A Comprehensive Formal Framework for Analyzing the Behavior of Nature-inspired Routing Protocols

Saira Zahid, Muhammad Shahzad, Syed Usman Ali, Muddassar Farooq

***Abstract*—Nature-inspired routing protocols are becoming an active area of research. Researchers in the community follow a well known engineering philosophy: inspire, abstract, design, develop and validate. As a consequence, the protocols are designed on the basis of heuristics and then their performance is evaluated in a network simulator. To the best of our knowledge, virtually no attention has been paid in developing a formal framework that provides an analytical insight into the behavior and performance of such algorithms. The lack of formal treatment of Nature-inspired routing protocols is often criticized in the networking community. In this paper we propose a formal framework that helps in analyzing a Nature-inspired routing protocol, *BeeHive*. We have verified the correctness of our model by comparing its estimated values with the results obtained from extensive network simulations. An important outcome of the work is that the estimated and measured values only differ by a small deviation. We believe that this work will be instrumental for *Nature-inspired Telecommunications*.**

I. Introduction

Nature-inspired routing protocols have received a significant amount of attention by researchers in Nature-inspired Computing because they provide efficient, scalable, robust, fault-tolerant, dynamic, decentralized and distributed solutions to the traffic engineering within the existing connectionless model of IP [1]. The objectives are achieved through a population of agents, inspired by Natural colony systems i.e ant or bee colony, which have simple learning behavior [2]. As a result, such robust agent based systems can be easily embedded in real world networks because they do not require additional hardware or software resources for their deployment in real world networks [3]. *AntNet* [4], *BeeHive* [5] and *DGA* [6] are considered to be state-of-the-art routing algorithms for fixed packet switched networks.

The results of extensive experiments reported in [7] clearly demonstrated the superior performance of such algorithms as compared to classical routing algorithms. The most important reason is that the agents explore multiple paths between all source/destination pairs and then stochastically distribute the network traffic on them. As a consequence, the network performance is significantly enhanced [7]. Researchers working in Nature-inspired routing protocols have mostly followed the same protocol engineering philosophy: inspire, abstract, design, develop and validate. As a result, Nature-inspired protocol engineering is predominately biased towards heuristics and their empirical evaluation in a simulation environment. To the best of our knowledge, no comprehensive research, except the preliminary work reported in [8], has been conducted in developing a formal framework that provides insight into the behavior of Nature-inspired routing protocols.

The lack of any formal treatment is often cited as a big shortcoming of Nature-inspired routing protocols. Researchers in the community have little formal understanding of the reasons behind the superior performance of such routing protocols. The obvious reason for lack of a formal treatment of a dynamic and adaptive routing algorithm is that it demands expertise in analyzing non-linear, time varying, real time and dynamic optimization problems.

The major contribution of the work presented in this paper is that we have developed a formal framework for analyzing the behavior of *BeeHive*. This is the first cardinal step towards development of a general framework for analyzing Nature-inspired routing protocols. With the help of this model, we are able to gain valuable insight into the impact of different design options adopted in *BeeHive* and provide formal logic for its behavior. We also used the formal model in representing relevant performance parameters: throughput and end-to-end delay. The comprehensive experiments were conducted on two topologies in a well known OMNeT++ [9] network simulator to verify the semantic and functional correctness of our formal model. The behavior of *BeeHive* and relevant performance parameters estimated by our formal model match the behavior and results obtained from OMNeT++ simulations respectively.

Section II gives an overview of *BeeHive* routing protocol. In section III, we present a formal treatment of two different formulas utilized in *BeeHive* for evaluating the quality of a path. Section IV presents our comprehensive formal model of *BeeHive* protocol. We have modeled goodness, total delay and throughput of the links in the network. In Section V we verify the correctness of our model by comparing its results with the results obtained from OMNeT++. Finally, we conclude our work with an outlook to the future work.

