Optimization of Semantic Routing Table

Amitava Biswas, Suneil Mohan, Rabi Mahapatra
Department of Computer Science
Texas A&M University,
College Station, TX, USA 77840
amita_bi@tamu.edu, suneil@cs.tamu.edu, rabi@cs.tamu.edu

Abstract—In a semantic routed network (SRN), messages are routed based on the meaning of the message key. This means network nodes can be addressed by the meaning of the data content. Providing fast and successful semantic routing for any key is a challenging task due to conflicting performance demands. We present semantic routing table optimizing techniques to realize small world topology for fast and successful message routing. Our simulation shows that a SRN of 1000 nodes can achieve a competitive 57% routing success within 6 messaging delays, and have message delivery response of 3.3 messaging delays.

Keywords—Semantic content addressable network; small world.

I. INTRODUCTION

In the future, Internet users would prefer to access network resources based on their intentions rather than using IP addresses or network resource names. For example, users would prefer to access a weather station sensor by using a conjunction of descriptions: “WeatherStation”, “location: Chicago”, etc. In addition users would prefer to use a variety of interchangeable descriptions like “thermometer” or “MeasurementDevice” AND “temperature” instead of “thermometer” or “MeasurementDevice”, all of which can not be enumerated a-priori to set up an address resolution system.

A Semantic Routed Network (SRN) which routes messages based on their meaning instead of IP addresses can reasonably satisfy this aforementioned human need. To send a message to a particular network node, the message is loaded with a semantic key (denoted by “K”, Fig. 1) which have meaning similar to the description of the targeted node in the SRN (denoted by “D”, Fig 1). Once the message is injected in the SRN, it is delivered to the intended destination and then the destination can respond back to the message source. A SRN can be realized by a network of interconnected semantic routers which compare similarity between message keys and resource description stored in semantic routing tables to resolve the next hop destination (semantic lookup). This SRN does not obviate the existing Internet and IP network, on the contrary it would be an overlay network on top of the existing Internet and IP fabric. The semantic messages will be IP packet payloads and the connectivity between semantic routers (Internet hosts) will be over TCP/UDP spanning across several IP routers on the Internet. To allow physical co-location of multiple resources each of which may be a distinct SRN destination, URI names may be used as the underlying physical addresses, and SOAP can be used as the messaging transport over TCP.

Semantic routing and SRN have various applications and advantages. When a SRN is available, searching for resources in a grid will be possible without centralized indexing/directory services. The distributed nature of the SRN will render better scalability and coverage for searching/discovering resources in a grid. Users need not know the exact description (matching keywords) of service/data or its schema, they can simply send a ‘call back’ request with appropriate semantic key which describe the desired service/data. The semantic routers will tolerate variability of keys used as long as the keys contain same meaning and enough information to disambiguate that meaning (or comply with a standard ontology). This kind of semantic data retrieval system is novel because the available content addressable networks [1], intentional naming systems [2], or distributed search networks [3-6] only support routing/searches with exact matching keys.

The real challenge in a SRN is how to provide scalable routing performance in terms of high end-to-end routing success and low response time for any meaningful key. Quality of routing table content will determine routing performance which in turn affects search recall rate and response time when SRN is used as a semantic data retrieval system. The routing table needs to be optimized to improve the routing performance. In this regard the available proposals [1-6], do not attempt to improve routing table (or hash data structure) content quality and the suggested semantic routing mechanisms [7] are not extensible, and will require additional mechanisms to improve routing performance.

Simulation results in [8] indicate that wide variety of small-world networks are searchable or navigable, which
means that messages can be routed through them. If the network can self-organize to form a small-world topology that has small hop distances (number of hops required to traverse the network) between all node pairs, then any message can reach the intended destination in a small number of messaging delay. However, it is not a trivial task to generate a small world topology for semantic routing with modest routers.

This paper presents some techniques to optimize semantic routing tables so that a small-world topology is generated for an overlay semantic routed network (SRN) using modest sized semantic routers. Semantic lookup is a resource intensive process, therefore practical semantic routers will support only small routing tables having limited number of key-destination map entries and service limited message traffic rates. We address this scalability issue by using a large number of small routers, instead of small number of large routers. To make the small semantic routing table effective we applied several techniques that: (1) improve the quality of their content; and (2) impart a self-organizing behavior to the SRN so that the routing performance improves over time and usage. Our simulated SRN having 1000 nodes, has a expected end-to-end routing response (message delivery time) of less than ~3.3 messaging delays and routing success of ~57% when observed within a time frame of 6 messaging delays. This performance is competitive compared to 37% success rates reported for non-semantic distributed search networks [5].

