Energy-aware Path-planning for a Mobile Data Collector in Wireless Sensor Network

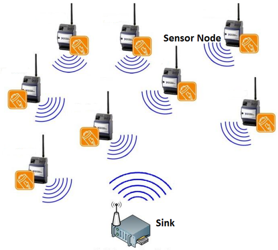
Md. Shaifur Rahman (0409052028)

M.Sc. Engineering Pre-Defense
Supervised By: Dr. Mahmuda Naznin

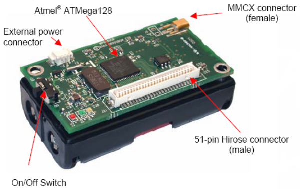
January 7, 2013

Outline

A wireless sensor network consists of some sensors and one or more sinks.



(a) A typical WSN



(b) A Sensor Mote (MICA)

(c) A sink

Data Gathering Problem in a WSN

Collecting data packets from the sensors by the sink

Introduction (Contd.)

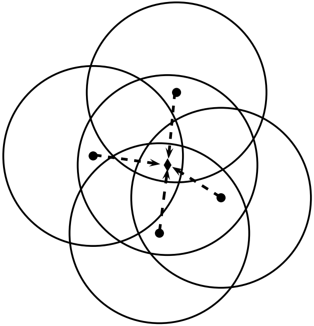
Types of Data Gathering Methods

1. Direct Contact
2. Multi-hop Forwarding
3. Mobile Elements (ME)

Introduction (Contd.)

Types of Data Gathering Methods

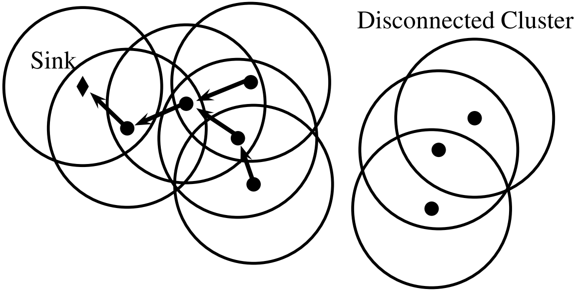
1. Direct Contact
2. Multi-hop Forwarding
3. Mobile Elements (ME)



Introduction (Contd.)

Types of Data Gathering Methods

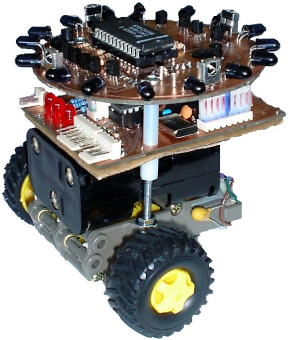
1. Direct Contact
2. Multi-hop Forwarding
3. Mobile Elements (ME)



Introduction (Contd.)

Types of Data Gathering Methods

1. Direct Contact
2. Multi-hop Forwarding
3. Mobile Elements (ME)



Introduction (Contd.)

Advantages of Using ME

1. Full Connectivity and Coverage
2. Cost Reduction
3. Minimization of Funneling Effect
4. Maximization of Network Lifetime
5. More Reliability

Challenges of Using ME

1. Path-planning for the ME
2. Contact Detection

Motivation

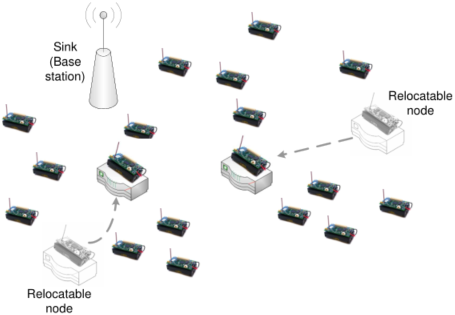
Different Roles of ME in a WSN

1. Relocatable Node
2. Mobile Peers
3. Mobile Data Collector (MDC)

Motivation

Different Roles of ME in a WSN

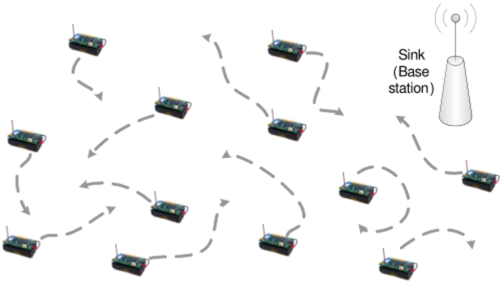
1. Relocatable Node
2. Mobile Peers
3. Mobile Data Collector (MDC)



Motivation

Different Roles of ME in a WSN

1. Relocatable Node
2. Mobile Peers
3. Mobile Data Collector (MDC)



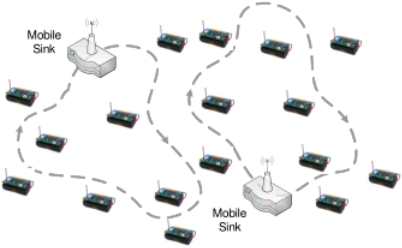
Motivation

Different Roles of ME in a WSN

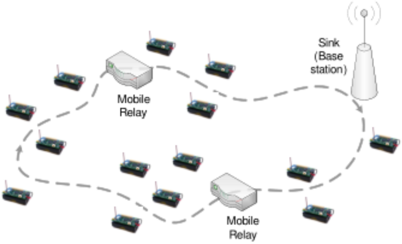
1. Relocatable Node
2. Mobile Peers
3. Mobile Data Collector (MDC)

Motivation (Contd.)

MDC are of two types: (a) Mobile Sink and (b) Mobile Relay



(a)



(b)

Motivation (Contd.)

Why MDC as a Data Collector

1. Cheaper than Mobile Relays & Mobile Peers
2. No Coordination Overhead
3. No Disruption to Sensing Activity

Related Work

- *Zhao and Ammar* introduced the concept *Message Ferry* in 2003
- *Message Ferry* not suitable for data gathering in WSN- lack of energy saving, requirements too demanding for sensor nodes
- *Shah et al.* introduced the concept of *Data Mule* in 2003
- *Data Mule* not suitable for data gathering in WSN- due to random motion, tour may not cover all the nodes
- Path-planning for MDC is done by *Wang et al.*(2005), *Rao et al.*(2008), *Rao and Biswas*(2010), *Anastasi et al.* (2011)
- None of the above works consider energy-efficiency & latency at the same time

Related Work (Contd.)

Work by [Ma and Yang, 2007]

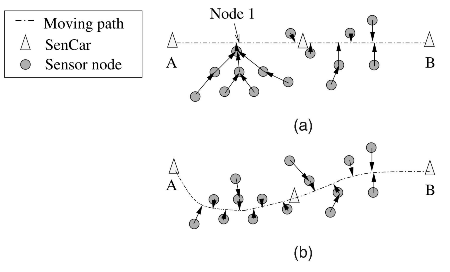


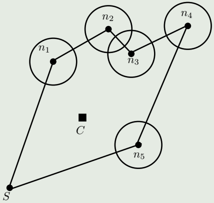
Figure: (a) The straight line path of SenCar, (b) The curved path of SenCar

Limitations:

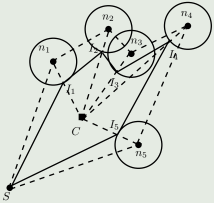
- The path is not a cycle
- Heuristics produce redundant paths
- *Funnelling Problem* still exists

Related Work (Contd.)

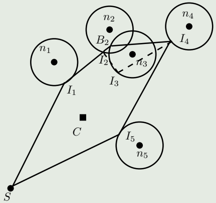
Work by [Yuan and Peng, 2010]



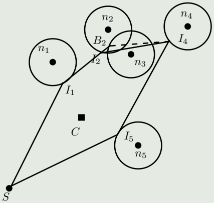
(a) Centroid



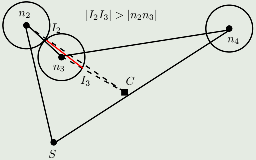
(b) Inner Bends



(c) Concave Bends



(d) Making shortcut



(e) Flaws

Related Work (Contd.)

Work by [Bhadauria et al., 2011]

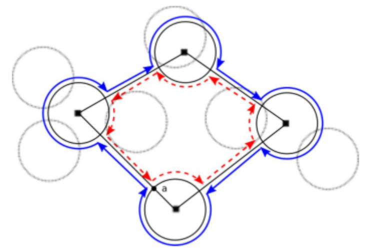


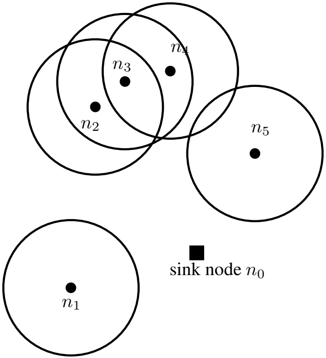
Figure: Approximate TSPN

Limitations:

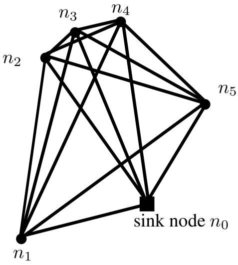
- Traversing boundaries add up to path-length
- Inefficient for sparse network

System Model

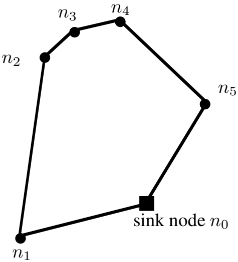
- A WSN with n nodes is represented by a complete graph K_n
- Weights of the edges are Euclidian distances.
- A *TSP-tour* visits all the nodes exactly once such that tour-length is the minimum.



(a) A sample of sensor network



(b) K_6

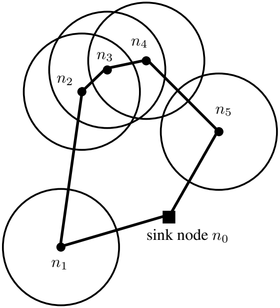


(c) The *TSP-tour* by the MDC

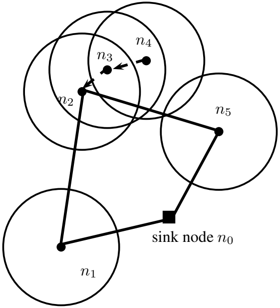
Figure: *TSP-tour* by MDC in sensor network

System Model (Contd.)

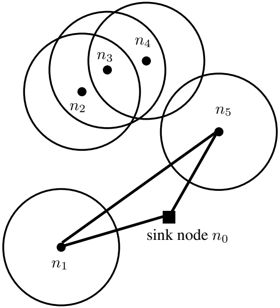
- Complete Tour: Each sensor can send data to the sink
- Incomplete Tour: One or more sensors are missed by the MDC



(a) TSP-tour



(b) A tour with a 2-hop MF-tree



(c) An incomplete tour

Observation

A *TSP-tour* is a complete tour (by definition)

System Model (Contd.)

Energy Model [Anastasi et al., 2009]

$E_{i,j} = k_0 + [(h(n_i, n_j)]^w$

Total Energy of a Sensor

$$E_{n_j} = \sum_{\forall \text{ node } m \in T_{n_j}} (k_0 + 1^w)$$
$$= |T_{n_j}|(1 + k_0)$$

Observation

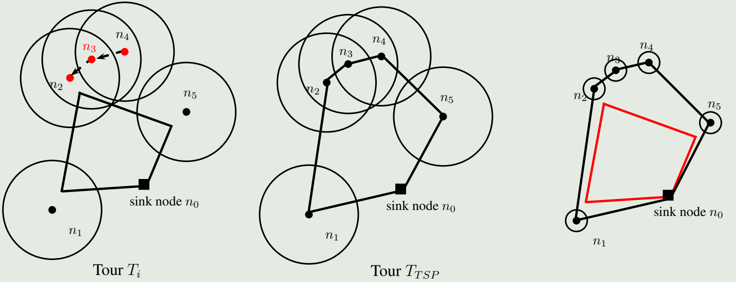
The total energy consumption by a sensor node is directly proportional to the number of packet forwarding.

System Model (Contd.)

m-lifetime: period after which exactly m nodes of a sensor network die due to energy depletion

Lemma: *TSP-tour* has the maximum m-lifetime of all the complete tours by the MDC

Proof: The maximum hop-count of all the MF-trees of tour T_i is either greater than 1 (Case a) or, it is 1 (Case b)



Case (a)

Case (b)

System Model (Contd.)

Advantages of *TSP-tour*

- Complete
- No packet forwarding, thus no path-finding overhead
- Invariant to TXR
- Allows data gathering in sparse or disconnected network
- Ensures the maximum m-lifetime of the network

Challenge of *TSP-tour*

- High tour-time, thus high data delivery latency

System Model (Contd.)