II. Overview of *BeeHive*

BeeHive has been proposed by Wedde, Farooq and Zhang in [5]. The algorithm has been inspired by the communication language of honey bees. Each node periodically sends a *bee agent* by broadcasting the replicas of it to each neighbor site. The replicas explore the network using priority queues and they use an estimation model to estimate the propagation and queuing delay from a node, where they are received, to their launching node. Once the replicas of the same agent

Saira Zahid, Muhammad Shahzad and Syed Usman Ali are students at the Department of Electrical Engineering, College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Pakistan.

Muddassar Farooq is a Professor at the Department of Computer Engineering College of Electrical and Mechanical Engineering, National University of Sciences and Technology, Pakistan.

TABLE I
SYMBOLS USED IN THE PAPER

g_{in}	goodness of a link (i,n)
ld_{in}	delay of a packet from node i to n
Ld	matrix whose entries are ld_{in}
p_{in}	propagation + transmission delay for link (i,n)
q_{in}	queuing delay for the link (i,n)
$\mathfrak{b}_{in}$	proportion in which the bees traverse the link (i,n)
x	iteration index
y	q_{in}/p_{in}
β_i	rate of the bees entering the network at node i
$\boldsymbol{\beta}$	matrix whose entries are β_i
γ_i	flow of bees at node i
$\boldsymbol{\gamma}$	matrix whose entries are γ_i
ξ_i	rate of the data packets entering the network at node i
$\boldsymbol{\xi}$	matrix whose entries are ξ_i
η_i	flow of data at node i
$\boldsymbol{\eta}$	matrix whose entries are η_i
$\mathfrak{bf}_{in}$	flow of bees on link (i,n)
$\mathfrak{df}_{in}$	flow of data on link (i,n)
υ_{in}	total flow of traffic on link (i,n)
S_{in}	service rate of the link (i,n)
tx_{in}	transmission delay for the link (i,n)
pd_{in}	propagation delay of the link (i,n)
$cd_{n\mathcal{D}}$	cumulative delay from node n to node $\mathcal{D}$
Cd	matrix whose entries are $cd_{n\mathcal{D}}$
td_{in}	total delay from node i to $\mathcal{D}$ through n
Td	matrix whose entries are td_{in}
α	ratio of size of bee packet to data packet
$\mathcal{T}_{in}$	throughput of the link (i,n)
$\mathcal{A}$	set of all nodes in netwrok
$\mathcal{N}_i$	set of neighbours of node $i \in \mathcal{A}$
$\mathcal{L}$	set of links in the network

arrive at a node via different neighbor sites of the node, they exchange routing information to model the network state at this node. Through this exchange of information by the replicas at a node, the node is able to maintain a quality metric for reaching destinations via its neighbor sites. The algorithm utilizes just forward moving agents and, as opposed to *AntNet*, no statistical parameters are stored in the routing tables. In *BeeHive* a network is divided into *Foraging Regions* and *Foraging Zones*. Each node belongs to only one *Foraging Region*. Each *Foraging Region* has a representative node. A *Foraging Zone* of a node consists of all the nodes from whom a replica of an agent could reach this node in 6 hops. This approach significantly reduces the size of the routing table as compared to *AntNet* because each node maintains detailed routing information only about reaching the nodes within its *Foraging Zone* and for reaching the representative nodes of the *Foraging Regions*. In this way, a data packet, whose destination is beyond the *Foraging Zone* of a node, is forwarded in the direction of the representative node of the *Foraging Region* containing the destination node. The next hop for a data packet at a node is selected in a probabilistic fashion depending upon the goodness of each neighbor for reaching the destination. *BeeHive* is also fault-tolerant to crashing of routers. The interested reader will find more details in [5][3].

III. GOODNESS

Goodness of a link from node i to node n, g_{in}, is the probability that a packet will be switched to neighbor n. In [3], the authors have used two different formulas for calculating the goodness of a link. They did not provide any formal relation between the two. In this section we show the relation between the two formulas.

