Delay-Tolerant Networking
Internet-Draft
Intended status: Informational
Expires: May 06, 2020

Y. W. Chung
M. W. Kang
D. Y. Seo
Y. Kim
Soongsil University
November 04, 2019

# Extension of Probabilistic Routing Protocol using History of Encounters and Transitivity for Information Centric Network

draft-chung-dtn-extension-prophet-icn-05.txt

## Abstract

This document proposes extension of probabilistic routing protocol using history of encounters and transitivity (PRoPHET) for information centric network.

Table of Contents



# 1. Introduction

In Information centric network (ICN), a node requests *Data* by sending *Interest* packet and this *Interest* packet is forwarded through ICN routers. A router with the requested *Data* replies to the *Interest* to the requester and the *Interest* is delivered through a reverse path of the forwarded *Interest*. ICN router manages content store (CS), pending interest table (PIT), and forwarding information base (FIB) [George2014]. In CS, cached data is stored for future use. In PIT, the information of *Interest*, the incoming and outgoing faces of the *Interest* are stored, and this information is used to deliver *Data* to the requester using the reverse path of forwarded *Interest*. FIB is used to forward *Interest* to appropriate faces.

ICN is considered important for communication of urgent messages in disaster situations [Edo2014]. In disaster situations, communication infrastructure is destroyed and networks are fragmented. In fragmented networks where connectivity between the nodes at different fragmented networks is not possible, opportunistic network such as delay tolerant networks (DTN) can be used to deliver messages. In DTN, a message is delivered to a destination node via opportunistic contacts between intermediate nodes in a store-carry-forward way.

Since forwarding of *Interest* and *Data* should be carried out opportunistically using DTN in fragmented networks, forwarding schemes of *Interest* and *Data* in connected ICN networks should be extended to accommodate the disruptive characteristics of DTN. In this draft, we consider probabilistic routing protocol using history of encounters and transitivity (PRoPHET)[RFC6693] for extension. Then, we propose forwarding schemes for *Interest* and *Data* of ICN.

# 2. Conventions and Terminology

## 2.1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2.2. Terminology

TBD

# 3. Forwarding of *Interest* and *Data* for ICN

## 3.1. Delivery predictability of PRoPHET

In PRoPHET, delivery predictability is defined between any two nodes. The delivery predictability between node A and node B i.e., P(A,B), increases whenever node A and node B contact as follows:

$$P(A,B)=P(A,B)\_old+(1-delta-P(A,B)\_old)*P\_encounter, (1)$$

where delta sets an upper bound for P(A,B) and P_encounter is a scaling factor to control the rate of increase [RFC6693].

Also, it decreases as time elapses since the last contact as follows:

P(A,B)=P(A,B)_old*gamma^K,(2)

where 0<=gamma<=1 is an aging constant and K is the elapsed time.

Finally, the delivery predictability has a transitive property i.e., if node A and B encounter frequently, and node B and node C encounter frequently, then node A probably encounters node C as follows:

P(A,C)= MAX(P(A,C)_old,P(A,B)*P(B,C)*beta),(3)

## 3.2. Extension for Interest forwarding

Conventional DTN routing protocol is based on push model and the destination of a message is a specific node. However, pull model is used in ICN and *Interest* is forwarded based on content name, rather than node ID. In order to forward *Interest* to appropriate nodes which have the requested *Data* in its CS, the delivery predictability of a node A for the *Interest i* corresponding to the requested *Data* is defined as P(A,N(d_i)), similar to Eq. (1) as follows:

P(A,N(d_i))

=P(A,N(d_i))_old+(1-delta-P(A,N(d_i)_old)*P_encounter,(4)

where N(d_i) represents a set of nodes with the *Data* corresponding to *Interest* i in its CS.

In Eq. (4), P(A,N(d_i)) increases whenever node A contacts another node which has d_i in its CS, where the number of nodes having *Data* d_i is generally larger than 1, since d_i can be cached in multiple nodes by adopting the ICN approach. Similar to Eq. (2), the delivery predictability of a node to a node set N(d_i) decreases as time elapses since the last contact. We note that if node A has *Data* d_i, P(A,N(d_i))=1.

