Computer Science & Engineering Department

Universitatea de Vest din Timişoara

SYNASC 2011

Published by: CPS Conference Publishing Services

A Data Dissemination Algorithm for Opportunistic Networks

Radu-Ioan Ciobanu, Ciprian Dobre, Valentin Cristea

E-mail: ciprian.dobre@cs.pub.ro

University "*Politehnica*" of Bucharest
Faculty of Automatic Control and Computer Science

Outline

- Scope and Motivation
- Dissemination algorithm
- Scenarios and experimental results
- Conclusions

Context and motivation

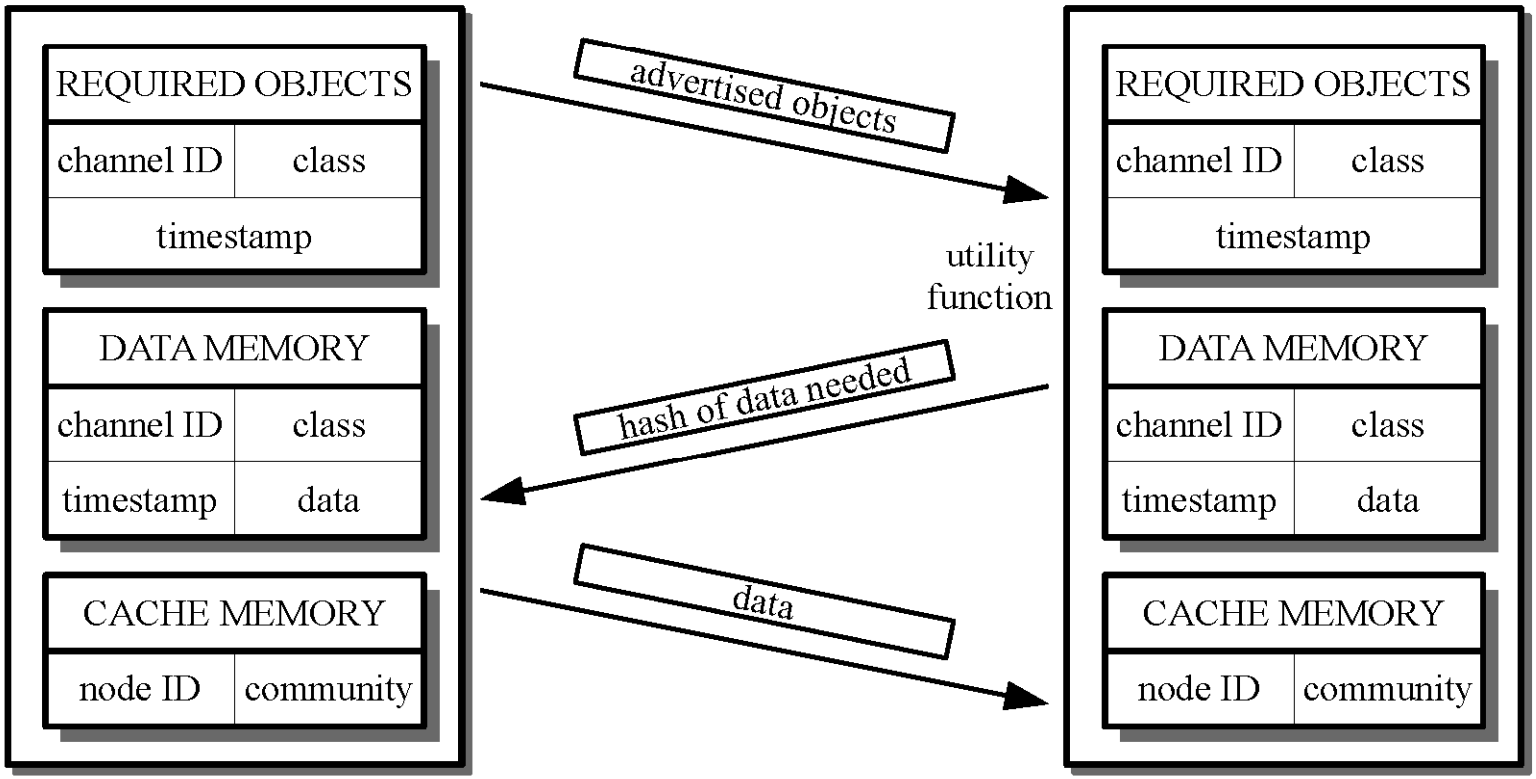
- Opportunistic networks:
  - networks composed of mobile nodes that may not have a direct connection between them
  - routes are build dynamically
  - nodes act according to the store-carry-and-forward paradigm
  - data dissemination is usually based on a publish/subscribe model
  - nodes move according to social relationships

