OntoNet: Scalable Knowledge-Based Networking

Joseph B. Kopena
Drexel University

Boon Thau Loo
University of Pennsylvania

Abstract

Recent years have seen a proliferation of work on the *Semantic Web*, an initiative to enable intelligent agents to reason about and utilize World Wide Web content and services. Concurrently, the networking community has developed a concept of the *knowledge plane*, using artificial intelligence to reason about and manage network behavior. These two efforts have progressed independently despite potential synergies. This paper presents early work on *OntoNet*, a knowledge-based middleware which aims to integrate those visions and provide flexible, scalable *knowledge-based networking* with ontologies. We focus on supporting multicast messaging in mobile ad-hoc networks using description logic advertisements and requests in a subset of the Web Ontology Language (OWL). We explore a novel hybrid tree-mesh protocol which enables efficient and robust propagation of potentially verbose queries and descriptions. We demonstrate the potential viability of this protocol via simulations in the newly developed NS-3 simulator.

1 Introduction

In this paper, we explore *OntoNet*, a scalable knowledge-based middleware that supports content-based reasoning and routing in networked and mobile systems. *OntoNet*'s usage scenarios are wide-ranging, including logistics management, infrastructure monitoring, and ubiquitous computing. We are primarily motivated by real-world systems for first-responders to emergency incidents such as chemical, biological, and radiological hazards. In these settings, many teams from a variety of organizations are rapidly deployed for managing a wide variety of emergencies, including transportation accidents, industrial mishaps, and natural or intentional disasters. Current solutions have limited interoperability capabilities, due to the reliance on manual processes and proprietary tools. These emergency scenarios require real-world systems with sensing capabilities and mobility, devices that are rapidly deployed, yet incrementally extended as new software versions, data types and devices are added over time.

To tackle these challenges, we argue for *automated* networked systems to enable emergency-response teams to perform more effectively and safely. Such systems require significant interaction with the real-world, including a wide variety of fixed, portable and mobile sensors, as well as robots and tele-operated effectors for probing, cleanup, and other tasks. They would also need to incorporate human decision making, collaboration, and situational awareness tools.

The central tenet of our work is that many of the major challenges of such real-world systems—including *scalability*, *robustness*, *interoperability*, and *system design*—may be at least partially addressed through *knowledge-based* communications middleware. *OntoNet* draws several techniques on knowledge representation and reasoning from the *Semantic Web* [4], which was originally proposed as an extension to the World Wide Web. In the Semantic Web, intelligent software agents automatically extract information from and reason about Web content, services, and their relationships with each other. This work is deeply rooted in research on knowledge representation in the artificial intelligence community. Its central promise is flexible, autonomous software with sophisticated abilities to self-organize and integrate. While that remains a goal to be achieved, component technologies such as XML, the Resource Description Framework (RDF) [8], and the Web Ontology Language (OWL) [6], are rapidly gaining adoption for organizing and reasoning on Web content.

In *OntoNet*'s knowledge-based framework, software processes, messages, hosts, the environment, and other components are represented and reasoned on using declarative, formal languages and logical inference to provide capabilities such as service discovery and composition, content-based messaging, and distributed querying. Specifically, this paper makes the following contributions:

Language: We propose the *OWL-Net* language, a description logic based on a subset of OWL, and demonstrate via examples that *OWL-Net* aims to achieve a sweet spot between expressiveness, ease-of-use, and efficient reasoning. By decoupling system components via capabilities described in *OWL-Net*, we enable flexible, autonomous software with sophisticated abilities to self-organize and integrate. Design and implementation is eased through adoption of modular architectures. In addition, interoperability is supported by encapsulation and formal interface specifications.

Scalable propagation and routing: We propose a novel hybrid tree-mesh protocol for decentralized distribution and binding of *OWL-Net* advertisements and requests based on the multicast delivery model. To reduce communication overhead, we propose multi-query optimizations, in which formal reasoning that capitalizes on the nature of description logic semantics as a mechanism towards aggregating descriptions, resulting in decreased forwarding state and control traffic.

