

COSC418 Project:

Load balancing in a CTP based network

Henry Jenkins and Regan Gunther

(<https://github.com/steakunderscore/COSC418-Assignment>)

Department of Computer and Electrical Engineering,
University of Canterbury,
Christchurch,
New Zealand

September 30, 2011

- 1 Design
 - Overview
- 2 Implementation
 - Code
- 3 Figures
 - CTP
- 4 Results & Conclusions
 - Testing
 - Conclusions

Our Design

- $p^* = \arg \min p \in \{ \text{direct neighbours} \} [\alpha \cdot (ETX_{s,p} + ETX_p) + \beta \cdot L_p^s]$
- Our design is to keep L_p^s locally
- We increment L_p^s as:
- $L_p^s = n_p^s - t_p^s$
- $n_p^s \geq t_p^s$
- Where t_p^s is incremented periodically through a timer
- t_p^s is included to stop L_p^s getting too large.

Why do this instead of transmitting link load data

- Allows us to use the unmodified CTP routing packet
- Means we can add load balancing to an existing CTP network
- No extra data transmission

UnicastNameFreeRouting

- We took the original UnicastNameFreeRouting interface and turned this into UnicastNameFreeLoadBalRouting by adding extra load balancing features.

Code Implementation

```
interface UnicastNameFreeLoadBalRouting {  
  
    command am_addr_t nextHop();  
    command bool hasRoute();  
    //Triggers a packet notification  
    command void packetSent();  
  
    event void routeFound();  
    event void noRoute();  
}
```

- Interface provided by CtpRoutingEngine and used by MultiHopOscilloscope

UnicastNameFreeLoadBalRouting Wiring

- We rewired the CtpForwardingEngine and CtpRoutingEngine to be connected by UnicastNameFreeLoadBalRouting rather than the original UnicastNameFreeRouting

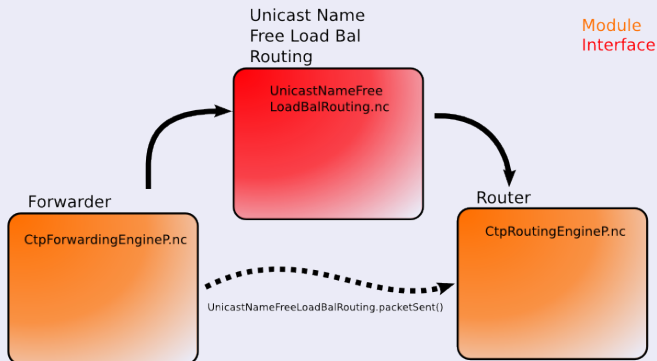
Code Implementation (CtpP.nc)

```
implementation {  
  ...  
  
  components new CtpForwardingEngineP() as Forwarder;  
  components new CtpRoutingEngineP(...) as Router;  
  
  Forwarder.UnicastNameFreeLoadBalRouting -> Router.Routing;  
  
  ...  
}
```

UnicastNameFreeLoadBalRouting Wiring (Contd.)

- The rewiring of UnicastNameFreeLoadBalRouting is shown below

Wiring block diagram



Our Design

- $p^* = \arg \min_{p \in \{\text{direct neighbours}\}} [\alpha \cdot (ETX_{s,p} + ETX_p) + \beta \cdot L_p^s]$
- Our design is to keep L_p^s locally

Code Implementation

```
// ETX for load balancing
uint16_t loadEtx;

//Complete Equation:
beaconMsg->etx = routeInfo.etx +
call LinkEstimator.getLinkQuality(routeInfo.parent)
+ (loadEtx/LOAD_EFFECT_THRESHOLD);
```

t_p^s

Code Implementation

```
/*
 * Timer for the load balancing algorithm
 */
event void LoadTimer.fired() {
    //First decrement loadEtx to ensure decay
    if (loadEtx > 0) {
        loadEtx--;
    }
    //If there is a large change in loadEtx tell neighbours
    if (radioOn && running) {
        if (loadEtx > oldLoadEtx + 10 ||
            (oldLoadEtx > 10 && loadEtx < oldLoadEtx - 10)) {
            post sendBeaconTask();
        }
    }
}
```


n_p^s

Code Implementation

```
/*  
 * This is to be called when ever a packet is sent via the radio.  
 */  
command void Routing.packetSent() {  
    loadEtx++;  
    printf("P\n");  
    //printf("Load ETX Incremented. It is now: %d\n",loadEtx);  
    //printf fflush();  
}
```

Parameter Definitions

In TreeRouting.h:

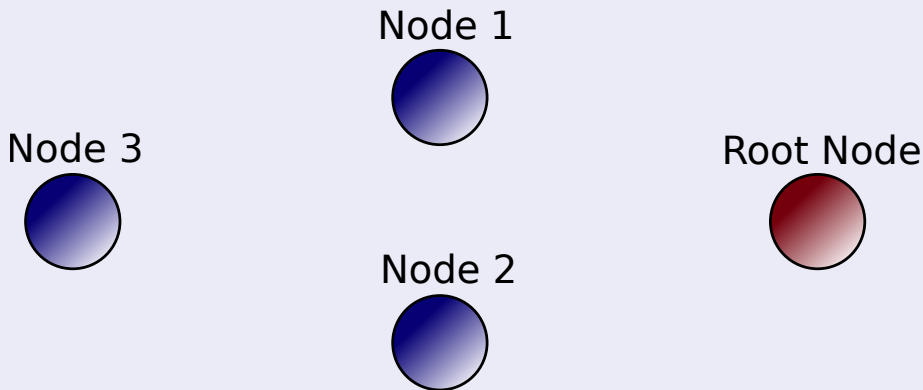
Code Implementation

```
enum {  
    ...  
  
    // Load balancing timer  
    LOAD_INTERVAL = 100,  
  
    //Number of packets per rollover  
    LOAD_EFFECT_THRESHOLD = 1,  
  
    ...  
};
```

- This allowed for easy access of values during the testing stage

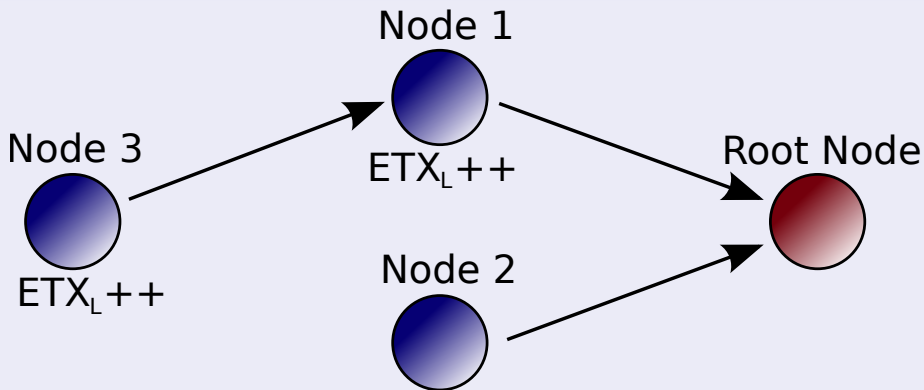
Initialisation

Data flow



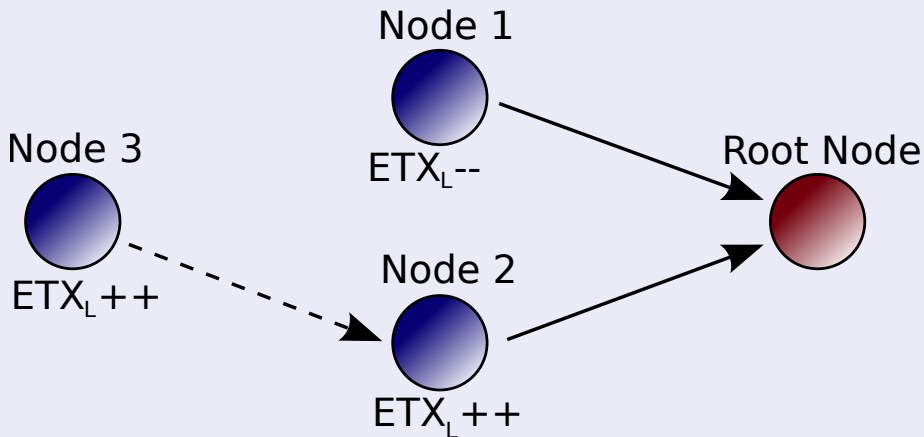
Initial route

Data flow



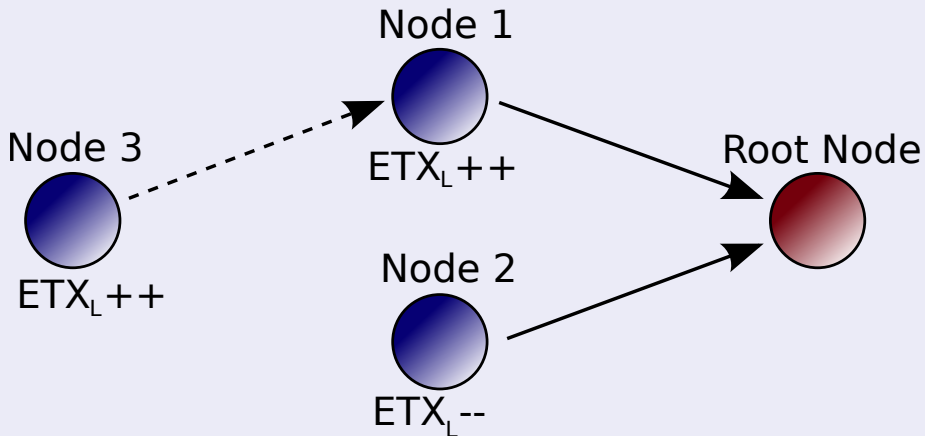
Load balanced

Data flow



Load balanced (again)

Data flow



Testing

- Using lower transmit power
- *printf* client to debug
- Listen client to view raw traffic
- Transmitting routing beacon every data packet
- Couldn't get Multihop oscilloscope working
- MViz showed our shifting ETX values

Conclusion

- Still working on finding correct thresholds
- Currently implementing and testing code
- Can visualise operation using the MViz application
- We have implemented load balancing as a component to plug into standard CTP
- Simply include our CTP to use

