Routing in a Delay Tolerant Network*

Sushant Jain (Univ. of Washington),

Kevin Fall (Intel Research Berkeley)

Rabin Patra (UC Berkeley)

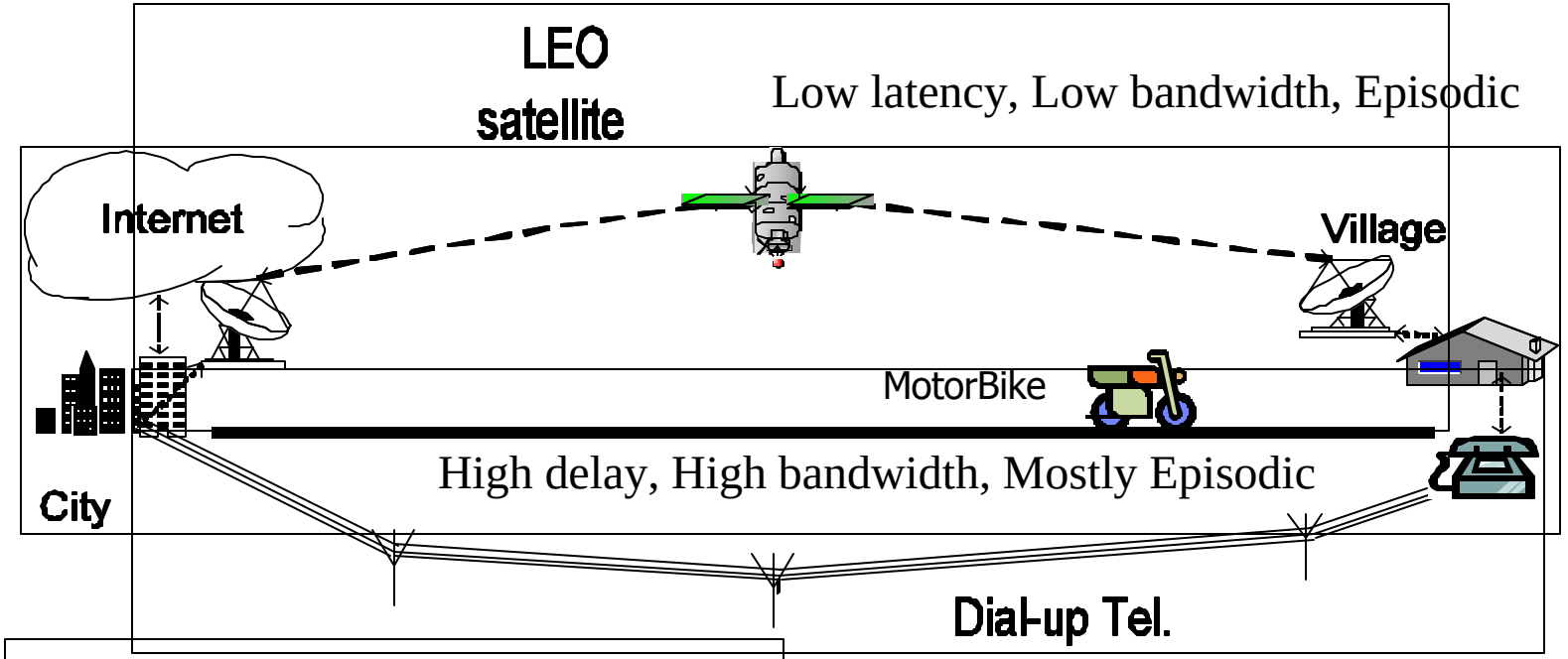
ICSI/ICIR – 12-May-04

*To appear, SIGCOMM 2004

Outline

- Why do this? (a motivating example)
- What is routing in a DTN?
  - Why it is different (model assumptions)
  - Formulation
- Evaluation Framework
  - Optimal solution
  - Oracle construction
- Current status and Future Work

Example: Connecting a Remote Village



Which paths?

When? Which data?