In addition, we present an inexpensive scheme to compute semantic similarity which is used to resolve route. This scheme imparts more sophisticated functionality and extendibility because the semantic knowledge is not hardwired to the hash data structures as in other distributed search networks [4-6]. By addressing this problem we addressed the core issue of semantic routing in contrast to the available proposals [3,4,6,7]. To our knowledge this is the first work that optimizes the routing performance using self-organizing topology and which simulates a SRN for semantic routing performance evaluation. The understanding gained here can be utilized to design routers for grids which will provide semantic services.

II. DESIGN OF SRN: BASIC PRINCIPLES & CHALLENGES

A. Abstract Model of SRN & Concept of Semantic Routing

SRN consists of semantic router nodes and resource nodes that represent network resources. Routers stores addresses of all the resource and router nodes registered to it for message forwarding. A resource node sends messages to another resource or router node via the router node to which it is registered to. The route resolution (semantic lookup) essentially involves choosing a single (or few) forwarding destinations, based on which destination descriptor “D_i” is closest (most similar) to the message key descriptor “K”. We denote the semantic similarity metric between “D_i” and “K” by the notation KâD_i. If KâD_i > KâD_j then the message will be forwarded to the destination having description D_i.

B. Semantic Descriptor Data-structure : Motivation & Challenges

If we choose a network organization scheme that groups destinations based on how similar their descriptors are (analogous to forming sub-nets), then this allows us to replace multiple key-destination map entries (routing table rows) with a single entry (row). However this requires identifying all the destinations whose descriptors are similar to each other. For route resolution we also need the same semantic similarity comparison capability. Thus we need to: (1) design a semantically scalable and computationally tractable data structure to represent meaning; and (2) design a method to compare two descriptor data structures and generate a semantic similarity metric. This data structure will be used to define the semantic key and the destination descriptors and the comparison method is contingent on the data structure design. Traditional approaches to capture, convey and compare meanings using lexical [9], ontological [10] and semantic network based approaches [11] have limitations in SRN application. These schemes do not semantically scale up, which means either these schemes can not correctly capture and compare moderately complex meanings that are required, or they are computationally expensive for a router.

C. Network Organization: Motivation & Challenges

The other set of problem deals with network organization and topology which have effect on scalability. Hierarchical topology was sufficient to tackle the scalability issue in IP networks primarily because IPv4/v6 addresses have rather limited range (0 to 2^{32} or 2^{128}) and router could support larger routing tables. But hierarchical organization is insufficient to tackle the scalability challenges in SRN because the semantic routing space is quite large whereas routing tables have to be smaller. The range of routing space, which represents all possible mappings between all points in the address space and points in the actual destination space, is product of: (i) number of all possible resources that has to be individually addressed (destination space), and (ii) number of possible ways to describe each resource (address key space). Since users are likely to use multiple descriptions to address a resource, the resulting combinatorial address is extremely large compared to the IP address space. At the same time the number of semantic objects (destination nodes) is likely to be several orders of magnitude larger than possible IP destinations, because availability of the SRN will definitely encourage applications which will need large number of individual semantic objects (destinations). Therefore the challenge is how to generate an alternative scalable network topology that would: (1) cover large semantic routing space to yield higher routing success rate (% of messages that successfully reach destinations for a variety of key-destination combinations); (2) limit response times by limiting number of semantic routing hops for a given length of routing table; (3) minimize congestion; and (4) minimize number of routers.
III. SEMANTIC DESCRIPTOR AND SIMILARITY COMPARISON

A. Descriptor Data Structure for Complex Concepts

Meaning is actually a concept in the human mind. Human beings compose complex concepts (e.g. “Weather Station at Chicago” as in Fig. 2) in their mind using an associative network (known as “semantic network” [11]) of elementary concepts (e.g. “Weather Station”, “Chicago”). Therefore a semantic network can be used as the semantic descriptor data structure to represent/store, transmit and compare the intended meaning as understood by human users. We propose to represent the weather network by a simple flat set of elementary concepts. In this scheme the descriptor “D2” for of a complex concept is represented by a set of elementary concepts: \{C_{11}, C_{12}, \ldots, C_{1n}\}.