Evaluation: We demonstrate the viability of the tree-mesh protocol via simulations in NS-3[1]. Our paper

[1] An ongoing major revision of NS-2; `http://www.nsnam.org/`.

presents some of the first published experiments that utilizes NS-3.

Our work should be viewed as small steps toward integrating the Semantic Web and the knowledge plane and are not "drop-in" solutions to enabling either. In the long run, we hope that the continued exploration of this work will result richer naming, querying, and reasoning in knowledge-based networks.

2 Motivating Scenario

To motivate *OntoNet*, we first present an emergency-response scenario, followed by an outline of other possible scenarios.

2.1 Emergency-response networks

Many organizations maintain response teams for incidents involving chemical, biological, or radiological (CBR) hazards. Scenarios include industrial disasters, transport accidents, and intentional acts. Unfortunately, current best practices largely use non-integrated and redundant tools applied via manual processes. Even among well provisioned teams, critical data collection is frequently performed via clipboard and pen—a difficult task in a full protective suit. The few electronic sensor interfaces in use as a rule are rudimentary and based on stovepiped mechanisms. These deficiencies hinder rapid deployment, data collection, decision making, cost, and safety.

Figure 1 depicts a CBR response scenario using knowledge based networking. In Figure 1a, permanent sensors detect an accident creating a CBR hazard and generate an alert. Response teams arriving in Figure 1b automatically integrate feeds from the permanent sensors with their personal sensors and displays. CBR specialists deploy more sensor stations as they investigate with mobile sensors. Eventually, operations bases and a network uplink are established in Figure 1c. Importantly, from the start all of the sensor stations and personnel form a mesh network enabling communication over low power, low cost radios with better coverage and mobility than typical single hop or star topologies. A diverse set of information is produced and consumed, including command, sensor, and background data, as shown in Figure 1d. Many generations of equipment are present, from sensors placed years previously to a mix of modern and dated equipment fielded by response teams, hence motivating the need for mechanisms to achieve inter-operability. In addition, the system has to operate from the moment the first personnel arrive to deployment of operations bases, uplinks, and many more personnel and sensors. Both mobile and fixed nodes exist; in addition, teams must be able to operate independently if disconnected, but use infrastructure if available.

2.2 Other Scenarios

Besides emergency response, other potential scenarios involve monitoring buildings, factories, airport controls, ubiquitous computing environments, and logistics management. To illustrate,a corporation may own a large fleet of taxi cars and vans in a metropolitan area. Those vehicles may be equipped with a variety of sensors for monitoring position, engine conditions, wear and tear, and other factors. Each of those sensors may generate data to be processed and transformed by a variety of software throughout the enterprise. The vehicle may also have interfaces for the driver and passengers to present maps, weather, local attractions, and other information. Enterprise-level data stores and query services must be provided and accessible to support those displays.

Further, *OntoNet* enables several less traditional communication patterns. Fleet garages and bridge or highway tolls may generate queries for credentials which must be routed to the appropriate authentication agent on the vehicle. Messages may also be generated by authorities within the corporation which must be delivered to specific groups of staff or vehicle types. More immediately, and less easily and efficiently supported by current systems, drivers may generate messages such as notifications to all taxis in a neighborhood that a crowd has been observed and more vehicles are needed. Another message would be to request a handicapped accessible taxi at a given location, which must then be delivered to an appropriate vehicle under some optimization parameters, such as proximity and availability.

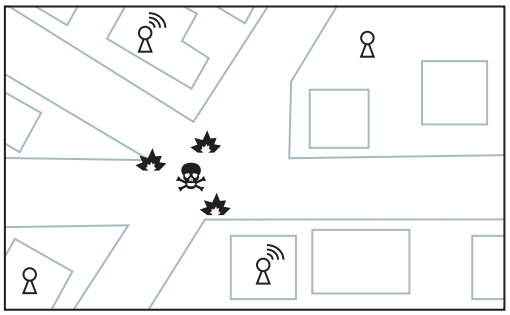
3 Service Model and Language

In this section, we present *OntoNet*'s language and service model. *OntoNet*'s message delivery model is based on matching *descriptions* with *queries*, using either *multicast* and *anycast* routing. First, we assume that all *OntoNet* nodes are pre-initialized (either via flood or installed out-of-band) *ontologies* which can be used for reasoning and routing. At a high-level, ontologies represent concepts within a domain, and relationships between these concepts. Ontologies are used for reasoning about objects within a domain. Unlike descriptions and queries which are dynamic, ontologies are relatively static and hence can be pre-propagated.

