OntoNet: Scalable Knowledge-Based Networking

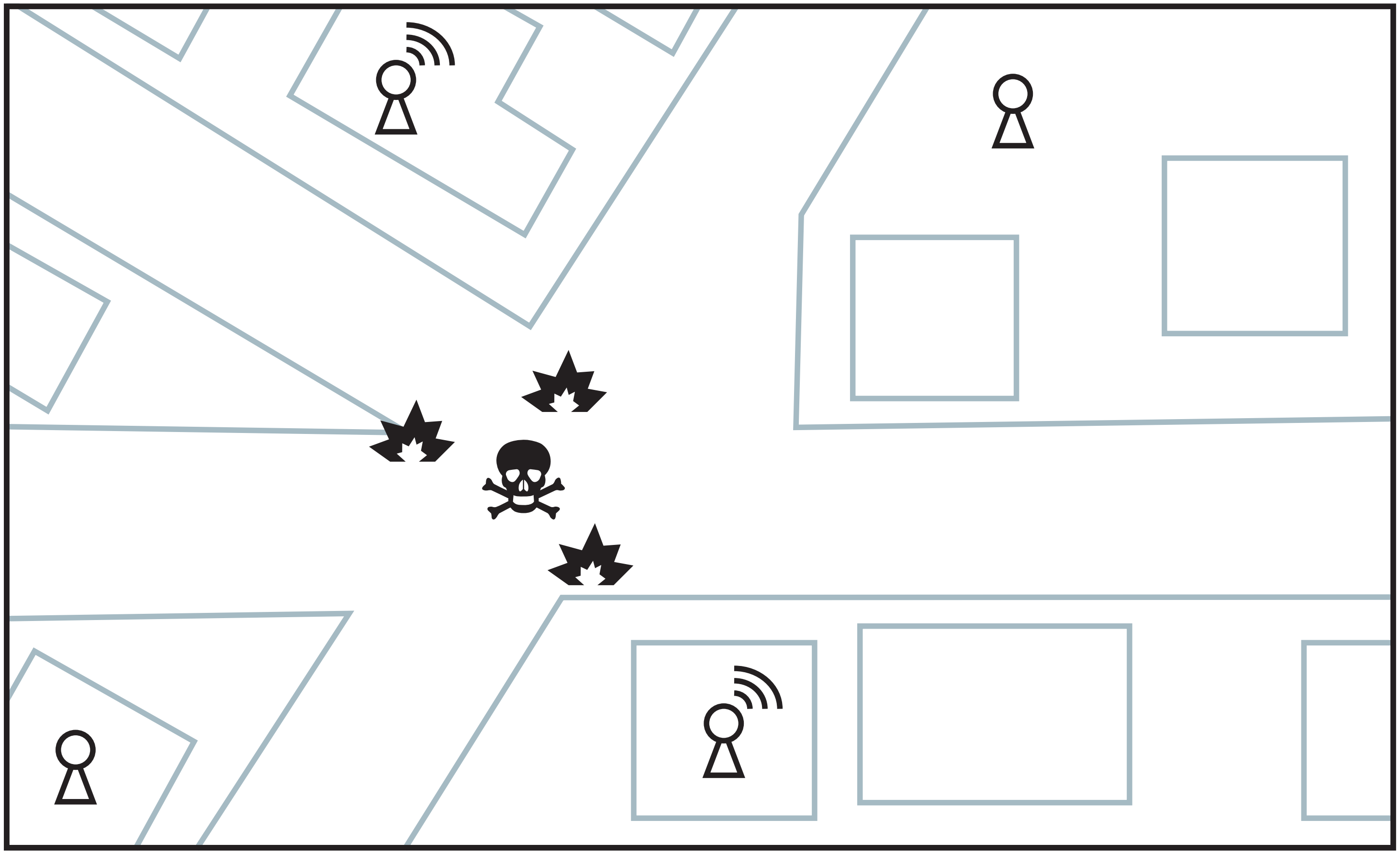
Joseph B. Kopena
tjkopena@cs.drexel.edu
Dept. of Computer Science
Drexel University

Boon Thau Loo
boonloo@cis.upenn.edu
Dept. of Computer and Information Science
University of Pennsylvania