- Previous solutions for data dissemination: Socio-Aware, DTN, *ContentPlace* (Most Frequently Visited, Most Likely Next, Future, Present, Uniform Social, …)
- Social Dissemination: a socially-aware *data dissemination algorithm* that takes advantage of the social grouping of nodes in communities (inspired by the caveman model)

Algorithm Protocol

- In Social Dissemination each node stores data using:
  - Required Objects - information about data objects *required by the node*
  - Data Memory – data previously *received from other nodes* (necessary or stored for future use)
  - Cache Memory - information about *nodes most recently encountered*
  - Published Data - data *published by the node*
- Communication between two nodes that are in range has several steps:
  1. advertise stored data
  2. analyze received data to find needed objects
  3. if needed objects are not found, analyze received data to find objects that can be disseminated (using *utility function*)
  4. send back hash of needed data objects to encountered node
  5. receive requested objects

Algorithm Protocol (2)



Utility Function

- Computed according to the community a node belongs to, and to the communities it had contacts with so far
- Based on the caveman model (“home" and “acquainted“ communities) – describes realistic mobility and social patterns
- Computed as:

$$u = \sum_{i=1}^{n} \underbrace{p_i^c * (1 - p_i^e)}_{\text{computed over the cache}} + freshness + c_v$$

- Components:
  - n - number of communities
  - p_i^c - percentage of nodes from community i encountered so far
  - p_i^e - the percentage of nodes from community i that the encountered node has been in contact with so far
  - *freshness* - higher value of utility for newer objects
  - c_v - an additional element added if the nodes belong to different communities

Mobility Model

- Developed a mobility model simulator based on HCMM:
  - assumes that nodes are not driven only by the social relationships, but also by the attraction of physical locations
  - each community has a home cell
  - each node is attracted to its home cell according to the social attraction exerted by all nodes that are part of its community
  - attraction of an external cell computed based on the relationships with nodes that have their home in that cell
  - adapted to permit a graphical representation of the movement of nodes, interactions between them, and division of area into cells
  - various parameters can be set: size of movement area, number of cells, number of nodes, number of communities, speed of the nodes, simulation time
  - statistical data is collected and written to a le at the end of the simulation
  - allows the addition of a dissemination algorithm through the implementation of a Java interface

Test Scenarios

- Two series of experiments:
- Compare with experimental conditions from ContentPlace:
  - 45 nodes grouped into 3 communities
  - 1000x1000 grid, divided into 16 cells
  - 3 channels, each generates 99 data objects
  - a node can only be subscribed to one channel
  - interests of nodes is distributed according to a Zipf's law within each community
  - nodes ask for data objects according to a Poisson distribution, with λ=200
  - speed of nodes varies between 1 m/s and 1.86 m/s
  - transmission range of 20 meters
  - length of simulation time is 50 seconds
  - data memory size is 33 objects
  - C1 has relationships with C2 and C3 through two “traveller" nodes (subscribed to channel 1)
- Perform experiments to analyze scaling capabilities:
  - 120 nodes, communities varying from 2 to 6
  - 4 communities, number of nodes varying (40, 60, 120, 240)