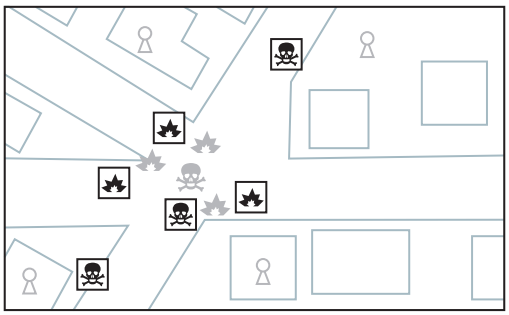
Descriptions are matched with queries using a publish/subscribe (pub/sub) model. In the *query-centric* approach, nodes publish descriptions describing their services, and messages containing queries are routed to all nodes (for multicast) that matches the description, or the best node (for anycast) based on metrics defined as part of the description. As an alternative, in the *data-centric* approach, nodes register continuous queries in *OntoNet*, and messages containing the descriptions are routed to nodes that satisfies the queries. For ease of exposition, we will describe the examples and routing strategies in terms of the *data-centric* approach. We further limit our discussion on multicast delivery model.

3.1 *OWL-Net* by Examples

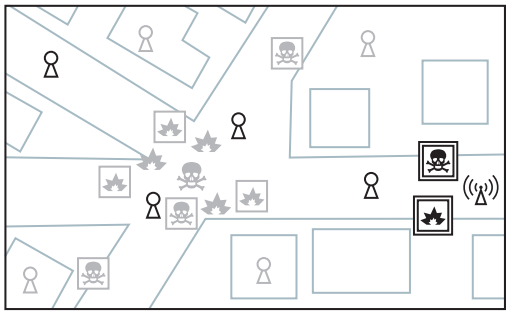
We present *OWL-Net*, which is our initial language design, using examples from the emergency-response scenario from Section 2.1. In this scenario the basic ser-



(a) An accident occurs involving CBR components. Permanent sensors detect the incident and transmit alerts over long range links.



(b) Firefighting personnel deploy in small teams and are warned of the CBR danger by the sensors. CBR specialists deploy as fires diminish.



(c) CBR specialists deploy portable sensors in addition to their mobile devices. Operations bases and a backhaul link are established.

Sensor Data
Chromatography; Mass & IR Spectrometry; Radiation Rate and Energy Levels; Photo and Flame Ionization; Gas Detectors; Weather

Planning and Execution
Diagrams; Text & Voice Messages; GPS; Automatic Alerts

Device and Personnel Capabilities
Physical Configurations; Sensor Types; Temporal & Spatial Resolution; Training; Tools & Equipment; Organization; IDs

Data Repositories
Simulation Results; Maps; Blueprints; Medical and Materials Libraries; Intelligence; Medical Data & Records

(d) Some of the information types produced and consumed by a wide variety of hardware, software, and organizations in the scenario.

Figure 1: Major elements of a conceptual response to a chemical, biological, or radiological (CBR) incident.

vice required is multicast dissemination of messages using the *data-centric* approach. message dissemination. In our example scenario, a spectrometer sensor periodically publishing findings for delivery to displays, analyzers, and archives. Publishers annotate messages with *descriptions*, which is the metadata describing the content, source, and other properties. Messages are then routed to all reachable nodes which have registered a *query* matching that description.