April 7, 2008

Motivating Scenario: CBR Response

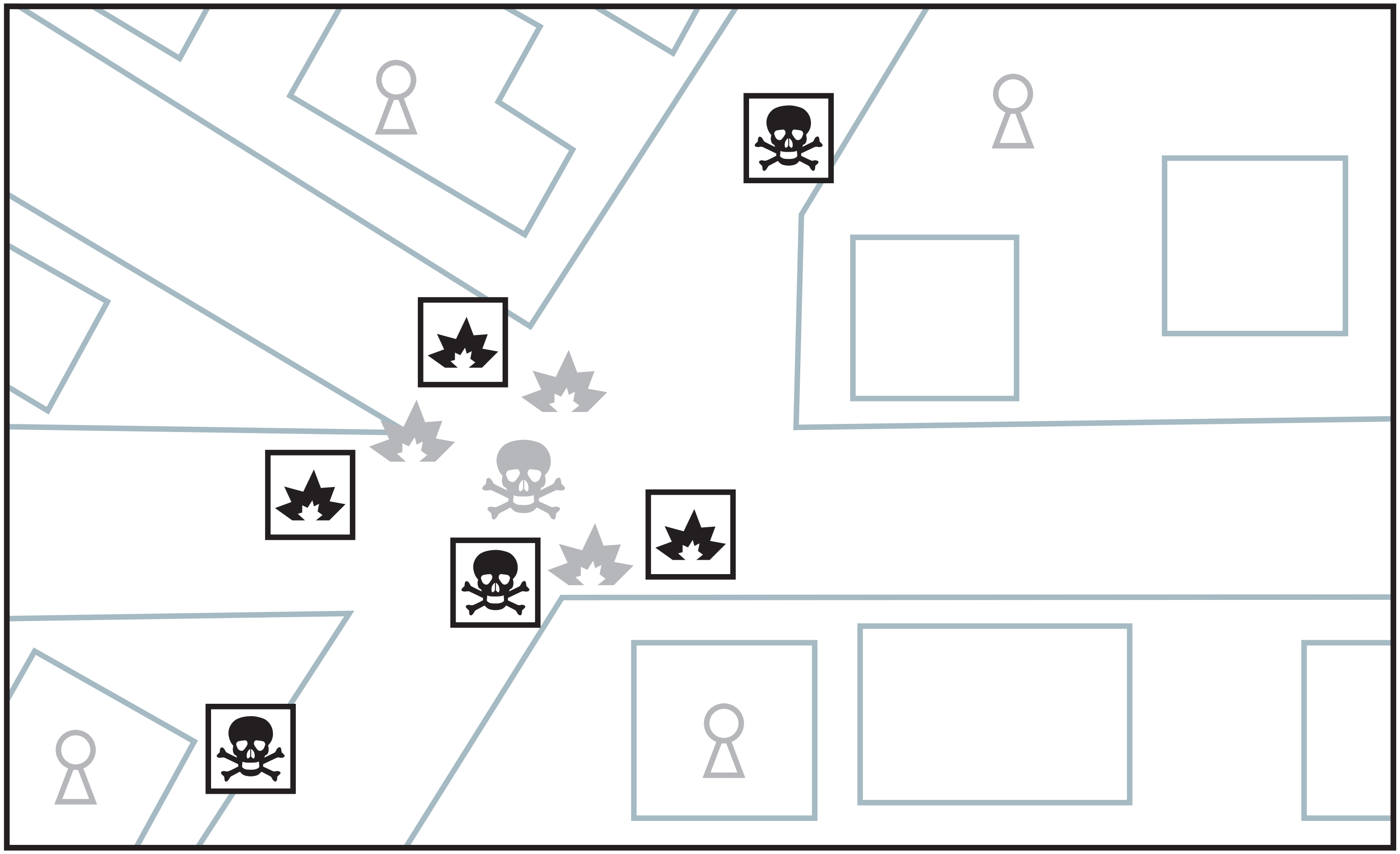
Response to a chemical/biological/radiological (CBR) incident:



An event is detected and reported by permanently placed sensors

Motivating Scenario: CBR Response

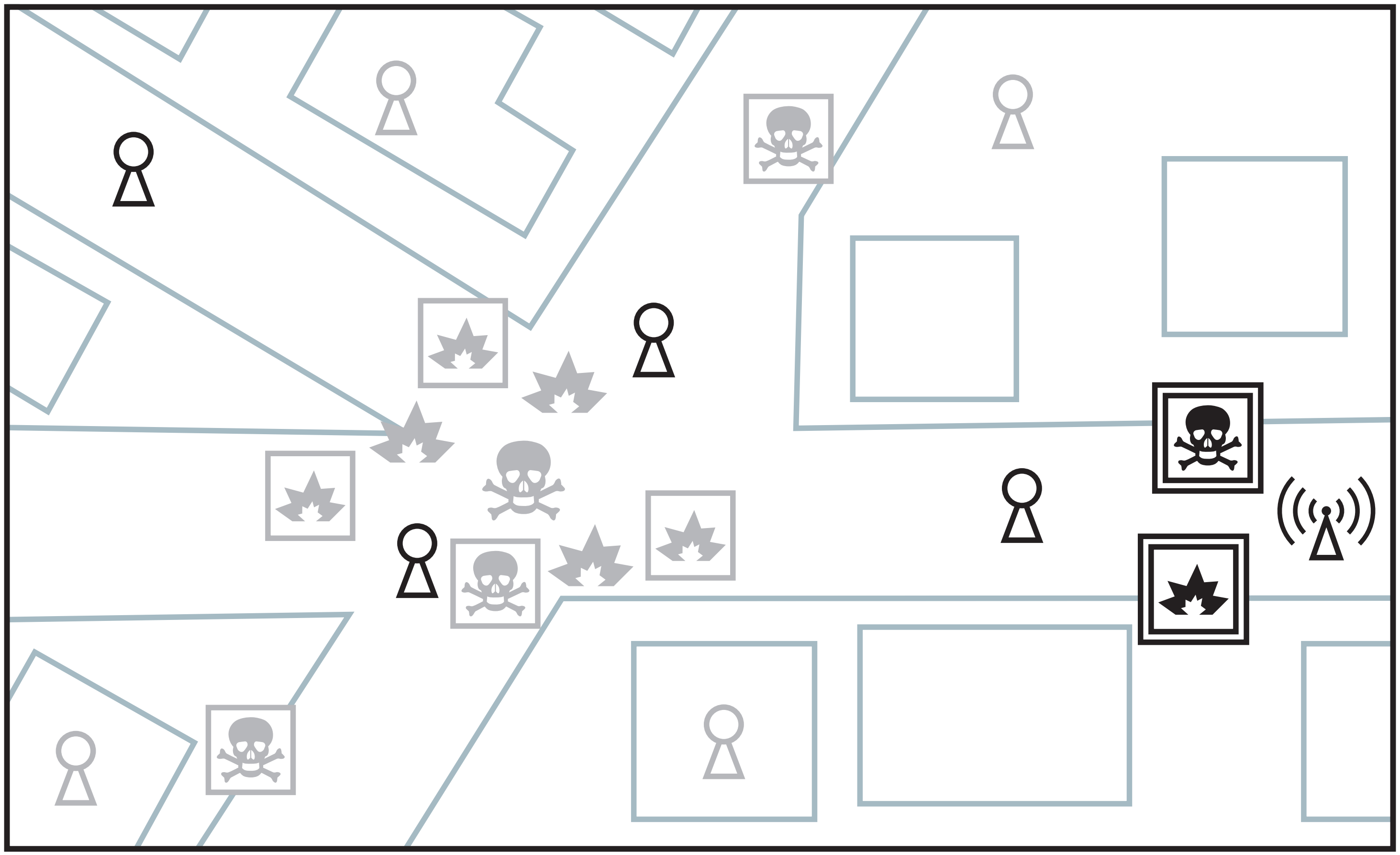
Response to a chemical/biological/radiological (CBR) incident:



Response teams deploy, integrate with on-site sensors, and collaborate

Motivating Scenario: CBR Response

Response to a chemical/biological/radiological (CBR) incident:



More sensors and backhaul are established, command & support units deploy

Key Scenario Points

Realistic, desired functionality for the near term future

- Based on deployed projects of Drexel's Secure Wireless Agent Testbed (SWAT) laboratory and discussions with first responder organizations

Two primary challenge areas:

- Integrating very heterogenous sets of software & hardware
    - Variety of data, vendors, authorities, versions, hosts, apps
- Operating on constrained, disrupted wireless networks
    - Particularly peer-to-peer mobile ad hoc networks (MANETs)
        - Focusing on PDAs, tablets, robots, some fixed sensors
    - Limited bandwidth, frequent link disconnects

Need to support rapid, automatic, on-site communications integration

Overview

OntoNet challenges and contributions:

- *Integrating very heterogenous sets of software & hardware*
  - Adapting Semantic Web concepts to networking problems
    - Using practical subset of Ontology Web Language (OWL)
- *Operating on constrained, disrupted wireless networks*
  - Novel hybrid tree-mesh protocol for expressive multicast

Current focus: Multicast messaging using expressive address scheme

- Messages are tagged with descriptions of contents
- Destinations register queries for desired messages
- Background ontologies are shared on- and off-line
- All written in simple declarative language

Notable out-of-scope topic: Ontology and schema integration

- Authors have previous work in this area, but not addressed here

Related Work

Several notable, similar projects and areas exist:

Area/System	Addresses	Concern	Delivery Model	Routers
MANET Routing	IP labels	Inflexible	$1 \rightarrow n$	Peer to peer
Basic Pub-Sub	URI labels	Inflexible	$1 \rightarrow n$	Infrastructure
Expressive Pub-Sub	XQuery	No implicit data	$1 \rightarrow n$	Infrastructure
DHTs	Labels	Inflexible; partial match inefficient	$n \rightarrow n$	Peer to peer; best with many, many nodes
INS	Attributes	Ad hoc, no implicit data	$n \rightarrow n$	Peer to peer, much state on broker nodes
GSD	OWL	Computational complexity	$n \rightarrow n$	Peer to peer, iterative flood, label-based aggregation

Message Model and Language

OntoNet multicast model has three components:

- **Each message m is associated with object m' and description d**
- **Each destination process p is associated with at least one query q**
- **There are known or retrievable background ontologies B**

With $dest$ relating messages to destinations, the model is:

$$\forall\,(m,m',d) \in M,\ (p,q) \in D$$

$$\left[d \bigwedge_{b \in B} b \models q(m')\right] \Rightarrow (m,p) \in dest$$

Message Model and Language

OntoNet multicast model has three components:

- **Each message m is associated with object m' and description d**
- **Each destination process p is associated with at least one query q**
- **There are known or retrievable background ontologies B**

With $dest$ relating messages to destinations, the model is:

$$\forall\,(m, m', d) \in M,\ (p, q) \in D$$
$$\left[d \bigwedge_{b \in B} b \models q(m') \right] \Rightarrow (m, p) \in dest$$

OntoNet applies description logic to define that entailment

- **A lightweight subset of the Ontology Web Language (OWL) is used**
  - **Favorable computational results, practical usability**
- **OWL is a product of the Semantic Web Community**
  - **Aims to integrate disparate data on the World Wide Web**
  - **As well as make it accessible to machine reasoning**

Sample Message Descriptions

A GPS position report from squad leader Joe:

```
<gis:GPSUpdate>
  <gis:unit>
    <role:SquadLeader rdf:about="#Joe">
      <role:squad rdf:resource="#Squad4" />
      <role:platoon rdf:resource="#AlphaPlatoon" />
    </role:SquadLeader>
  </gis:unit>
</gis:GPSUpdate>
```

A localized position report from squad leader Boon:

```
<gis:LocalizedUpdate>
  <gis:unit>
    <role:SquadLeader rdf:about="#Boon">
      <role:squad rdf:resource="#Squad1" />
      <role:platoon rdf:resource="#AlphaPlatoon" />
    </role:SquadLeader>
  </gis:unit>
</gis:LocalizedUpdate>
```