Low latency, Low bandwidth, On-Demand

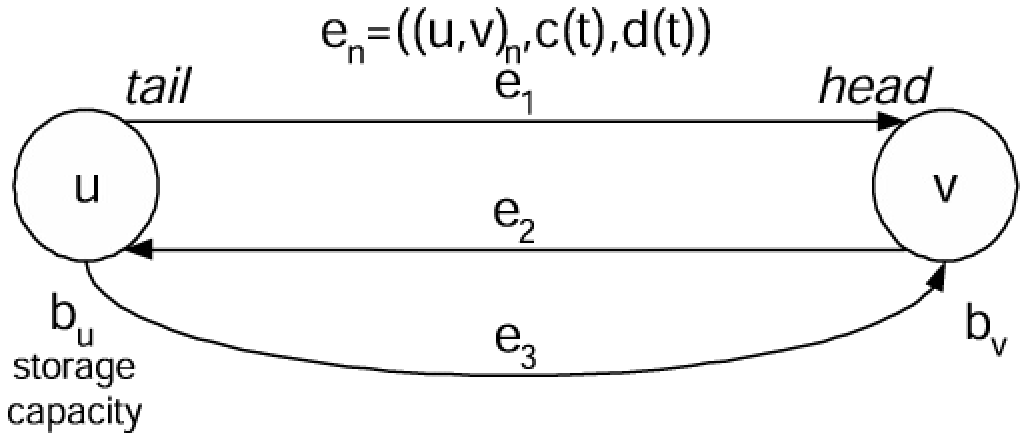
Other DTN networks

- Space Networks
  - Satellites
  - Deep space
- Sensor networks
  - Scheduled power outages
  - Service via *data mules*
- Mobile networks
  - Ad-hoc / mesh
  - Wandering nodes (e.g. ZebraNet)

What is Routing in a DTN?

- Traditional routing
  - Inputs*: G=(V,E), (s,d).* Find a shortest path from *s* to *d* in *G*. Dynamic: update as *G* changes, but still assume some path *p(s,d)* exists. "Shortest" can vary.
- DTN Routing
  - Inputs: Nodes with buffer limits, Contact List, Traffic Demand
    - Contact list may contain periods of capacity zero
  - Problem: given (some) metric of goodness, compute the path and schedule so as to optimize the metric. Multiple paths may be ok.
  - Assumption: paths are not lossy (replication not used)

DTN Routing MultiGraph



- One edge per (phys) link
- c(t) : capacity [piecewise constant]
- d(t): delay [piecewise constant]
- b_u: storage at node u

Notation:

$s(u,v)=u$

$t(u,v)=v$

DTN Routing Objective

- A DTN Message k is an ordered tuple *(u,v,t,m)*
  - *u*: source, *v*: destination, *t*: inject time, *m*: size [bytes]
- DTN Routing Objective
  - Without violating these constraints:
    - Do not overrun buffer capacity
    - Do not overrun edge capacity
  - Minimize average message delay
    - Optimal case will require multi-path
    - (other objectives are possible, but this helps most of them)

DTN Routing (optimal solution)

- LP Formulation uses *time intervals:*
  - $\boldsymbol{I}_e = \{I_1, \ldots, I_h\}$, $I_q = [t_{q-1}, t_q)$ $(t_{q-1} < t_q)$
    - *See other paper for how this is determined…*
  - $I_q + R = [t_{q-1}+R, t_q+R)$
- Topology Definitions
  - (V,E) [usual], $c_{e,t}$ [capacity of e at time t], $d_{e,t}$ [delay of e at time t], b_v [buffer at v], I^v [incoming edges to v], O^v [outgoing from v]

DTN Routing (optimal 2)

- Traffic Demand Definitions
  - K [set of all messages (commodities)]
  - K^v [set of messages destined for v]
  - $N^k_{v,t}$ [amount of k residing in v at time t]
  - $X^k_{e,I}$ [amount of k placed into e during I]
  - $R^k_{e,I}$ [amount of k received from e during I]

DTN Routing (optimal 3)

- Constraints: $\sum_{e \in \mathrm{I}^v} R^k_{e,I_q} - \sum_{e \in \mathrm{O}^v} X^k_{e,I_q} =$