Assume a network in which all the nodes are connected, either directly or indirectly with one another. Thus the network is a connected graph. The network consists of a set of nodes $\mathcal{A}$. Neighbors of each node $i \in \mathcal{A}$ are in corresponding set $\mathcal{N}_i$. All links in the network belong to the set $\mathcal{L}$. In our model we send data to a single destination, $\mathcal{D}$, at a time and we assume that two nodes are not directly connected by more than one link.
The goodness of link from node i to node n is given by:

$$g_{in} = \frac{\frac{1}{ld_{in}}}{\sum_{k\in\mathcal{N}_i}(\frac{1}{ld_{ik}})} \tag{1}$$

where

$$ld_{in} = \Big(\frac{1}{p_{in}}(e^{-\frac{q_{in}}{p_{in}}}) + \frac{1}{q_{in}}(1-e^{-\frac{q_{in}}{p_{in}}})\Big)^{-1} \tag{2}$$

In (2) p_{in} represents the sum of propagation delay and transmission delay for the link (i,n), q_{in} represents the queuing delay from node i to n in order to reach the destination $\mathcal{D}$ and ld_{in} is the total link delay between nodes i and n. One can easily conclude from (1) and (2) that the probability that a packet traverses a certain path depends on the delays of the path. Queuing delays under low loads are small. As a result, the goodness of a path is determined by the propagation delay of the path.

Mathematical functions like exponentials take significantly large number of cycles to compute, therefore, in [3] the authors have instead used a simpler version of (2). Its processing complexity is 10 times smaller as compared to (2) but it did not degrade the performance of *BeeHive*. The goodness formula is same as (1) where ld_{in} is given by

$$ld_{in} = p_{in} + q_{in} \tag{3}$$

In our model the values of ld_{in} are maintained in a matrix Ld of dimensions $\big(|\mathcal{A}|-1\big) \times \big(|\mathcal{A}|-1\big)$.

Lemma 1:
The two goodness formulas (2) and (3) converge to the same goodness value under high traffic load.

Proof :
Substituting

$$y = \frac{q}{p} \tag{4}$$

in (2), we get

$$\frac{1}{p}(e^{-\frac{q}{p}}) + \frac{1}{q}(1-e^{-\frac{q}{p}}) = \frac{1}{q}\left\{1 + ye^{-y} - e^{-y}\right\} \tag{5}$$

Power series of e^{-y} is given by

$$e^{-y} = 1 - y + \frac{y^2}{2!} - \frac{y^3}{3!} + \cdots \tag{6}$$

Substituting (6) in (5) and on simplification, the R.H.S of (5) becomes

$$= \frac{1}{q}\left\{\sum_{n=1}^{\infty}(-1)^{n+1}\left(\frac{n+1}{n!}\right)y^n\right\} \tag{7}$$

Similarly using (3), we obtain

$$\frac{1}{p+q} = \frac{1}{q}\left\{\frac{y}{y+1}\right\} \tag{8}$$

On applying binomial expansion, L.H.S of (8) becomes

$$= \frac{1}{q}\left\{\sum_{n=1}^{\infty}(-1)^{n+1}y^n\right\} \tag{9}$$

Now we can see that (7) and (9) differ only by the factor $\frac{n+1}{n!}$.

Using another approach to show the similarity between the two goodness formulas in (5) and (8), let

$$f_1(y) = 1 + ye^{-y} - e^{-y} \tag{10}$$

$$f_2(y) = \frac{y}{y+1} \tag{11}$$

Applying limit $y \to \infty$ on (10)

$$\begin{aligned} lim_{y\to\infty}\ f_1(y) &= lim_{y\to\infty}\left(1 + ye^{-y} - e^{-y}\right) \\ &= 1 + lim_{y\to\infty}\left(\frac{y}{e^y}\right) - 0 \end{aligned} \tag{12}$$