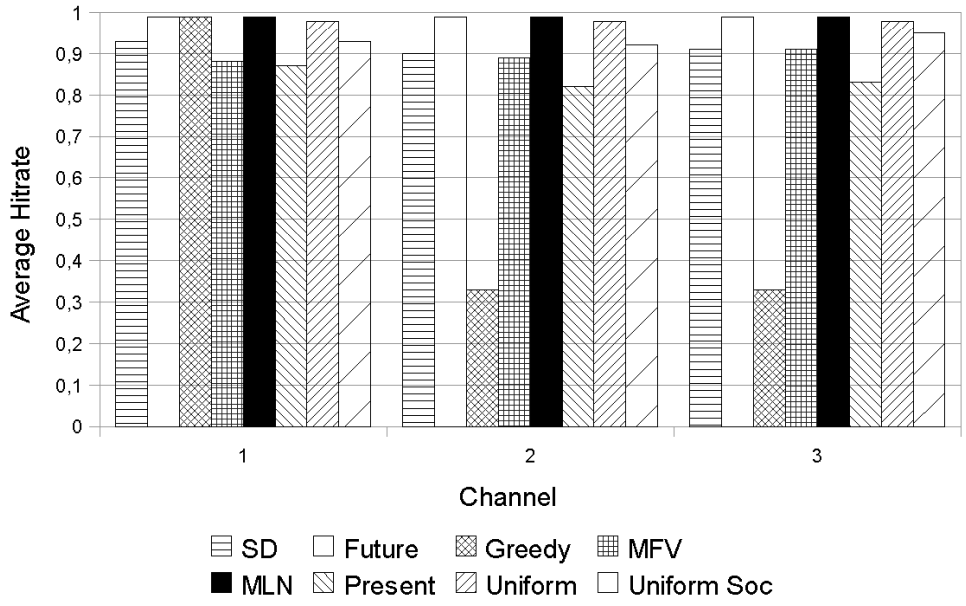
Test Results

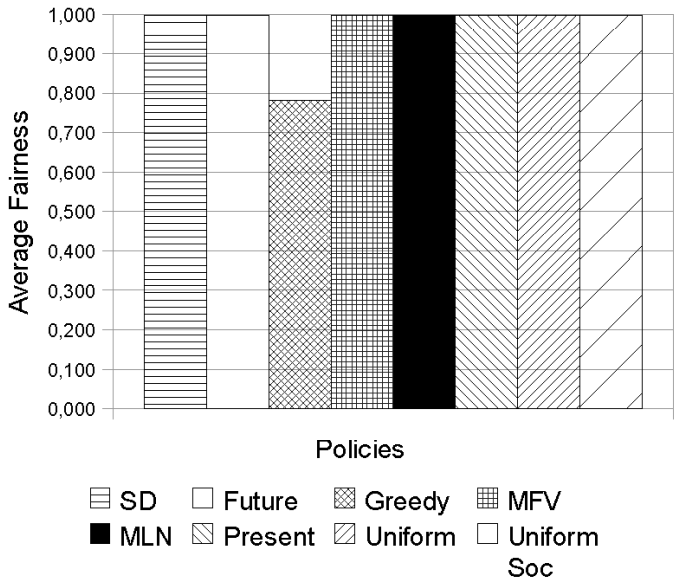
- Based on four metrics:
  - hit rate
  - fairness
  - resource consumption
  - latency
- Hit rate:
  - ratio between data objects that have successfully arrived at requesting nodes, and total number of requests
  - suggests the efficiency of the dissemination algorithm
  - shows fraction of requests that can't be served by a dissemination algorithm
- Fairness:
  - computed according to Jain's fairness index → this metric identifies underutilized channels and is not unduly sensitive to atypical network flow patterns

Test Results (2)

91% hit rate for SD on all channels

SD is fair for all channels





(a) Hit rate

(b) Fairness

Figure: Hit rate and fairness for Social Dissemination and ContentPlace

Test Results (3)

- Resource consumption:
  - traffic generated in the network by the dissemination algorithm
  - Social Dissemination is outperformed only by Greedy (due to *freshness* and c_v)
- Latency - difference between the time a data object request has been satisfied, and the time the request was made