Data is Stored or forwarded at vertex v over interval I_q

$$\begin{cases} N^k_{v,t_q} - N^k_{v,t_{q-1}} + \mathrm{m(k)} & \text{if } \mathrm{s(k)} = v,\ \omega(k) = t_q \\ N^k_{v,t_q} - N^k_{v,t_{q-1}} & \text{otherwise} \end{cases}$$

$$k, v, I_q \qquad (2)$$

Received = Sent

$$R^k_{e,I_q \oplus d_e, t_{q-1}} = X^k_{e,I_q}, \qquad k, e, I_q \qquad (3)$$

Not beyond buffer

$$\sum_{k \in K} N^k_{v,t_{q-1}} \leq b_v, \qquad v, I_q \qquad (4)$$

Not beyond edge capacity

$$\sum_{k \in K} X^k_{e,I_q} \leq c_{e,t_{q-1}} \cdot |I_q|, \qquad e, I_q \qquad (5)$$

Only sources start w/data

$$N^k_{v,t_0} = \begin{cases} \mathrm{m(k)} & \text{if } v = \mathrm{s(k)},\ t_0 = \omega(k) \\ 0 & \text{otherwise} \end{cases} \qquad k, v \qquad (6)$$

Only dests end w/data

$$N^k_{v,t_h} = \begin{cases} \mathrm{m(k)} & \text{if } v = \mathrm{d(k)} \\ 0 & \text{otherwise} \end{cases} \qquad k, v \qquad (7)$$

DTN Routing (optimal 4)

(start time for k)

$$\min \sum_{v \in V} \sum_{k \in K^v} \sum_{I_q \in I_E} (t_{q-1} - \omega(k)) \cdot (\sum_{e \in I^v} R^k_{e,I_q} - \sum_{e \in O^v} X^k_{e,I_q})$$

- This summand accounts for the fraction of delay at one vertex. The overall summation (over all vertices, commodities, intervals) minimizes the time in network. (The X term is an odd case called ‘move aside.’)

