# A Knapsack-Based Message Scheduling and Drop Strategy for Delay-Tolerant Networks

**En Wang [1,2], Yongjian Yang [1], and Jie Wu [2]**

1 Department of Computer Science and Technology, Jilin University, Changchun, China

2 Department of Computer and Information Sciences, Temple University, Philadelphia, USA

wangen0310@126.com ; yyj@jlu.edu.cn ; jiewu@temple.edu

# Outline

- **1. Introduction**
- 2. Model Description
- 3. Scheduling and Drop Strategy
- 4. Evaluation
- 5. Future Work

# 1. Introduction

## 1.1 Motivation

- **The dramatic change of topology and the frequent interruption of connections make it difficult to forward the message to the destination in DTNs**

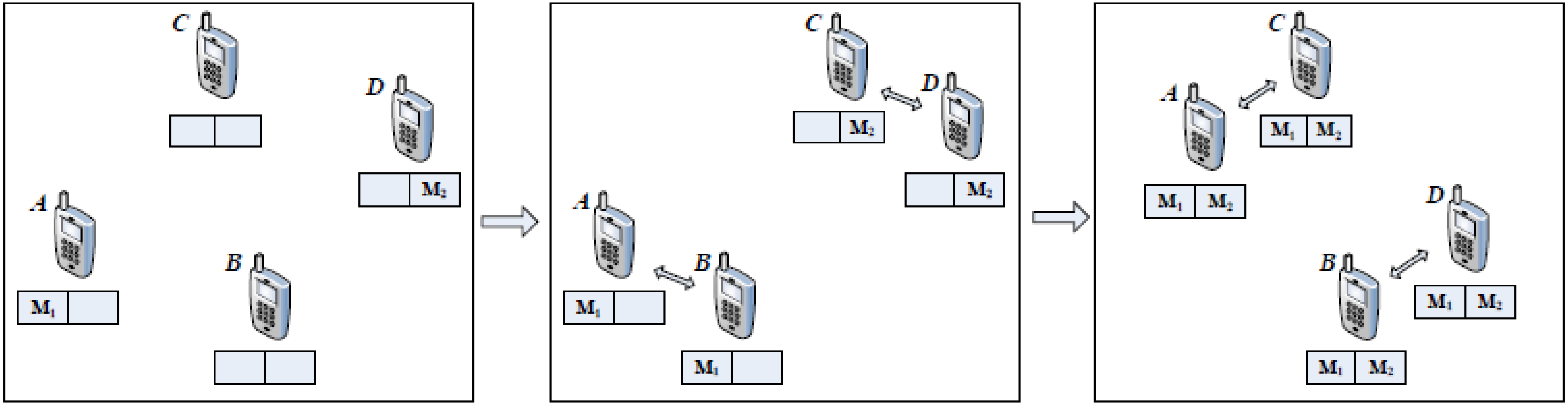
- **Routing protocols seek to improve the delivery ratio through increasing the number of message copies**

- **Excessive message copies lead to the occurrence of buffers overflowing because of limited storage space**

# 1. Introduction

## 1.2 Problem

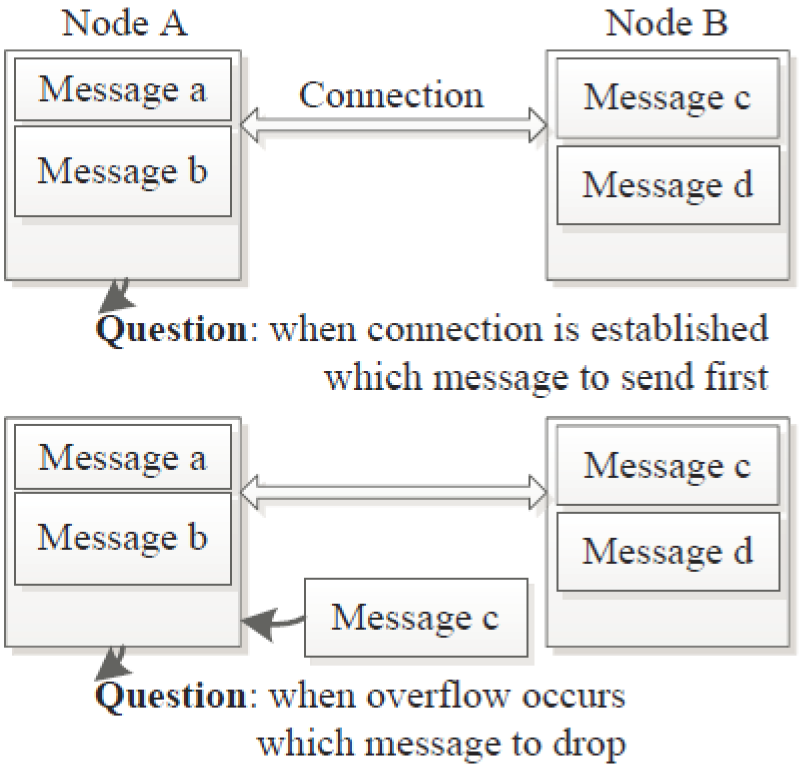
- **The message flooding process**



# 1. Introduction

## 1.2 Problem

- **When a connection is established, which message to send first**
- **When overflow occurs, which message to drop**