Data Delivery Latency: The time-difference between packet generation (at the sensor) and delivery (to the sink)

Why Shortcut of a Tour

$$t_l(i) = t_T - t_g(i)$$

$$t_{avg} = \frac{\sum_{i=1}^{n} [t_T - t_g(i)]}{n} = t_T - \frac{\sum_{i=1}^{n} t_g(i)}{n}$$

$t_{TSP} = t_h + t_m$, $t_h << t_m$ in a sparse network

$t_m = \frac{|t_{TSP}|}{v_{MDC}}$ Minimizing $|t_{TSP}|$ reduces t_{avg}

Goal

Problems Statement

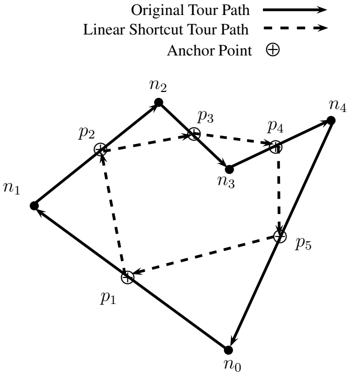
Given a TSP-tour by the MDC, we find a tour T_d that is complete and shorter than the TSP-tour

Strategy

1. The number of nodes visited in the *TSP-tour* can be decreased by making *shortcut* of the *TSP-tour*
2. The length of the edges in the resulting tour can be decreased by taking into consideration the value of TXR

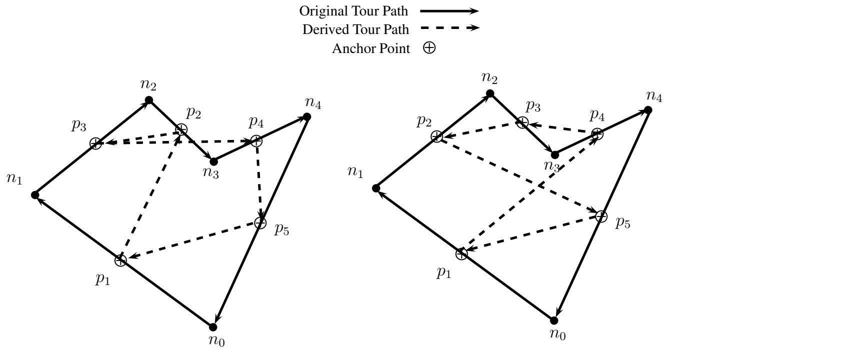
Linear Shortcut

Linear Shortcut of a tour is generated by selecting some points on the tour-edges and joining those points successively in the order of their visits on the given tour. The Selected Points are called Anchor Points.



$< p_1, p_2, p_3, p_4, p_5, p_1 >$ is a Linear Shortcut

Linear Shortcut (Contd.)



(a) Anchor point p_2 and p_3 are chosen in a wrong way

(b) Anchors not connected by edges according to the order of p_i's

Figure: Examples of derived tours which are not Linear Shortcut tours

Strategy of Finding Linear Shortcut

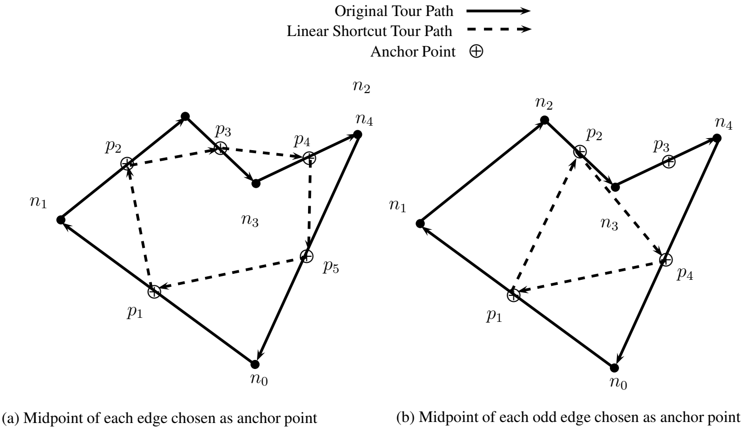
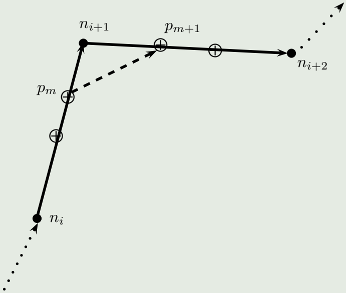


Figure: Example of different strategies for finding Linear Shortcut Tour

Property of Linear Shortcut

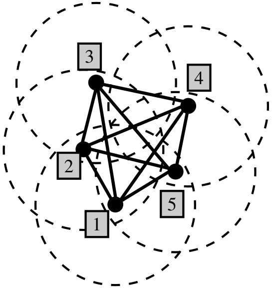
Lemma: The length of a Linear Shortcut Tour is at most that of the given tour.



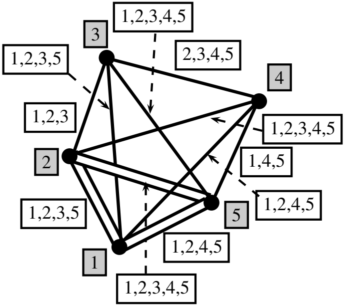
Proof: Using *Triangle Rule*, we get

$$|n_i p_m| + |p_m p_{m+1}| + |p_{m+1} n_{i+2}| \leq |n_i n_{i+1}| + |n_{i+1} n_{i+2}|$$

Label Covering Tour (*LC-tour*) From *TSP-tour*



(a) Complete Graph derived from the Connectivity Graph

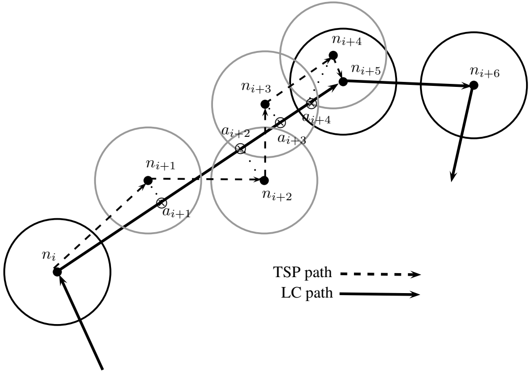


(b) Label Covering Tour (tour-edges makred in double-line)

Figure: Label Covering tour in a network with five nodes

$$L(1) \cup L(2) \cup L(5) = \{1,2,3,5\} \cup \{1,2,3,4,5\} \cup \{1,2,4,5\} = \{1,2,3,4,5\}$$

LC-tour (Contd.)



- Given a *TSP-tour*, the *LC-tour* can be computed in $O(n^3)$ time
- If TSP-tour is not given, finding an optimal *LC-tour* is *NP-hard*
- According to Linear Shortcut strategy, 0 or 2 Anchor Points are chosen on the tour-edge

Tight Label Covering Tour (*TLC-tour*)

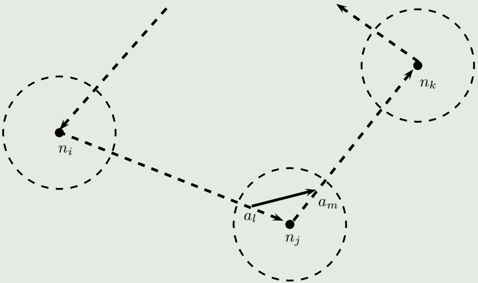
Tight Label Covering Tour (*TLC-tour*): Any tour derived by making a Linear Shortcut of a complete tour

Observation

Is it possible to make *LC-tour* shorter by finding Linear Shortcuts ??

Shortcutting *LC-tour*

Lemma: If TXR of only the visited nodes in the *LC-tour* is zero, any tour derived by making linear shortcut of the LC-tour will not be complete.



Proof:

$E_d = E_{LC} - \{< n_i, n_j >, < n_j, n_k >\} \cup \{< n_i, a_l >, < a_l, a_m >, < a_m, n_k >\}$

Node n_j is missed by the MDC

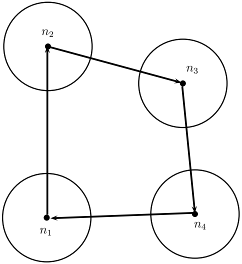
Problem Statement

Shortening of Label Covering Tour

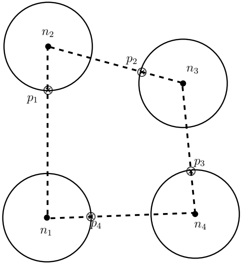
Given a Label Covering Tour T_{LC} for the MDC in a sensor network with non-zero transmission radius TXR, derive a tour via linear shortcutting that is complete.

Case I

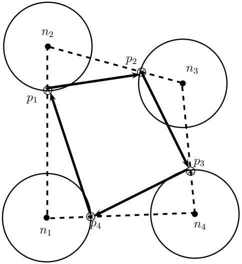
There are no overlapping intermediate nodes.



(a) LC-tour



(b) Anchor Points in LC-tour



(c) TLC-tour generated from the LC-tour

Generating *TLC-tour* (Contd.)

Case II

There are overlapping intermediate nodes.

We derive a shorter tours iteratively by using techniques which involve:

- *Contact Interval* (CI)
- *Critical Contact Interval* (CCI)

Contact Interval

The part of tour-edge within the transmission range of a node.

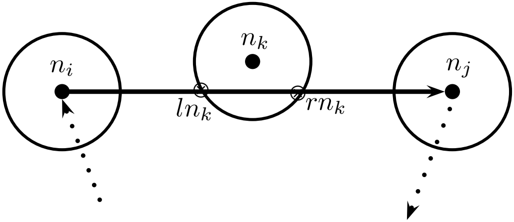


Figure: Contact Interval for node n_k: $< ln_k, rn_k >$

Contact Interval of Node n_k is the line segment between points ln_k and rn_k

Preliminaries (Contd.)

Different Contact Intervals (CI's):

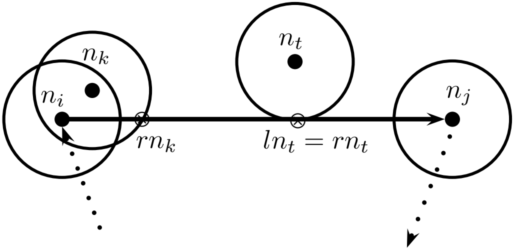
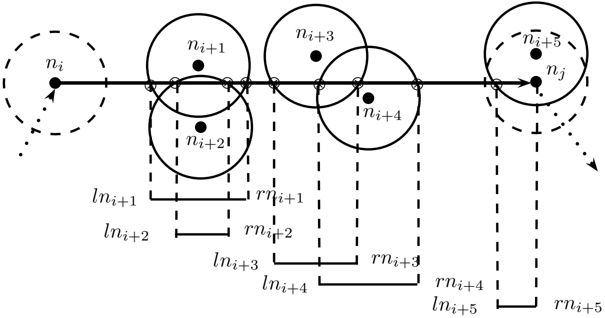


Figure: Different *CI*'s on a tour-edge

- $ln_t = rn_t$ (Tangent point of a circle)
- Either of rn_k or ln_k not on the tour-edge

Representation of the CI's



Node	ln(x,y)	rn(x,y)
n_i	N/A	N/A
n_{i+1}	$(316, 120)$	$(398, 120)$
n_{i+2}	$(337, 120)$	$(382, 120)$
n_{i+3}	$(422, 120)$	$(494, 120)$
n_{i+4}	$(461, 120)$	$(554, 120)$
n_{i+5}	$(613, 120)$	$(647, 120)$
n_j	N/A	N/A

Critical Contact Interval

- The minimum interval contributes to finding the shortcut path

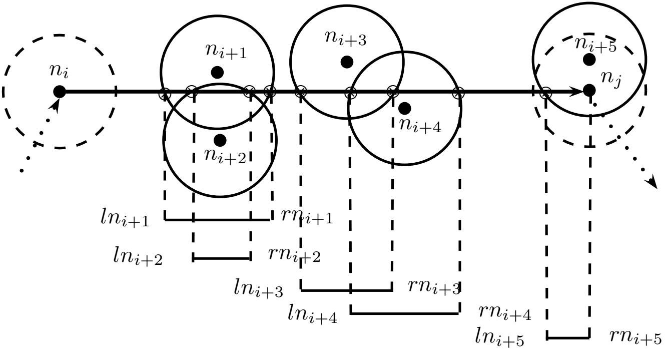


Figure: Finding the Critical Contact Interval Covering all CI's

MDC must cover at least the line segment called *Critical Contact Interval* or CCI

- The minimum interval contributes to finding the shortcut path

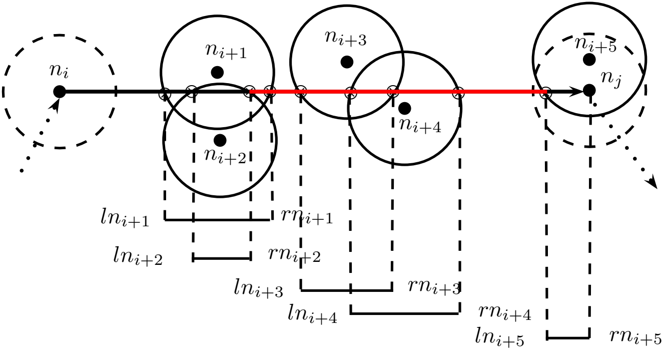


Figure: Finding the Critical Contact Interval Covering all CI's

MDC must cover at least the line segment called *Critical Contact Interval* or CCI

Finding the CCI

Problem

Given the list of contact intervals (in terms of l and r points), what is the minimum interval that covers all the contact intervals of a particular edge?