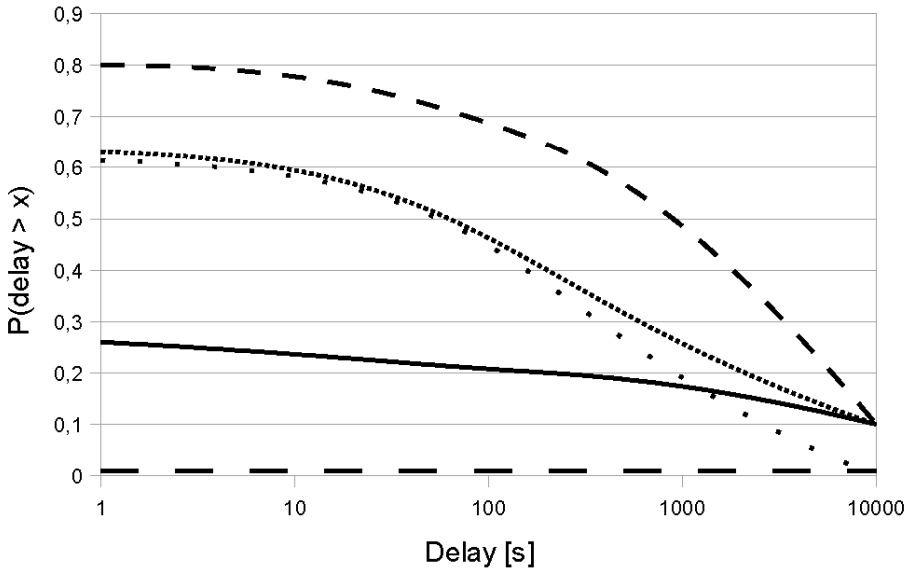


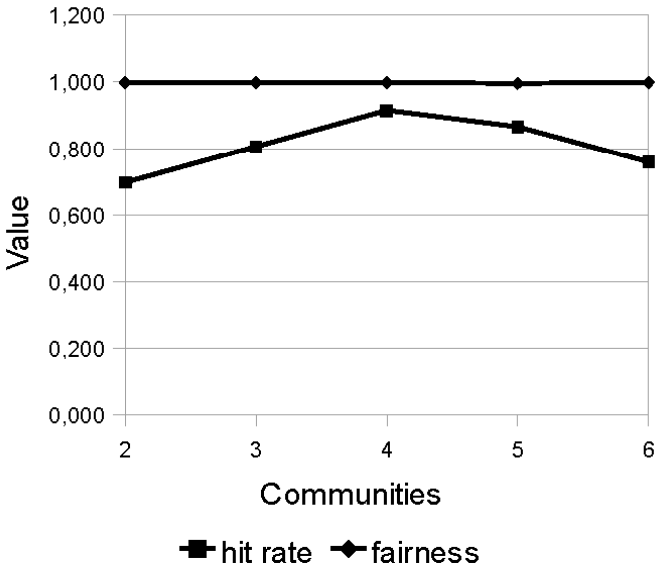
Figure: Complementary Cumulative Distribution Function for the delay of requests in case of Social Dissemination and ContentPlace.

Partial summary

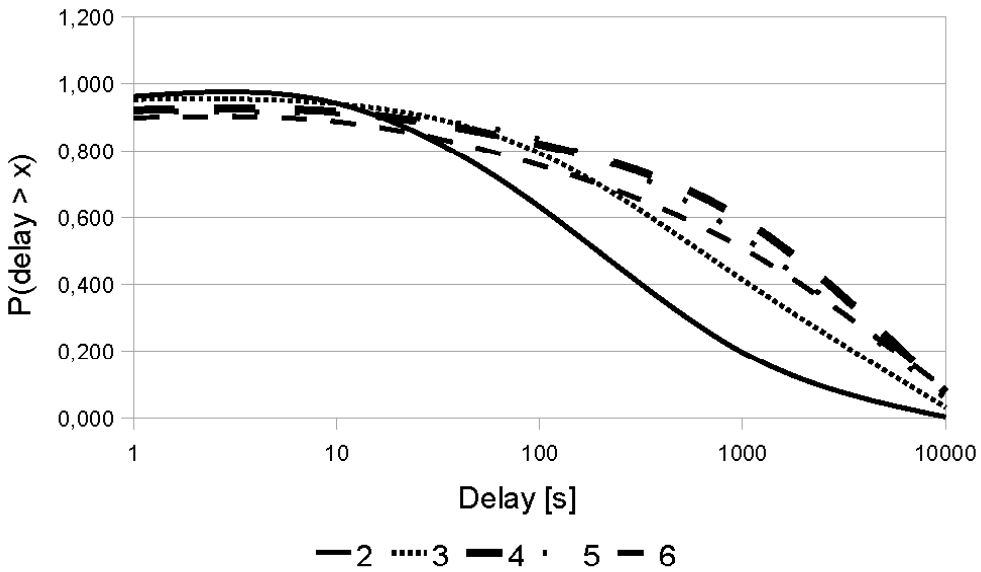
- Analyzing the results of SD compared to the seven ContentPlace policies, there is no policy that outperforms SD from the standpoint of all evaluated four metrics.
- The best two policies from ContentPlace are considered in literature to be Future and Most Likely Next
  - do slightly outperform the Social Dissemination technique in regard to *hit rate* and *latency*,
  - but not from the standpoint of bandwidth overhead
  - also, Social Dissemination can obtain a hit rate close to 100%.

