Chloé Desdouits 1 février 2012

Résumé de l'article de Jae-Hwan Chang et Leandros Tassiulas Energy Conservating Routing in Wireless Ad-hoc Networks

La définition de la durée de vie du réseau est le temps qui s'écoule jusqu'au moment où le premier capteur tombe en panne de batterie (TTFF). Les réseaux considérés comprennent une ou plusieurs composantes où chaque composante contient un ensemble de nœuds sources et un ensemble de nœuds destination.

Voici la formule qui donne le temps de vie du réseau :

$$T_{sys}(q) = min_{i \in N} \frac{E_i}{\sum\limits_{j \in S_i} e_{ij} \sum\limits_{c \in C} q_{ij}^{(c)}}$$

avec :
- q le flot
- N l'ensemble des nœuds du graphe (les capteurs)
- E_i la quantité d'énergie initiale du nœud i
- S_i l'ensemble des voisins du nœud i
- e_{ij} l'énergie requise pour transmettre une unité d'information du nœud i au nœud j
- C l'ensemble des composantes du réseau
- $q_{ij}^{(c)}$ le taux d'information de la composante c transmise du nœud i au nœud j

L'auteur voit le problème comme un problème de flot maximum et propose donc un algorithme d'augmentation de flot (FA). Il étend également un algorithme de redirection de flot au cas où il y a plusieurs composantes dans le réseau.

1 Algorithme $FA(x_1, x_2, x_3)$ d'augmentation de flot

L'idée consiste à utiliser un algorithme de plus court chemin existant (comme l'algorithme distribué de Bellman-Ford) et de lui passer trois paramètres (x_1, x_2 et x_3) qui détermineront sa fonction de poids sur les arcs. Ces paramètres coefficienteront les trois facteurs suivants : le coût de transmission du nœud i au nœud j (e_{ij}) ; l'énergie résiduelle du nœud i ($\underline{E_i}$), et l'énergie initiale du nœud i (E_i). Voici la formule qui donne le poids de l'arc (i, j) en fonction des paramètres :

$$c_{ij} = e_{ij}^{x_1} \underline{E_i}^{-x_2} E_i^{x_3}$$

Algorithm 1 $FA(x_1, x_2, x_3)$

Require: a network $G(N, A)$
  while each node i has some energy **do**
    for each commodity $c \in C$ **do**
      for each origin node $o \in O^{(c)}$ **do**
        compute the shortest cost path from o to $D^{(c)}$
        augment the flow on this path by an amount of $\lambda Q_i^{(c)}$
      end for
    end for
  end while
  return the total flow q in the network

2 Algorithme FR de redirection de flot

Cet algorithme est basé sur le fait que le flot trouvé est optimal ssi pour tous les chemins de l'origine à la destination portants un flot positif, la durée de vie minimum est la même. Le principe consiste donc à rediriger une partie du flot de chaque nœud à travers un autre chemin vers la destination tel que la durée de vie de chaque chemin portant du flot augmente (ou au moins reste stable). Le flot initial a pour valeur la quantité totale d'informations générées par le réseau.

Pour chaque nœud dans le réseau, on compare les chemins sortants de ce nœud vers la destination. On construit un ordre total sur ces chemins en les classant en fonction de leur nœud qui possède la plus courte durée de vie (puis par nombre de nœuds).

Si le nœud dont la durée de vie est la plus courte sur le chemin est le nœud source (i), alors on a deux choix. On regarde les voisins sortants de (i). Soit on redirige un des flots qui passe par le voisin (j) dont le coût de transmission e_{ij} est le plus grand vers un des chemins qui passe par le voisin (j) dont le coût de transmission e_{ij} est le moindre. Soit on redirige un flot quelconque vers le chemin le plus petit dans l'ordre total parmi ceux qui passe par le voisin (j) dont le coût de transmission e_{ij} est le moindre.

Si, en revanche, le nœud dont la durée de vie est la plus courte sur le chemin n'est pas le nœud source, on a également deux possibilités. Dans les deux cas on redirige le flot le plus petit dans l'ordre total. Soit on redirige vers le chemin le plus grand dans l'ordre total. Soit on redirige vers le chemin qui passe par le voisin (j) dont le coût de transmission e_{ij} est le moindre parmi les chemins supérieurs dans l'ordre total au chemin redirigé.

Une fois que l'algorithme a défini pour un nœud donné les deux chemins, il détermine la quantité de flot à rediriger.