Finding the CCI

Problem

Given the list of contact intervals (in terms of l and r points), what is the minimum interval that covers all the contact intervals of a particular edge?

We need to sort contact intervals first

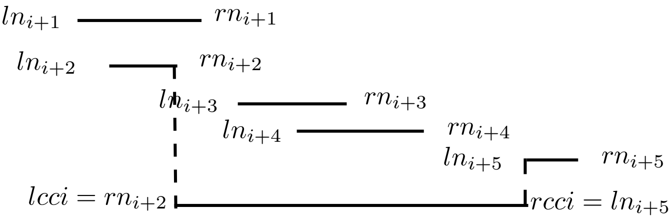


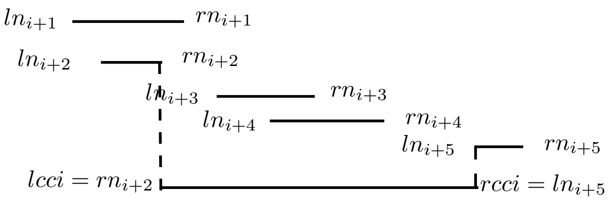
Figure: CCI for a given list of intervals

Finding the CCI (Contd.)

- $lcci \leftarrow rn$ Point closest to the starting endpoint of the edge
- $rcci \leftarrow ln$ Point farthest from the starting endpoint of the edge

The intervals have been already sorted based on l point's distance from the edge's starting point

Time required for finding CCI: $O(1)$

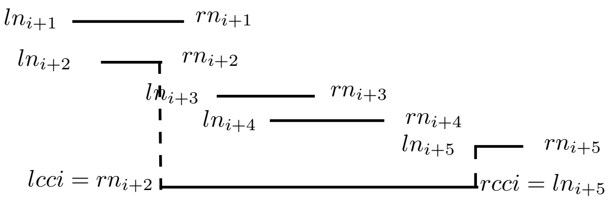


Finding the CCI (Contd.)

- $lcci \leftarrow rn$ Point closest to the starting endpoint of the edge
- $rcci \leftarrow ln$ Point farthest from the starting endpoint of the edge

The intervals have been already sorted based on l point's distance from the edge's starting point

Time required for finding CCI: $O(1)$



- CCI has left endpoint $lcci = rn_{i+2}$ and right end point $rcci = ln_{i+5}$
- If $lcci = rn_{i+1}$ is chosen, node n_{i+2} will be missed by the MDC

Finding the CCI (Contd.)

Node	ln(x,y)	rn(x,y)
n_i	N/A	N/A
n_{i+1}	$(316, 120)$	$(398, 120)$
n_{i+2}	$(337, 120)$	$(382, 120)$
n_{i+3}	$(422, 120)$	$(494, 120)$
n_{i+4}	$(461, 120)$	$(554, 120)$
n_{i+5}	$(613, 120)$	$(647, 120)$
n_j	N/A	N/A

- Running time for each edge $O(n + n \log n) = O(n \log n)$
- The number of edges in LC-tour $O(n)$
- Running time for the generating CCI for total graph: $O(n^2 \log n)$

Generating the Sorted CI's

Input: An edge $e \in E$ that connects node n_i and node n_j in LC-tour and its list of intermediate nodes I_e

```
CI_e ← {}
for all node n_k ∈ I_e do
    Find intersections (ln_k, rn_k) of edge e and circle of radius TXR centered at n_k
    if ln_k is outside of line segment of edge e then
        ln_k ← n_i
    end if
    if rn_k is outside of line segment of edge e then
        rn_k ← n_j
    end if
    CI_e ← CI_e ∪ {(n_k, ln_k, rn_k)}
end for
sort CI_e using ln_k as key
```

Output: CI_e is the sorted contact interval

Finding Shortcut Bypassing the Visited Nodes

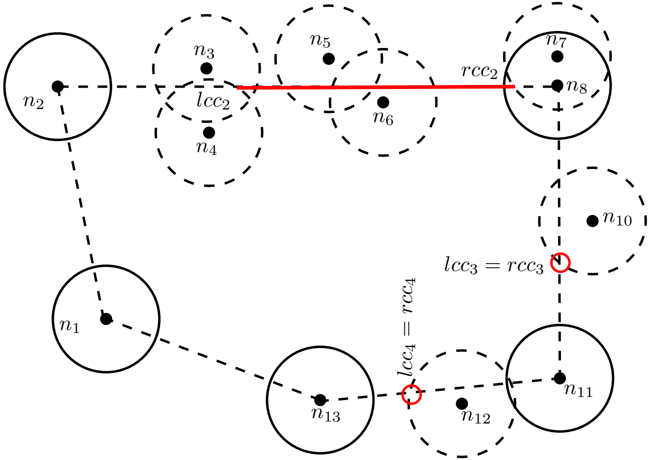


Figure: Connecting $rcci$ and $lcci$ Points

Finding Shortcut Bypassing the Visited Nodes

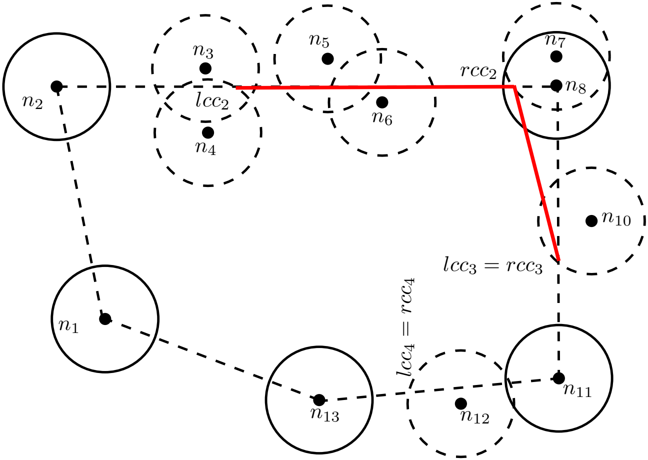


Figure: Connecting $rcci$ and $lcci$ Points

Finding Shortcut Bypassing the Visited Nodes

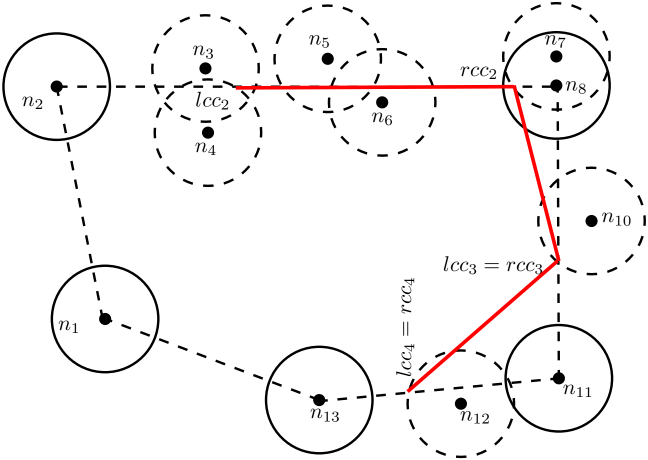


Figure: Connecting $rcci$ and $lcci$ Points

Finding Shortcut Bypassing the Visited Nodes

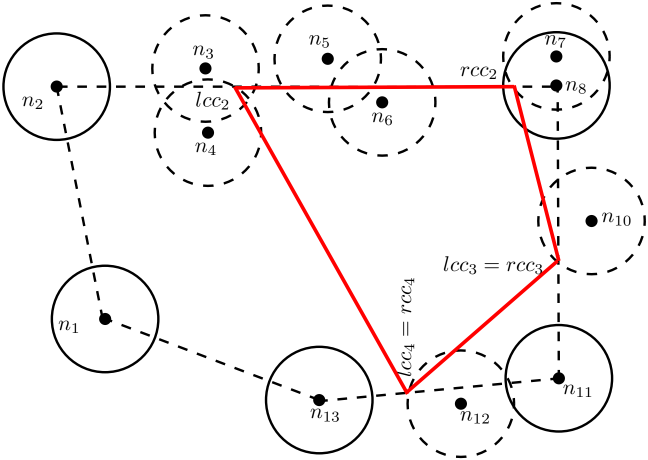
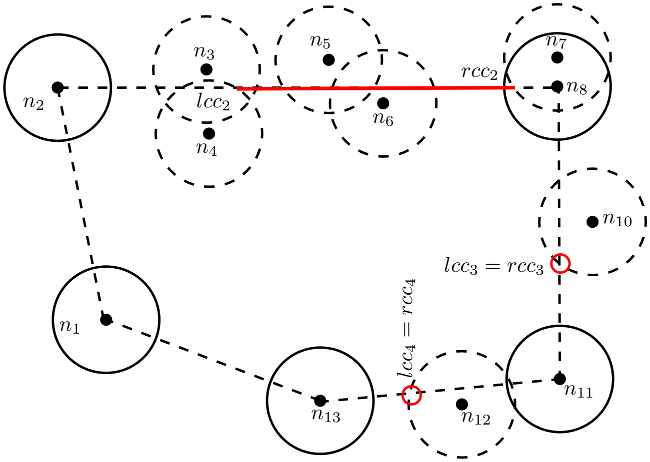