Scalability when increasing the number of communities

- The highest hit rate value achieved when no of channels equals no of communities
  - data objects from each channel are uniformly spread across communities.
- The hit rate for 6 communities is better than the hit rate for 2 communities
- SD maintains a fairness close to 100% for all channels, no matter how many communities are in the network.



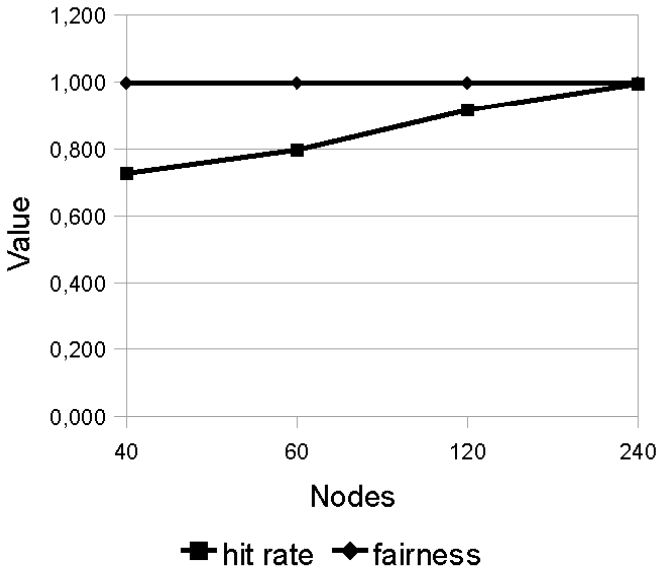
(a) Hit rate and fairness



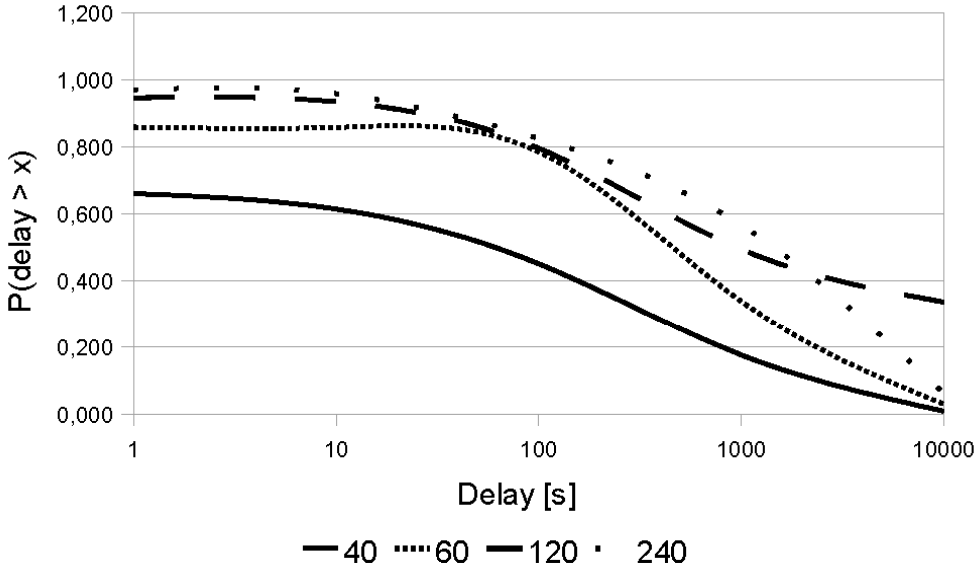
(b) Delay CCDF

Scalability when increasing the number of nodes

- In the second experiment the number of communities is 4, while the number of nodes varies (with values of 40, 60, 120 and 240).
- The fairness remains very close to 100%
  - every community receives an almost equal percentage of data objects from the ones they requested.
  - hit rate obtained when testing with 4 communities grows with the number of users in the network, reaching 100% for 240 nodes.
  - our proposed algorithm scales extremely well even when the number of nodes in the network is increased.



(a) Hit rate and fairness



(b) Delay CCDF

Conclusions

- A social-based dissemination algorithm for opportunistic networks
- Experiments show there is no ContentPlace policy that outperforms Social Dissemination from the standpoint of all four metrics
- Experiments show that SD scales well when the number of nodes or communities is increased, obtaining good hit rates and an acceptable growth in node request latency
- Future Work:
  - extend the results and improve the utility function to take into account the communities a node is more likely to visit in the future
  - improve the latency, in order to obtain an opportunistic network that can be useful in a practical way
  - implement the algorithm for real-life use, using smartphones

Q&A

Thank you!