# 1. Introduction

## 1.3 Challenge

- **The utility model to decide the priority of a message**

- **The communication bandwidth is limited**

- **The message sizes are different**

# Outline

- 1. Introduction
- **2. Model Description**
- 3. Scheduling and Drop Strategy
- 4. Evaluation
- 5. Future Work

# 2. Model Description

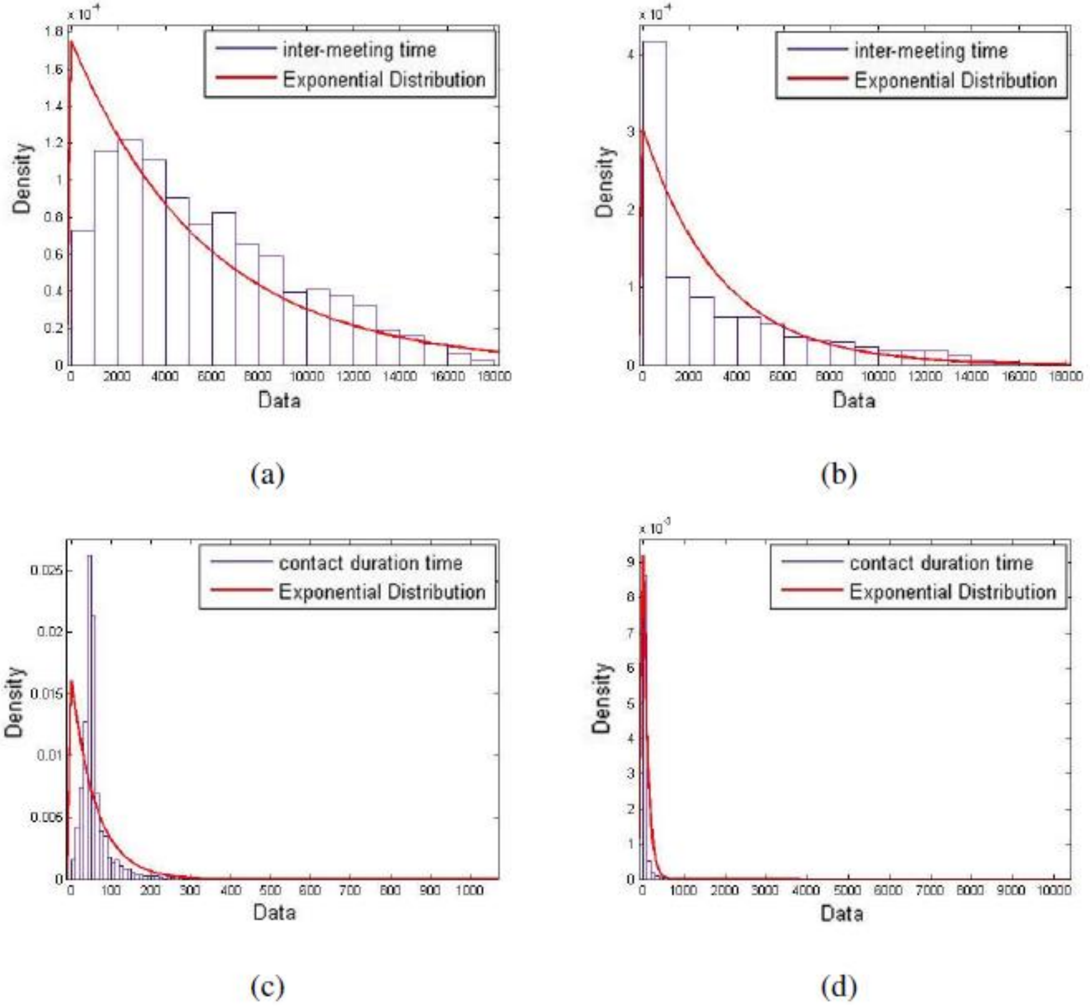
## 2.1 Mobility Model

- ***Definition 1*: Intermeeting time: the elapsed time from the end of the previous contact to the start of the next contact between nodes in a pair**
- ***Definition 2*: Contact duration: the duration time during which the nodes in a pair stay in each other's communication range**
- **Intermeeting times and contact durations are exponentially distributed under many popular mobility patterns such as random walk, random waypoint, and random direction.**

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x > 0 \\ 0 & x \leq 0 \end{cases}$$