Figure: Connecting $rcci$ and $lcci$ Points

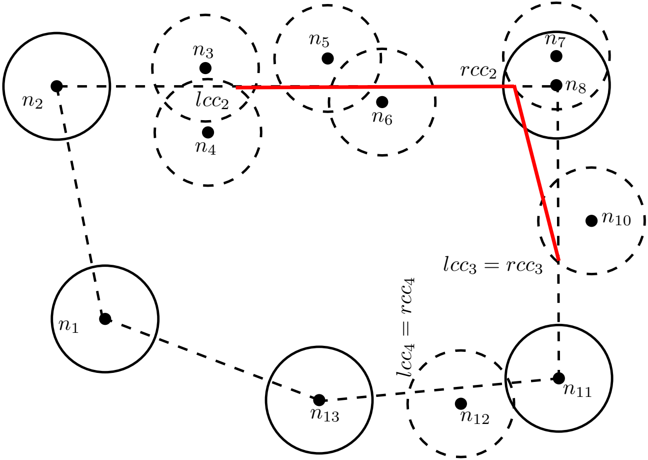
Deriving a Complete Tour



Case 1: Adjacent edges have non-empty CCI

Connect the r and l Points, call this line $r - l$ *Line*

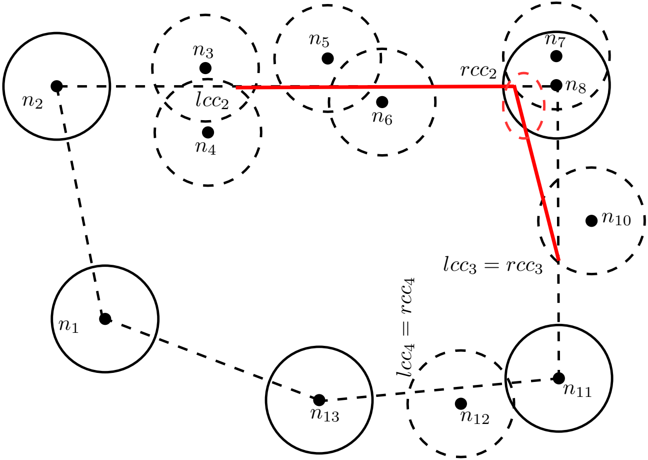
Deriving a Complete Tour



Case 1: Adjacent edges have non-empty CCI

Connect the r and l Points, call this line $r - l$ *Line*

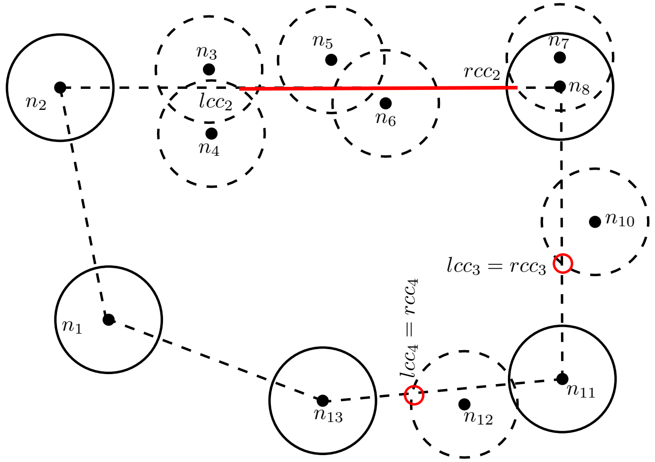
Deriving a Complete Tour



Case 1(a): The $r - l$ *Line* intersects the uncovered circle

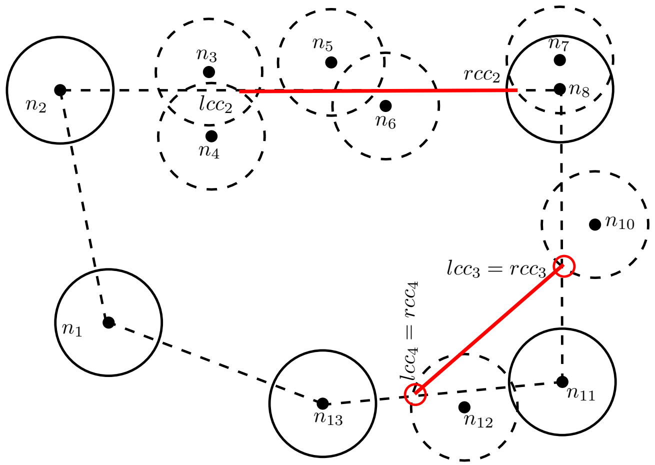
Add the $r - l$ *Line* segment in the edge-set

Deriving a Complete Tour



Case 1(b): The $r - l$ *Line* does not intersect the uncovered circle
Draw tangent to the circle parallel to the $r - l$ *Line*

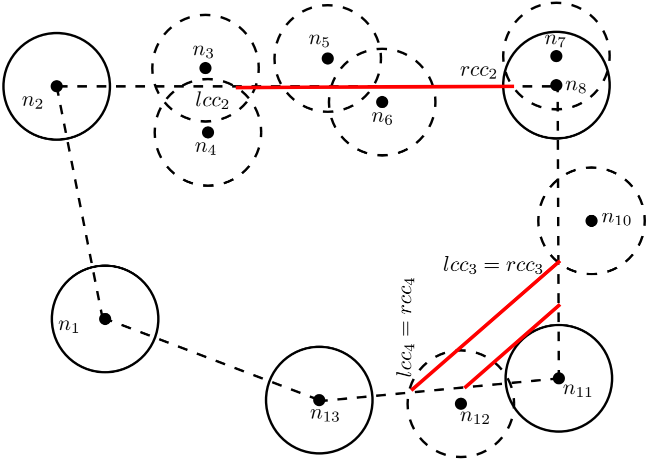
Deriving a Complete Tour



Case 1(b): The $r - l$ *Line* does not intersect the uncovered circle

Draw tangent to the circle parallel to the $r - l$ *Line*

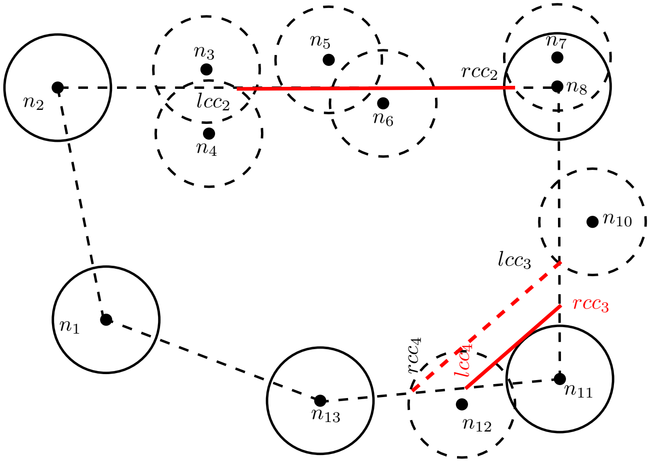
Deriving a Complete Tour



Case 1(b): The $r - l$ *Line* does not intersect the uncovered circle

Draw tangent to the circle parallel to the $r - l$ *Line*

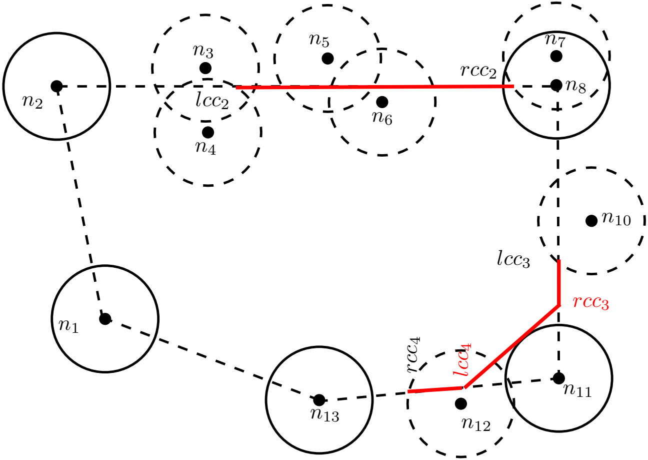
Deriving a Complete Tour



Case 1(b): The $r - l$ *Line* does not intersect the uncovered circle

Draw tangent to the circle parallel to the $r - l$ *Line*

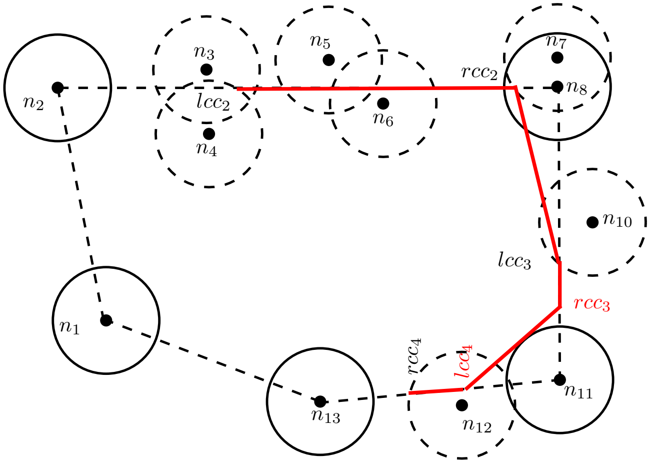
Deriving a Complete Tour



Case 1(b): The $r - l$ *Line* does not intersect the uncovered circle

Draw tangent to the circle parallel to the $r - l$ *Line*

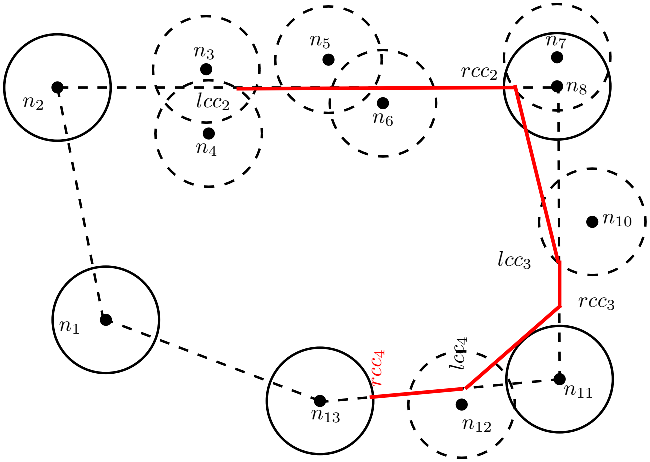
Deriving a Complete Tour



Case 2(a): The CCI of *Incoming Edge* does not reach the circle

Set $rcci$ of CCI as the intersection point with the circle

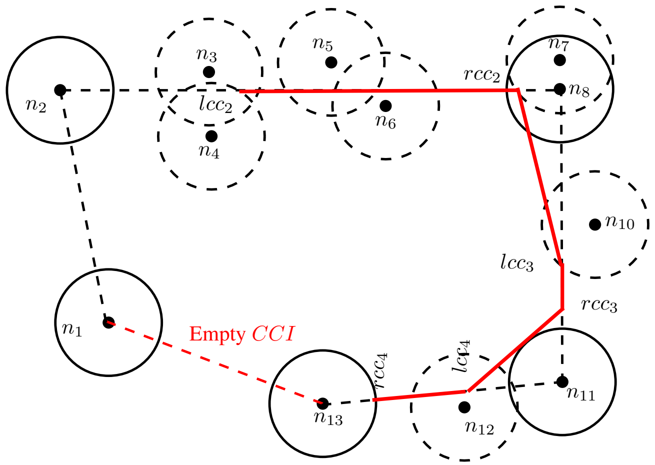
Deriving a Complete Tour



Case 2(a): The CCI of *Incoming Edge* does not reach the circle

Set $rcci$ of CCI as the intersection point with the circle

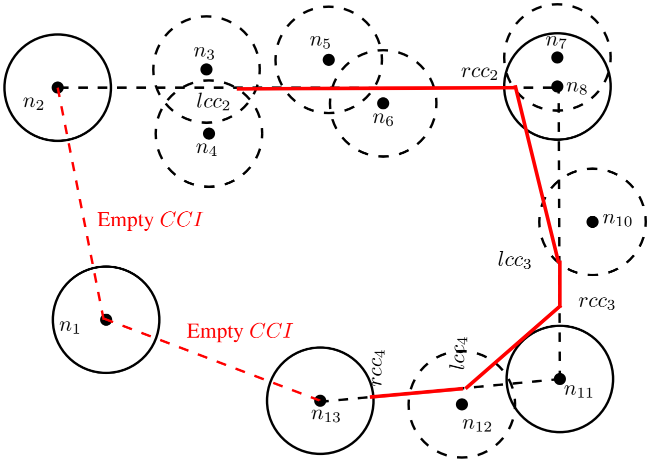
Deriving a Complete Tour



Case 2(b): The *Incoming Edge* has Null CCI

Set $lcci$ and $rcci$ of *Incoming Edge* as the intersection point with the circle

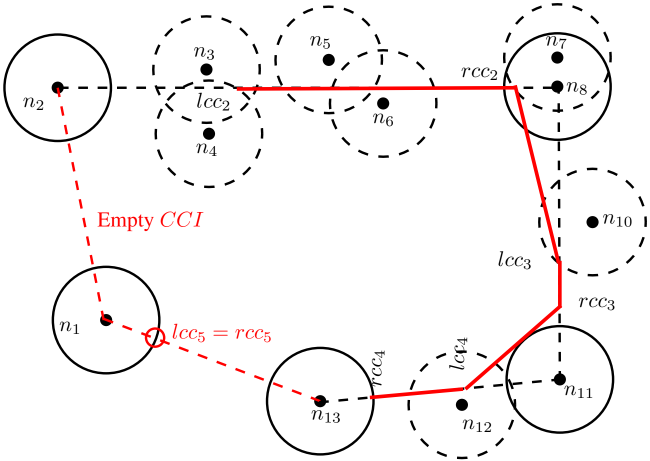
Deriving a Complete Tour



Case 2(b): The *Incoming Edge* has Null CCI

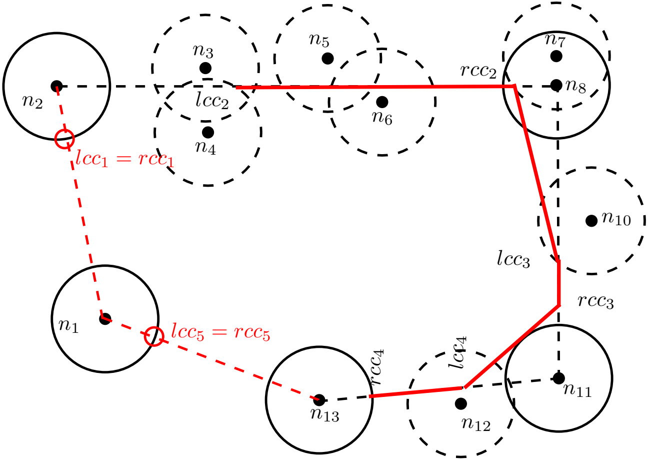
Set $lcci$ and $rcci$ of *Incoming Edge* as the intersection point with the circle