When node A and node B contact, *Interest* i stored in node A is forwarded to node B, if P(A,N(d_i)) < P(B,N(d_i)), since node B is a more probable node to deliver *Interest* i to a node having d_i than node A. In this case, the information of requester nodes for *Interest* i is also delivered to node B. The information of requester nodes for the same *Interest* i stored in both node A and node B is shared, irrespective of the comparison of delivery predictabilities. For example, if node A has *Interest* i with requester R1 and if node B has *Interest* i with requester R2, both node A and node B have information of requesters R1 and R2 for *Interest* i after contact.

## 3.3. Extension for Data forwarding

For the delivery of *Data* in DTN, there is no known reverse path like the one using PIT in ICN. Therefore, *Data* also should be delivered using DTN routing protocol, too. In the proposed extension, the information of requesters for the considered *Data* is used to forward the *Data*. If the number of requesters for the Data corresponding to *Interest* i is only one, the forwarding scheme of conventional PRoPHET can be applied directly since the destination of the *Data* is a requester node and forwarding is carried out based on node ID. That is, if P(B,R(d_i)) is larger than P(A,R(d_i)), the *Data* d_i is forwarded to node B, where R(d_i) is defined as the requester node for the *Data* corresponding to *Interest* i.

If there are multiple requesters for the *Data* corresponding to *Interest* i, current forwarding scheme of PRoPHET should be extended, too, based on the delivery predictability relationship of two contact nodes for each requester. In this draft, three forwarding schemes for multiple requesters are presented in as examples. If node A and B contact and node A has *Data* with multiple requesters, the *Data* can be forwarded to node B if any of the following condition is met depending on the selected policy:

1) if the delivery predictability between node B and a requester is larger than that between node A and the corresponding requester for any requester,

2) if the delivery predictability between node B and a requester is larger than that between node A and the corresponding requester for all requesters,

3) if the average of the delivery predictabilities of node B and requesters are larger than that between node A and the corresponding requesters.

For example, if node A has *Data* d_i with requesters R1 and R2 and if node B does not have *Data* d_i already when node A and node B contact, *Data* d_i in node A will be forwarded to node B depending on a *Data* forwarding policy as follows:

1) if P(A,R1(d_i))<P(B,R1(d_i)) or if P(A,R2(d_i))<P(B,R2(d_i));(5)

2) if P(A,R1(d_i))<P(B,R1(d_i)) and if P(A,R2(d_i))<P(B,R2(d_i));(6)

3) if Average(P(A,R1(d_i)),P(A,R2(d_i)))

< Average(P(B,R2(d_i)),P(B,R2)(d_i)).(7)

Information on requesters is also delivered if *Data* is forwarded. If both node A and node B have the same *Data*, the information of requesters is shared between node A and node B.

## 3.4. Extension for caching

In ICN, *Data* can be cached at the CS of nodes for future use. However, due to the limited memory size of CS of mobile nodes, it is necessary to restrict the lifetime of the cached Data. In this draft, a TTL(time-to-live) value is defined for each cached Data. For simplicity, TTL of cached Data can be defined as a predefined constant value. For performance enhancement, however, the value of TTL can be defined as a dynamic value. For example, the value of TTL of cached Data can be determined depending on the delivery predictability to the requester node. If the number of requesters for the Data corresponding to *Interest* i is only one, the TTL value can be defined based on the delivery predictability of a node to the requester node. If the delivery predictability of a node to a requester node is higher, the node should cache the Data longer for a better delivery, and a higher value of TTL should be set. On the other hand, if the delivery predictability is lower, the TTL value should be set as a lower value. Therefore, TTL value can be a function of delivery predictability and various functions can be defined. For example, a linear function for TTL can be defined based on the delivery predictability as shown in Eq. (8) when the Data is initially cached:

TTLinit = (TTLmax - TTLmin)* P(A,requester) + TTLmin (8)

where TTLmax and TTL min are predefined maximum TTL value and minimum TTL value, respectively.

As time elapses, the value of TTL decreases and if it expires, the cached Data are removed from the CS. Since the delivery predictability increases according to Eqns. (1) and (3), we need to increase the current TTL value depending on the current delivery predictability value. This is because if the delivery predictability increases according to Eqns. (1) and (3), it is more probable to deliver the cached Data to the destination and thus, TTL should be extended for better delivery. The amount of increased TTL value can be defined in various ways. For example, if

TTLnew = TTLcurrent

+(TTL_max - TTL_min)*(P(A,requester)new - P(A,requester)old) (9)

where TTLnew and TTLcurrent are updated TTL value and current TTL value, respectively, and P(A,requester)new and P(A,requester)old are updated delivery predictability value and current delivery predictability value, respectively. We note that since TTL value naturally decreases as time elapses, the effect of decreasing delivery predictability based on Eq. (2) on TTL value is not considered to additionally decrease the current TTL value.