# 2. Model Description

## 2.1 Mobility Model: random-waypoint (a,c), EPFL (b,d)



# 2. Model Description

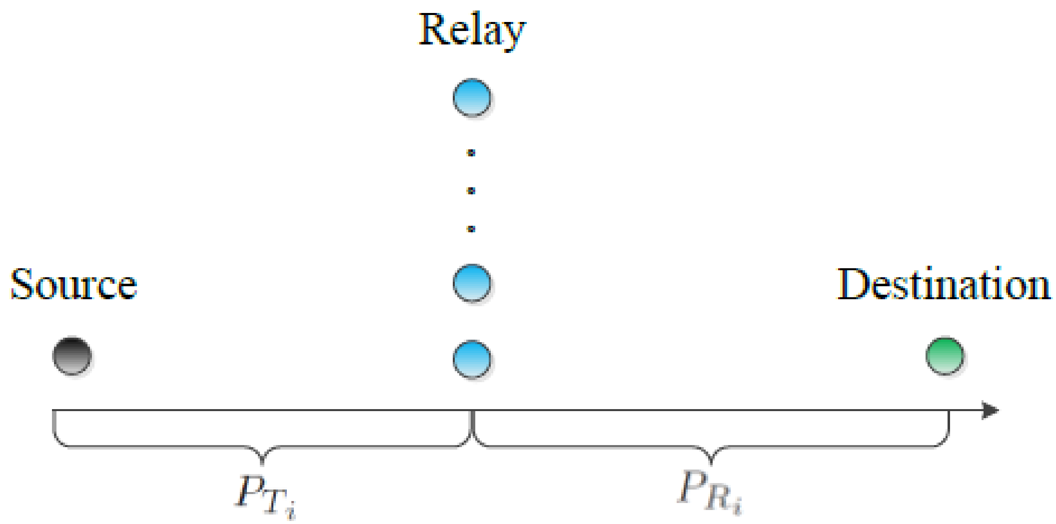
## 2.2 Utility Model

| | |
|---|---|
| $N$ | Total number of nodes in the network |
| $n_i(T_i)$ | The copy number of message $i$ in the network after the elapsed time $T_i$ |
| $m_i(T_i)$ | Number of nodes (excluding source) that have seen message $i$ from its creation time to the elapsed time $T_i$ |
| $P_{T_i}$ | Probability that message $i$ has been successfully delivered at the present moment |
| $P_{R_i}$ | Probability that undelivered message $i$ will reach the destination within time $R_i$ |
| $P_i$ | Probability that message $i$ can be successfully delivered |
| $\varepsilon_i$ | Probability that message $i$ can be forwarded successfully during contact |
| $M_i$ | Size of message $i$ |
| $W$ | Bandwidth of the contacts between nodes in a pair |

- **The copy number is less than the number of nodes that have seen message *i* as some copies are discarded**

# 2. Model Description

## 2.2 Utility Model




- **The probability of message *i* being delivered is given by the probability that message *i* has been delivered and the probability that message *i* has not yet been delivered, but will be delivered during the remaining time $R_i$**

$$P_i = (1 - P_{T_i})P_{R_i} + P_{T_i}$$

# 2. Model Description

## 2.2 Utility Model

- **Due to the reason that all the nodes including the destination have an equal chance of seeing the message *i:***

$$P_{T_i} = \frac{m_i(T_i)}{N}$$

- **Probability that message *i* can be forwarded successfully during a contact:**

$$\varepsilon_i = e^{-\lambda_2 \frac{M_i}{W}}$$

# 2. Model Description

## 2.2 Utility Model

- **1-$P_{R_i}$ gives the probability that the $n_i(T_i)$ nodes at $T_i$ will not contact the destination node during $R_i$, and the new infected nodes will also not reach the destination node.**

$$1 - P_{R_i} = e^{-\lambda n_i(T_i)R_i} \prod_{t=0}^{R_i} e^{-\lambda n_i'(T_i+t)(R_i-t)}$$

$$= \frac{e^{-\lambda n_i(T_i)R_i} \prod_{t=0}^{R_i} e^{-\lambda n_i'(T_i+t)(R_i)}}{\prod_{t=0}^{R_i} e^{-\lambda n_i'(T_i+t)(t)}}$$

$$= \frac{e^{-\lambda n_i(T_i+R_i)R_i}}{e^{-\lambda \int_0^{R_i} n_i'(T_i+t)t\mathrm{d}t}}$$

# 2. Model Description

## 2.2 Utility Model

- **The probability $P_{R_i}$ represents the likelihood that the undelivered message *i* at $T_i$ can reach the destination in the remaining time $R_i$**

$$P_{R_i} = 1 - \frac{N^{\frac{1}{\varepsilon_i}}}{e^{\lambda N R_i}[n_i(T_i) - n_i(T_i)e^{-\varepsilon_i \lambda N R_i} + N e^{-\varepsilon_i \lambda N R_i}]^{\frac{1}{\varepsilon_i}}}$$

- **The probability that message *i* can be successfully delivered:**

$$P_i = \frac{m_i(T_i) - N}{N} N^{\frac{1}{\varepsilon_i}} \frac{1}{e^{\lambda N R_i}[n_i(T_i) - n_i(T_i)e^{-\varepsilon_i \lambda N R_i} + N e^{-\varepsilon_i \lambda N R_i}]^{\frac{1}{\varepsilon_i}}} + 1$$

# 2. Model Description

## 2.2 Utility Model

- **Three cases：**

$$\begin{cases} \Delta(n_i) = 1 & \text{If replicate message } i \text{ during contact.} \\ \Delta(n_i) = 0 & \text{If no action for message } i \text{ is taken.} \\ \Delta(n_i) = -1 & \text{If drop an already existing message } i. \end{cases}$$

- **Therefore, the utility of message *i* is precisely the derivative of the delivery ratio $P_i$**

$$U_i = [N - m_i(T_i)]N^{\frac{1-\varepsilon_i}{\varepsilon_i}} e^{-\lambda N R_i} \frac{1}{\varepsilon_i}(1 - e^{-\varepsilon_i \lambda N R_i})$$