Deriving a Complete Tour



Case 2(b): The *Incoming Edge* has Null CCI
Set $lcci$ and $rcci$ of *Incoming Edge* as the intersection point with the circle

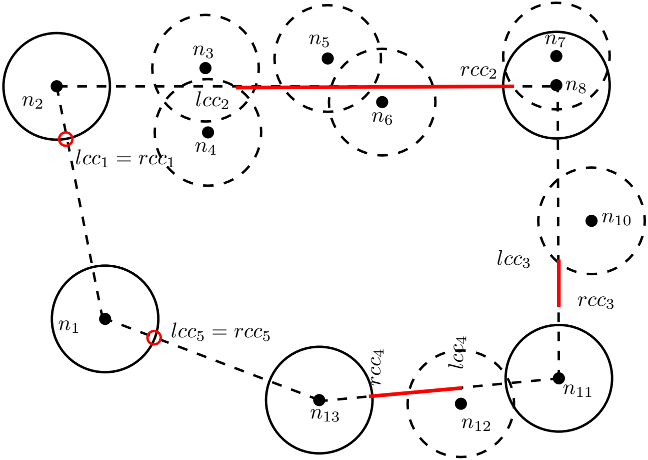
Deriving a Complete Tour



Case 2(b): The *Incoming Edge* has Null CCI

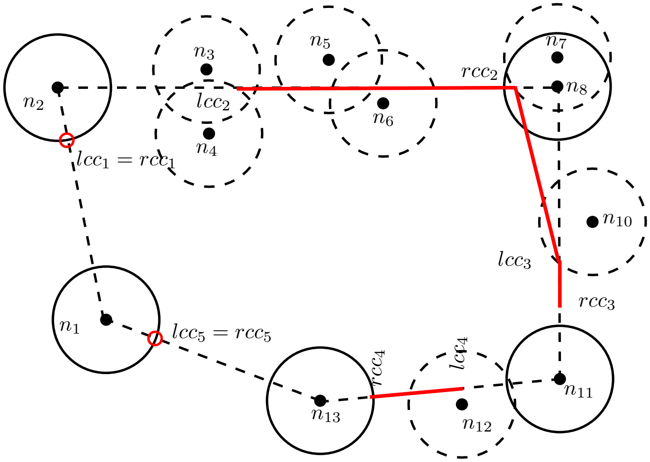
Set $lcci$ and $rcci$ of *Incoming Edge* as the intersection point with the circle

Deriving a Complete Tour



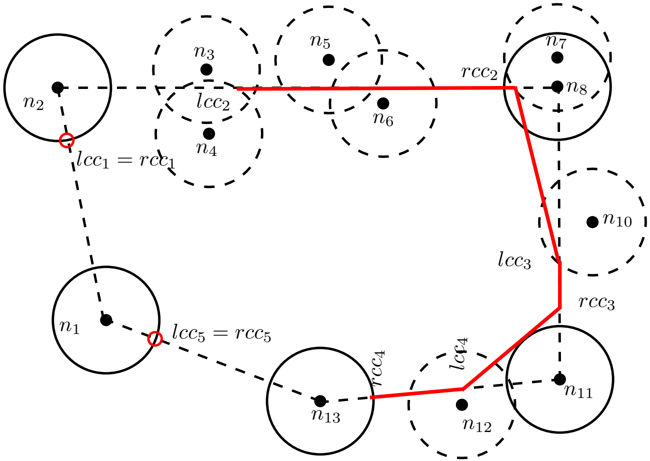
1. Connect One Edge's rcc Point to the next Edge's lcc Point (New edge)
2. Include the updated CCI in the edge-set (Shortented edge)

Deriving a Complete Tour



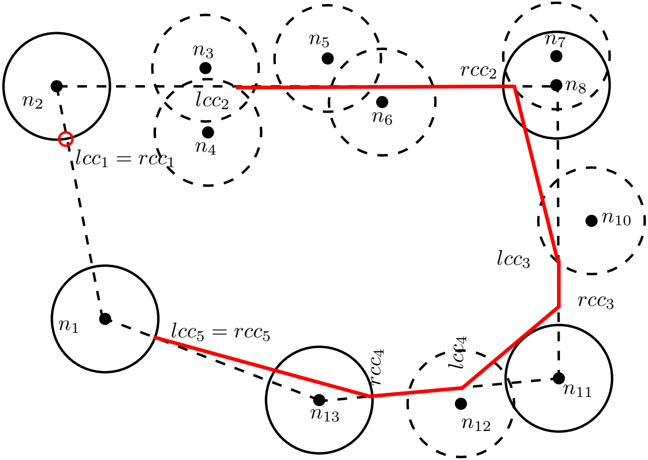
1. Connect One Edge's rcc Point to the next Edge's lcc Point (New edge)
2. Include the updated CCI in the edge-set (Shortented edge)

Deriving a Complete Tour



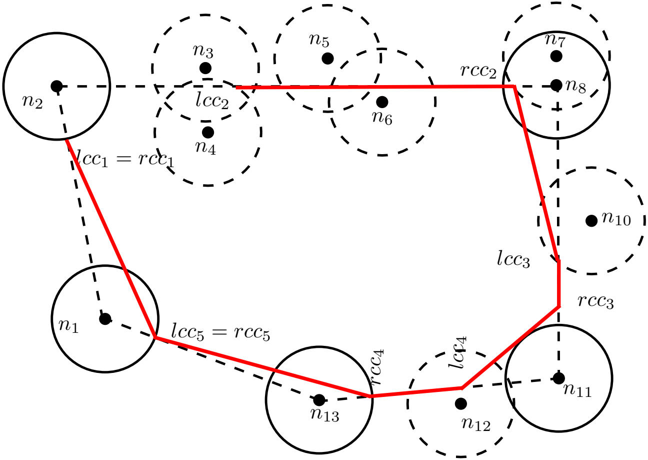
1. Connect One Edge's rcc Point to the next Edge's lcc Point (New edge)
2. Include the updated CCI in the edge-set (Shortented edge)

Deriving a Complete Tour



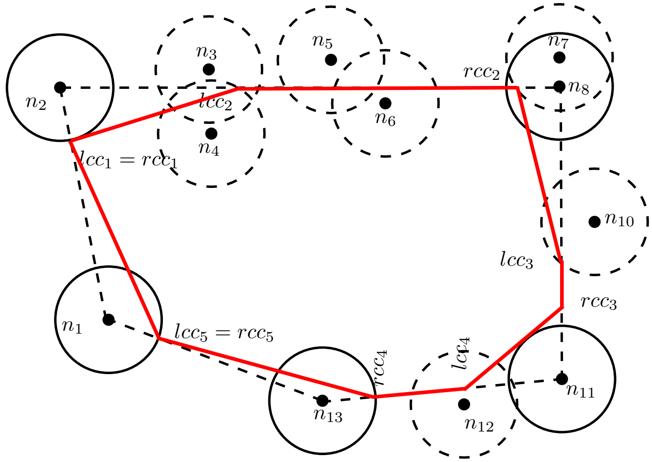
1. Connect One Edge's rcc Point to the next Edge's lcc Point (New edge)
2. Include the updated CCI in the edge-set (Shortented edge)

Deriving a Complete Tour



1. Connect One Edge's rcc Point to the next Edge's lcc Point (New edge)
2. Include the updated CCI in the edge-set (Shortented edge)

Deriving a Complete Tour



1. Connect One Edge's rcc Point to the next Edge's lcc Point (New edge)
2. Include the updated CCI in the edge-set (Shortented edge)

Algorithm for Generating *TLC-tour*

Input: A tour t with CCI's associated with each edge

for all node n_i visited in the tour t **do**
  if both the edges e_s (incoming) and e_t(outgoing) incident with n_i have CCI **then**
    if line l_{st} connecting r point and l point of the CCI's of edges e_s and e_t respectively does not intersect circle centered at n_i **then**
      l_{n_i} is the line parallel to l_{st} and tangent to the circle centered at n_i
      update r point of edge e_s as the intersection of l_{n_i} and e_s
      update l point of edge e_t as the intersection of l_{n_i} and e_t
    end if
  else
    p_{n_i} is the intersection of the incoming edge e_s and circle centered at n_i
    if p_{n_i} is closer to n_i than r point of the CCI of incoming edge e_s **OR** CCI for incoming edge e_s does not exist **then**
      update r point of edge e_s as p_{n_i}
      if CCI for incoming edge e_s does not exist **then**
        update l point of edge e_s as its r point
      end if
    end if
  end if
end for
$t_{TLC} \leftarrow \{\}$
Join r point of an edge to the l point of the next edge successively and add it to tour t_{TLC}

Output: t_{TLC} is a TLC tour =0

Algorithm for Generating *TLC-tour*

Input: A tour t with CCI's associated with each edge

for all node n_i visited in the tour t **do**
  if both the edges e_s (incoming) and e_t(outgoing) incident with n_i have CCI **then**
    if line l_{st} connecting r point and l point of the CCI's of edges e_s and e_t respectively does not intersect circle centered at n_i **then**
      l_{n_i} is the line parallel to l_{st} and tangent to the circle centered at n_i
      update r point of edge e_s as the intersection of l_{n_i} and e_s
      update l point of edge e_t as the intersection of l_{n_i} and e_t
    end if
  else
    p_{n_i} is the intersection of the incoming edge e_s and circle centered at n_i
    if p_{n_i} is closer to n_i than r point of the CCI of incoming edge e_s **OR** CCI for incoming edge e_s does not exist **then**
      update r point of edge e_s as p_{n_i}
      if CCI for incoming edge e_s does not exist **then**
        update l point of edge e_s as its r point
      end if
    end if
  end if
end for
$t_{TLC} \leftarrow \{\}$
Join r point of an edge to the l point of the next edge successively and add it to tour t_{TLC}

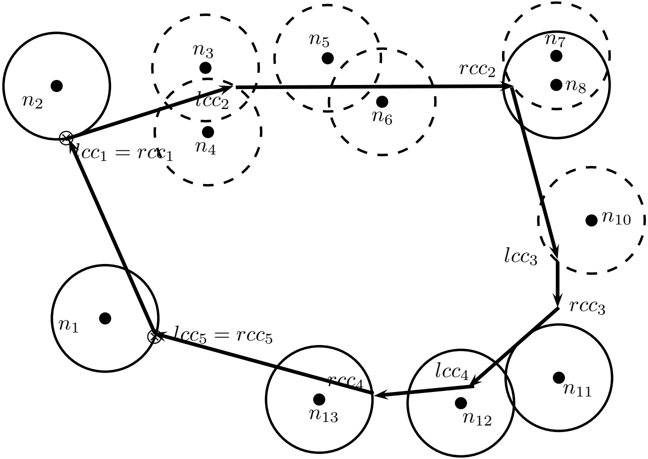
Output: t_{TLC} is a TLC tour

Iterative Shortening

- Iterative shortening is done using Linear Shortcut
- 0, 1 or 2 points are selected from tour-edges according to the following steps:

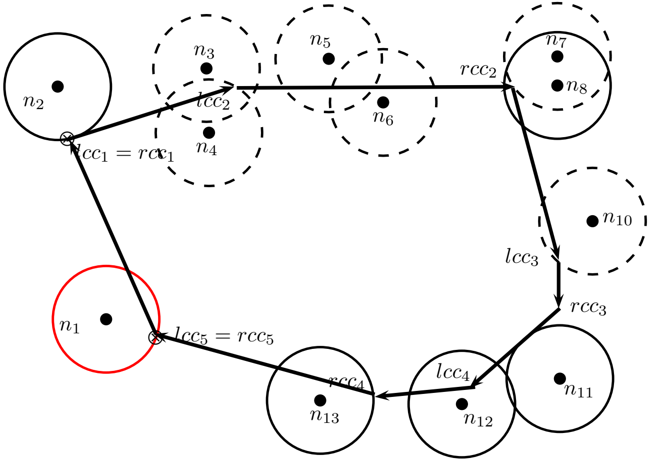
Step 1: Connect $rcci$ of a tour-edge and $lcci$ of the *next* tour-edge with non-Null CCI
Step 2: *Re-associate* the circles with the new set of edges
Step 3: *Re-compute* the CCI for each edge