Using L'Hopital's rule for 2nd term on R.H.S

$$\begin{aligned} lim_{y\to\infty}\left(\frac{y}{e^y}\right) &= lim_{y\to\infty}\left(\frac{\frac{d(y)}{dy}}{\frac{d(e^y)}{dy}}\right) \\ &= lim_{y\to\infty}\frac{1}{e^y} \\ &= 0 \end{aligned} \tag{13}$$

So (12) becomes

$$lim_{y\to\infty}\ f_1(y) = 1 \tag{14}$$

Again applying limit $y \to \infty$ on (11)

$$\begin{aligned} lim_{y\to\infty}\ f_2(y) &= lim_{y\to\infty}\left(\frac{y}{y+1}\right) \\ &= lim_{y\to\infty}\left(\frac{1}{1+\frac{1}{y}}\right) \\ &= 1 \end{aligned} \tag{15}$$

From (14) and (15), one can easily deduce that the two goodness formula give same results under heavy traffic load. This can be verified graphically as well. In Figure 1 we plot $f_1(y)$ and $f_2(y)$ as these functions represent the only difference between (5) and (8).

In Figure 1 one can see that the difference between the two factors decreases as y increases. This behavior is expected because an increase in y is possible only due to an increase in q (see Lemma 1) that in turn results due to heavy network traffic load.

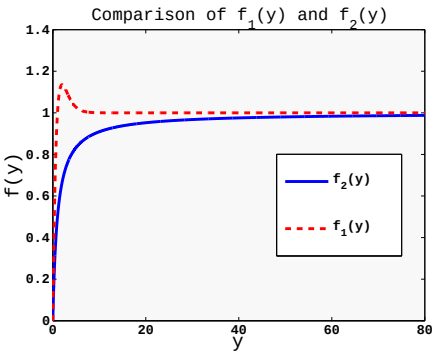

Fig. 1. Comparison between (10) and (11)

IV. Analytical Model

In this section we present a comprehensive formal model that can be of significant help in analyzing the behavior of *BeeHive* Protocol [5]. We will utilize relevant concepts of deductive mathematics and queuing theory in our formal model. Markov transition matrices coupled with stochastic processes [10][11][12] serve as the foundation of our framework. We model the network traffic by calculating three parameters:

- goodness of a link from node i to node n, g_{in}
- total delay experienced by a packet (propagation+queuing+transmission) to traverse a certain path in the network
- throughput of the links in the network.

The packet routing probabilities (goodness) can be collected using the Markov Transition Matrix in which each entry represents routing probability between the corresponding two nodes. This is $|\mathcal{A}| \times |\mathcal{A}|$ matrix and follows the properties of a Markov Transition Matrix which are:

- $\sum_n g_{in} = 1$
- It is a square matrix.
- Its entries are non-negative.

We also use a discrete iteration index $x \in \mathbb{N}$. Furthermore we maintain a matrix B of dimensions $\left(|\mathcal{A}|-1\right) \times \left(|\mathcal{A}|-1\right)$. The entries of this matrix actually determine the proportion in which a node forwards the *bee agents* towards its neighbors. This matrix is defined as:

$$B(x) = \mathfrak{b}_{in}(x) \qquad _{i,n=1,2,\cdots,|\mathcal{A}| \wedge i \neq \mathcal{D}} \tag{16}$$

where

$$\mathfrak{b}_{in}(x) = \begin{cases} \frac{1}{n(\mathcal{N}_i)} & (i,n) \in \mathcal{L} \\ 0 & i = n \vee (i,n) \notin \mathcal{L} \end{cases} \tag{17}$$