![Semantic Network Representation](image)

![Concept Lattice Representation](image)

Figure 2. Representation of complex and elementary concepts

B. Similarity Comparison of Complex Concepts

The method to compare semantic similarity between two descriptors is based on semantic similarities between their elementary concepts. For example, when D1 = \{C_{11}, C_{12}\} and D2 = \{C_{21}, C_{22}\}, then D1 \rightarrow D2 is represented by a ordered set S_{12} = \{s_1, s_2, \ldots, s_n\}, where s_1 > s_2 > \ldots > s_n, and each element “s_i” belongs to a set \{C_{11}, C_{12}, C_{21}, C_{22}\}. Suppose D1 \rightarrow D3 = S_{12} = \{5,5,4,3,1,1,0\} and D1 \rightarrow D4 = S_{34} = \{5,5,4,3,1,1,0\}. The set of common prefix elements from the two ordered sets S_{12} and S_{34} is \{5,5,4\}. Once we take out \{5,5,4\} from S_{12}, the set of remaining ordered elements is S_{12}' = \{3,1,1,0\}. Similarly S_{34}' = \{4,3,1,1,0\}. The maximum elements from sets S_{12}' and S_{34}' are 3 and 4 respectively. As 3 < 4, hence we assert that D1 \rightarrow D2 < D1 \rightarrow D4. Here the closest relationships influence the comparison most. Using a flat set of elementary concepts is a pragmatic choice because it is inexpensive. Next, we present the elementary concept data structure (e.g. C_{11}) and the associated similarity (e.g. C_{12} \rightarrow C_{22}) comparison technique which render sufficient expressiveness.

C. Data Structure for Elementary Concepts

Human beings conceptualize and remember objects by their attributes and associated subsumption relationships. Hence we use a data structure known as concept lattice [12,13] to represent elementary concepts in term of attributes. A concept lattice is composed of multiple unit concepts, each of which is represented by a pair \{O_k, A_k\}, where “O” denotes a set of objects which have all attributes belonging to a set denoted by “A”. Therefore, meaning of an object belonging to O can be conveyed by this unit concept representation \{O_k, A_k\}. We now explain the concept lattice with an over simplified hypothetical example, but in real applications the data structures would be more complex and sufficiently expressive. If the intent is to allow addressing the weather station thermometer with more generic alias names like: “MeasurementDevice” (an IEEE SUMO [14] term) in conjunction with additional qualifiers like “weather”, “temperature”, etc, then the objects of interests are: “thermometer”, “Weather Station”, “MeasurementDevice”, and the list of attributes of interest (based on the SensorML [15] descriptions) are: “temperature”, “weather”, “weatherMeasurements”. In real application these lists will be much larger. The object-attribute set pair \{\{Weather Station, thermometer\}, \{weather, weatherMeasurements\}\} will denote the weather station concept, which is represented as an unit concept node in the concept lattice (Fig. 2). We can conceive a unit concept node “measurement device” which is represented as \{\{Weather Station, thermometer, MeasurementDevice\}, \{weatherMeasurements\}\}, and which is a super concept of the weather station. The super concept node is placed above the concept node and connected to it by an edge. By adding more super and sub concept nodes and edge relationships we can generate the entire lattice graph. Algorithms to generate concept lattices [13] and techniques to generate them from textual description [16] are available. We are only interested in constructing parts of the concept lattice that is essential and sufficient for the given application. Evidence of usefulness of concept lattice can be found in [16].

When the meaning of an attribute can be unambiguously communicated by a smaller set of alternative symbols (lexical representations) under all possible contexts, then those are considered as lowest level or primitive attributes. Otherwise attributes will require further decomposition (definition) in the concept lattice to get a larger, decomposed lattice, where these attributes themselves are considered as objects for the next lower level of analysis. Primitive attributes will be represented by their lexical representations. For example if the attribute “location” is considered primitive attribute, then it will simply represented by its textual representation (i.e. “location”).