Figure 2 depicts an OWL/RDF description for a typical message in this scenario[2]. It declares that the message contains sensor data, the source is a spectrometer deployed by the NEAir-GM organization, contents are in the IEEE/ANSI N42.42 radiation data format[3], and the sensor has detected a particular nuclide[4]. Notably, this markup does not replace message contents or formats, such as SensorML, N42.42, CAP, or DoD CBRN in this scenario. Instead it captures metadata and key content, rendering datasets, images, and other opaque messages accessible to general inference.

To receive messages, nodes register queries defined using OWL class expressions. Figure 3 gives an example that might be registered by an analysis package or hand-

```
<msg:Message rdf:about="#MSG74">
  <msg:source>
    <cbr:Sensor rdf:about="#Sensor43UX">
      <rdf:type rdf:resource="&cbr;#Spectrometer" />
      <rdf:type rdf:resource="&cbr;#Fixed" />

      <msg:org rdf:resource="&orgs;#NEAir-GM" />
    </cbr:Sensor>
  </msg:source>

  <msg:format rdf:resource="&nist-ansi-n4242xsd;#"/>

  <cbr:nuclideDetected>
    <cbr:SuspiciousNuclide>
      <cbr:name>Am-241</cbr:name>
      <cbr:confidence>98.0</cbr:confidence>
    </cbr:SuspiciousNuclide>
  </cbr:nuclideDetected>
</msg:Message>
```

Figure 2: OWL/RDF description of a message containing spectometry data, noting a particular finding.

held display to receive messages such as in Figure 2. The class consists of all objects with a source that belongs to the NEAir organization and is reporting a suspicious nuclide using the ANSI N42.42 format. Delivering messages requires forwarders to apply background ontologies and OWL semantics to match descriptions against queries. Those ontologies define domain classes and properties as well as description format elements such

[2]Namespace declarations, ontology imports, and `rdf:RDF` wrappers have been removed from these examples for clarity.

[3]`http://standards.ieee.org/getN42/`

[4]This example is based on `http://units.nist.gov/Divisions/Div846/Gp4/ANSIN4242/2005/annexC.n42`

```
<owl:Class rdf:about="#Query">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource="&msg;#source" />
      <owl:someValuesFrom>
        <owl:Restriction>
          <owl:onProperty rdf:resource="&msg;#org"/>
          <owl:someValuesFrom
            rdf:resource="&orgs;#NEAir" />
        </owl:Restriction>
      <owl:someValuesFrom>
    </owl:Restriction>

    <owl:Restriction>
      <owl:onProperty
          rdf:resource="&cbr;#nuclideDetected" />
      <owl:someValuesFrom
          rdf:resource="&cbr;#SuspiciousNuclide" />
    </owl:Restriction>

    <owl:Restriction>
      <owl:onProperty
          rdf:resource="&msg;#format" />
      <owl:hasValue
          rdf:resource="&nist-ansi-n4242xsd;#" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Figure 3: OWL query for reports of suspicious nuclides in N42.42 format from a particular organization.

as `msg:Message` and `msg:format`. Domain specific knowledge in these examples includes the taxonomy of sensors and properties such as `cbr:nuclideDetected`. For example, matching Figure 2 and Figure 3 requires knowing that the `orgs:NEAir-GM` object is an instance of the `orgs:NEAir` class.

Figure 4 contains a description from an updated or different model of sensor using extended ontologies. To match Figure 3, inferences must be made from those ontologies, particularly the excerpts in Figure 5. Forwarders must reason about subclasses, such as `cbr2:Am-241` implies `cbr:SuspiciousNuclide`, apply the `orgs:NEAir-Haz` object's membership in `orgs:NEAir`, and infer a value for the `msg:format` property based on explicit membership in `cbr2:N4242Spectrometry`. This inference and deduction of implicit information promotes interoperability, evolution, and byte savings. It also differentiates the ontological, knowledge-based approach from relational database, XML-based pub/sub systems [7], and other data-oriented networks [3].

3.2 Formal Model and Semantics of *OWL-Net*

The *OWL-Net* language aims to achieve a sweet-spot between simplicity, expressiveness, and ease-of-computation. *OWL-Net* is simple compared to the full OWL specification, but very expressive as a network addressing scheme and sufficient to greatly aid application development and extension. Importantly, it is also decidable and tractable, with subsumption polynomial in the combined size of the axioms involved [1].