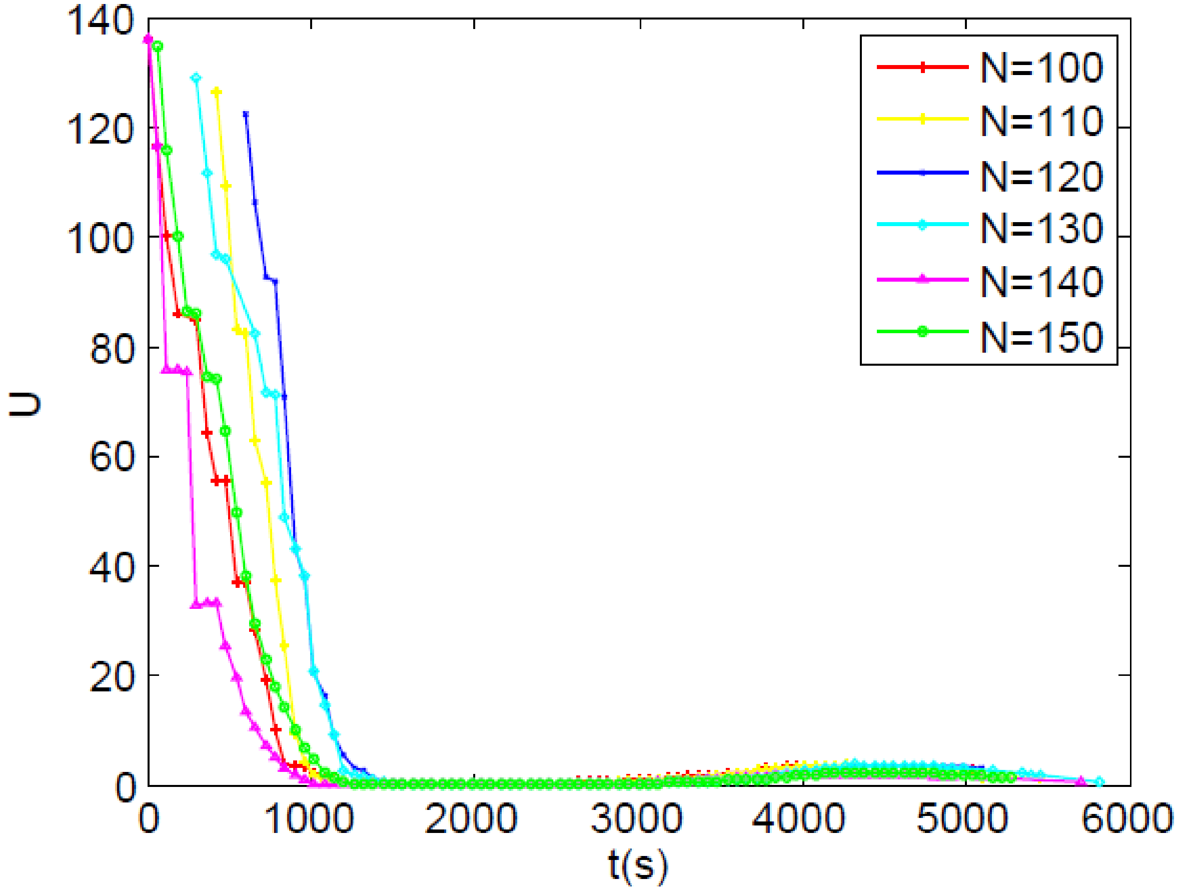
$$[n_i(T_i) - n_i(T_i)e^{-\varepsilon_i \lambda N R_i} + Ne^{-\varepsilon_i \lambda N R_i}]^{\frac{-\varepsilon_i - 1}{\varepsilon_i}}$$

- **The higher $U_i$ indicates that the message *i* is more important**

# 2. Model Description

## 2.2 Utility Model

| Parameter | Value |
|---|---|
| Mobility Pattern | random-waypoint |
| Simulation Area | 4500m×3400m |
| Number of Nodes | 100-150 |
| Moving Speed | 2m/s |
| Transmission Speed | 250Kbps |
| Buffer Size | 25MB |
| TTL | 6000s |

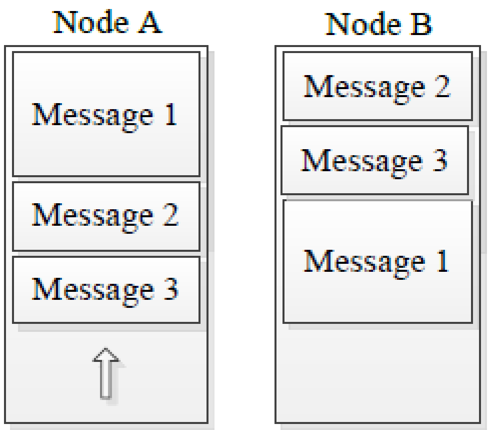


# Outline

- 1. Introduction
- 2. Model Description
- **3. Scheduling and Drop Strategy**
- 4. Evaluation
- 5. Future Work

# 3. Scheduling and Drop Strategy

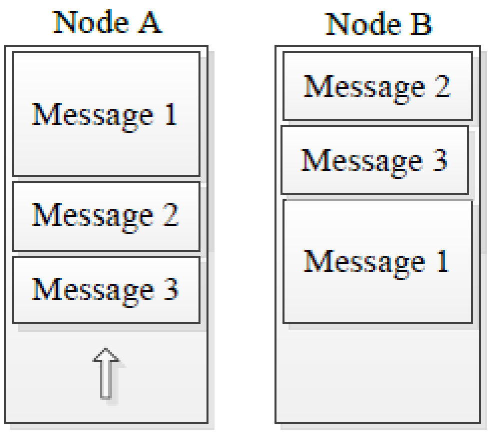
## 3.1 Scheduling Strategy




- **The utilities satisfy:** $U_1 > U_2 > U_3$, and $U_2 + U_3 > U_1$.
- **The message sizes satisfy:** $M_1 = 2M_2$ and $M_2 = M_3$.
- **Condition: only messages with no greater than $M_1$ can be successfully forwarded due to the limited contact duration and bandwidth**