D. Similarity Comparison of Elementary Concepts

We modify the technique to compare two concept lattices as suggested in [16,17]. The similarity metric is derived based on how many nodes in the two concept lattices share common elements from both object set “O” and attribute set “A”. First a set of candidate objects and attributes that are common to both latices are identified (step 1). Then for every common object (and attribute) in this candidate object (attribute) set, the most specific concepts (the concepts that at lowest position in the concept lattice) are extracted from both concept lattices (step 2). These extracted candidate unit concepts are compared against each other to find whether they match or not (step 3). A match indicates that there is at least a common object and a common attribute between the two unit concepts \{O_k, A_k\} and \{O_l, A_l\} belonging to these two concept lattices. The total number of such matches found is normalized (step 4) by the total number of all extracted concepts pairs considered for matching to get the similarity metric. By using compact data structures like Bidirectional Associative Memory [17] and bloom filters [18] this similarity comparison technique can be parallelized and implemented in low level hardware to materialize fast response semantic router hardware.
IV. NETWORK ORGANIZATION MECHANISMS

A. Small World Topology: Motivation

We chose small world to implement a SRN because small world topology renders better congestion behavior and requires less number of routers to interconnect a given number of resources compared to over hierarchical topology (Table 1). The congestion comparison is made by considering worst case probability that a message will pass though a particular router. In hierarchical topology the top most (core) router is almost always hit (probability ≈ 1), whereas the traffic load is uniformly distributed among all small world routers. In absence of exact analytically solution, empirical simulation of small world graphs [19] indicated for expected path length to be quite small for large graph sizes which is found to be comparable to that of hierarchical topology (Table 1).

![Figure 3. Hierarchical and small world topologies](image)

**TABLE I. PERFORMANCE COMPARISON OF NETWORK TOPOLOGIES**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Hierarchical</th>
<th>Small world</th>
</tr>
</thead>
<tbody>
<tr>
<td>Congestion (message probability)</td>
<td>Very high</td>
<td>Very low</td>
</tr>
<tr>
<td></td>
<td>$k^* (N - 1)^*N/k \approx 1$</td>
<td>$k^*w^{-d-r} \approx k^*w \times N/N$</td>
</tr>
<tr>
<td>Number of routers necessary</td>
<td>Higher</td>
<td>Lower</td>
</tr>
<tr>
<td></td>
<td>$\log \sum_{i=1}^{\text{number of} \text{interests}}(N^<em>k^{-i})/k^{N^</em>}$</td>
<td>$\log(N^<em>k^{-i})/k^{N^</em>}$</td>
</tr>
<tr>
<td>Expected path length</td>
<td>$2 \log(N/k) + c$</td>
<td>$c \log(N/(k^*w^{-d-r}))$</td>
</tr>
</tbody>
</table>

$N$ = number of resources, $k = number of routing table rows (length), w = number of columns (width), r = number of peer routers registered by each router, $d =$ number of long edges (shortcuts) per router, $c =$ a constant, $N >> k > w >> r > 1$, $1 < c < 5$.

B. Key Principles of Small World Topology

Small world network topology is characterized by small expected path length (hop distances) and large clustering coefficient (probability that two nodes are connected if they have a common peer) [19]. This topology is generated if a very small fraction of edges in a uniform lattice graph are disconnected from immediate neighbors and randomly connected to a node which is far away (Fig. 3) [19]. Next we present how topology generation is materialized in SRN.

C. SRN Node Behavior & Generation of Small World Topology

The router and resource nodes have dynamic (agent like) behavior by virtue of which they rapidly interconnect with each other to organize a small world topology. Resource nodes have a single descriptor called resource description, whereas routers have small number of multiple dissimilar descriptors, called router’s interests, which are semantically further away from each other. Other routers perceive interests of a router as its descriptions. Routers prefer to register only those resource and router nodes to its routing table, whose descriptions are close to either one of its own interests. Whereas a resource prefer to register itself to a single router whose one of the interest is semantically closest to its description. Each row of the routing table has multiple columns (as in Fig. 4) to allow message forwarding to multiple destinations which effectively increase the clustering coefficient. To further increase this coefficient we might choose to forward messages to destinations in “t” closest matching rows in the routing table. This can cause duplicate messages, hence our routers detect and drop duplicate messages.

