The Collection Tree Protocol

Authors: Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, Sukun Kim, Philip Levis and Alec Woo

Presenter: Zoltán Kincses

Introduction

- The Collection Tree Protocol (CTP) is a tree based protocol with some tree root nodes
- CTP is address free
- Nodes generate routes to root using rooting gradient
- CTP assumes that the data link layer provides:
  - efficient local broadcast address
  - synchronous acks for unicast packets
  - protocol dispatch field (support higher-level protocols)
  - single-hop source and destination fields

- CTP assumes that it has link quality estimates of some number of nearby neighbors
- CTP has several mechanisms in order to improve delivery reliability (not promise 100%)
- CTP designed for relatively low traffic

Collection and CTP

- CTP uses Expected Transmissions (ETX) as a routing metric
- $ETX_{root}=0$ and $ETX_{node}=ETX_{parent}+ETX_{linktoparent}$
- CTP should choose the route with the lowest ETX
- CTP represents ETX as 16-bit fix-point real number with precision of hundredths
- Two main problem
  - Rooting loops
  - Packet duplication

Rooting Loops

- Occur when a node choose a new route with higher ETX than its old one
- Two mechanisms to address this problem
  - CTP packet contains a node’s current gradient value
    - the data frame with lower gradient value indicates inconsistency
    - try to solve inconsistency by broadcasting a beacon frame
  - not consider routes with an ETX higher than a reasonable constant

Packet Duplication

- Occurs when a node receives a packet successfully but the ack is not received by the sender
- The sender retransmits the packet and the receiver receives it a second time
- The duplication is exponential
- CTP data frames have Time Has Lived (THL) field which was incremented by routing layer

CTP Data Frame

<table>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th></tr>
<tr><td>P</td><td>C</td><td colspan="6">reserved</td><td colspan="8">THL</td></tr>
<tr><td colspan="16">ETX</td></tr>
<tr><td colspan="16">origin</td></tr>
<tr><td colspan="8">seqno</td><td colspan="8">collect_id</td></tr>
<tr><td colspan="16">data...</td></tr>
</table>

- P (Routing Pull)
  - allows nodes to request routing information
  - if P is set the node should transmit a routing frame
- C (Congestion notification)
  - if a node drops a CTP data frame it must set the C bit field on the next data frame
- THL (Time Has Lived)
  - if a node generates a CTP data frame, it must set THL to 0
  - if a node receives a CTP data frame must increment the THL

- ETX (Expected Transmissions)
  - the ETX is the routing metric of the single-hop sender
  - node send a CTX data frame must put the ETX of its routes
  - node receives a packet with lower gradient must schedule a routing frame
- Seqno
  - origin sequence number
- Collect_id
  - Higher-level protocol identifier
- Data
  - the data payload, of zero or more bytes

CTP Routing Frame

<table>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th></tr>
<tr><td>P</td><td>C</td><td colspan="6">reserved</td><td colspan="8">parent</td></tr>
<tr><td colspan="8">parent</td><td colspan="8">ETX</td></tr>
<tr><td colspan="8">ETX</td></tr>
</table>

- P (Routing Pull)
  - same as data frame
- C (Congestion Notification)
  - same as data frame
- Parent
  - the node's current parent
- Metric
  - the node’s current routing metric value

- After a node hears a routing frame, it must update its routing table
- If a node ETX value changes significantly, should transmit broadcast frame to notify other nodes
- The parent can detect when a child ETX is significantly below its own
  - parent must schedule a routing frame

Implementation

- Three major subcomponents
  - link estimator
    - responsible for estimating the single-hop ETX of communication with single hop neighbors
  - routing engine
    - uses link estimates to decide which neighbor is the next hop routing hop
  - forwarding engine
    - maintains queue of packets to send
    - decides when and if to send them

Link Estimation

- Two mechanism to estimate the link quality
  - periodic Link Estimation Extension Protocol (LEEP) packets
    - sends routing beacons as LEEP
    - seeds the neighbor table
    - similar to Trickle based dissemination
  - data packets
    - direct measure of ETX
    - estimator produces ETX estimate after 5 successfully acknowledged packet transmission

Routing Engine

- Picking the next hop for data transmission
- Keeps track of the path ETX of the subset of the nodes
- The minimum cost route has the smallest sum
  - the path ETX from that node
  - the link ETX of that node

Forwarding Engine

- Transmitting, retransmitting packets to the next hop and passing ack based information to the link estimator
- Deciding when to transmit packets to the next hop
- Detecting routing inconsistencies and informing the routing engine
- Maintaining a queue of packets to transmit (local and forwarded)
- Detection singe-hop transmission duplicates

- SubReceive.receive()
    - decides whether or not the node should forward a packet
    - calls the forwarding functions
- forward()
    - formats the packet for forwarding
    - checks if there is a possibly loop in the network
    - checks if there is space in the queue
        - if not C bit is set
        - if yes post send task

- sendTask()
  - examines the packet in the send queue, formats it and submits it to the AM layer
- sendDone()
  - examines the packet to see the results
    - if ack received pulls of the packet from the queue
    - if packet was locally generated is signals sendDone()
    - if there are packets remaining in the queue it starts a randomized timer that reposts this task

Thank you for your kind attention!