```
<cbr2:N4242Spectometry rdf:about="#MSG1134">
  <msg:source>
    <cbr2:Fixed-FT-IR rdf:about="#Sensor03NG">
      <msg:org rdf:resource="&orgs;#NEAir-Haz" />
    </cbr2:Fixed-FT-IR>
  </msg:source>

  <cbr:nuclideDetected>
    <cbr2:Am-241>
      <cbr:confidence>93.0</cbr:confidence>
      <cbr2:concentration>0.002</cbr2:concentration>
    </cbr2:Am-241>
  </cbr:nuclideDetected>
</cbr2:Spectometry>
```

Figure 4: Description for a report similar to Figure 2.

```
<owl:Class rdf:about="&cbr2;#N4242Spectrometry">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="&msg;#Message" />

    <owl:Restriction>
      <owl:onProperty rdf:resource="&msg;#format" />
      <owl:hasValue
        rdf:resource="&nist-ansi-n4242xsd;#" />
    </owl:Restriction>
  </owl:intersectionOf>
  <rdfs:subclassOf rdf:resource="">
</owl:Class>

<orgs:NEAir rdf:about="&orgs;#NEAir-Haz" />

<owl:Class rdf:about="&cbr2;#Am-241">
  <rdfs:subclassOf
    rdf:resource="&cbr;#SuspiciousNuclide" />
</owl:Class>
```

Figure 5: Excerpts of ontologies required to match the description in Figure 4 against the query in Figure 3.

While formal semantics are beyond the scope of this paper, we provide some high-level intuitions on the formal model and semantics of *OWL-Net* as a comparison to full OWL. Focusing on the *data-centric* multicast delivery model, each message m is associated with a message object m' and description d in relation M. Each destination process p is associated with at least one query q in the relation D. There is also a set of known or retrievable background ontologies B.

Using *dest* to denote which messages should be delivered to which nodes, the delivery model is then:

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

Entailment in that model is defined by the semantics of the logic used. OntoNet employs the description logic shown in Figure 6. It is a lightweight subset of OWL, using the constructs given in Figure 6a. Those syntactic elements are then formally defined within a slight extension of the description logic $\mathcal{EL}$ [1], shown in Figure 6b.

Basic Elements
- `owl:Class`
- `owl:Thing`
- `owl:Nothing`

Class Axioms
- `rdfs:subclassOf`
- `owl:intersectionOf`

Value Constraints
- `owl:hasValue`
- `owl:someValuesFrom`

(a) Permitted major OWL syntactic elements.

Name	Notation	Interpretation
Top Concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom Concept	$\bot$	$\emptyset$
Primitive Concept	A	$A^{\mathcal{I}}$
Nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Full Existential Restriction	$\exists \mathrm{r}.\mathrm{C}$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}} : (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
General Concept Inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

(b) Notation and interpretation of slightly extended $\mathcal{EL}$.

Figure 6: OWL constructs, abstract semantics, and description logic notation used in *OntoNet*.

4 Description and Message Propagation

To efficiently propagate descriptions and deliver messages, we present one possible solution, based on a novel hybrid tree-mesh protocol that ensures fault-tolerance, yet is amenable to efficient multicast and multi-query optimizations. The basic algorithm is shown in Figure 7. It has three conceptual steps: construction of localized trees; creation of an overlay mesh between adjacent trees; forwarding of messages across that mesh and delivery to hosts with matching destination processes.

4.1 Tree-Mesh Topology: One Possible Solution

Tree construction begins with nodes periodically self-electing to become beacons and regularly transmit tree construction messages. Upon receiving such a message, nodes join that tree by choosing the sender as their parent if they have no parent or it is closer to its root than their current parent. After selecting a parent, each node also begins transmission of tree construction messages. Included in the message are the identifiers of the node's root, distance to the root, and parent node.