We can visualize a semantic routing space (based on an ordinal scale) where a physical SRN node can be represented as a virtual semantic routing graph node. A resource node is represented as a single virtual graph node, whereas a router node is represented by multiple virtual graph nodes, one for each interest descriptor. The virtual graph nodes are connected by bi/uni-directional edges depending on the physical connectivity between the respective SRN nodes (as per the routing table content). The distance between the virtual graph nodes vary depending on the semantic similarity between the SRN node descriptions/interests, thus the preference to connect to similar nodes will generate short edges between their virtual graph nodes to form clusters/cliques and a lattice fabric as in (Fig. 3). The diversity in router’s interests allows its virtual graph nodes to be part of multiple cliques, which effectively generates the long edges (bold lines, Fig.3) in the routing graph making it a small world graph. This is because a message can arrive at a router based on one of its descriptions and then it may be forwarded to another peer router whose description is quite dissimilar (and far away). This is equivalent to a traversal over two short edges (corresponds to two physical hops) and a long edge (no physical hops) in between (bold dotted lines, Fig. 3). Thus the corresponding physical SRN also behaves as a small world network. Next we present the mechanisms that ensure this formation.

D. Node Clustering Algorithm

To identify nodes with similar descriptions, a router periodically sends query messages one for each interest description, along with a similarity comparison threshold. The nodes whose descriptions are similar to the query key (the similarity metric is greater than the threshold) respond back to the router and the router registers them. If the routing table still has vacancy, the router progressively expands its horizon by sending queries with smaller threshold (allow more dissimilar nodes). A resource node similarly sends queries to identify the closest matching router to register itself.

E. Routing table entry eviction policy

Each router has a limited number of routing table rows and columns (limited length and width capacity). Therefore our semantic routers strive to enhance the space-quality efficiency of the routing table contents over time by replacing the routing table entries with more relevant ones, so that messages are more correctly steered towards the destination through the shortest possible route. The destination addresses in the routing table columns are either conserved or evicted.
based on the “relevancy” of the destination node’s description w.r.t. the row’s key. The address, whose description is semantically farthest from the row’s key is evicted. Similarly the candidate row for eviction is chosen based on router’s relative interest on the row’s key, by comparing similarity between the row’s key and router’s interest descriptors.

F. Routing table Reorganization Algorithms

Reallocation of column entry: When a destination is evicted from a row, an attempt is made to reallocate it in another row which is the next best possible place for it (i.e. whose key is semantically closest to the current row from which the destination is being evicted) and which has vacancy. The check for vacancy is made by the following logic. If there is a free column space, then the vacancy test returns a straight forward “true”. If there is no free column space then a check is made to ascertain whether this destination being reallocated is likely to be evicted from this row in near future or not. If it appears that the reallocated destination is likely to be evicted in near future, then the next best matching row is considered. The check for possible future eviction is made by comparing the relevance of the reallocated destination node’s description with the lowest relevant destination’s description that is already in that row. This “relevancy” test (as previously described) is carried out with respect to the key of the row that is being considered.

If the vacancy test returns true for a row which does not actually have a free column space, then the lowest relevant destination is evicted from this row to make space for the destination which being reallocated. The destination which is swapped out is again reallocated to the best matching row using the same principle. This leads to a recursion of reallocation column entries (domino effect). To identify the best matching row, only those rows are considered which haven’t suffered a destination eviction so far in the current recursion tree. The recursion may terminate either when a free column space is found (which doesn’t lead to another destination being reallocated) or when there are no rows left to be consider. This is explained with an example routing table (Fig. 4), where the columns are arranged from left to right and rows has been arranged from top to bottom in order of increasing “relevancy”. The destination 5 from row with key value 11 is being reallocated. First destination 1 from row 9 is swapped out, then finally destination 83 from row 6 is swapped out.

Step 1: Destination 5 being reallocated, from row “11”

<table>
<thead>
<tr>
<th>Key</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 78 13 75</td>
</tr>
<tr>
<td>9</td>
<td>5 78 13 75</td>
</tr>
<tr>
<td>6</td>
<td>83 1 43 18</td>
</tr>
<tr>
<td>8</td>
<td>13 2 11 18</td>
</tr>
</tbody>
</table>

Step 2: Dest. 1 has been swapped out from row “9”

<table>
<thead>
<tr>
<th>Key</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 78 13 75</td>
</tr>
<tr>
<td>9</td>
<td>5 78 13 75</td>
</tr>
<tr>
<td>6</td>
<td>83 1 43 18</td>
</tr>
<tr>
<td>8</td>
<td>13 2 11 18</td>
</tr>
</tbody>
</table>

Step 3: Dest. 83 has been swapped out from row “6”

<table>
<thead>
<tr>
<th>Key</th>
<th>Destinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 78 13 75</td>
</tr>
<tr>
<td>6</td>
<td>1 78 13 75</td>
</tr>
<tr>
<td>8</td>
<td>13 2 11 18</td>
</tr>
</tbody>
</table>

Figure 4. Column reallocation algorithm

Reallocation of an entire row: When an entire row is evicted (to make space for a new row, so that a router can expand its horizon), an attempt is made to reallocate all its destinations using the column entry reallocation algorithm. Row with key 11 is an example of row eviction in Fig. 4.