Sample Query

A query for position updates, GPS or otherwise, from squad leaders, in OWL:

```
<owl:Class rdf:about="#Query">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="&gis;#PositionUpdate" />

    <owl:Restriction>
      <owl:onProperty rdf:resource="&gis:unit" />
      <owl:someValuesFrom rdf:resource="&role;#SquadLeader" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

In abstract description logic notation:

$$\textbf{Query} \equiv \textbf{gis:PositionUpdate} \sqcap \exists\, \textbf{gis:unit}.\textbf{role:SquadLeader}$$

The equivalent query in first order logic:

$$\forall x\, \exists y \;\; [\textbf{gis:PositionUpdate}(x) \wedge \textbf{gis:unit}(x,y) \wedge \textbf{role:SquadLeader}(y)] \supset \textbf{Query}(x)$$

The Query Logic

Query should match both messages

- **But neither is explicitly labeled gis:PositionUpdate**

Background ontologies fill in implied information

- **Assumed to be relatively stable, but extended over time**
- **Shared a priori & pulled a la active network capsule code (future work)**
- **Simple facts needed for this query:**

$$\text{gis:LocalizedUpdate} \subseteq \text{gis:PositionUpdate}$$

$$\text{gis:GPSUpdate} \subseteq \text{gis:PositionUpdate}$$

Result: The query matches both messages

- **Such inference, though simple, aids system evolution and deployment**
- **Has a learning curve, but makes it easy to integrate new components**

Sample Message Description 2