If the number of requesters for the Data corresponding to *Interest* i is multiple, the TTL value can be determined based on the delivery predictability of a node to the requester nodes. In this draft, three schemes are proposed to determine the TTL value using delivery predictability in Eq. (9) for multiple requesters are presented as follows:

1) TTL value is defined based on the minimum value of delivery predictabilities to the requester nodes,

2) TTL value is defined based on the maximum value of delivery predictabilities to the requester nodes,

3) TTL value is defined based on the average value of delivery predictabilities to the requester nodes,

The TTL value for multiple requesters can be updated corresponding to the varying values of delivery predictability in the selected scheme, too, similar to the case where the requester node is only one.

## 3.5. Operation of the proposed extension

In the proposed forwarding scheme, whenever node A and node B contact, they exchange *Interest* list and *Data* list. *Interest* list contains all the *Interests* that they receive from other nodes, where information for the requesters for *Interest* i is also managed in *Interest* list. *Data* list contains all *Data* that they cache in their CS for future delivery. Also, the information for the destination nodes of the *Data*, i.e., requesters, is also managed in *Data* list. Then, node A compares its *Interest* list with node B’s *Interest* list and forwards *Interest* i to Node B if node B does not have the *Interest* and P(B,N(d_i)) is larger than P(A,N(d_i)). The information of requester nodes for the same *Interest* i stored in both node A and node B is shared between both node A and node B after the contact.

```
+-------------------------------------------------------------------+
|  +=============================+  +=============================+  |
|  |  Interest List in Node A    |  |  Interest List in Node B    |  |
|  +=============================+  +=============================+  |
|  |  ID  | Data ID | Requester |  |  ID  | Data ID | Requester |  |
|  +======+=========+===========+  +======+=========+===========+  |
|  |  i_1 |   d_1   |    R1      |  |  i_3 |   d_1   |    R3      |  |
|  +------+---------+-----------+  +=============================+  |
|  |  i_2 |   d_2   |    R2      |  +=============================+  |
|  +------+---------+-----------+  |     Data List in Node B     |  |
|  |  i_4 |   d_4   |    R1      |  +=============================+  |
|  +=============================+  |  ID  |      Requester       |  |
|                                   +======+======================+  |
|                                   |  d_3 |          R4          |  |
|                                   +=============================+  |
|                                    ___  ___                       |
|                                   /   \/   \                      |
|                                  (  A () B  )                     |
|                                   \___/\___/                      |
|                                                                   |
|                        <Node A contacts node B>                   |
+-------------------------------------------------------------------+
```

Fig 1. Interest Forwarding Procedure (at time t)

Each node has a table for delivery predictability to a set of nodes with *Data* corresponding to *Interest* in each node, as shown in Tables 1 and 2.

Table 1. Delivery predictability to a set of nodes with *Data* corresponding to *Interest* in node A(at time t)

| Node set | Delivery Predictability |
|---|---|
| N(d_1) | 0.5 |
| N(d_2) | 0.6 |
| N(d_4) | 0.8 |

Table 2. Delivery predictability to a set of nodes with *Data* corresponding to *Interest* in node B(at time t)

| Node set | Delivery Predictability |
|---|---|

| | |
|---|---|
| N(d_1) | 0.3 |
| N(d_2) | 0.7 |

After the contact of node A and node B, the requester information for the same *Data* ID in *Interest* table is shared and thus requesters R1 and R3 are stored in both node A and node B. Since the delivery predictability of N(d_2) of node B is higher than that of node A, requester information R2 is forwarded to node B.