Upon receiving a construction message from a tree other than its own and choosing not to join, nodes insert the root identifier into a set of overheard roots. That set is also included in each construction message. When a node receives a message from one of its children, it adds the received overheard root set into its own set. In this way, information about adjacent trees is propagated to the roots. The result is a mesh of tree partitions logically overlaid on the network, as in Figure 7a.

Query propagation: Registered queries are also included in tree construction messages. Nodes maintain a set of queries received from their children and include it in their messages. Along the way, descriptions are potentially aggregated as discussed in Section 4.2. Eventually all nodes have a picture of requests from their descendants, as in Figure 7b.

Message forwarding: As depicted in Figure 7c, nodes perform three actions upon receiving an application message to multicast: (1) Deliver to matching local processes; (2) Forward to their parent. (3) Forward to children with potentially matching local or descendant queries. Note that false positives may occur in step 3 due to aggregation.

Roots additionally unicast application messages received from their children to all adjacent roots. In this case, the unicast can be supported by an underlying MANET routing protocol. Those in turn follow the steps above and unicast to all of their overheard roots except the root from which the message was received. Additionally, no actions are taken if the message has already been received, as detected by a <root, sequence number> pair or message hashing.

4.2 Multi-query Optimizations

Reducing the quantity of messages which must be propagated and reasoned over is an important step toward scalability. Description logics support that via *least common subsumer* (LCS) inference [2]. LCS inference applies formal semantics of the logic to rewrite a set of class definitions into a single class definition of which each input is a subclass, and for which no other such class up to equivalence may exist in the given logic.

For example, the OWL class in Figure 3 may be written in description logic notation as:

$$\begin{aligned}\text{Query1} \equiv\ & \exists \text{msg:source}.[\exists \text{msg:org}.\text{orgs:NEAir}] \sqcap \\ & \exists \text{cbr:nuclideDetected}.\text{cbr:SuspiciousNuclide} \sqcap \\ & \exists \text{msg:format}.\{\text{nist-ansi-n4242xsd}\}\end{aligned}$$

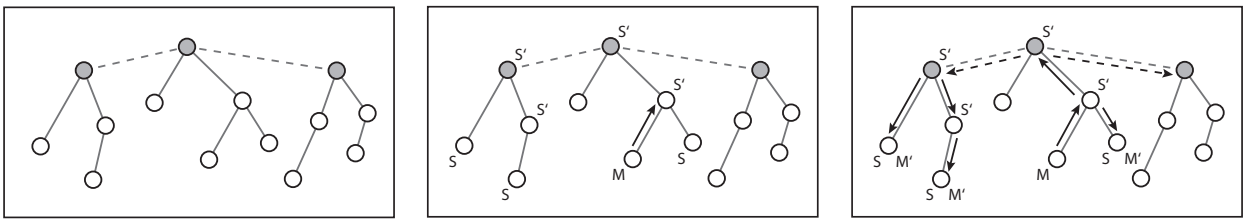
A similar but different query may be:

$$\begin{aligned}\text{Query2} \equiv\ & \exists \text{msg:source}.[\exists \text{msg:org}.\{\text{orgs:NEAir-Haz}\}] \sqcap \\ & \exists \text{cbr:nuclideDetected}.\text{cbr:IndustrialNuclide} \sqcap \\ & \exists \text{msg:authentication}.\text{msg:X509Certificate}\end{aligned}$$

The LCS of those two classes would be:

$$\begin{aligned}\text{Query3} \equiv\ & \exists \text{msg:source}.[\exists \text{msg:org}.\text{orgs:NEAir}] \sqcap \\ & \exists \text{cbr:nuclideDetected}.\text{cbr:Nuclide}\end{aligned}$$

Using the LCS, multiple destination queries may be summarized into a less discriminating query that is prop-



(a) Some nodes self-elect to be beacons; others form trees rooted at those beacons. Nodes overhearing adjacent trees pass that information on to their root, forming a mesh of trees.

(b) Nodes pass destination queries (S) to their root, possibly aggregating along the way (S'). Messages (M) generated by applications are also forwarded by the origin node.

(c) Received messages are forwarded to local matches as well as potentially matching descendants. Root nodes also unicast forward to adjacent roots, excluding the previous root.

Figure 7: Overview of the query propagation and message delivery process.

agated and reasoned on in remote network regions.

5 Preliminary Evaluation