```
<message:Message>
  <message:sender>
    <role:SquadLeader rdf:about="#Joe">
      <role:squad rdf:resource="#Squad2" />
      <role:platoon rdf:resourc="#PlatoonC">
    </role:SquadLeader>
  </message:sender>
  <message:deliveryModel>
    <message:Reliable>
      <message:window>
        <message:Duration>
          <message:minutes>20</message:minutes>
        </message:Duration>
      </message:window>
    </message:Reliable>
  </message:deliveryModel>
  <message:content>
    <int:OverheadImageRequest>
      <int:location>
        <gps:Coordinate>
          <gps:lat>xx.xxxxxx</gps:lat>
          <gps:lon>yy.yyyyyy</gps:lon>
        </gps:Coordinate>
      </int:Location>
      <image:resolution rdf:resource="&image;#Meter" />
    </int:OverheadImageRequest>
  </message:content>
</message:Message>
```

Queries & descriptions may be large
$\Rightarrow$ Propagation is a challenge

Message Propagation

Two main approaches in paper:

- **Naive scheme: Flood descriptions, perform local query matching**
- **OntoNet: Partition network to constrain state and traffic generated**

Naive, Flooding-Based Scheme

Simple, baseline approach for comparision:

- Flood queries throughout network
- Local query matching for each message
- Unicast forward to destinations

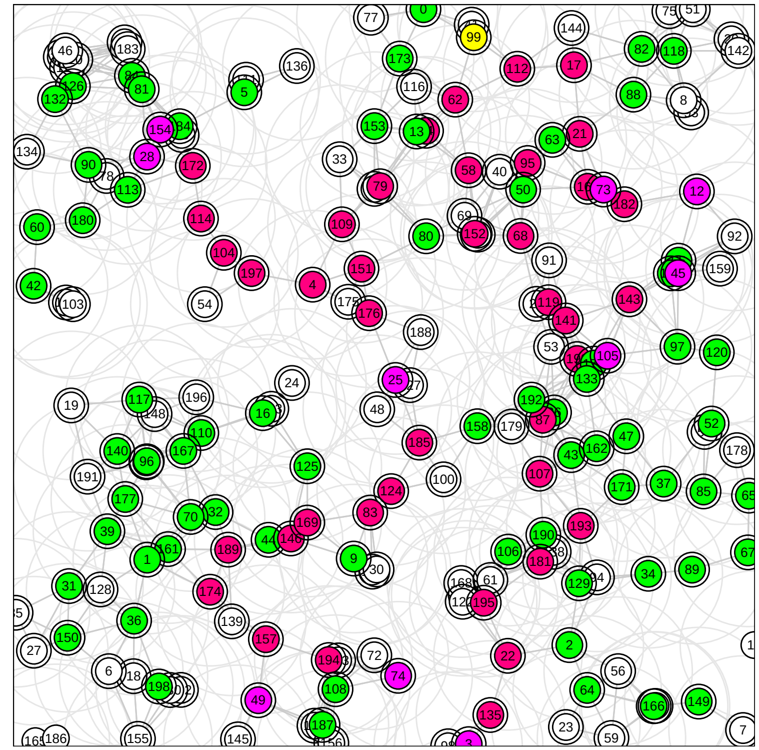
Flooding done via multi-point relays (MPRs)

- Well studied in MANET community
- Reduces amount of traffic transmitted
- Especially efficient at high densities

But, basic problems remain:

- Lots of state stored throughout network
- Lots of traffic maintaining registrations

Simulation Visualization



Legend:

- Green: MPRs
- Red: Unicast routes
- Purple: Destinations
- Yellow: Message generator

OntoNet: Hopefully Less Naive...

OntoNet employs a hybrid tree-mesh protocol:

- Construct set of trees
  - Each a small subset of nodes
- Create mesh overlay among tree roots
- Propagate queries up trees
- Forward messages across mesh

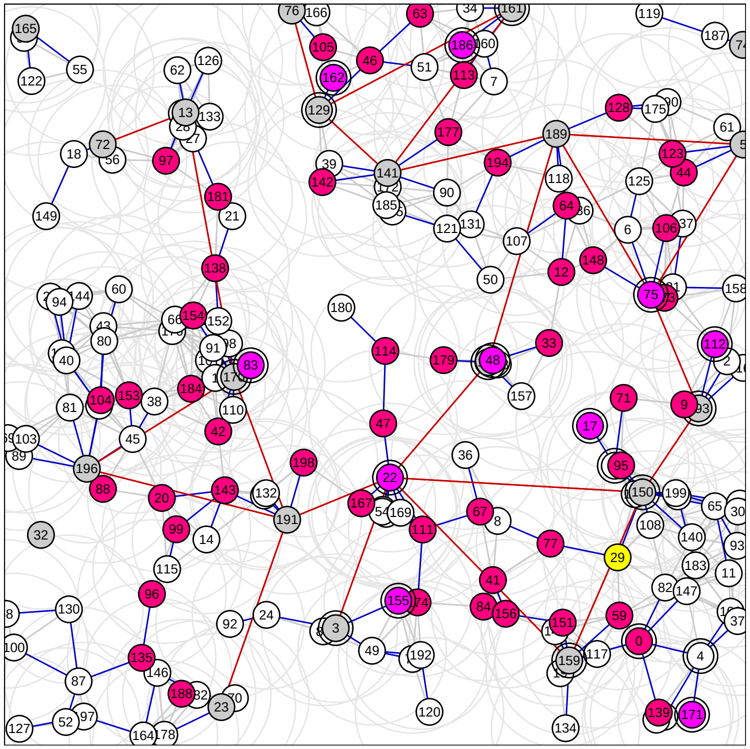
Basic motivations:

- Trees partition network into regions
  - Constrain query propagation