Re-associating Process



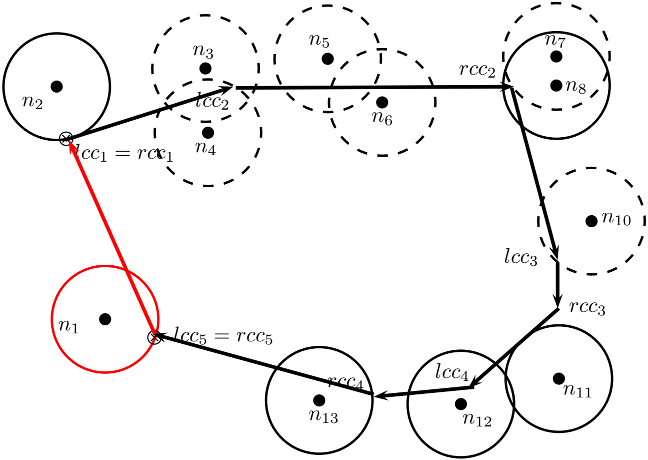
Policy 1: Associate with the edge for which CI is longer

Re-associating Process



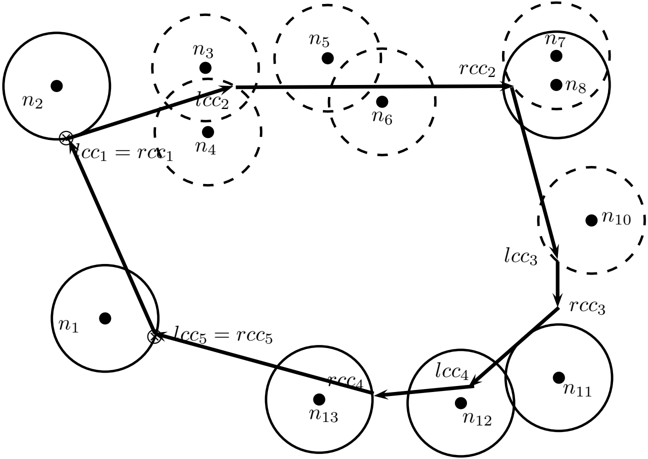
Policy 1: Associate with the edge for which CI is longer

Re-associating Process



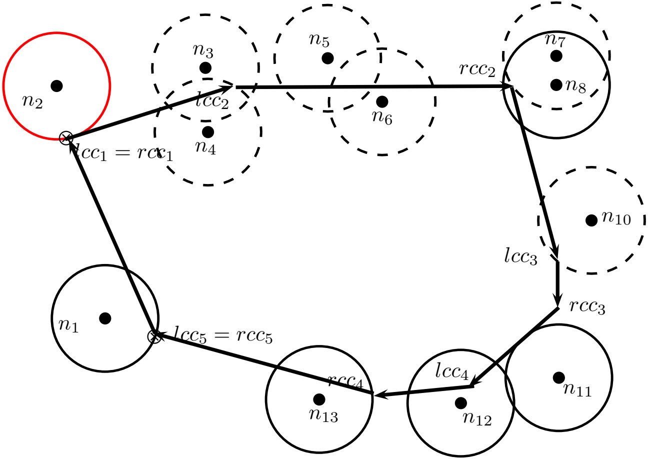
Policy 1: Associate with the edge for which CI is longer

Re-associating Process



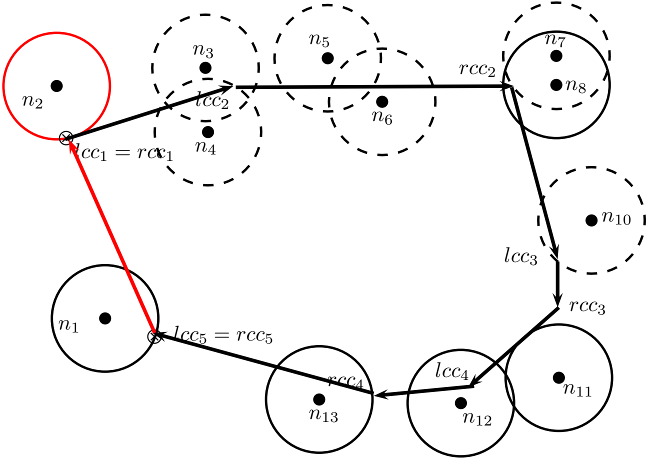
Policy 2: If CI's are equal, associate with the *incoming edge*

Re-associating Process



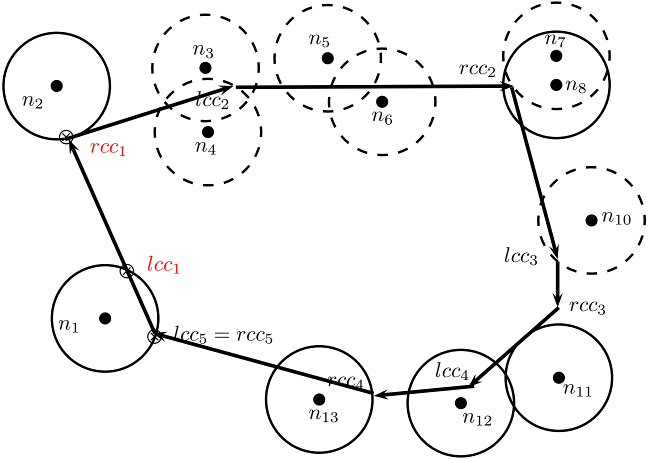
Policy 2: If CI's are equal, associate with the *incoming edge*

Re-associating Process



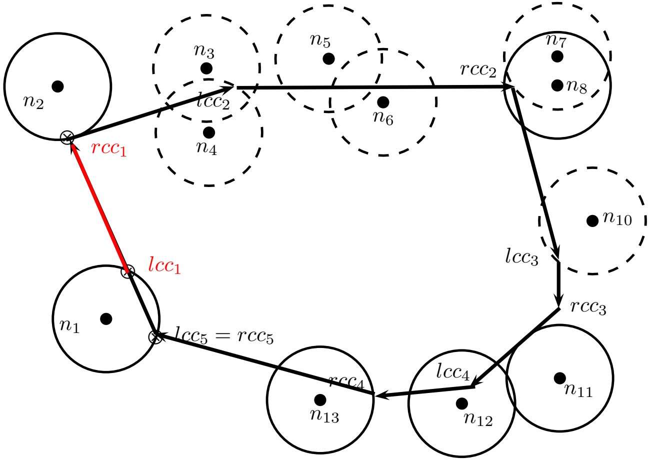
Policy 2: If CI's are equal, associate with the *incoming edge*

Re-computing CCI



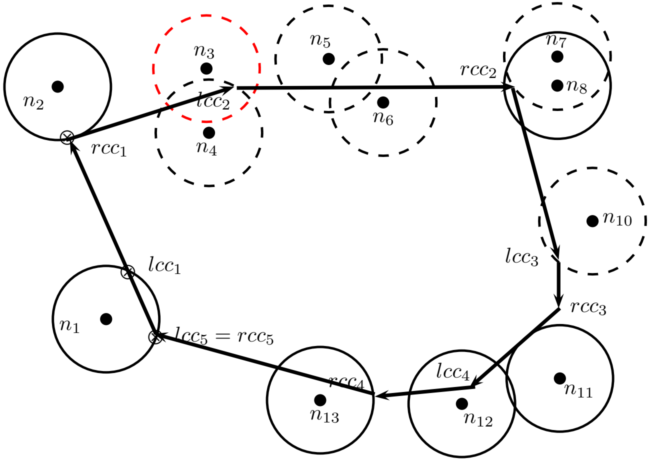
Update the lcc and rcc Points

Re-computing CCI

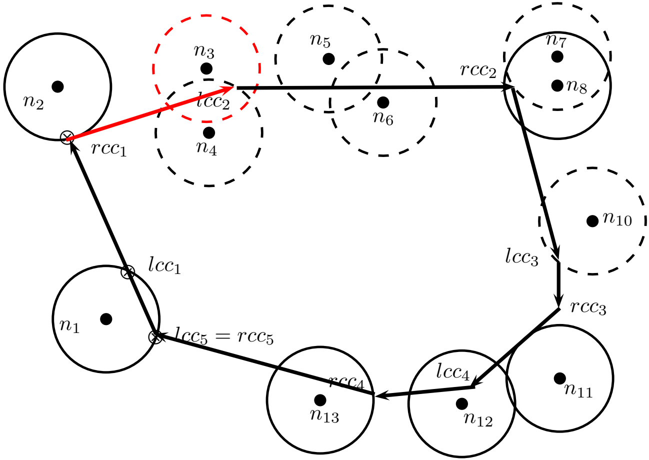


Update the lcc and rcc Points

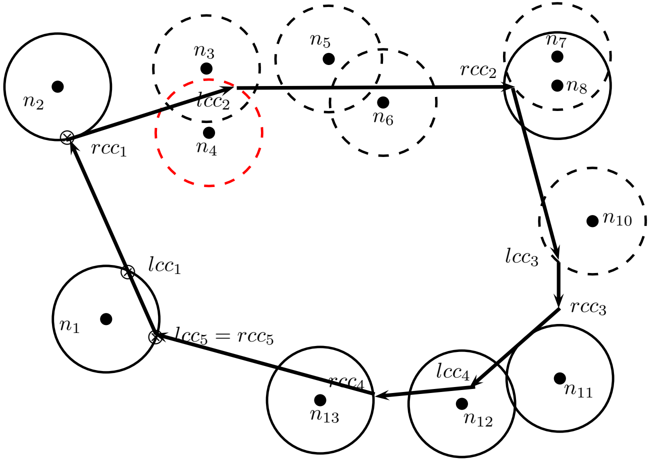
Re-associating Process (Contd.)



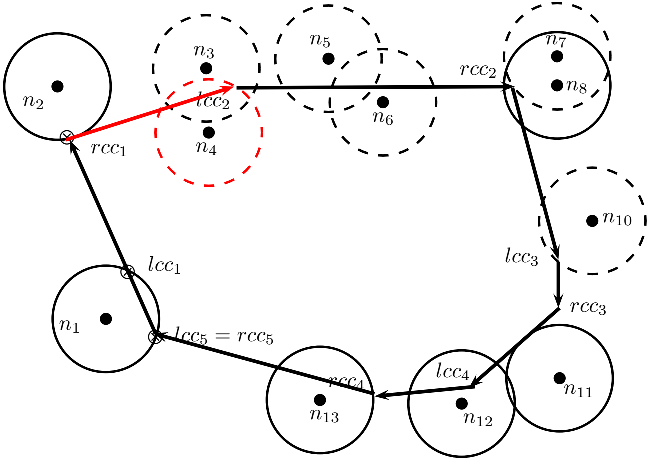
Re-associating Process (Contd.)



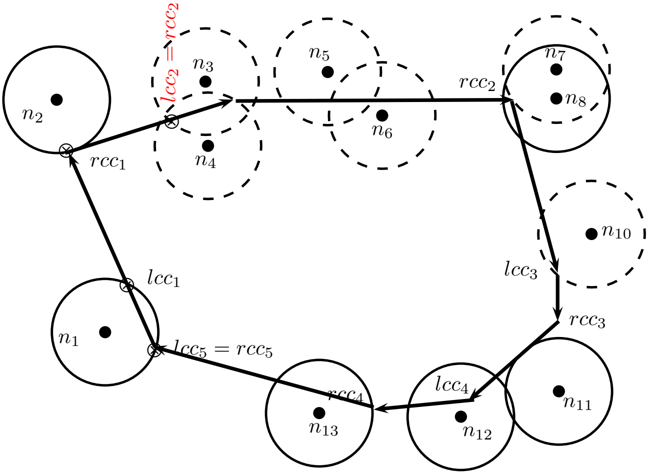
Re-associating Process (Contd.)



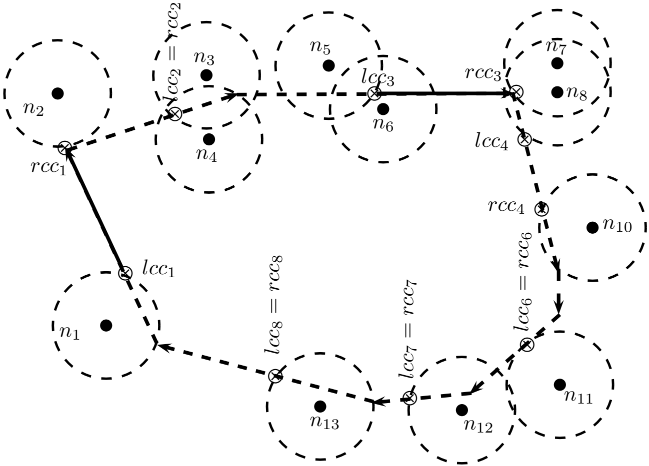
Re-associating Process (Contd.)



Re-associating Process (Contd.)