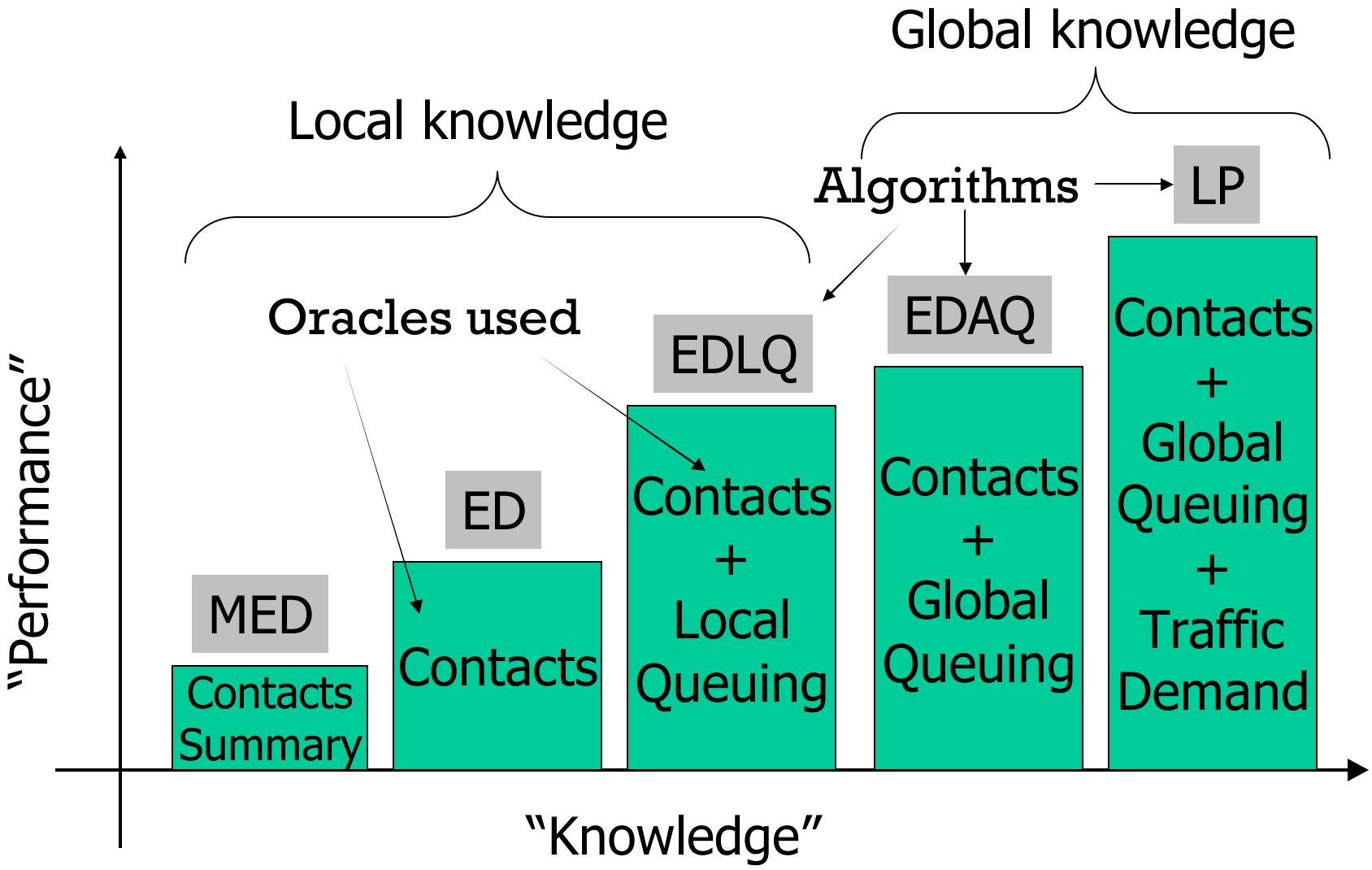
LP Results

- Pretty horrible performance
  - The interesting constraints grow as the product of |V| x |E| x |K| x |***I***|
  - Can lead to millions of equations for small problem
- But a useful bound and driver of insight
  - Move aside: cycles can be optimal routes
  - Several places we might relax the need for global information…

Knowledge ‘Oracles’

- Contacts Oracle
  - Complete link availability schedule (c(t), d(t))
  - Time dependent information
- Contacts summary Oracle
  - Average link availability
  - Time independent information
- Queuing Oracle:
  - Link queues, available storage
  - Two versions: Local vs Global
- Traffic Demand Oracle

Conceptual Performance



Algorithmic Menu

- FC (no knowledge), MED (average topology knowledge)
- ED (topology knowledge), EDLQ (ED w/local queuing)
- EDAW (ED w/all queuing), LP (perfect knowledge)

Abbr.	Name	Description	Oracles Used
FC	First Contact	Use any available contact	None
MED	Minimum Expected Delay	Dijkstra with time invariant link costs based on averages of link waiting time	Contacts Summary
ED	Earliest Delivery	Modified Dijkstra with cost function based on waiting time	Contacts
EDLQ	Earliest Delivery with Local Queue	ED with cost function incorporating local queuing	Contacts
EDAQ	Earliest Delivery with All Queue	ED with cost function incorporating queuing information at all nodes and using reservations	Contacts and Queuing
LP	Linear Program	-	Contacts, Queuing and Traffic

Modified Dijkstra

Input: $G = (V, E)$, s, T, $w(e, t)$
Output: L

1: $Q \leftarrow \{V\}$
2: $\mathrm{L}[s] \leftarrow 0$, $\mathrm{L}[v] \leftarrow \infty \; \forall \; v \in V \; s.t \; v \neq s.$
3: **while** $Q \neq \{\}$ **do**
4: Let $u \in Q$ be the node s.t $\mathrm{L}[u] \leq \forall_{x \in Q} \mathrm{L}[x]$
5: $Q = Q - \{u\}$
6: **for** each edge $e \in E$, s.t. $e = (u, v)$ **do**
7: **if** $\mathrm{L}[v] > (\mathrm{L}[u] + w(e, \mathrm{L}[u] + T))$ **then**
8: $\mathrm{L}[v] \leftarrow \mathrm{L}[u] + w(e, \mathrm{L}[u] + T)$
9: **end if**
10: **end for**
11: **end while**