Similarly we maintain another matrix $\mathcal{G}$ of dimensions $\left(|\mathcal{A}|-1\right) \times \left(|\mathcal{A}|-1\right)$. The entries of this matrix determine the goodness of corresponding links. This matrix is given as:

$$\mathcal{G}(x) = g_{in}(x) \qquad _{i,n=1,2,\cdots,|\mathcal{A}| \wedge i \neq \mathcal{D}} \tag{18}$$

where

$$g_{in}(x) = \begin{cases} \frac{\frac{1}{td_{in}(x-1)}}{\sum_{k\in\mathcal{N}_i}\left(\frac{1}{td_{ik}(x-1)}\right)} & (i,n) \in \mathcal{L} \\ 0 & i = n \vee (i,n) \notin \mathcal{L} \end{cases} \tag{19}$$

where td_{in} is the total delay from node i to $\mathcal{D}$ through neighbor n. The values of td_{in} are maintained in matrix Td of dimensions $\big(|\mathcal{A}|-1\big) \times \big(|\mathcal{A}|-1\big)$. The values for $\mathfrak{b}_{i\mathcal{D}}(x)$ and $g_{i\mathcal{D}}(x)$ cannot be directly calculated from (16) and (18) respectively. They can be calculated using the following formulas:

$$\mathfrak{b}_{i\mathcal{D}}(x) = 1 - \sum_{n \in (\mathcal{N}_i \setminus \mathcal{D})} \mathfrak{b}_{in}(x) \quad (20)$$

$$g_{i\mathcal{D}}(x) = 1 - \sum_{n \in (\mathcal{N}_i \setminus \mathcal{D})} g_{in}(x) \quad (21)$$

A. Node Traffic

Flow rate of node and link is the amount of the traffic (bee-agents and data packets) passing through the node and the link respectively per unit time.

1) Bee traffic: In *BeeHive*, every node generates *bee agents* that traverse all possible paths available to a node in the network and update routing tables at every visited node. The size of these packets is small as compared to data packets [5]. As a result, they use less than 1% of the bandwidth. For this purpose we introduce a constant α which is the ratio of the size of a bee packet to the data packet. Therefore, we multiply the bee generation rate of every node with this constant to specify the degree to which *bee agents* load the network.

Let β_i be the rate of bee traffic entering the network at node i. The values for all $\beta_i, i = 1, 2, \cdots, |\mathcal{A}| \wedge i \neq \mathcal{D}$ are collected in $\big(|\mathcal{A}|-1\big)$ dimensional vector represented by $\boldsymbol{\beta}$. The bee generation rate is assumed to be constant during an iteration x. *Bee agents* explore the complete network instead of just sampling the best paths. The flow of *bee agents* at node i, $\gamma_i(x)$, is given by:

$$\gamma_i(x) = \beta_i + \sum_{m \in \mathcal{N}_i} \gamma_m(x)\mathfrak{b}_{mi}(x) \qquad i=1,2,\cdots,|\mathcal{A}| \wedge i \neq \mathcal{D} \quad (22)$$

The solution of the recursive or iterative functions is complex because a large number of simultaneous equations need to be reduced and simplified by algebraic techniques. Therefore, we use matrices which are simpler to handle. Solution to (22) is given by:

$$\boldsymbol{\gamma}(x) = \boldsymbol{\beta}\big(I - B^t(x)\big)^{-1} \quad (23)$$

where $\mathcal{B}^t(x)$ represents the transpose of $\mathcal{B}(x)$ and $\boldsymbol{\gamma}(x)$ is an $\big(|\mathcal{A}|-1\big)$ dimensional vector that contains the flow rates of *bee agents* of all nodes except $\mathcal{D}$. The proof of (23) is given in the appendix.

2) Data traffic: The nodes in the network also generate data packets that are destined to $\mathcal{D}$. Different packet generation rates can be assigned to each node in the network. Let ξ_i be the rate at which data packets are sent into the network by node i. The values for all $\xi_i, i = 1, 2, \cdots, |\mathcal{A}| \wedge i \neq \mathcal{D}$ are collected in an $\big(|\mathcal{A}|-1\big)$ dimensional vector represented by $\boldsymbol{\xi}$. The data generation rate is considered constant during different iterations (x).