Since node A contacts with node B which has *Data* d_3 in its cache, delivery predictability of node A is updated, as shown in Table 3. Since node B does not have delivery predictability to a node set N(d_4) before contact, the delivery predictability of node B to a node set is updated using transitivity property.

```
+-----------------------------------------------------------------+
|  +============================+  +============================+  |
|  |  Interest List in Node A   |  |  Interest List in Node B   |  |
|  +============================+  +============================+  |
|  |  ID  | Data ID | Requester |  |  ID  | Data ID | Requester |  |
|  +======+=========+===========+  +======+=========+===========+  |
|  | i_1 |   d_1   |  R1, R3   |  | i_3 |   d_1   |  R1, R3   |  |
|  +------+---------+-----------+  +------+---------+-----------+  |
|  | i_2 |   d_2   |    R2     |  | i_2 |   d_2   |    R2     |  |
|  +------+---------+-----------+  +============================+  |
|  | i_4 |   d_4   |    R1     |  +============================+  |
|  +============================+  |       Data List in B       |  |
|                                  +============================+  |
|                                  |  ID  |      Requester      |  |
|                                  +======+=====================+  |
|                                  |  d_3 |          R4         |  |
|                                  +============================+  |
|                                                                 |
|                        ___          ___                         |
|                       /   \        /   \                        |
|                      (  A  )      (  B  )                       |
|                       \___/        \___/                        |
|                                                                 |
|                    <Node A disconnects node B>                  |
+-----------------------------------------------------------------+
```

Fig 2. Interest Forwarding Procedure (at time t+dt)

Table 3. Delivery predictability to a set of nodes with *Data* corresponding to *Interest* in node A(at time t+dt)

| Node set | Delivery Predictability |
| --- | --- |
| N(d_1) | 0.5 |
| N(d_2) | 0.6 |
| N(d_4) | 0.8 |
| N(d_3) | 0.5 |

Table 4. Delivery predictability to a set of nodes with *Data* corresponding to *Interest* in node B(at time t+dt)

| Node set | Delivery Predictability |
| --- | --- |
| N(d_1) | 0.3 |
| N(d_2) | 0.7 |
| N(d_4) | 0.36 |

For *Data* forwarding, node A checks *Data* list. If node A has only one requester information for the considered *Data*, node A forwards *Data* d_i, which corresponds to *Interest* i, if node B does not have the *Data* and P(B,R(d_i)) is larger than P(A,R(d_i)). If node A has multiple requesters information for the considered *Data*, *Data* can be forwarded to node B if any of forwarding condition for multiple requesters defined in this draft is met, as proposed in Eqns. (4)-(6). Information on requesters is delivered if *Data* is forwarded. If both node A and node B have the same *Data*, the information of requesters is shared between node A and node B after the contact.

Figures 3 and 4 show an example of the proposed *Data* forwarding procedure. Each node has a *Data* list table, where the information of *Data* and requester who requested the *Data* is stored.

```
+-------------------------------------------------------------------+
|  +============================+  +============================+   |
|  |       Data List in Node C  |  |     Data List in Node D    |   |
|  +============================+  +============================+   |
|  |  ID  |      Requester      |  |  ID  |      Requester      |   |
|  +======+=====================+  +======+=====================+   |
|  |  d_1 |       R1, R3        |  |  d_2 |         R4          |   |
|  +------+---------------------+  +============================+   |
|  |  d_2 |         R2          |                                   |
|  +============================+                                   |
|                                   ___  ___                        |
|                                  /   \/   \                       |
|                                 (  C () D  )                      |
|                                  \___/\___/                       |
|                                                                   |
|                        <Node C contacts node D>                   |
+-------------------------------------------------------------------+
```

Fig 3. Data Forwarding Procedure (at time t)

Table 5 and Table 6 show delivery predictability to requester node for corresponding data in each node.

Table 5. Delivery predictability to requester node for corresponding *Data* in node C (at time t)

| Node ID | Delivery Predictability |
|---|---|
| R1 | 0.9 |
| R2 | 0.6 |
| R3 | 0.2 |
| R4 | 0.7 |

Table 6. Delivery predictability to requester node for corresponding *Data* in node D (at time t)