- Mesh connects partitions together
  - Forward messages across regions

Future work motivation:

- Trees well suited for aggregation
  - I.e., multi-query optimization

Simulation Visualization

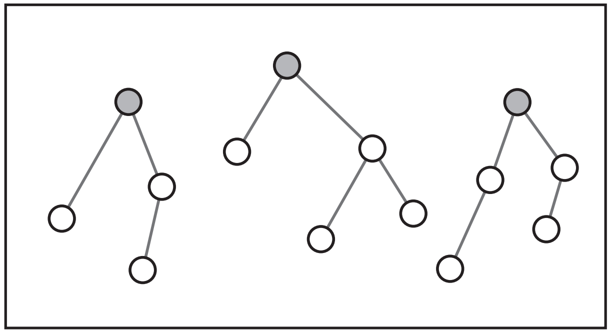


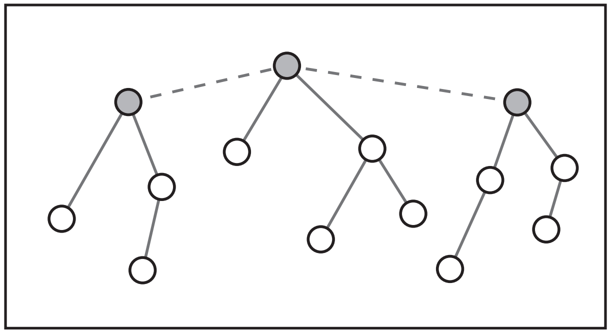
Legend:

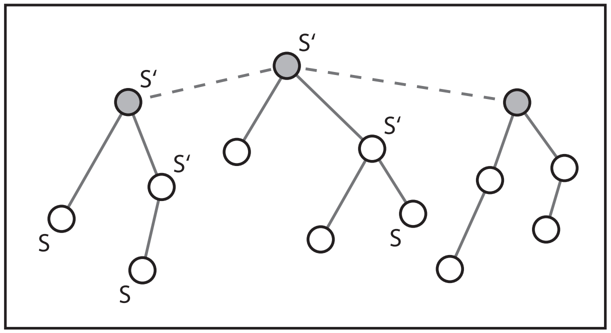
- Grey: Root
- Red: Unicast routes
- Purple: Destinations
- Yellow: Message generator

Tree & Mesh Formation

Phase 1: Partition network into trees, create overlay mesh on roots







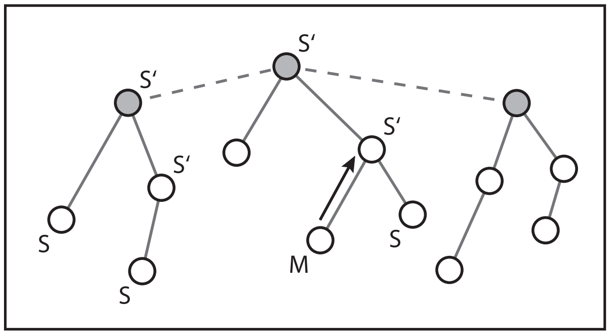
• Some nodes self-elect to be beacons
• Other nodes join trees rooted at beacons

• Nodes eavesdrop on adjacent trees
• Pass neighbor root information upward

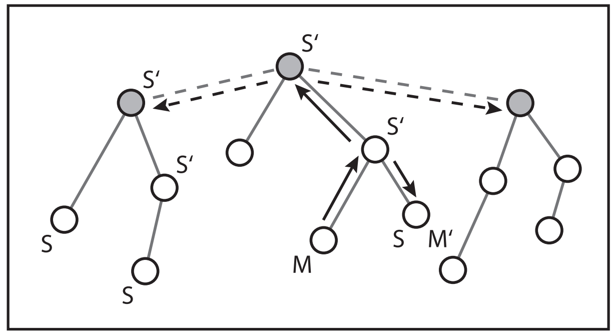
• Destination queries are also passed upward
• Possibly aggregated along the way.

Message Propagation

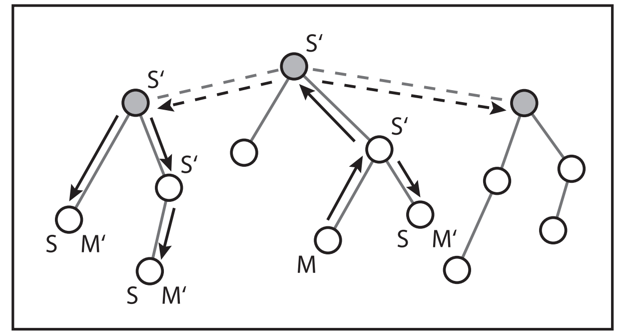
Phase 2: Forward messages up trees, across mesh, and down to destinations



• Messages and their descriptions are created by applications, propagated up tree



• Descriptions are matched against queries
• Delivered locally & forwarded downward if descendants might match
• Root nodes forward to neighboring roots



• Roots propagate flood over root overlay mesh
• Messages forwarded if previously unseen, skipping previous root
• Roots forward down their tree if descendants might match

Preliminary Results

Conducting evaluations in NS-3 (`http://nsnam.org`)

- **An NSF-primed, open source initiative for a new network simulator**
- **Clean slate, all-C++ design incorporating IP stacks, mobility, etc**
- **Heavy focus on software engineering for extensible architecture**
- **This is one of the first publications using NS-3**

Preliminary Results

Conducting evaluations in NS-3 (`http://nsnam.org`)

- **An NSF-primed, open source initiative for a new network simulator**
- **Clean slate, all-C++ design incorporating IP stacks, mobility, etc**
- **Heavy focus on software engineering for extensible architecture**
- **This is one of the first publications using NS-3**

