# COSC418 Project:
# Load balancing in a CTP based network

Henry Jenkins and Regan Gunther

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

September 30, 2011

1. Design
   - Overview
2. Implementation
   - Code
   - Drawbacks
3. Results & Conclusions
   - Testing
   - Conclusions

# 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

# Our Design

- $p^* = \text{arg min} p \in \{ \text{ direct neighbours}\} \, [\alpha \cdot (ETX_{s,p} + ETX_p) + \beta \cdot L_p^s]$
- 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 nabours
  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);
  //printfflush();
}
```

# 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

# Drawbacks

What if $ETX_p + ETX_{p,s} + L_p^s$ (i.e. $ETX_s$) increases to be $\geq 2^{16}$?

- Bad things...
- This is managed by including the $t_p^s$ in the loading parameter $L_p^s$
- (Remember $L_p^s = n_p^s - t_p^s$ )
- Periodic timer.

# Drawbacks

ETX changing with each packet transmission causes routing updates all the time. All these updates would be inefficient.

- A solution to this is to only update routes when ETX changes by a given threshold

# Drawbacks

Exponential ETX growth of a node

- In a path of 5 nodes to the root, $L_p^s$ increases with every packet. The more hops to the root, the more pronounced this effect

# Conclusion

- No results as yet
- Currently implementing and testing code