| Node ID | Delivery Predictability |
|---|---|
| R1 | 0.7 |
| R2 | 0.7 |

| | |
|---|---|
| R3 | 0.6 |
| R4 | 0.9 |

As shown in Figure 4, requester information is shared between two nodes. Thus requester information for *Data* d_2 is shared as R2 and R4 and the requester information for *Data* d_1 of node A is transferred to node B.

```
+-------------------------------------------------------------------+
|  +=============================+  +=============================+  |
|  |     Data List in Node C     |  |     Data List in Node D     |  |
|  +=============================+  +=============================+  |
|  |  ID  |      Requester       |  |  ID  |      Requester       |  |
|  +======+======================+  +======+======================+  |
|  | d_1 |        R1, R3         |  | d_2 |        R4, R2         |  |
|  +------+----------------------+  +------+----------------------+  |
|  | d_2 |        R2, R4         |  | d_1 |        R1, R3         |  |
|  +=============================+  +=============================+  |
|                                                                   |
|                          ___          ___                        |
|                         /   \        /   \                       |
|                        (  C  )      (  D  )                      |
|                         \___/        \___/                       |
|                                                                   |
|                    <Node C disconnects node D>                    |
+-------------------------------------------------------------------+
```
Fig 4. Data Forwarding Procedure (at time t+dt)

Table 7 and Table 8 show delivery predictability to requester node for corresponding data in node A and node B, respectively after the contact, where the delivery predictability is updated.

Table 7. Delivery Predictability to requester node for corresponding data in node C (at time t+dt)

| Node ID | Delivery Predictability |
|---|---|
| R1 | 0.9 |
| R2 | 0.6 |
| R3 | 0.27 |
| R4 | 0.7 |

| | |
|---|---|
| D | 0.5 |

Table 8. Delivery Predictability to requester node for corresponding data in node D (at time t+dt)

| Node ID | Delivery Predictability |
|---|---|
| R1 | 0.7 |
| R2 | 0.7 |
| R3 | 0.6 |
| R4 | 0.9 |
| C | 0.5 |

## 3.6. Extension for overload control

In the proposed forwarding scheme, a requester node which issues an *Interest* message does not know whether the Interest message has been delivered to a node which has the requested Data until it receives a requested Data. Therefore, unnecessary Interest messages may be forwarded further even though it has been successfully delivered to a node which has the requested Data already. Also, unnecessary Data may be forwarded further even though it has been delivered to a requester node already. Therefore, it is necessary to limit this unnecessary overload of Interest and Data efficiently. In this draft, we propose an extension for overload control, which is basically based on the schemes proposed in the work in [Hass2006].

In the proposed overload control, we manage delivered Interest and Data list in the pending anti-Interest and Data (PAID) table. If node A forwards an Interest message i_1 to a node B which has the requested Data d_1, we can apply one of the following three schemes to limit the forwarding of the satisfied Interest message efficiently as follows:

1) Scheme A: the node A removes the delivered Interest i_1 from its Interest list and sets anti-Interest flag for the Interest message i_1 in PAID table. Then, node A does not accept the i_1 again.

2) Scheme B: the node A removes the delivered Interest i_1 from its Interest list and sets anti-Interest flag for the Interest message i_1 in PAID table, and does not accept the i_1 again. Further, if node A contacts another node C which has the same Interest i_1, it shares anti-Interest flag with node C. Then, node C removes the Interest i_1 from the Interest list and sets anti-Interest flag for the Interest message i_1 in PAID table. The node C does not accept the i_1 again.

3) Scheme C: the node A removes the delivered Interest i_1 from its Interest list and sets anti-Interest flag for the Interest message i_1 in PAID table, and does not accept the i_1 again. Further, if node A contacts any node, it shares anti-Interest flag with the contact node. If the contact node has the Interest i_1 already, it removes the Interest i_1 from the Interest list and sets anti-Interest flag for the Interest message i_1 in PAID table, and does not accept the Interest i_1 again. Otherwise, it just sets anti-Interest flag for the Interest message i_1 in PAID table and does not accept the i_1 again.

```
+-------------------------------------------------------------------+
|  +=============================+  +=============================+  |
|  |  Interest List in Node A    |  |  Interest List in Node C    |  |
|  +=============================+  +=============================+  |
|  |  ID   | Data ID | Requester |  |  ID   | Data ID | Requester |  |
|  +======+=========+===========+  +======+=========+===========+  |
|  |  i_1 |   d_1   |    R1     |  |  i_3 |   d_1   |    R3     |  |
|  +=============================+  +=============================+  |
|  +=============================+  +=============================+  |
|  |     PAID table in Node A    |  |     PAID table in Node C    |  |
|  +=============================+  +=============================+  |
|  |Data ID| Requester ID| Flag |  |Data ID| Requester ID| Flag |  |
|  +=============================+  +=============================+  |
|  |  d_1  |      R1      |  0   |  |  d_1  |      R1      |  1   |  |
|  +=============================+  +       +--------------+------+  |
|                                   |       |      R3      |  0   |  |
|                                   +=============================+  |
|                                                                   |
|                            ____          ____                    |
|                           /    \        /    \                   |
|                          (  A   )      (  C   )                  |
|                           \____/        \____/                   |
|                                                                   |
|                      <Node A contacts node C>                     |
+-------------------------------------------------------------------+
```