Different

T: start time; L[]: path cost from s to all nodes; w(e,t): cost (time) on e at time t

Adapting Dijkstra

- Using this framework we can assign w(e,t):
  - w(e,t) = msgsize/c(e,t) + Q(e,t)/c(e,t) + d(e,t)

    cost = transmission + queuing/waiting + propagation
- Q(e,t): amount of data queued for edge e at time t
  - Q(e,t) = 0 (for ED: earliest delivery)
  - Q(e,t) = amnt of data queued locally on e at time t (for EDLQ: ED with local queuing information)
  - Q(e,t) = amnt of data queued anywhere for e at time t (for EDAQ: ED with all queuing information)

DTN Simulations

- Developed our own DTN simulator (Java)
  - Special focus on link disconnection:
    - Complete failure (all transiting msgs dropped)
    - Close at source (all transiting msgs are delivered)
    - Modular way to attach up/down patterns
  - Dynamic fragmentation (see DTN architecture)
- Simulated two scenarios
  - **Village network (first slide)**
  - Bus network in San Francisco (see paper)

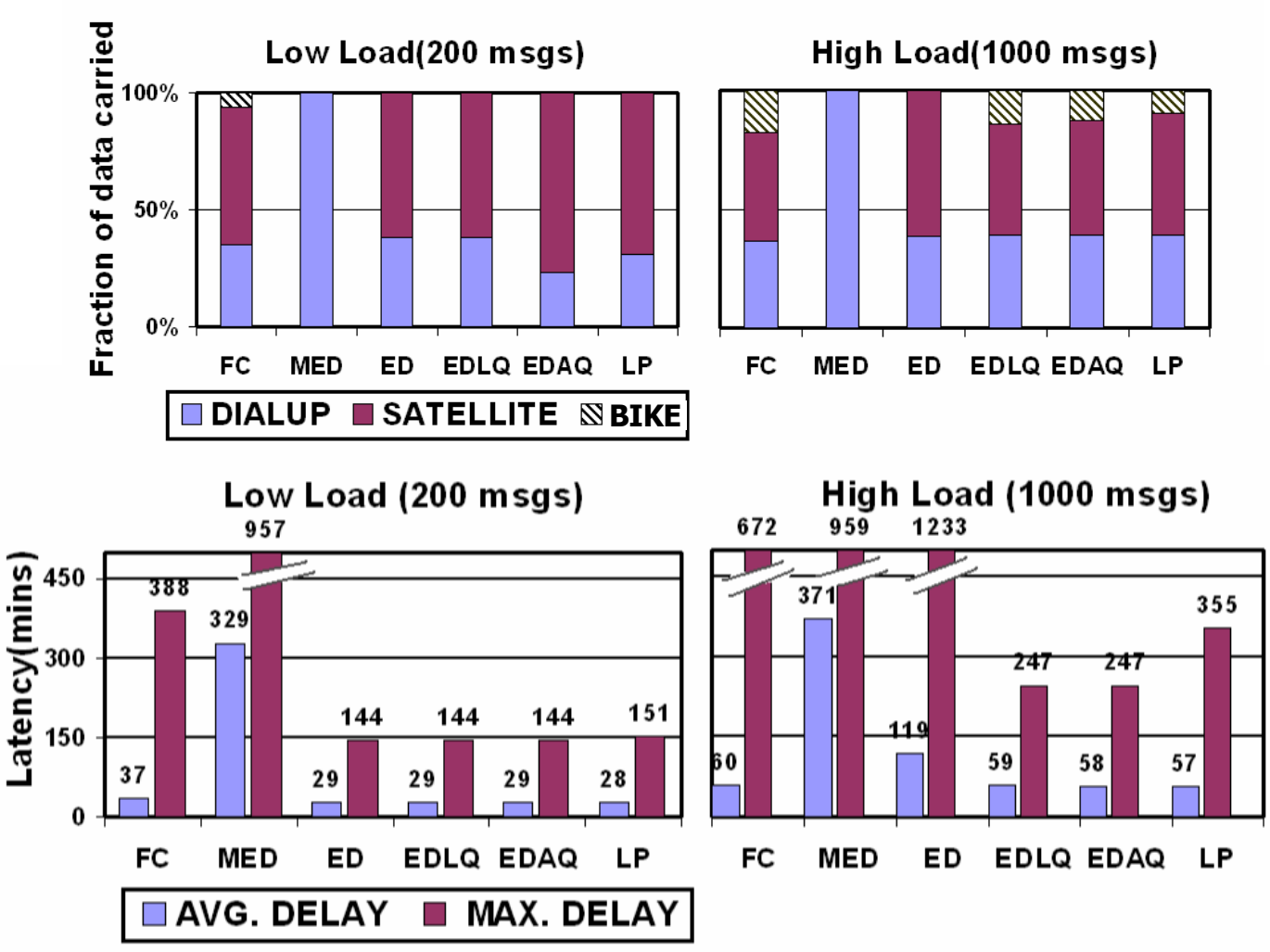
Village Simulation

- Locations
  - Kwazulu-Natal (Village) [see **`http://wizzy.org.za`**]
  - Capetown, S. Africa (City)
- Network (based on a true story…)
  - Dialup (4kbps at night 23:00-06:00 local time, 20msec)
  - 3 PACSATs (bent pipes, 4-5 passes/day, 10 min/pass, 10kbps, 25msec)
  - 3 Motorbikes (2hr journey, 1Mbps to bike, 128MB storage capacity, 5 min contacts)

Village Simulation 2

- Traffic Pattern
  - V? C traffic is small (1KB avg, ~web requests)
  - C ? V traffic is larger (10KB avg, ~web pages)
  - Two loadings: 200 msgs/day (low), 1000 msgs/day (high)
- Traffic injected uniformly over 1st 24-hours of 48-hour simulation run (no traffic remains)

Village Results



Some Observations

- A simplistic yet rich “routing” scenario
- MED: dialup always used during high or low load
- ED: most traffic over sat (60%), the rest uses dialup (low or high load)