V. EXPERIMENTS & RESULTS

A. Simulation Setup

Depending on the experiments, following parameters were used: number of resource nodes = 200 to 400, number of routers = 200 to 800, routing table length = 5 to 10, routing table width = 4 to 9, “t” (as defined in sec. IV. C) = 1 to 3, periodicity of resource and router nodes were 10 virtual clock cycles, the timeout period after which the message delivery success was noted = 20 and network link delay = 3 clock cycles respectively. Routing table size was kept very small compared to number of nodes to simulate the scalability hurdle. When not specified the routing table length, width and “t” values are 5, 5 and 2.

B. Results & Analysis

We compare the performance of the SRN against two criteria: (i) estimated expected end-to-end routing response time or message delivery time (hops required by a message to reach destination, lesser is better); and (ii) end-to-end routing success rate (instances of destinations reached if they exist, higher is better) observed within a time frame of 6 messaging delays. The network is dynamic because the nodes are constantly seeking each other and thereby the network is using its own capabilities to evolve, therefore virtual clock cycles indicate network maturity and usage.

Fig. 5 compares the routing success rate for two SRN paradigms: (a) large number (200) of small semantic routers having optimized routing table size of 5 X 5; (b) small number (20) of large routers having large size of 50 X 50 non-optimized routing tables. This shows that small optimized routers (35% success rate, response time of 3.3 messaging delays) outperform large routers (17.5% success rate, response time 4.4 messaging delays). Therefore it is possible to replace large routers with large numbers of small routers.

Fig. 6 compares the success rate in three different SRNs: (1) the first one with routers which neither enjoy prioritized eviction, nor have routing table reorganization algorithms (“RT w/ no optimization”); (2) the second one with routers that only have prioritized eviction of routing table (“RT w/ prioritization”); and (3) the third one which has routers with full routing table optimization (“RT fully optimized”).

Figure 5. Performance comparison between small & large semantic routers

Figure 6. Role of different routing table optimization algorithms
In the first case hardly any successful semantic routing took place. In the second case some successful semantic routing took place, and the routing performance improved when the destination reallocation algorithm was applied in the third case. This comparison demonstrated the roles of the prioritized eviction and row and column reallocation algorithms.

Fig. 7 shows how the expected (average) end-to-end routing (delivery) time gradually improves over the life-time of SRN. The “expected” number of hops to traverse before reaching destination drops below 2 hops after the SRN has self-organized for 120 cycles (routing table rows = cols = 6).

Figure 7. Improvement of expected delivery time with network maturity

Fig 8 reveals the trends of routing success rate with network maturity. When “t” = 1, the routing table capacity is reached at around 40 cycles which corresponds to an end-to-end routing success rate of ~10%. This capacity limit is relaxed when higher values of “t” are used (for “t” = 2, success rate is ~36%, for “t”= 3, it is beyond 57%) (Fig. 8). As the capacity limit is relaxed it takes more time to reach the limiting value.

Figure 8. Success rate trends for different “t” parameter

Fig. 9 shows the trends in routing table capacity (length = 4, width = 3) utilization. Even though the routing table length is constant after 40 cycles, but success rate continues to grow till 80 cycles. Over time, routers fill up their routing tables by selective discovery and finally once they hit the space limit (at 40 cycles) they improve the space-quality efficiency of the routing table (during the period of 40 to 80 cycles) by prioritizing and reallocating their contents. After 80 cycles the routing table capacity is reached and thus no more improvement in the end-to-end routing success rate is noticed.

Figure 9. Routing table capacity utilization

VI. CONCLUSION

We have presented optimization techniques for a semantic routed network (SRN), where messages are routed depending on the meaning of the message key. The presented performance evaluation results show that this system has a small response time and competitive success rate. We also demonstrated that these techniques allow SRN to self-organize and continuously improve the routing performance. These techniques generate topologies that have small world behavior and yet it can be supported by routers which have modest resources. This means that such semantic routers can be practically built to implement semantic service grids and web enabled sensor applications. To our knowledge this is the first work on optimization of a Semantic Routed Network.

REFERENCES