Fig 5. Overload control for *Interest* (at time t)

```
+-------------------------------------------------------------------+
|  +=============================+  +=============================+  |
|  |  Interest List in Node A    |  |  Interest List in Node C    |  |
|  +=============================+  +=============================+  |
|  |  ID   | Data ID | Requester |  |  ID   | Data ID | Requester |  |
|  +======+=========+===========+  +======+=========+===========+  |
|  |  i_1 |   d_1   |    R3     |  |  i_3 |   d_1   |    R3     |  |
|  +=============================+  +=============================+  |
|  +=============================+  +=============================+  |
|  |     PAID table in Node A    |  |     PAID table in Node C    |  |
|  +=============================+  +=============================+  |
|  |Data ID| Requester ID| Flag |  |Data ID| Requester ID| Flag |  |
|  +=============================+  +=============================+  |
|  |  d_1   |      R1     |  1   |  |  d_1   |      R1     |  1   |  |
|  +        +-------------+------+  +        +-------------+------+  |
|  |        |      R3     |  0   |  |        |      R3     |  0   |  |
|  +=============================+  +=============================+  |
|                                                                   |
|                        ____         ____                          |
|                       /    \       /    \                         |
|                      (  A   )     (  C   )                        |
|                       \____/       \____/                         |
|                                                                   |
|                     <Node A disconnects node C>                   |
+-------------------------------------------------------------------+
```

Fig 6. Overload control for *Interest* (at time t+dt)

Similar approaches can be applied to delivered Data, too. If Data d_2 is delivered to a node E from a node D, which requested the Data d_2 before, we can apply one of the following three schemes to limit the forwarding of the delivered Data efficiently as follows:

1) Scheme D: the node D removes the delivered Data d_2 from its Data list and sets anti-Data flag for the Data d_2 in PAID table. Then, node D does not accept the d_2 again.

2) Scheme E: the node D removes the delivered Data d_2 from its Data list and sets anti-Data flag for the Data d_2 in PAID table, and does not accept the d_2 again. Further, if node D contacts another node F which has the same Data d_2, it shares anti-Data flag with node F. Then, node F removes the Data d_2 from the Data list and sets anti-Data flag for the Data d_2 in PAID table. The node F does not accept the d_2 again.

3) Scheme F: the node D removes the delivered Data d_2 from its Data list and sets anti-Data flag for the Data d_2 in PAID table, and does not accept the d_2 again. Further, if node D contacts any node, it shares anti-Data flag with the contact node. If the

contact node has the Data d_2 already, it removes the Data d_2 from Data list and sets anti-Data flag for the Data d_2 in PAID table, and does not accept the Data d_2 again. Otherwise, it just sets anti-Data flag for the Data d_2 in PAID table and does not accept the d_2 again.

```
+------------------------------------------------------------------+
|  +=============================+  +=============================+  |
|  |     Data List in Node D     |  |     Data List in Node F     |  |
|  +=============================+  +=============================+  |
|  |  ID  |       Requester      |  |  ID  |       Requester      |  |
|  +======+======================+  +======+======================+  |
|  |      |                      |  | d_2  |          D           |  |
|  +------+----------------------+  +=============================+  |
|  +=============================+  +=============================+  |
|  |     PAID table in Node D    |  |     PAID table in Node F    |  |
|  +=============================+  +=============================+  |
|  |Data ID| Requester ID| Flag |  |Data ID| Requester ID| Flag |  |
|  +=============================+  +=============================+  |
|  |  d_2  |      D       |  0   |  |  d_2  |      D       |  0   |  |
|  +=============================+  +       +-------------+------+  |
|                                   |       |     R4      |  1   |  |
|                                   +=============================+  |
|                                                                  |
|                         ____         ____                        |
|                        /    \       /    \                       |
|                       (  D   )     (  F   )                      |
|                        \____/       \____/                       |
|                                                                  |
|                      <Node D contacts node F>                    |
+------------------------------------------------------------------+
```

