A Data-First Architecture for Unstructured Wireless Networks

Michael Meisel

Ph.D. Dissertation Prospectus
October 12, 2010
Unstructured Wireless Networks

• Multi-hop

• No controlled mobility

• Topology can be highly dynamic

• Can be connected or disconnected

• Examples: MANETs, VANETs, disruption-tolerant networks, combinations thereof
Goals

• One architecture that will work on any unstructured network

• Follow the Named Data Networking (NDN) philosophy
  • Data first: delivery based on “what”, not “where”

• Get rid of holdovers from the wired domain
Current Paradigm

- Provide data delivery from source to destination node
- Treat connected and disconnected networks separately

You may ask: why is it bad to treat connected and disconnected nets separately? See next slide.
Connected or Disconnected?

(suppose the nodes are mobile)
A protocol that is efficient in ANY unstructured network doesn’t need to decide
Current Approaches
Connected Nets: The Wired Approach

1. Each node is assigned an IP address

2. Applications communicate using destination IPs

3. The routing protocol finds a single best path from source to destination

4. At each hop along the path, the sender determines which single node (based on step 3) is allowed to forward the data
Issues with The Wired Approach

• IP addresses lose their meaning, aggregatability in mobile nets

• Applications care about data, not location

• Finding, maintaining hop-by-hop paths is expensive

• Pre-determined paths don’t take advantage of the broadcast nature of wireless
Alternative: Opportunistic Routing

• Improvements:
  • Takes advantage of the broadcast nature of wireless

• Shortcomings:
  • Focused on stationary mesh networks
  • Still dependent on IP addressing, location-based delivery

• Examples: ExOR [1], MORE [2]

Disconnected Network Routing

- **Improvements:**
  - Can take advantage of the broadcast nature of wireless
  - No IP addressing

- **Shortcomings:**
  - Inefficient for connected networks (or network segments)

- **Examples:** Epidemic routing [3], Spray and Wait [4]

Named Data Networking (NDN)
NDN Architecture

- Routing/forwarding is based on data names instead of node addresses

Applications can be built directly on top of NDN data delivery, use names to comm.

Any communication media that can provide best effort delivery
NDN Communication

• *(Optional Step 0: Use a routing protocol to announce names)*

• **Step 1:** An application sends an *Interest* packet containing a request for data by name. It can be flooded or routed.

• **Step 2:** Any node that has the data can send a *Data* packet back towards the source of the Interest. Intermediate nodes cache the data.

• Future Interests for the same name can be serviced by caches
Assume we flood Interests.
By forwarding the Interests, the intermediate nodes have established a path from responder to requester.
The nodes that forward the data also cache it.
Suppose another node requests the same data name.
Its immediate neighbors have cached the appropriate data, so they can respond.
NDN Advantages for Unstructured Nets

• Applications can communicate based on data names only, no need to worry about location

• Unlike IP addresses, data names are always meaningful

• Built-in caching for disconnected networks

• Can interoperate with wired NDN infrastructure

Data name movement is orthogonal to node movement
Listen First, Broadcast Later (LFBL)
What is LFBL?

- A forwarding protocol for *connected* wireless networks
- A proof-of-concept for NDN in multi-hop wireless networks
LFBL Goals

- Name-based communication at the application layer
- Broadcast-only communication at the MAC layer
- No control packets
- Use the best *available* path *on the fly*
  - No path selection in advance
Communication

- At first, requests are flooded
- Requests contain the desired name
- Any number of responders may respond with a data packet
- Responses take the best available path back to the requester
- Further requests for the same name take the best available path to the responder(s)
Broadcast-Only Forwarding

• Forwarding decisions must be made by the receiver

• **Step 1:** Determine if I am *eligible* to forward the packet. If so:

• **Step 2:** Listen to see if another node closer to the intended destination forwards the packet. If not:

• **Step 3:** Forward the packet
Follow-up Questions

• How does a receiver know if it’s eligible to forward?

