# Scalable Event Matching for Overlapping Subscriptions in Pub/Sub Systems

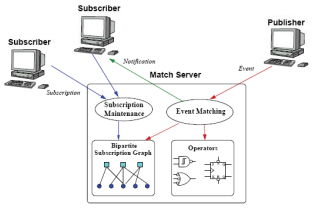
Zhen Liu, Srinivasan Parthasarthy, Anand Ranganathan, Hao Yang

Presented by Vamsidhar Thummala
5th Feb 2008

Instructor: Prof. Jun Yang

## Preliminaries

- What is a publish/subscribe system?




## Preliminaries

- Applications
  - Stock information delivery
  - Auction system
  - Air traffic control
  - etc..
- Pub/Sub systems are of two types
  - Topic-based
  - Content-based
- Selectivity
  - the probability that an event satisfies a given predicate
- Popularity
  - the number of subscriptions containing the given predicate

## Problem

- Current content-based publish/subscribe systems has to deal with huge number of subscriptions
- Scalability?
- Optimal subscription evaluation is NP-Hard
- Solution
  - Approximation algorithm by exploiting the overlap in the subscriptions

## Example

- Monitoring metropolitan area using UAV's, subscriptions can be of type:

```
S1={locIn(Bronx), chasePattern(policeCar,anyCar)}
S2={locIn(Bronx), mobPattern(?people), (?people>50)}
S3={locWithin(TimesSquare,5000m),
   trafficPattern(congestion),
   pedestrianCount(?ped), ?ped>250)}
S4={locIn(Bronx), trafficPattern(congestion),
    trafficPattern(accident), policeCarCount(?pol),
   (?pol<2)}
S5={distanceFrom(StatueOfLiberty,?x), (?x<1000m),
   tourBoatCount(?y), coastGuardBoatCount(?z),
   (?y>2*?z)}
```

- Naïve Approach: Evaluate each subscription one by one

## Observations

- Users often share common interests
  - subscriptions share common predicates
  - reuse the evaluated predicates
- A notification has to be sent only if
  - all the predicates in the subscription satisfies the event
- Else all the subscriptions containing the not satisfied predicate can be discarded

## Analysis of problem

- Theorem:
  - Subscription evaluation problem is NP-Hard (NP-Complete?)
- Proof:
  - Reduce subscription evaluation problem to set-cover problem

## Sequential Algorithm

**Algorithm 1** Sequential Evaluation Algorithm

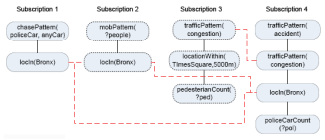
```
1: procedure SEQUENTIAL(T, Q)                          ▷
   The subset of predicates in T which must be evaluated
   to decide subscriptions Q
2:     U ← Q                          ▷ U is the current set of undecided
   subscriptions
3:     E ← ϕ ▷ E is the set of predicates evaluated so far
4:     while U ≠ ϕ do                 ▷ some subscriptions are still
   undecided
5:         Select any subscription q ∈ U
6:         S ← the set of predicates subscribed by q which
   have currently not been evaluated
7:         while S ≠ ϕ do
8:             t ← any predicate in S
9:             if t is subscribed by some subscription in U
   then
10:                evaluate t
11:                for all subscription q ∈ U which is now
   decided do
12:                    U = U \ {q}
13:                end for
14:            end if
15:            E = E ⋃ {t}
16:            S = S \ {t}
17:        end while
18:    end while
19:    return E                       ▷ the set of evaluated predicates
20: end procedure
```

## Sequential Algorithm

- Returns the set of subscriptions whose predicates are all satisfied individually.
  - Perform variable bindings and value checks to determine whether the subscription is completely satisfied
- Running time
  - at-most d*optimal
  - d is the maximal number of predicates across all subscriptions

## Sub-Order Algorithm - Preliminaries

- Linked chains structure




## Sub-Order Algorithm - Preliminaries

- Cost of evaluating a single chain

$$C = 1 + p_1 + p_1 p_2 + \ldots + \prod_{i=1}^{n-1} p_i$$
$$= 1 + \sum_{i=1}^{n-1} \prod_{j=1}^{i} p_j$$

- Cost of evaluating two chains independently

$$C = 1 + \sum_{i=1}^{n-1} \prod_{j=1}^{i} p_{1,j} + 1 + \sum_{i=1}^{m-1} \prod_{j=1}^{i} p_{2,j}$$

- Order of evaluation is important!!