# 3. Scheduling and Drop Strategy

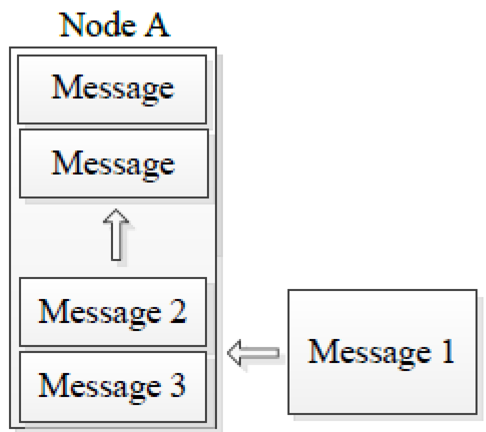
## 3.1 Scheduling Strategy




- **Traditional strategy: schedule the messages in decreasing order of $U_i$. Gain: $U_1$**
- **Our strategy: schedule the messages in decreasing order of $\frac{U_i}{M_i}$. Gain: $U_2 + U_3$**
- **Conclusion: due to the reason that $U_2 + U_3 > U_1$, our strategy is better than the traditional strategy**

# 3. Scheduling and Drop Strategy

## 3.1 Drop Strategy




- **The utilities satisfy:** $U_1 > U_2 > U_3$, and $U_2 + U_3 > U_1$.
- **The message sizes satisfy:** $M_1 = 2M_2$ and $M_2 = M_3$.
- **Condition: the local buffer is already full**

# 3. Scheduling and Drop Strategy

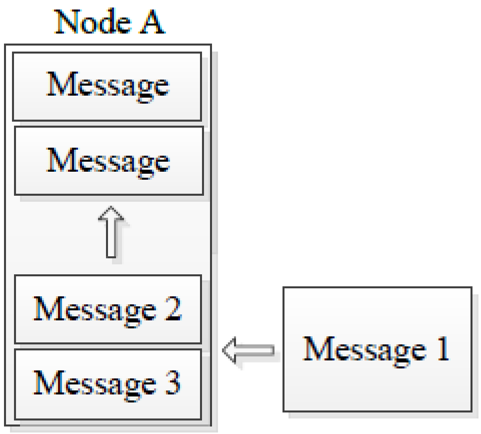
## 3.1 Drop Strategy

**Algorithm 1.** Dynamic programming to solve 0-1 knapsack problem

```
for j = 0;j ≤ totalWeight; j++ do
  for i = 0;i ≤ n; i++ do
    if (i = 0||j = 0) then
      bestValues[i][j]=0;
    else
      if j < sizes[i − 1] then
        bestValues[i][j]=bestValues[i − 1][j];
      else
        iweight=sizes[i − 1];
        ivalue=values[i − 1];
        bestValues[i][j]=MAX(bestValues[i − 1][j]);
        ivalue=ivalue+bestValues[i − 1][j − iWeight];
if bestSolution=null then
  bestSolution=int[n];
tempWeight=totalWeight;
for i = n;i ≥ 1; i−− do
  if bestValues[i][tempWeight] > bestValues[i − 1][tempWeight] then
    bestSolution[i − 1]=1;
    tempWeight=sizes[i − 1];
  if tempWeight=0 then
    break;
bestValue=bestValues[n][totalWeight];
```

# 3. Scheduling and Drop Strategy

## 3.1 Drop Strategy




- **Traditional strategy: drop the message with lowest utility**
  **Loss:** $U_2 + U_3$
- **Our strategy: drop the message according to the solution of the following 0 − 1 knapsack problem Loss:** $U_1$

Restriction: $\sum_{k=1}^{n} M_k x_k \leq M,\ x_k=\{0,1\}, k=1,2,3\cdots n,$

Objective: $Max \sum_{k=1}^{n} U_k x_k.$

# Outline

- 1. Introduction
- 2. Model Description
- 3. Scheduling and Drop Strategy
- **4. Evaluation**
- 5. Future Work

# 4. Evaluation

## 4.1 Simulation parameters (random-waypoint)

| Parameter | Value |
|---|---|
| Simulation Time | 18000s |
| Simulation Area | 4500m×3400m |
| Number of Nodes | 100 |
| Moving Speed | 2m/s |
| Transmission Speed | 250Kbps |
| Transmission Range | 100m |
| Buffer Size | 10MB,15MB,20MB,25MB,30MB |
| Interval of Message Generation | [5,15][15,25][25,35],[35,45] |
| TTL | 300 |