• How long should a receiver listen, waiting for someone else to forward?
Distances and Eligibility

- The network shares a single *distance metric*
  - (Could be: hop count, receive power, geo distance...)
- In every packet, senders broadcast their distance to the requester and/or responder
- Nodes track their distance to active endpoints
- **Only nodes closer to the destination are eligible forwarders**
Listening Periods

• Eligible forwarders choose their listening period based on the network’s delay metric

  • Tells the node how long to wait before forwarding

• Only forward if a closer node does not forward before the listening period is over
R broadcasts a request, all nodes forward, record their distance from R

d(A,R) = 10

d(B,R) = 8

d(C,R) = 10

d(S,R) = 14

d(D,R) = 16

d(E,R) = 3

d(F,R) = 3
S broadcasts a response. D will never forward. A, B, or C may forward... after some delay.

If any node closer to R forwards the message, progress will be made.
The delay depends on the network’s delay metric.

If any node closer to R forwards the message, progress will be made.
Simplest delay metric: random.

If any node closer to R forwards the message, progress will be made.
But the receivers have some useful information: their own distance to R and S’s distance to R.

If any node closer to R forwards the message, progress will be made.
C can calculate its listening period by comparing S’s claimed distance to R with its own prediction, assuming the packet were to travel through C. Its listening period will be proportional to the difference.

distance traversed from S = 5
\[ d(C,R) = 10 \]
\[ d(S,C,R) = 5+10 = 15 \]
wait \( \propto 15-14 \)

d\( (S,R) = 14 \)
Suppose all neighbors received the packet. B will forward immediately.

- Distance traversed from S = 6
  - \(d(C, R) = 8\)
  - \(d(S, R) = 6 + 8 = 14\)
  - \(wait \propto 14 - 14\)

- Distance traversed from B
  - \(wait \propto 1\)
  - (same as C)
A and C hear B before their listening period ends, so they do not forward.
Suppose B moved away.
A and C will forward the packet instead, once their listening period is over.
But A and C will try to forward at the same time, resulting in a collision!
Simple solution: include a random factor as well. Suppose \( y < x \). C will forward first, A will overhear and not forward.
Preliminary Handling of Stale State

- Distances will become stale, discard old ones
- Track variance in distance change
  - Make nodes with greater variance have longer listening periods
  - Allows us to implicitly prefer more stable paths
LFBL Simulation Results
Simulation Setup

- QualNet simulator
- 100 nodes
- 1500 x 1500 meter area
- Random waypoint using steady-state initialization [5]
- Bidirectional traffic; one request-response every 100 ms; multiple node pairs

Evaluation Metrics

- **Roundtrip time**: Time from request sent to response received

- **Response ratio**: Responses received over requests sent

- **Overhead**: Percent of bytes transmitted *not* in direct service of data delivery

- **Path length**: Average length of all paths used for **successful** data delivery

- **Total data transferred**: Total number of bytes successfully received at all endpoints (requesters and responders)

overhead includes control packets, all headers, colliding data packets
Distance and Delay Metric Comparison

Shows two things:
1. Hop count vs. Rx power
2. Distance+variance vs. random (Briefly explain delay metrics here)
Nodes either stationary or RW at 30 m/s
8 flows
Despite the fact that LFBL uses delays in the protocol, much shorter RTTs
Performance isn’t affected by mobility levels
Transitioning between multiple responders
Everything looks the same from the application layer
Problems with Stale State

8 flows. Stale state timeout fixed at 3 seconds.
At one second, stale state is used, response ratio plummets.
At 5 seconds, stale state is thrown out, everything is flooded.
Dissertation Goals
Goal 1: Caching Support

• Purpose:
  - Can improve performance in connected networks
  - Necessary to support disconnected networks

• Challenges:
  - Will require significant changes to how LFBL deals with names
  - Selection of cache replacement algorithm
Goal 2: Better Handle Stale State

• Purpose:
  • More reliable delivery, less flooding

• Challenges:
  • Mitigating the effects of stale state
  • Detecting delivery problems due to staleness quickly
Goal 3: Disconnected Network Support

• Purpose:
  • Unify connected and disconnected networks
  • The NDN architecture already provides caching

• Challenges:
  • NDN should work for disconnected networks/DTNs in theory, but has not been tried in practice
  • Any protocol adjustments necessary for disconnected networks must still work in connected networks
Timeline

• **November 2010**: Initial caching support

• **December 2010**: Various stale state handling techniques

• **January 2011**: Simulation and evaluation; understand behavior in connected and disconnected networks

• **February 2011**: Iteration on and selection of techniques developed above

• **March 2011**: Final results and dissertation ready