## Sub-Order Algorithm - Preliminaries

- Consider an equivalence link between nodes $n_{1,a}$ and $n_{2,b}$ where 1 <= a <= n and 1 <= b <=m
- Cost of evaluating s1 first and then s2 is

$C_1$ = (Cost of evaluating all nodes in $s_1$)
+ (Probability of nodes $n_{1,1}, n_{1,2}, \ldots n_{1,a-1}$ all having solutions) × [ (Probability of node $n_{1,a}$ having solutions) × ( Cost of evaluating $s_2$ without having to evaluate node $n_{2,b}$)
+ (Probability of node $n_{1,a}$ not having solutions) × ( 0 ) ]
+ (Probability of at least one of the nodes $n_{1,1}, n_{1,2}, \ldots n_{1,a-1}$ not having a solution) × ( Cost of evaluating $s_2$ including the evaluation of node $n_{2,b}$)

## Sub-Order Algorithm - Preliminaries

- Cost of evaluating s1 before s2

$$C_1 = \left[1 + \sum_{i=1}^{n-1} \prod_{j=1}^{i} p_{1,j}\right] + \prod_{i=1}^{a} p_{1,i} \times \left[1 + \sum_{i=1, i\neq b-1}^{m-1} \prod_{j=1, j\neq b}^{i} p_{2,j}\right] + \left(1 - \prod_{i=1}^{a-1} p_{1,i}\right) \times \left[1 + \sum_{i=1}^{m-1} \prod_{j=1}^{i} p_{2,j}\right]$$

- $C_2$ is the cost of evaluating s2 before s1
- Cost difference

$$C_1 - C_2 = (1 - p_{1,a}) \times \prod_{i=1}^{b-1} p_{2,i} \times \left(1 + \sum_{i=1}^{a-2} \prod_{j=1}^{i} p_{1,j}\right) - (1 - p_{1,a}) \times \prod_{i=1}^{a-1} p_{1,i} \times \left(1 + \sum_{i=1}^{b-2} \prod_{j=1}^{i} p_{2,j}\right) = (1 - p_{1,a}) \times (P_{2,b} \times C_{1,a} - P_{1,a} \times C_{2,b})$$

## Sub-Order Algorithm

- Arrange the predicates in the decreasing order of selectivity among each subscription chain
- Calculate the cost differences for each pair of subscriptions (Now, the problem is TSP for which has a k-approximation solution)
- Use Kruskal's algorithm (greedy) to form a minimum spanning tree
  - The weight of the edge (u,v) is the cost savings of evaluating u before v
- Evaluate the nodes in the order given by Kruskals (?)

## Assumptions

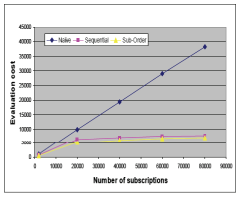
- Selectivity of predicates is conditionally independent of selectivity of other predicate
- All predicates have the same evaluation cost
- If predicates have different evaluation cost, then approximation guarantee increases from d to $d*C_{max}/C_{min}$.

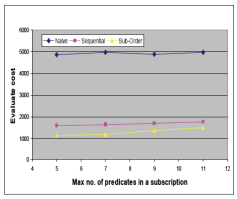
## Experiments

- Synthetic workload (?)

| parameter | description | value |
|---|---|---|
| $N_s$ | number of subscriptions | $2K \sim 80K$ |
| $N_p$ | number of predicates | $1K \sim 40K$ |
| $model$ | popularity distribution of the predicate | Uniform or Zipf |
| $min_{sl}$ | minimum number of predicates in a subscription | 3 |
| $max_{sl}$ | maximum number of predicates in a subscription | $5 \sim 11$ |
| $N_e$ | number of events | 1000 |
| $ratio_{match}$ | ratio of matched subscriptions among all subscriptions | 0.2% |

## Results




- Claim: The performance of Sequential is not sensitive to the subscription length

## Points of discussion

- Only selectivity is considered
  - How to order both on selectivity as well as popularity?
- Synthetic data set is used
- Metric used: Average number of predicates
- Are the results convincing?
  - There is hardly a noticeable difference between Sequential and Sub-Order algorithm from the graphs

## Points of discussion

- Unrealistic assumptions
  - Independent assumption on selectivity of predicates
- Cost model
- Prim's for MST?
- The current model provides sort of middle layer assuming that all the predicates are retrieved (from multi-media objects) and available. Can't we achieve better selectivity by pushing the selections down to multi-media objects?