- FC: sometimes chooses bike (10%), which explains its high maximum delay; avg delay is nominal– FC can’t make a terrible choice here
- EDAQ/EDLQ identical for low-load

Some Observations - 2

- At high load, some differences appear:
  - MED, ED same as low load (not queuing aware)
  - ED deteriorates rapidly as it tries to route all messages over a satellite
  - EDLQ/EDAQ now start using motorbike (~25%), leading to a significant reduction in delay
  - FC winds up routing more traffic over the bike which, interestingly, helps it out too
- LP took 7.5 min, for 16k iterations in CPLEX (8-proc PIII@700Mhz each with 3GB memory), producing about the same results as EDLQ/EDAQ (500k constraints)
  - Trades off higher max delay for the best minimum avg delay

Conclusions

- Interesting/challenging scenarios are easily generated
  - General problem formulation is non-trivial, but some guidance from operations research literature
  - Even simplistic networks may be difficult
- LP formulation explodes, but simpler schemes (e.g. based on Dijkstra) seem reasonable
  - Light load: moderate scheme (ED) optimal
  - Higher load: congestion aware scheme (EDLQ) ok
  - Not a profound benefit for going to EDAQ or LP (!)

Futures

- How to implement an oracle?
  - They represent the data collected by some routing protocol, but this will be a challenge to realize
- Lossy networks
  - What if an oracle lies?
  - Sensitivity to bad contact information and/or bad information regarding queing
  - Utility of data replication and/or erasure coding
- Improvements to the LP formulation
- More complex simulation scenarios

Acknowledgements

- Juan Alonso (SICS) for help with the LP formulation
- Carlos Guestrin for help with CPLEX
- DTNRG and ICT4B for motivating the problem

thanks

Is this like e-mail routing?

- Internet e-mail “routing”
  - Static set of mail exchangers via DNS MX-record construct
- Downsides of MX-based routing
  - No dynamics
  - No multiple-paths
  - Specific to internet (uses DNS resource record)