Update of CCI's after Iteration 1



Summary of Computations in Successive Iterations

Iteration 1

1. Compute CI for each circle
2. Compute CCI for each edge
3. Connect r and l Points
4. *Re-associate* circles with edges
5. Recompute CI for each circle for the new edge-set

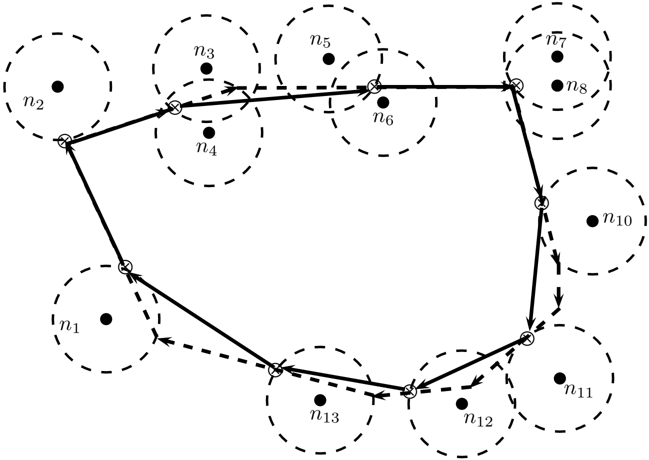
Iteration 2

1. Connect r and l Points
2. *Re-associate* circles with edges
3. Recompute CI for each circle for the new edge-set

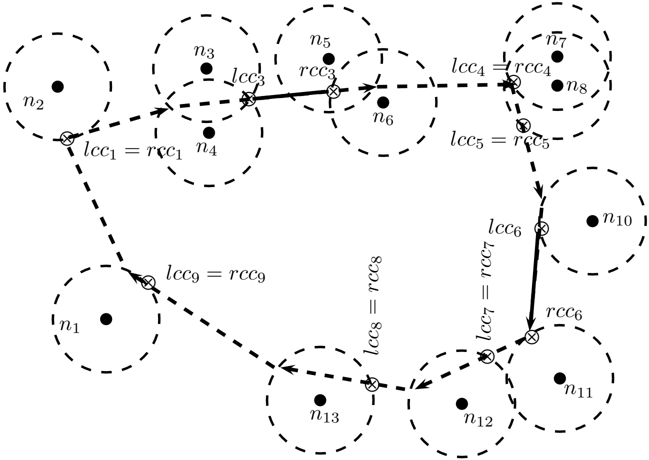
Iteration 3

1. Connect r and l Points . . .

Iteration 2

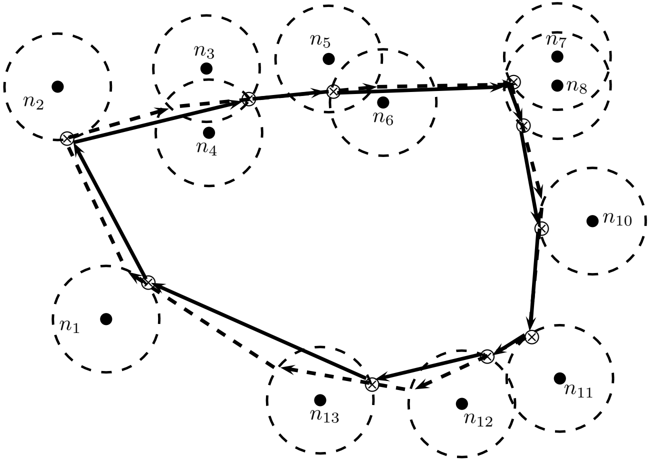


Iteration 2

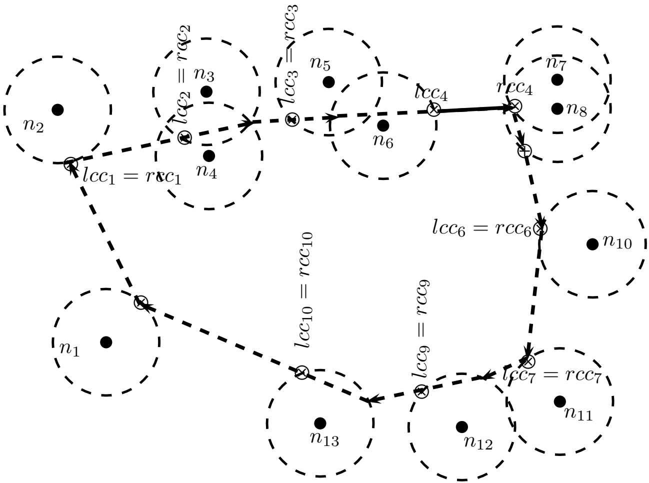


n_2
n_3
n_4
n_5
n_6
n_7
n_8
lcc_3
rcc_3
$lcc_4 = rcc_4$
$lcc_1 = rcc_1$
$lcc_5 = rcc_5$
lcc_6
n_{10}
$lcc_9 = rcc_9$
$lcc_8 = rcc_8$
$lcc_7 = rcc_7$
rcc_6
n_1
n_{11}
n_{13}
n_{12}

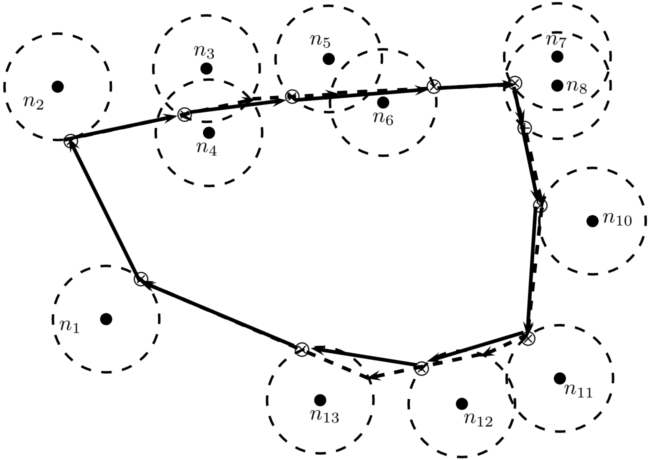
Iteration 3



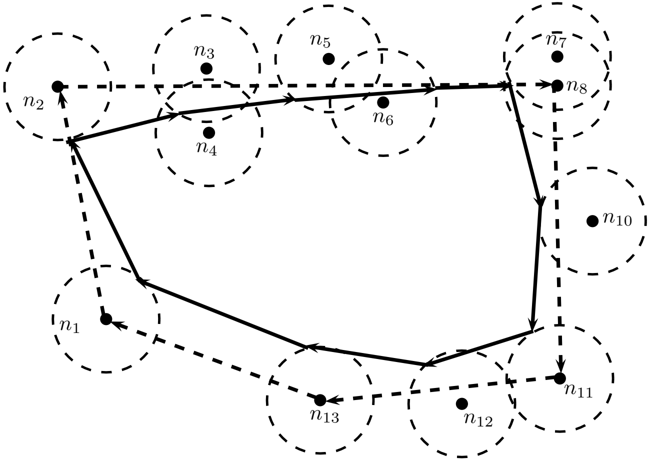
Iteration 3


n_2
$lcc_1 = rcc_1$
$lcc_2 = rcc_2$
n_3
n_4
$lcc_3 = rcc_3$
n_5
n_6
lcc_4
rcc_4
n_7
n_8
$lcc_6 = rcc_6$
n_{10}
n_1
$lcc_{10} = rcc_{10}$
$lcc_9 = rcc_9$
$lcc_7 = rcc_7$
n_{11}
n_{13}
n_{12}

Iteration 4



Iteration 4



Termination of Iterations

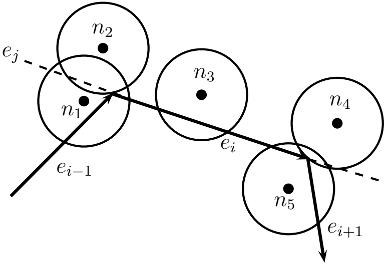
Path Gain g_i in Iteration i:

$$g_i(t_{TLC}) = \frac{|t_{TLC}|_{i-1} - |t_{TLC}|_i}{|t_{TLC}|_i}$$

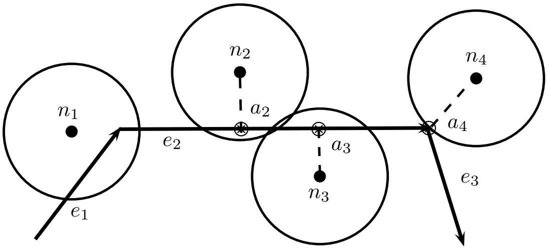
The iteration is stopped after *path gain* is below a threshold i.e. 5%, 1% etc.

Speed Hack

- A node-list for an edge contains all the associated nodes. Example $L(e_1) = \{n_1, n_2\}$
- An attribute of each node for the current associated edge. Example $A(n_1) = e_1$
- *Re-association* for a circle is done in $O(1)$ time
- Only circles which have CI's coincident with the endpoints of the associated edge are checked for *Re-association*. Example, Nodes n_1, n_2, n_4 and n_5



Anchor Points



- MDC halts at *Anchor Points* and collect packets form nearby nodes
- *Anchor Points* are calculated after final iteration

Sample Input and Output

Node	x-Coord.	y-Coord.
n_1	100.4	201.9
n_2	30	10
n_3	95	2
$\dots$	$\dots$	$\dots$
n_{100}	301	202

(a) Input

anchor-x	anchor-y	Node-list
302	308	starting point(sink)
290	205	n_{60}
202	192	n_{75}, n_{80}
$\dots$	$\dots$	$\dots$
302	308	starting point(sink)

(b) Output

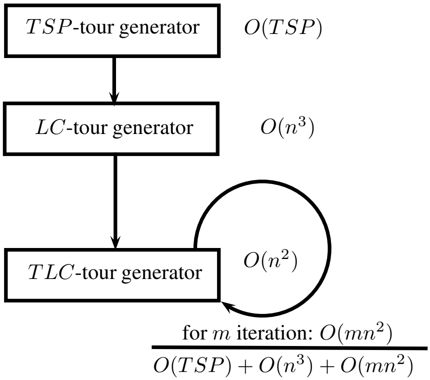
Algorithm for Generating *TLC-tour*

Input: Set of all nodes V and set of all edges E_{i-1} of TLC-tour of iteration $(i-1)$

```
E_i ← {}
for all edge e ∈ E_i with non-null CCI do
    add CCI of edge e to E_i
    connect r point of edge e to the l point of next edge with CCI
    and add it to E_i
end for
Re-associate nodes for the set of edge E_i
for all edge e ∈ E_i do
    Update CCI
end for
```

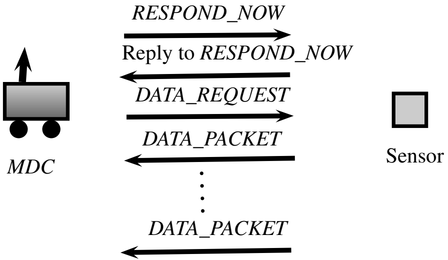
Output: Set of all nodes V and set of all edges E_i of TLC-tour of iteration i

Computational Complexity



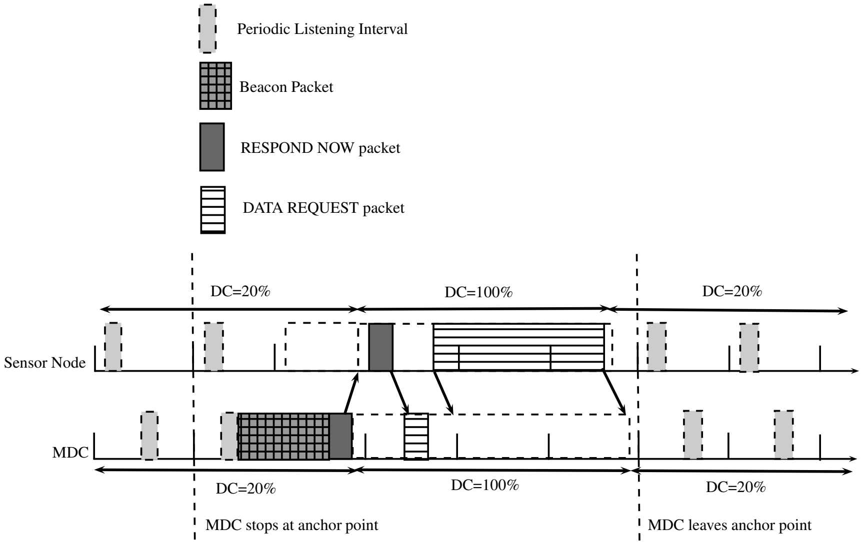
- Computation of *TSP-tour* dominates the running time
- If approximation algorithm for finding *TSP-tour* runs in $O(n^2)$ time, overall complexity is: $O(n^3 + mn^2)$
- If $O(m) = O(n)$, overall complexity is: $O(n^3)$

Energy Efficient MAC Layer



- Conventional MAC's are not applicable
- MDC wakes up the sensor node at the anchor point by sending *RESPOND_NOW* packet
- Sensor node replies to it with the number of packets it want to deposit
- MDC replies with *DATA_REQUEST* packet
- Sensor node sends the agreed number of data packets

Adaptive Duty Cycle of MAC Layer



Experimental Results

Test Bed Design

- Simulator Name: *Castalia 3.2* framework of *OMNET++ 4.4.2* WSN-simulator
- Random Traffic: Using Mersenne Twister type RNG
- Interval Between Packet Generation: Max: 30s and Min: 15s
- Speed of MDC: 1 m/s
- Radio Type: *CC2420*
- Data Rate: 250 kbps
- Packet Buffer Size: 120 packets

Experimental Results (Contd.)