# 4. Evaluation

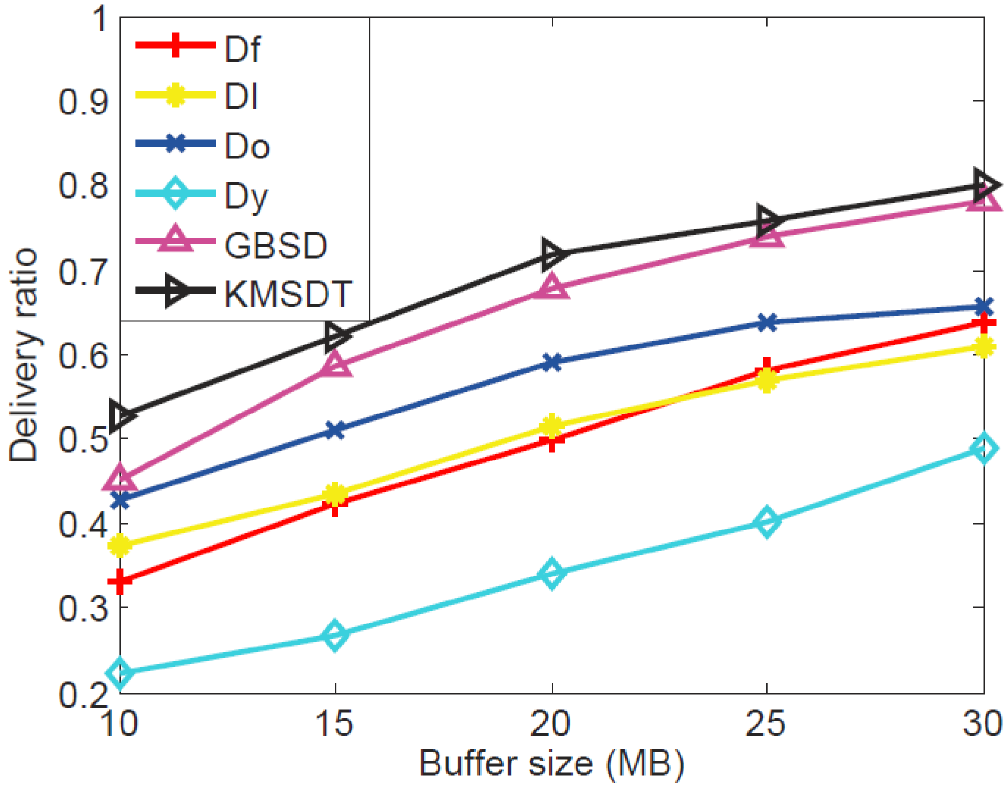
## 4.2 Comparative method (GBSD [10])

- **Without considering the following three problems:**
- **1. Different message sizes.**
- **2 . Limited contact duration.**
- **3. The impact on delivery ratio of message copies, which will be generated in $R_i$ (remaining *TTL)*.**

[10] Krifa, A, Barakat, C, "Message Drop and Scheduling in DTNs: Theory and Practice", IEEE Transactions on Mobile Computing, 11 (9) (2012) 1470-1483.