Initial evaluation shows favorable performance versus naive scheme

- **Scales well, up to thousands of nodes (data in paper)**
- **Total bytes generated grows slowly due to partitioning of network**
  - **Notable overhead for maintaining network partitions, registrations**
  - **But worthwhile if messages, descripions, or queries are large**
- **Loading remains fair, no nodes handling excessive amounts of traffic**

Preliminary Results

Conducting evaluations in NS-3 (http://nsnam.org)

- An NSF-primed, open source initiative for a new network simulator
- Clean slate, all-C++ design incorporating IP stacks, mobility, etc
- Heavy focus on software engineering for extensible architecture
- This is one of the first publications using NS-3

Initial evaluation shows favorable performance versus naive scheme

- Scales well, up to thousands of nodes (data in paper)
- Total bytes generated grows slowly due to partitioning of network
  - Notable overhead for maintaining network partitions, registrations
  - But worthwhile if messages, descripions, or queries are large
- Loading remains fair, no nodes handling excessive amounts of traffic

But must investigate robustness of tree maintenance

- Note: Unlike many MANET multicast protocols, OntoNet does not maintain structure across entire network
- OntoNet partition trees are small, with nearby alternatives
- Easy to repair, limited damage if tree breaks

Future Work

Aggregation policies, reasoning mechanisms are next major step

- **Can apply INS-style memory structures, but need further reductions**
- **Major motivation for using OWL/description logic**
  - **Least Common Subsumer (LCS) inference provides formal semantics for multiquery optimization of registrations**
- **Need intelligent, adaptive policies to apply aggregation**
  - **Tradeoff message false positives versus state, query bandwidth**
- **Effectiveness largely dependent on richness of services, ontology?**

Future Work

Aggregation policies, reasoning mechanisms are next major step

- Can apply INS-style memory structures, but need further reductions