Fig 7. Overload control for *Data* (at time t)

```
+------------------------------------------------------------------+
|  +=============================+  +=============================+  |
|  |     Data List in Node D     |  |     Data List in Node F     |  |
|  +=============================+  +=============================+  |
|  |  ID  |       Requester      |  |  ID  |       Requester      |  |
|  +======+======================+  +======+======================+  |
|  |      |                      |  |      |                      |  |
|  +------+----------------------+  +=============================+  |
|  +=============================+  +=============================+  |
|  |     PAID table in Node D    |  |     PAID table in Node F    |  |
|  +=============================+  +=============================+  |
|  |Data ID| Requester ID| Flag |  |Data ID| Requester ID| Flag |  |
|  +=============================+  +=============================+  |
|  |  d_2  |      D       |  1   |  |  d_2  |      D       |  1   |  |
|  +       +-------------+------+  +       +-------------+------+  |
```

```
   |  |       |     R4      |  1   |  |       |     R4      |  1   |  |
   |  +===========================+  +===========================+  |
   |                                                                 |
   |                       /    \       /    \                       |
   |                      (  D  )      (  F  )                       |
   |                       \____/       \____/                       |
   |                                                                 |
   |                  <Node D disconnects node F>                    |
   +-----------------------------------------------------------------+
```

Fig 8. Overload control for *Data* (at time t+dt)

3.7. Overload control based on context information

The overload control schemes in Section 3.6 can be applied dynamically, depending on the context information of Interest and Data, since forwarding of Interest and Data should be treated efficiently by considering context information. In the proposed scheme, a non-overload control scheme is basically applied and if a condition is met, overload control scheme proposed in Section 3.6 is applied. Although numerous context information can be used, we consider the number of hop counts, TTL, and the number of requester nodes as examples as follows:

1) Number of hop counts: If the number of hop counts of Interest and Data are not larger than threshold values, an overload control scheme is not applied. Otherwise, an overload control scheme is applied. The threshold value of Interest and Data can be defined differently depending on the urgency of the Interest and Data.

2) TTL: If the TTL values of Interest and Data are not lager than threshold values, an overload control scheme is not applied. Otherwise, an overload control scheme is applied. This is because if TTL of Interest and Data is larger, it means that it has been forwarded more, and thus overload control scheme is needed to avoid unnecessary forwarding.

3) Number of requester nodes: If the number of requester nodes of Interest and Data are not larger than threshold values, an overload control scheme is not applied. Otherwise, an overload control scheme is applied. This is because if the number of request nodes is smaller, more message dissemination is favorable and thus, overload control scheme should not be applied.

# 4. Security Considerations

TBD

# 5. IANA Considerations

TBD

# 6. References

## 6.1. Normative References

[RFC6693] Lindgren, A., Doria, A., Davies, E., Grasic, S, "Probabilistic routing protocol for intermittently connected networks", RFC 6693, August 2012.

## 6.2. Informative References

[George2014] Xylomenos, G. Ververidis, C. N., Siris, V. A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. V. Polyzos, G. C., "A Survey of Information-Centric Networking Research", IEEE Communications Surveys and Tutorials, Vol. 16, No. 2, 2014.

[Edo2014] Monticelli, E., Schubert, B. M., Arumaithurai, M., Fu, X., Ramakrishnan, K. K., "An Information Centric Approach for Communications in Disaster Situations," Proceedings of IEEE Local & Metropolitan Area Networks, USA, May 2014.

[Hass2006] Hass, Z. J., Small, T., "A new networking model for biological applications of ad hoc sensor networks", IEEE/ACM Transactions on Networking, Vol. 14, No. 1, pp. 27-40, Feb.,2006.

## Authors' Addresses

Yun Won Chung
Soongsil University
369, Sangdo-ro, Dongjak-gu,
Seoul, 06978, Korea

Email: ywchung@ssu.ac.kr

Min Wook Kang
Soongsil University
369, Sangdo-ro, Dongjak-gu,
Seoul, 06978, Korea

Email: goodlookmw@gmail.com

Dong Yeong Seo
Soongsil University
369, Sangdo-ro, Dongjak-gu,
Seoul, 06978, Korea

Email: seodong2da@nate.com

Younghan Kim
Soongsil University
369, Sangdo-ro, Dongjak-gu,
Seoul, 06978, Korea

Email: younghak@ssu.ac.kr