# 4. Evaluation

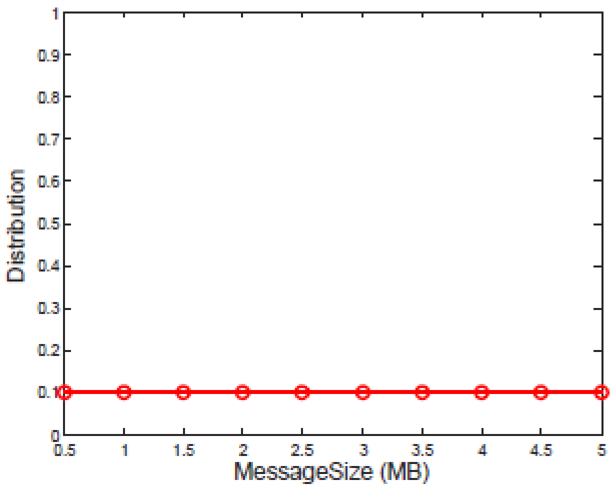
## 4.3 Same Message Size



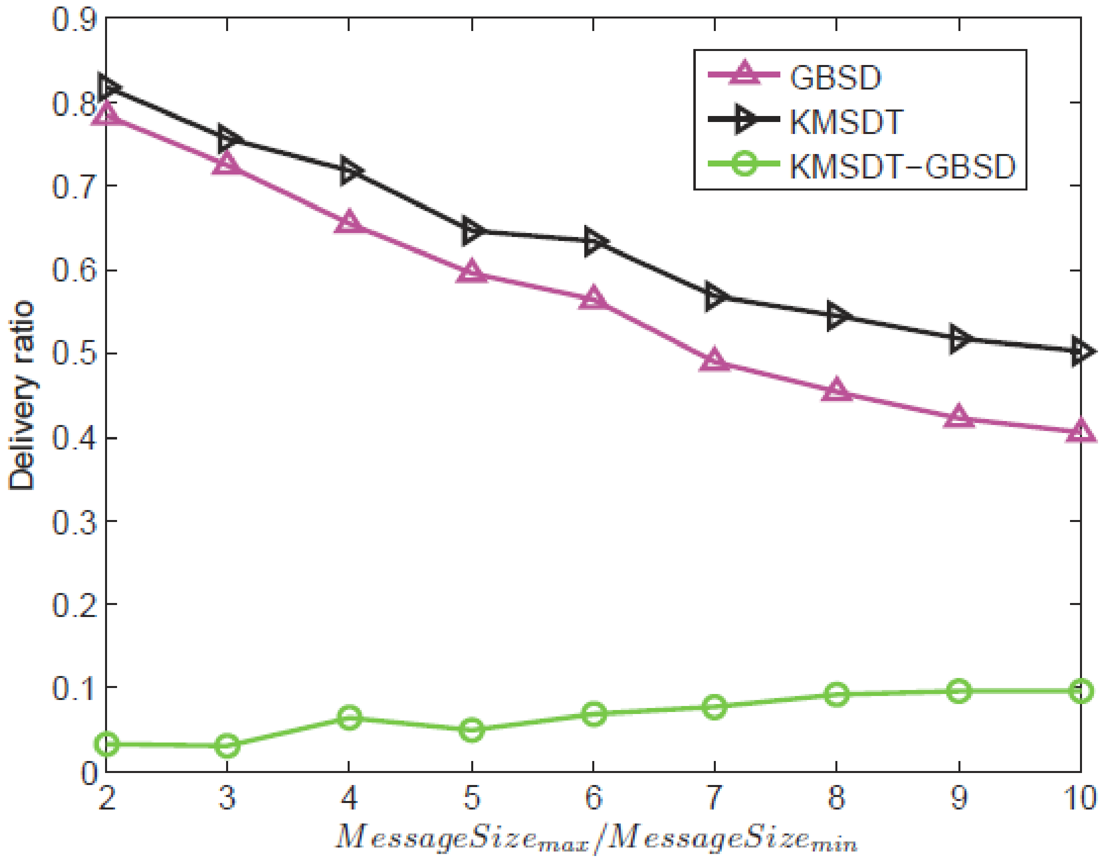

- **Df: drop the message first enters the buffer Dl: the last enters Do: the oldest Dy: the youngest**