- Major motivation for using OWL/description logic
  - Least Common Subsumer (LCS) inference provides formal semantics for multiquery optimization of registrations
- Need intelligent, adaptive policies to apply aggregation
  - Tradeoff message false positives versus state, query bandwidth
- Effectiveness largely dependent on richness of services, ontology?

Further investigation of delivery models

- Matching semantics has effect on message type, extensability
  - Destination queries: Broadcast to any interested application
  - Message queries: Impose requirements, i.e., for service request
- Would like to support exactly-one, expressive anycast

Future Work

Aggregation policies, reasoning mechanisms are next major step

- Can apply INS-style memory structures, but need further reductions
- Major motivation for using OWL/description logic
  - Least Common Subsumer (LCS) inference provides formal semantics for multiquery optimization of registrations
- Need intelligent, adaptive policies to apply aggregation
  - Tradeoff message false positives versus state, query bandwidth
- Effectiveness largely dependent on richness of services, ontology?

Further investigation of delivery models

- Matching semantics has effect on message type, extensability
  - Destination queries: Broadcast to any interested application
  - Message queries: Impose requirements, i.e., for service request
- Would like to support exactly-one, expressive anycast

Live evaluation, complementing mobility studies in ns-3

- Using Drexel's SWAT laboratory testbeds

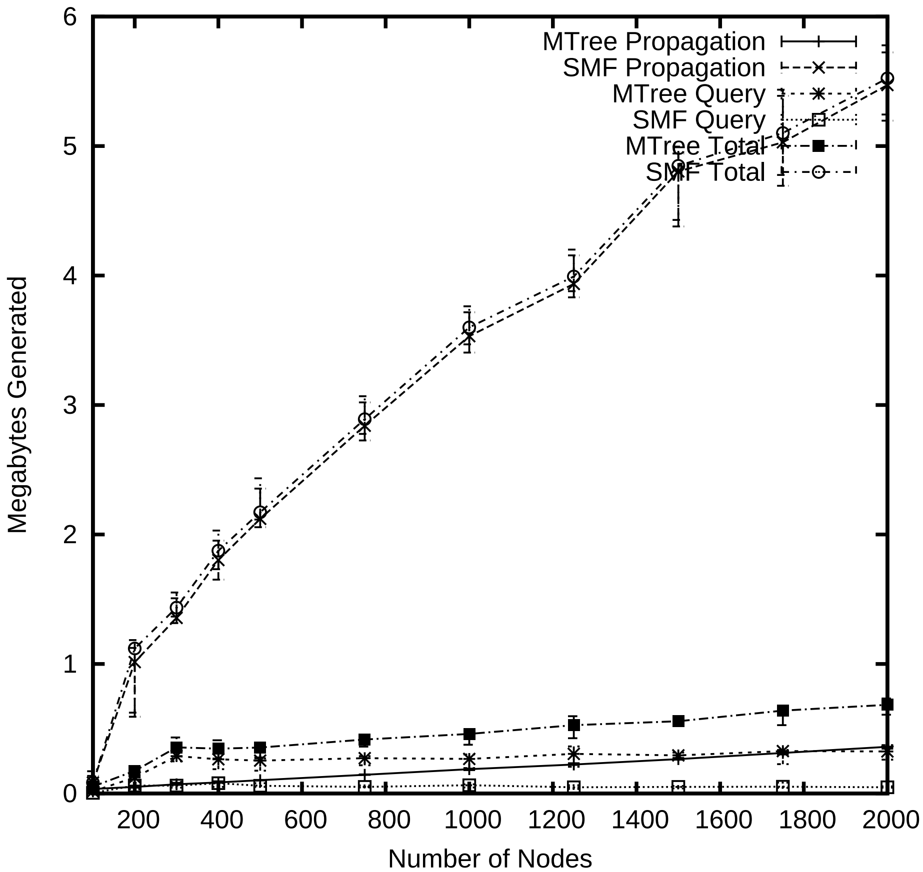
Wrap-Up

Contact:

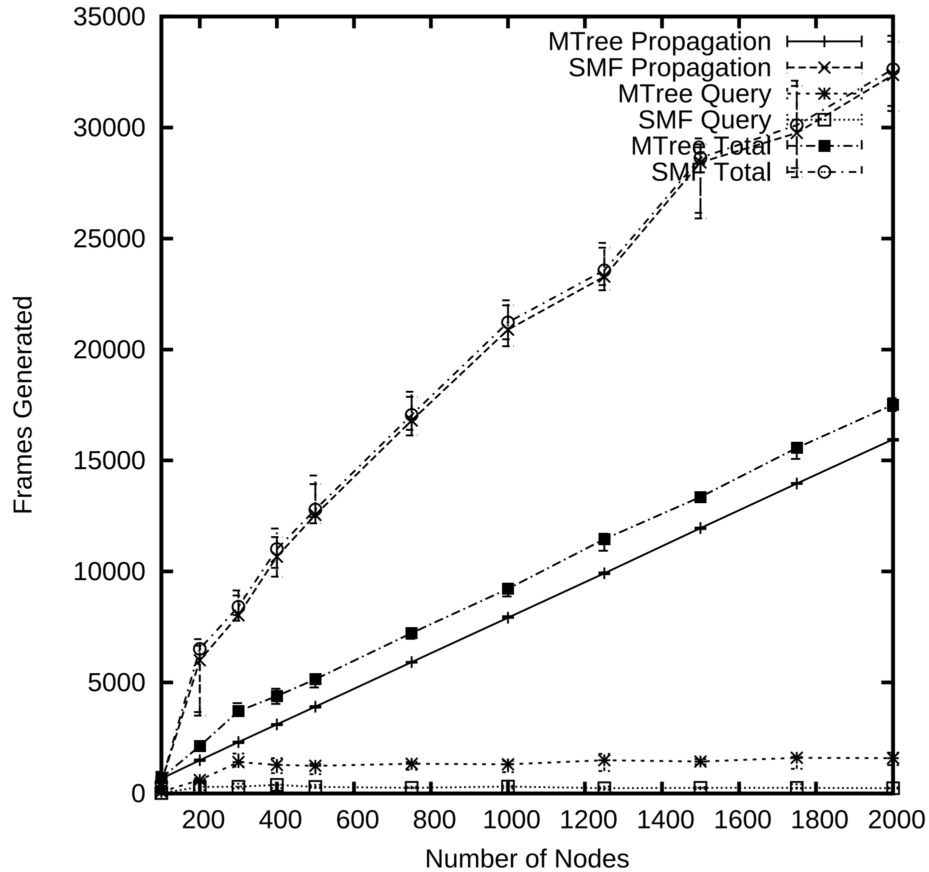
Joseph B. Kopena
tjkopena@cs.drexel.edu

Boon Thau Loo
boonloo@cis.upenn.edu

Preliminary Results: Traffic Volume

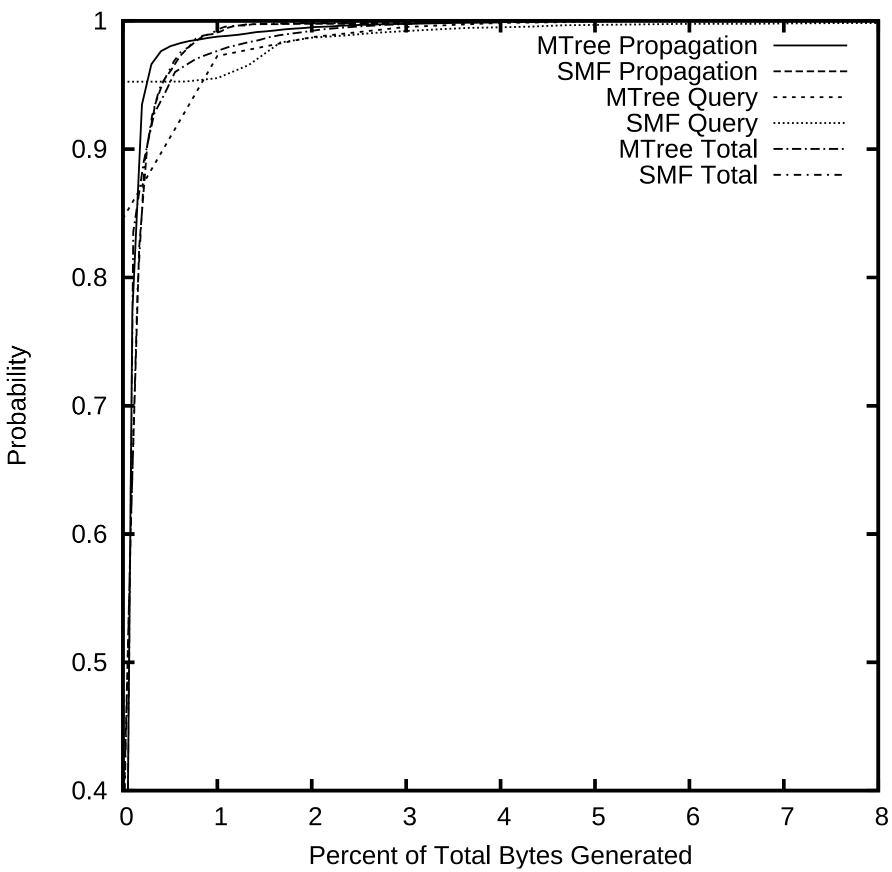


Total megabytes of traffic, not including packet headers.

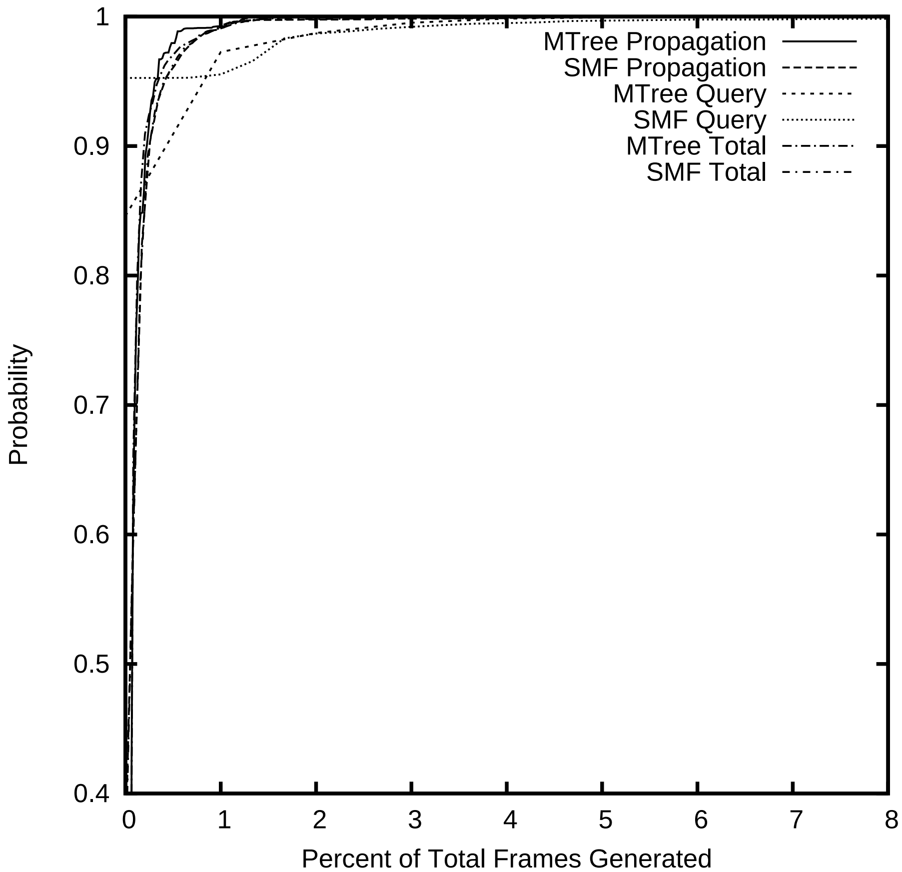


Total number of frames transmitted.

Preliminary Results: Traffic Loading



Loading on each node in terms of bytes.



Loading on each node in terms of transmissions.