Collection Tree Protocol

Chen Jiang

Collection Tree Protocol

- A routing protocol that computes anycast routes to a single or a small number of designated sinks(Collection point) in WSN
- Only support upstream routing

Goals

- Reliable - end-to-end delivery
- Robust - adjust to topologies change
- Efficient - minimum amount of transmissions
- Hardware Independence

Challenges

- Highly dynamics links
  - Use periodic beacons to maintain topology & estimate link qualities
  - Beacon rate: tradeoff
- Transient loops
  - Topology changes → repair
  - Cause packet drops or stop until repair done

Two design mechanisms

- Datapath validation
  - Quickly discover
  - Fixes routing inconsistencies
- Adaptive beaconing
  - Fast recovery: reduce route repair latency
  - Low cost: send fewer beacons

Datapath Validation

- Every node maintains an estimate of the cost of its route to a collection point
  - cost metric: ETX = $\sum$ 1 / Prob($TX_{success}$)
- Each data pkt contains transmitter's ETX info
- Detect routing loop(inconsistence)
  - transmitter's advertised cost < receiver's advertised cost

Adaptive Beaconing

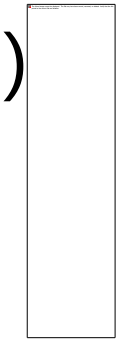
- Broadcast control beacons
- Adjust timer interval: (64ms ~ 1 hour)
  - Double timer interval every time until it reach the max value(low overhead)
  - Reset timer interval to min value when special events happen(quick recovery)

Control Plane Design

- Three events trigger to reset beacon interval
  - Detect a possible loop
    - Does not drop
    - Put a slight pause before forwarding
  - Routing cost decreases significantly
    - Quickly advertise cost changes
  - Receive packet with P bit set
    - Initial neighbor discovery

Data Plane

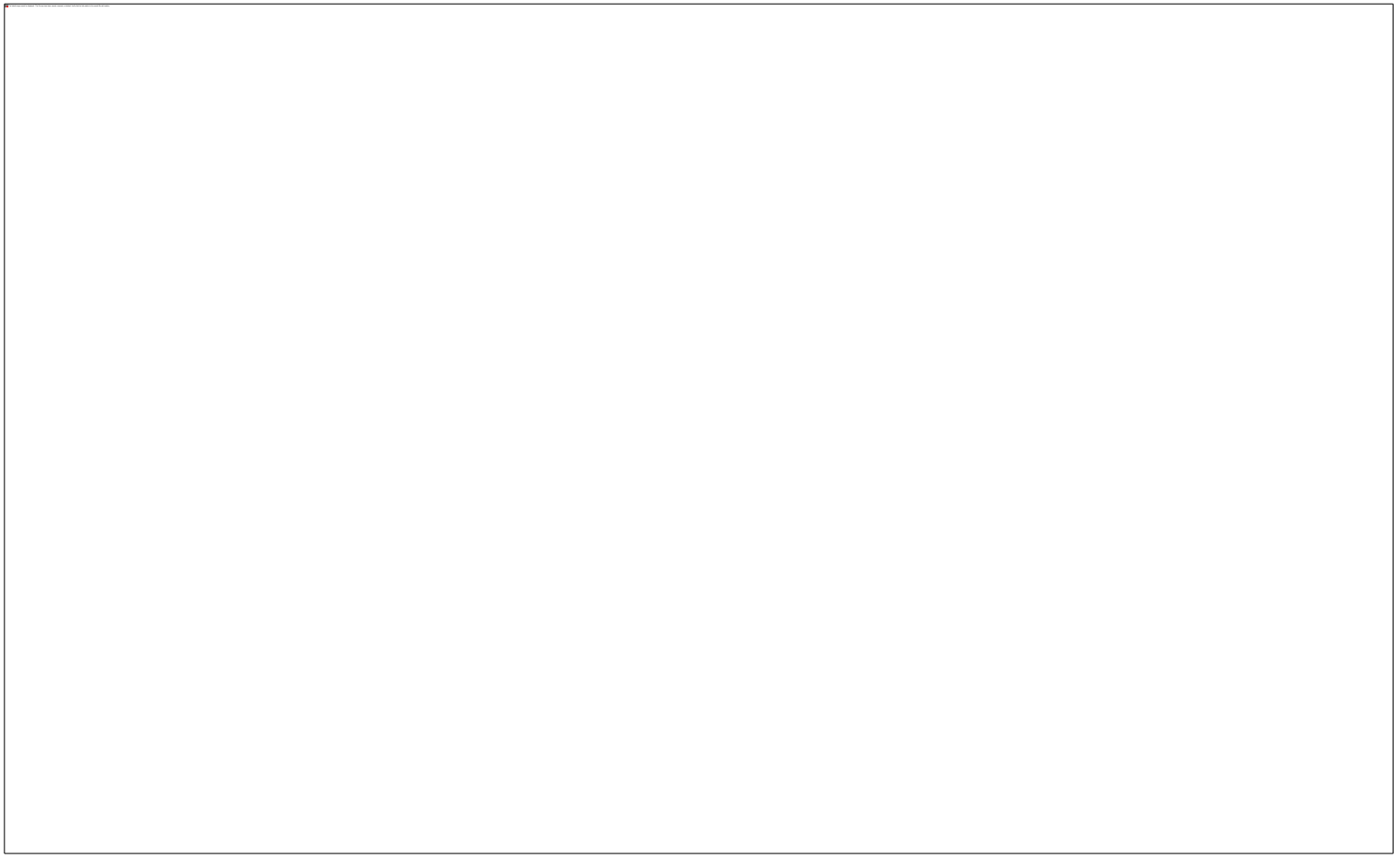
- Data frame: 8 byte header
  - Control field(P and C bit)
  - ETX field: Datapath validation
  - THL(Time has lived)
  - Origin Node ID
  - Sequence number
  - Collect ID