Algorithm 2 $FR()$

Require: a network G(N, A)
  add to G an imaginary super destination node $\tilde{d}^{(c)}$
  for all $d \in D^{(c)}$ **do**
    $\tilde{d^{(c)}} \in S_d$
    $e_{d\tilde{d}^{(c)}} = 0$
  end for
  let q be the initial flow
  for all $o \in O^{(c)}$ **do**
    $q_{o\tilde{d}^{(c)}} = Q_o^{(c)}$
  end for
  for each $c \in C$ **do**
    for each $i \in N - D^{(c)}$ **do**
      (Determine the Two Paths)
      (Calculate $\epsilon_i^{(c)}$)
      (Redirect the flow)
    end for
  end for
  return

Algorithm 3 Determine the Two Paths

Form $G_F^{(c)}(N, A_F^{(c)})$ of $G(N, A)$ where $A_F^{(c)} = \{(i,j)|q_{ij}^{(c)} > 0, (i,j) \in A\}$
Let $P_i^{(c)}$ be the set of all paths in $G_F^{(c)}$ from node i to any $d \in D^{(c)}$
for each path $p \in P_i^{(c)}$ **do**
  Define $L_p(q)$ as a vector of lifetimes of all the nodes in the path p
end for
Let's define a total order on these paths such that a path p is smaller than another path p' iff the smallest element in p is smaller than the smallest element in p'
Use the Bellman-Ford algorithm to obtain the shortest length paths distributively
Let $sp^{(c)}(i)$ be the shortest path from i to $\tilde{d}^{(c)}$ in $G_F^{(c)}$
Let $lp^{(c)}(i)$ be the longest path from i to $\tilde{d}^{(c)}$ in G
Let $g \in S_i$ be the giver : the node from which path the flow will be subtracted
Let $t \in S_i$ be the taker : the node from which path the flow will be added
if $T_i(q) \leq min[L_{sp^{(c)}(i)}(q)]$ **then**
  OR :
  $g = \underset{j:j \in S_i, q_{ij}^{(c)} > 0}{argmax} e_{ij}$
  $t = \underset{j:j \in S_i}{argmin} e_{ij}$
  OR :
  $t = \underset{j:j \in S_i, e_{ij} < e_{ig}}{argmax} L_{lp^{(c)}(j)}(q)$
  $g =$ any node with non-minimum energy expenditure
else if $T_i(q) > min[L_{sp^{(c)}(i)}(q)]$ **then**
  g is the next hop node of node i in $sp^{(c)}(i)$
  OR :
  $t = \underset{j:j \in S_i}{argmax} L_{lp^{(c)}(j)}(q)$
  OR :
  $t = \underset{j:j \in S_i, min[L_{lp^{(c)}(j)}(q] > min[L_{sp^{(c)}(j)}(q]}{argmin} e_{ij}$
end if
The path composed of (i,g) and $sp^{(c)}(g)$ will be re-routed to the path composed of (i,t) and $lp^{(c)}(t)$

Algorithm 4 Calculate $\epsilon_i^{(c)}$

Choose ϵ_i such that :
for each link (i,j) in $sp^{(c)}(g)$ **do**
  $\epsilon_i \leq q_{ig}^{(c)}$
  $\epsilon_i \leq q_{jk}^{(c)}$
end for
if $T_i(q) \leq min[L_{sp^{(c)}(i)}(q)]$ **then**
  for each link (j,k) in $lp^{(c)}(t)$ **do**

$$\frac{1}{T_j(q)} + \frac{e_{jk}\epsilon_i^{(c)}}{E_j} \leq \frac{1}{T_i(q)}$$

  end for
else if $T_i(q) > min[L_{sp^{(c)}(i)}(q)]$ **then**

$$\frac{1}{T_j(q)} + \frac{e_{jk}\epsilon_i^{(c)}}{E_j} \leq \frac{1}{min[L_{sp^{(c)}(i)}(q)]}$$

  if $e_{it} > e_{ig}$ **then**

$$\frac{1}{T_j(q)} + \frac{(e_{it} - e_{ig})\epsilon_i^{(c)}}{E_i} \leq \frac{1}{min[L_{sp^{(c)}(i)}(q)]}$$

  end if
end if

3 Résultats des tests

Soit R_X le ratio de la durée de vie du réseau obtenue par l'algorithme X sur la durée de vie optimale.

TABLE 1 – Performance des algorithmes dans un réseau à une seule composante

Algorithme X	avg R_X	min R_X	$PrR_X > 0.9$
MTE	0.7310	0.1837	33%
FR	0.9596	0.6878	88%
MREP	0.9572	0.8110	89%
FA(1,1,1)	0.9744	0.7347	94%
FA(1,50,50)	0.9985	0.9911	100%

TABLE 2 – Performance des algorithmes dans un réseau à plusieurs composantes

Algorithme X	avg R_X	min R_X	$PrR_X > 0.9$
MTE	0.6982	0.2201	25%
FR	0.8862	0.4297	54%
MREP	0.9349	0.7298	69%
FA(1,1,1)	0.9565	0.7178	86%
FA(1,50,50)	0.9974	0.9906	100%