In this section, we present preliminary evaluation of some of the basic *OntoNet* ideas and concepts described in this paper. We perform an initial simulation-based evaluation of the proposed tree-mesh multicast (*MTree* protocol described in Section 4.1), comparing its performance against the *SMF* [11] optimized flooding protocol used by OLSR [5]. We conduct our experiments in the NS-3 [12] simulator. Our simulations utilize a trivial loss-free radio model[5]. Experiments are varied from 100 to 2000 nodes placed at random with a uniform distribution in a 1km^2 arena, each with 100m radio range. After a 12s stabilization period, 10 randomly selected nodes register queries. Beginning at 24 seconds, a randomly selected node generates a multicast message every 1-3s to be delivered to those 10 nodes. The simulation stops at 32s.

Our simulation results demonstrate that *SMF* has significantly higher total bandwidth consumption compared to *MTree*, and its overhead is dominated by the flooding of descriptions. For a network of size 2000, *SMF* consumes an aggregate bandwidth of 5MB, compared to 0.5MB for *MTree*. Both approaches leads to similar load-balanced properties, where 95% of the nodes consuming less than 1% of the aggregate bandwidth. Since the root nodes consume the most resources, it is essential that we adapt our basic protocol to have nodes take turns to be roots, or assign powerful machines to be the root nodes.

6 Conclusion

In this paper, we explore *OntoNet*, a scalable knowledge-based middleware that supports content-based reasoning and routing in networked and mobile systems. We demonstrate the use of *OWL-Net* (a subset of OWL) and description logics in expressive, ontology-supported multicast querying and delivery. Aggregation and simplification mechanisms based on those logics have also been outlined, a key requirement for true scalability. In addition, a novel hybrid tree-mesh protocol has been given for efficiently propagating queries and delivering messages, along with initial evaluation in the NS-3 simulator. Our future work is progressing along several fronts. First, we would like to explore the limitations of the *OWL-Net* language via a variety of application scenarios. Second, we plan to experiment with different routing protocols for description and message propagation, including adding support for *anycast*. Third, we intend to utilize declarative networks [9, 10] as an extensible routing layer that enables a wide variety of protocols and scenario-driven propagation mechanism to be developed and deployed for *OntoNet*.

[5]More sophisticated radio models for NS-3 are in progress; this model was implemented for these experiments.

References

[1] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In *Int. Joint Conference on Artificial Intelligence*, pages 364–369, 2005.

[2] F. Baader, R. Kusters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Int. Joint Conference on Artificial Intelligence*, pages 96–101, 1999.

[3] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking Up Data in P2P Systems. *Communications of the ACM, Vol. 46, No. 2*, Feb. 2003.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):28–37, 2001.

[5] T. Clausen and P. Jacquet. RFC 3626: Optimized Link State Routing Protocol (OLSR). Technical report, The Internet Society, 2003. Experimental Track.

[6] M. Dean, G. Schreiber, et al. OWL Web Ontology Language reference. World Wide Web Consortium (W3C), February 2004. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`.

[7] Y. Diao, S. Rizvi, and M. J. Franklin. Towards an internet-scale xml dissemination service. In *VLDB*, 2004.

[8] O. Lassila, R. Swick, et al. Resource Description Framework (RDF) Model and Syntax Specification. World Wide Web Consortium (W3C), 1999. `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`.

[9] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative Networking: Language, Execution and Optimization. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, June 2006.

[10] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *Proceedings of ACM SIGCOMM Conference on Data Communication*, 2005.

[11] J. Macker, J. Dean, and W. Chao. Simplified multicast forwarding for MANET. Technical report, The Internet Society, 2007. Experimental Track Draft; `http://www.ietf.org/internet-drafts/draft-ietf-manet-smf-06.txt`.

[12] Network Simulator 3. 2007. http://www.nsnam.org/.