# 4. Evaluation

## 4.4 Different Distributions of Message Sizes

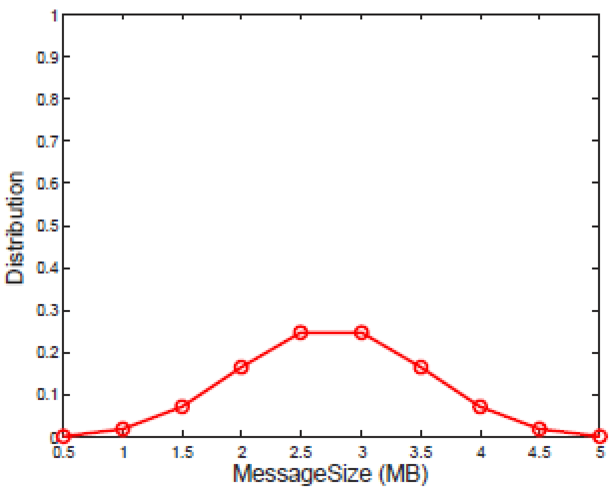



(a) Uniform distribution

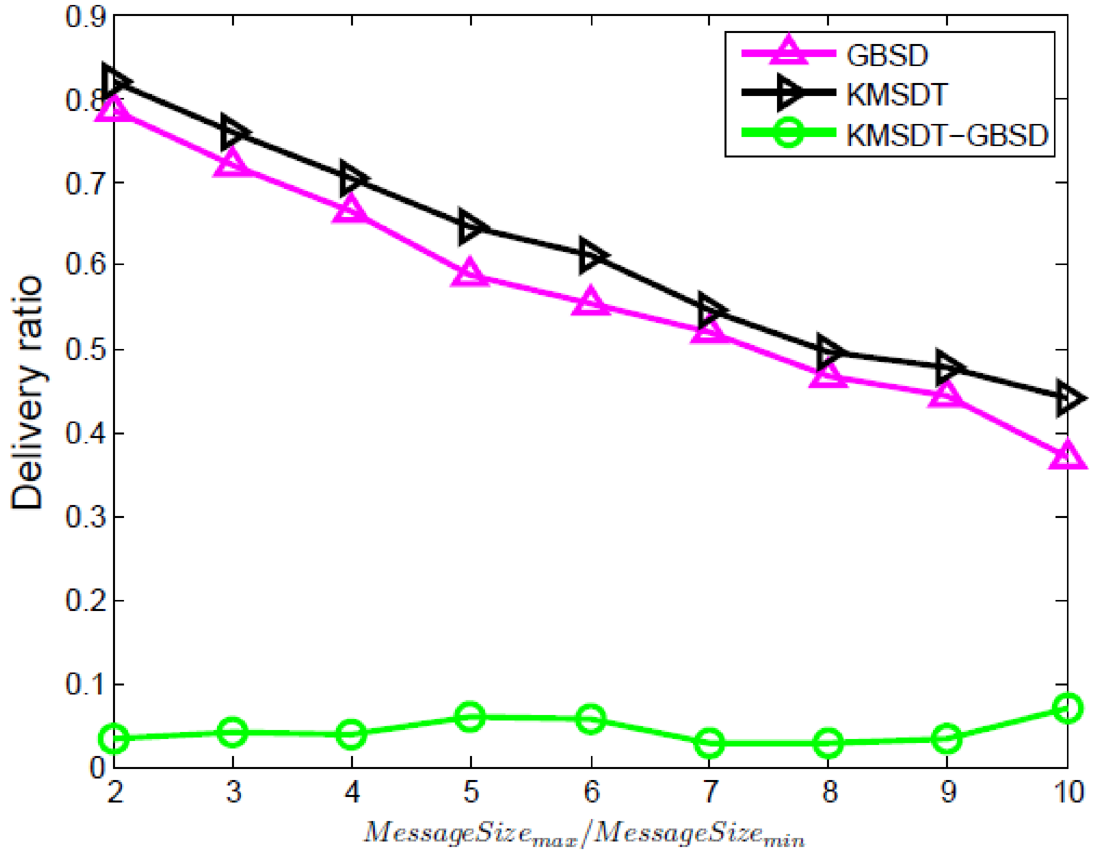


# 4. Evaluation

## 4.4 Different Distributions of Message Sizes

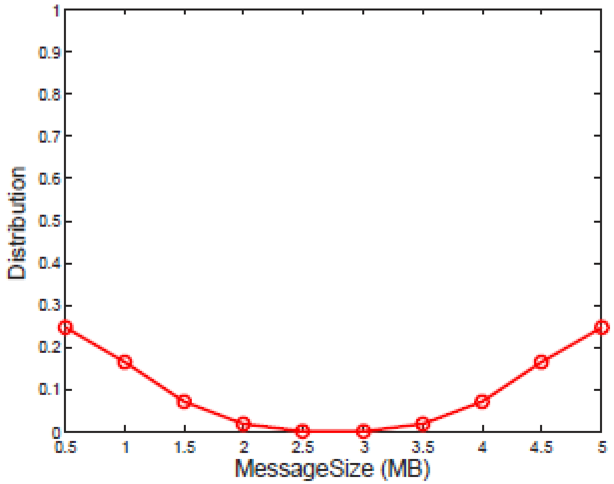



(b) ∩ distribution

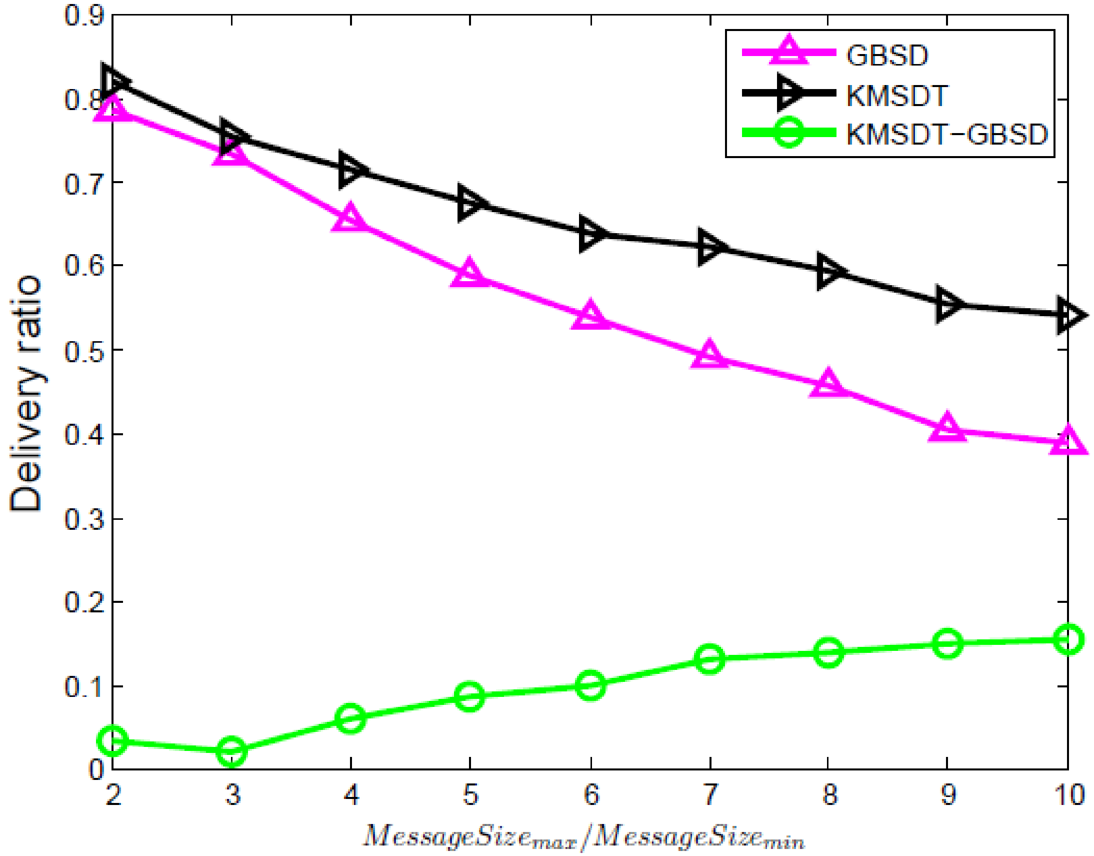


# 4. Evaluation

## 4.4 Different Distributions of Message Sizes




(c) ∪ distribution



# Outline

- 1. Introduction
- 2. Model Description
- 3. Scheduling and Drop Strategy
- 4. Evaluation
- **5. Future Work**

# 1. Future Work

- **Other replication-based routing schemes**
  - **Spread-and-wait, delegation forwarding, etc**

- **The fairness of messages in different sizes**

# Thank You