: $|S_1 \cap S_2|$
  - **XOR (inversed)**: $|S_1 \oplus S_2|^{-1}$
  - **IOS**: $\frac{|S_1 \cap S_2|^2}{|S_1| + |S_2|}$
  - **IOU**: $\frac{|S_1 \cap S_2|^2}{|S_1 \cup S_2|}$

  where S_1 and S_2 are two subscriptions.

Phase Two: CRAM Algorithm

Data: subscription pool, broker pool

Result: set of non-connected brokers (possibly) with subscriptions

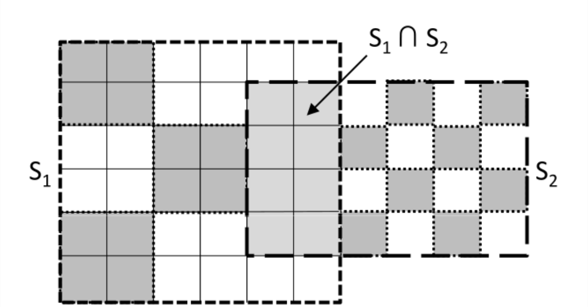
Run Bin Packing algorithm on Data

while *True* **do**
    s1, s2 = Pick two closest subscriptions (using metrics)
    if *s1 or s2 empty* **then**
        return broker pool
    end
    s' = s1 or s2 (bitwise)
    Remove s1 and s2 from pool
    Allocate s' using Bin Packing algorithm
    if *Allocation of s' fails* **then**
        s1, s2 = Revert s1 or s2 (bitwise)
        Note s1, and s2 should not be clustered again
    end
end

- Complexity: $O(S^3 log(S))$
- Optimizations proposed...

Phase Two: CRAM Optimizations

1. **Grouping of Equal Subscriptions**: Consider subscriptions of equal bit vectors identical. Reduce S.
2. **Search Pruning**: Reduce search space for closeness. $O(S^3 log(S)) \rightarrow O(S^2 log(S)$
3. **One-to-Many Clustering**: Cluster S_1 with shaded subscriptions before clustering with S_2 (even though closeness is worse).

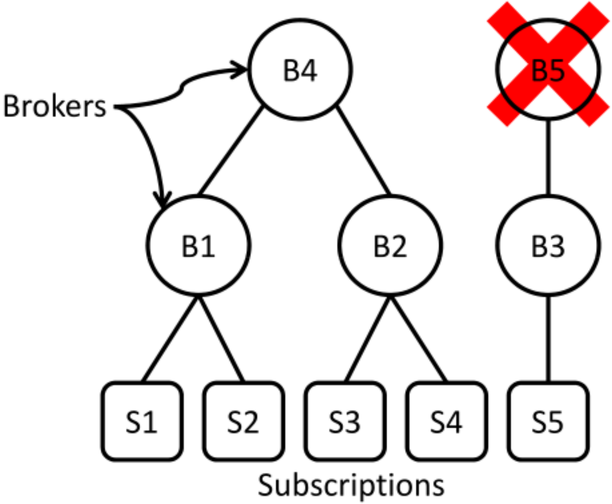


Phase Three

Phase Three: Recursive Overlay Construction

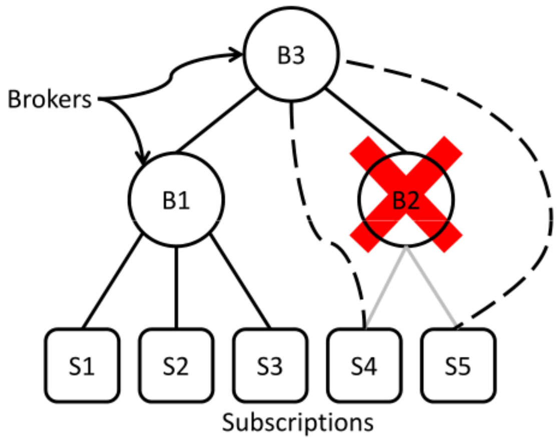
- Design a tree overlay of connected brokers.
- Use **CROC** to apply reconfiguration.
- Finally, use **GRAPE** algorithm to relocate publishers to the final overlay (out of scope of this paper).
- Three proposed optimizations for overlay construction.

Phase Three: Optimization One



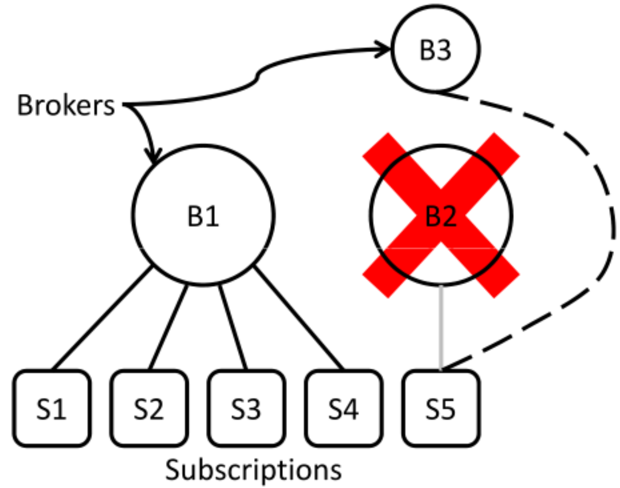
(a) Opt. 1: Deallocate B5 since it is a pure forwarding broker

Phase Three: Optimization Two



(b) Opt. 2: B3 has enough capacity to serve B2's subscriptions directly along with B1

Phase Three: Optimization Three



(c) Opt. 3: Replace under-utilized high resource brokers with low resource brokers

Experiment

Experiment

- Comparison of proposed framework with related approaches and baseline algorithms
- Usage of PADRES, an open-source distributed content-based publish/subscribe system implemented at UToronto

Conclusions

Conclusions

- The approach works on any implementation or variation of a pub/sub system.
- Advantages:
  1. CRAM reduces the average broker message rate by up to 92% comparing with prior work.
  2. CRAM reduces the number of allocated brokers by up to 91%, no publication hop.
  3. IOU metric heavily reduces computation time of CRAM
- Disadvantages
  1. CRAM is computationally heavier than all other approaches
  2. CRAM may set high delivery delays
- Possible Remedies
  1. Adjustments to bit vector length (reduce computation time)
  2. Strategically relocate publishers to reduce delivery delays

Thank you! Questions?