Experimental Setup

- Nodes are randomly distributed
- MDC starts from the sink, collects packets from the nodes and deposits to the sink
- The total time for each run is 7200s
- PHY Layer phenomena like signal fading, interference etc. are present
- TXR is varied (from $2m$ to $32m$)

Experimental Results (Contd.)

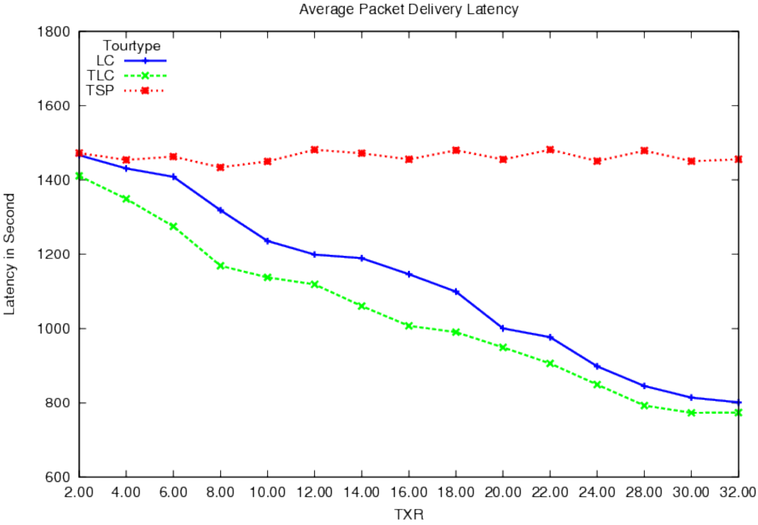


Figure: Impact on the average Packet Delivery Latency

Experimental Results (Contd.)

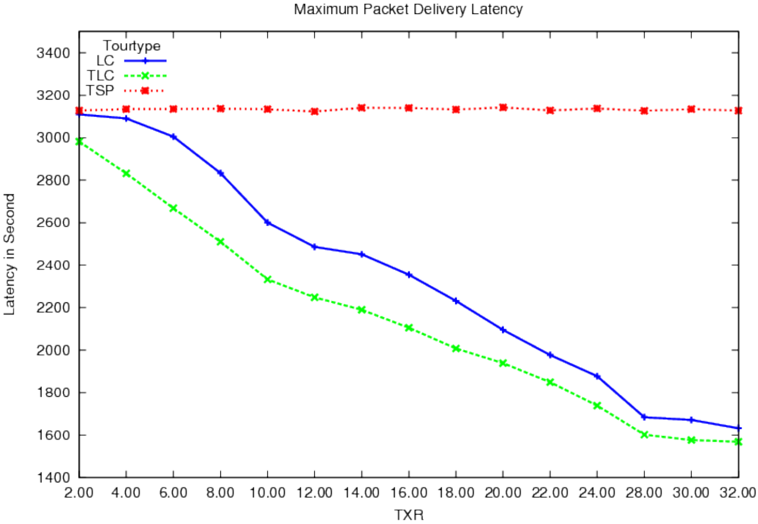


Figure: Impact on the maximum Packet Delivery Latency

Experimental Results (Contd.)

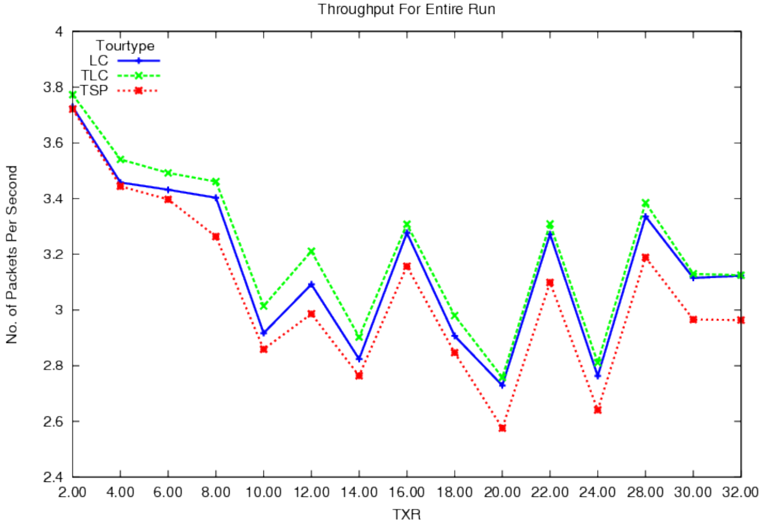


Figure: Impact on Packet Delivery Rate *PDR*

Experimental Results (Contd.)

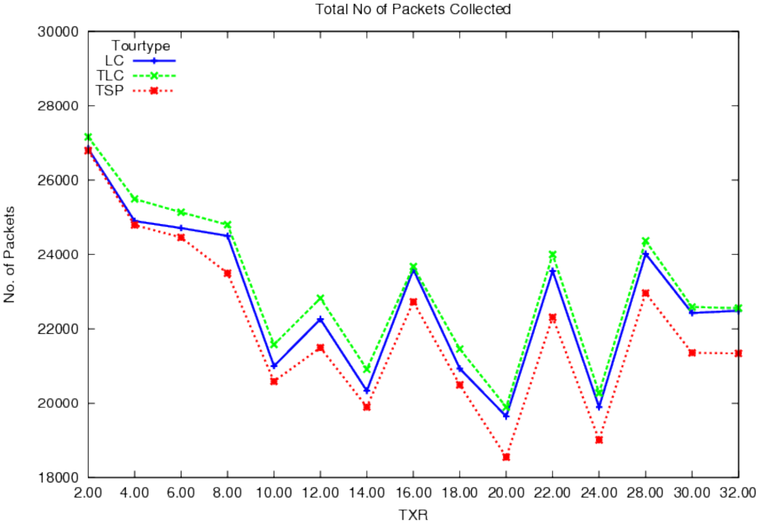


Figure: The total number of packets collected by the MDC

Experimental Results (Contd.)

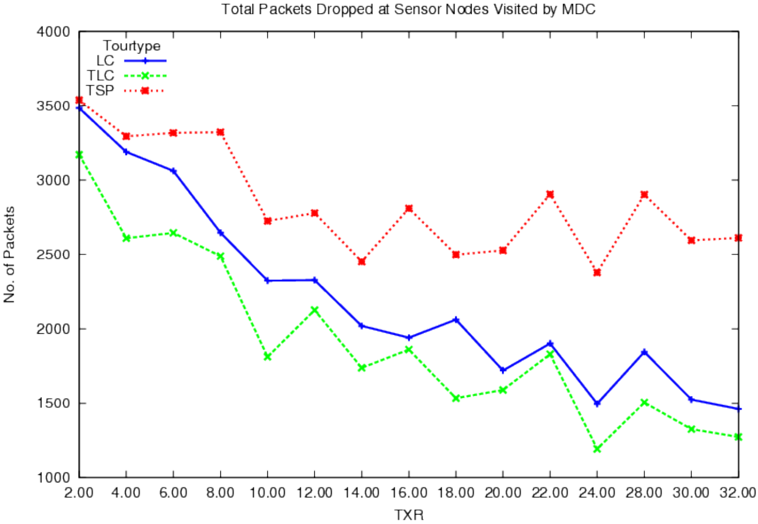


Figure: The total number of packets dropped by nodes

Experimental Results (Contd.)

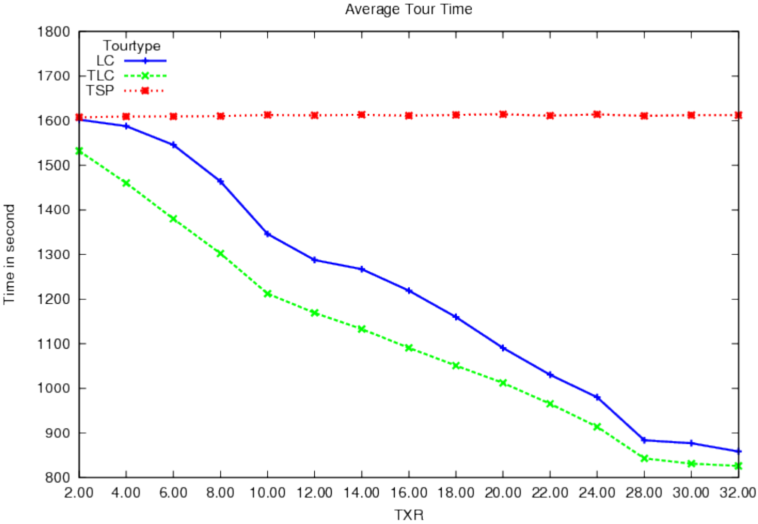


Figure: The average tour time of MDC

Experimental Results (Contd.)

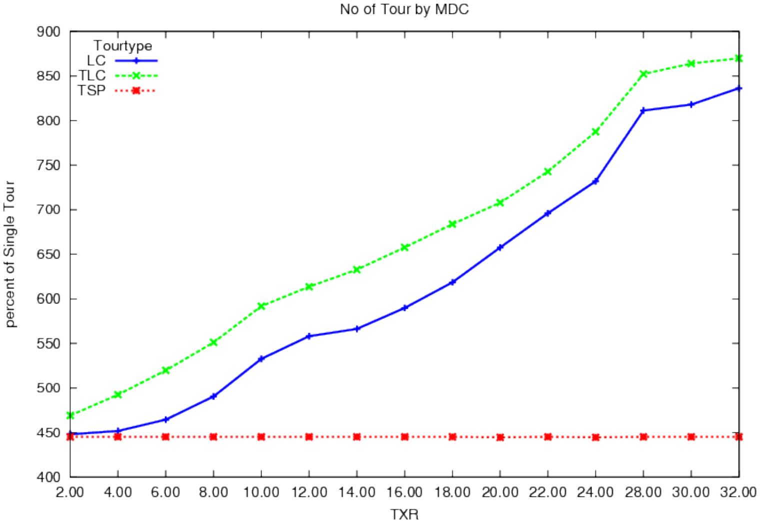


Figure: The total number of tours by MDC

Experimental Results (Contd.)

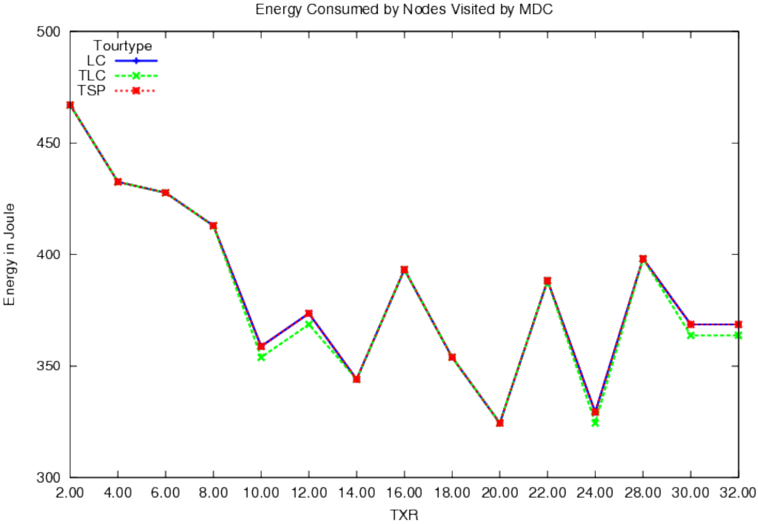


Figure: The average energy consumed by the sensor nodes

Conclusions

Summary of Results

Our Method of Linear Shortcut ensures:

- On the average, lower packet delivery latency
- Lower packet-drop rate
- Higher throughput
- Maximum m-lifetime of the network

Future Work

- The path-planning for multiple MDC's
- Possibly experiments with MICA motes (Wireless Network Lab, Dept. of CSE, BUET)

Thank You