Note that every node i acts as a forwarding node for the data packets, generated by the neighbors of i. However it will act only as a sink node if data packets are destined for it. As a result, we derive flows at nodes as probabilistic recursive functions. These functions model goodness of the links of a node. As a consequence, the rate of packet arrival and departure is dependent on the goodness of incoming and outgoing links respectively. The flow of data at node i, $\eta_i(x)$, is given by:

$$\eta_i(x) = \xi_i + \sum_{m \in \mathcal{N}_i} \eta_m(x) g_{mi}(x) \qquad i=1,2,\cdots,|\mathcal{A}| \wedge i \neq \mathcal{D} \quad (24)$$

Solution to (24) is given by:

$$\boldsymbol{\eta}(x) = \boldsymbol{\xi}\big(I - \mathcal{G}^t(x)\big)^{-1} \quad (25)$$

where $\boldsymbol{\eta}(x)$ is also an $\big(|\mathcal{A}|-1\big)$ dimensional vector that contains the flow rates of data packets of all nodes except $\mathcal{D}$.

B. Link flows

Once the node flow rate (number of data packets being forwarded by a node in unit time) of a node is known then its corresponding link flow rate (amount of data flowing on the link in unit time) can be calculated as a function of its neighbors and their goodness. These are represented as:

$$\mathfrak{bf}_{in}(x) = \gamma_i(x)\mathfrak{b}_{in}(x) \quad (26)$$

$$\mathfrak{df}_{in}(x) = \eta_i(x) g_{in}(x) \quad (27)$$

where $\mathfrak{bf}_{in}(x)$ and $\mathfrak{df}_{in\mathcal{D}}(x)$ represent the bee link flow rate and data link flow rate respectively from node i to node n. Total link flow rate, υ_{in}, can be calculated as:

$$\upsilon_{in}(x) = \mathfrak{bf}_{in}(x) + \mathfrak{df}_{in}(x) \quad (28)$$

C. Calculation of delays

In computer networks the arrival of packets is assumed to follow Poisson distribution and the queues that are built as a consequence are M/M/1 queues [13][14][15]. Moreover the service or departure rate can be pre-assigned because they are independent of the network conditions. We can calculate the queuing delay associated with every traversed link by utilizing the M/M/1 queuing theory if we know the arrival and departure rates. Representing the service rate of link (i, n) by S_{in}, we have the result [13]:

$$q_{in}(x) = \begin{cases} \frac{1}{S_{in} - \upsilon_{in}(x)} & \upsilon_{in} < S_{in} \\ \infty & \upsilon_{in} \geq S_{in} \end{cases} \quad (29)$$

The total link delay ld_{in}(x) experienced by the packets to traverse a link (i, n) can be found either by using the approach given in (2) or (3).
Using the approach given in (2) we get:

$$ld_{in}(x) = \Big(\frac{1}{p_{in}}(e^{-\frac{q_{in}(x)}{p_{in}}}) + \frac{1}{q_{in}(x)}(1 - e^{-\frac{q_{in}(x)}{p_{in}}})\Big)^{-1} \quad (30)$$

Using the approach given in (3) we get:

$$ld_{in}(x) = p_{in} + q_{in}(x) \quad (31)$$

where

$$p_{in} = tx_{in} + pd_{in}$$

tx_{in} and pd_{in} are the transmission and propagation delays respectively of the link between node i and node n.

Once the packet has reached at an intermediate node, n, again it might have multiple paths to reach the destination node through its neighbors and we need to calculate the delays for all possible routes from n to the destination $\mathcal{D}$. Again using the concept of recursive probabilistic functions we get:

$$td_{in}(x) = ld_{in}(x) + cd_{n\mathcal{D}}(x) \tag{32}$$

where $cd_{n\mathcal{D}}(x)$ is the cumulative delay experienced by a packet in order to reach $\mathcal{D}$ from n through any neighbors of n. It is given by:

$$cd_{n\mathcal{D}}(x) = \sum_{k\in\mathcal{N}_n} \Big(\alpha\mathfrak{b}_{nk}(x)+(1-\alpha)g_{nk\mathcal{D}}(x)\Big)\Big(ld_{in}(x)+cd_{k\mathcal{D}}(x)\Big) \quad n=1,2,\cdots,|\mathcal{A}|\wedge i\neq\mathcal{D} \tag{33}$$