Duplicate check

- Aggressive retransmission policy
  - 32 times

Forwarding path



Client queue

- What's client?
- A set of One-deep queues
- What would happen If it find the client queue is busy? Drop the packet or delay it?

Hybrid Send Queue

- size = Number of Client queue + size of forwarding buffer pool
- Check duplicate before put data into the queue
  - Transmit Cache
  - The send Queue itself
- Possible packet loss if this buffer is full

Transmit Timer

- To solve self-interference problem
- Try to avoid collisions while under high load
- A → B → C
- If A and C could hear each other, C's forwarding could influence A's next transmitting. Hold back A's second transmission until C finish forwarding
- Wait in (1.5p, 2.5p) ~ 2p to make sure C finish
- My question: what if there is another D, also in A's transmitting range, why 2p is enough

Transmit Cache

- Since we don't drop looping packets, need a way to distinguish link-layer duplicate from looping packets
- Looping packets: same orignal sequence number and nodeID, different THL
- 4 slots is enough in practice
- My question: I don't see how this could solve false positive ACK and false negative ACK

Evaluation

- Topology properties
  - Number of nodes
  - Degree(Min, Max) – sparse connected or not
  - Whether on an interfering channel

Evaluation(2)

- Quantify link stability and quality
  - PL(average path length)
    - Larger PL, wider network
  - Cost(transmission/delivery)
  - Churn(parent change rate)
    - High dynamic links, high density network, highly interfered network usually has high churn

Experimental Results

- Choose IPI(Inter-packet interval) values well below saturation
- CTP Noe maintains an average delivery ratio above 90%
- Use 99.8% less time to response to topology changes
- Transmit Cache decrease the cost by 9%
- Increase IPI could improve the delivery ratio

Compare with MultiHopLQI

(Figure 2 ~ Figure 10)

- Higher delivery ratio – Reliable
- Lower Delivery Cost/packet – Efficient
  - Less control packet overhead
- Agile respond to new nodes join and route inconsistencies
- 100% delivery even some nodes fail – Robust
- Consume much less power

Questions

- How does it detect a dead link and find a new parent faster than beacon rate?
- Why longer sleep intervals could cause node to choose longer routes of more reliable links
- What about compare it with other protocols rather than just MultihopLQI
- What if links are busty & heavier traffic load?
- Self-interference problem is still not solved.