The solution to (33) is given by:

$$\boldsymbol{cd}(x) = \Big(I - \alpha\mathcal{B}(x) - (1-\alpha)\mathcal{G}(x)\Big)^{-1}\boldsymbol{\delta}(x) \tag{34}$$

$\boldsymbol{\delta}(x)$ is given by:

$$\delta_n(x) = \sum_{k\in\mathcal{N}_n} \big(\alpha\mathfrak{b}_{nk}(x) + (1-\alpha)g_{nk\mathcal{D}}(x)\big)ld_{nk}(x) \quad n=1,2,\cdots,n(\mathcal{A})\wedge i\neq\mathcal{D} \tag{35}$$

where $\boldsymbol{\delta}(x)$ and $\boldsymbol{cd}(x)$ are vectors of dimensions $\big(|\mathcal{A}|-1\big)$ each. The solution similar to (34) is derived in the appendix.

The values of $\boldsymbol{cd}(x)$ can be obtained from (34), values of $td_{in}(x)$ can be obtained from (32) and we get a new value of goodness by using (1). Here we give it as:

$$g_{in}(x+1) = \begin{cases} \frac{\frac{1}{td_{in}(x)}}{\sum_{k\in\mathcal{N}_i}(\frac{1}{td_{ik}(x)})} & (i,n)\in\mathcal{L} \\ 0 & i=n\vee(i,n)\notin\mathcal{L} \end{cases} \tag{36}$$

So starting from a goodness value in (19) we reached (36) which is the goodness value for the next iteration.

In the case when link flow rate becomes greater than the service rate, the value of $q_{in}\to\infty$ and consequently $td_{in}\to\infty$. Thus the goodness factors given by (19) and (36) can not be calculated from these equations. In order to keep the model flexible under such circumstances, we take $g_{in}=\mathfrak{b}_{in}$ for this particular link (i,n) until link flow rate again becomes smaller than the service rate.

D. Throughput

Finally, we present our model for calculating the throughput of the algorithm.

$$\mathcal{T}_{in} = \frac{\text{effective data transferred on } (i,n)}{ld_{in}(x)}$$

The expression on right hand side is actually the data flow rate of the link (i,n) given by (27), so throughput is given by:

$$\mathcal{T}_{in} = \mathfrak{df}_{in}(x) \tag{37}$$

V. Empirical verification of formal model

We now verify our model on two topologies. The first topology is a 4 node square topology shown in Figure 2 while the other one is SimpleNet (see Figure 5). In first topology major emphasis is on studying the impact of relevant behavior of *BeeHive* by changing the data packet rates and service rates of nodes and links respectively on the goodness of the links. In the second example we compare the results obtained from our formal model with the ones obtained from extensive simulations of *BeeHive* protocol in OMNeT++. The values obtained in every iteration is averaged using exponential moving average. This is done to dampen the oscillations in the graph. This exponential moving average is taken in the simulation of formal model as well as in the results obtained from OMNeT++.

A. Example 1

The topology of Figure 2 is simple. For this particular

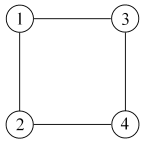

Fig. 2. Topology 1

example we assume that node 4 is the destination node and the remaining nodes can send data to this particular node. First we use the following parameters for our simulation:

- *transmission + propagation delay, p_{in} = 0.1*
- *ratio of size of bee packet to data packet,* $\alpha = 0.01$
- *bee generation rates of nodes,* $\boldsymbol{\beta} = [2\ 2\ 2\ 0]$
- *data generation rates of nodes,* $\boldsymbol{\xi} = [5\ 8\ 0\ 0]$
- *service rate of links,* $S_{in} = 10\ \ \forall\ \ (i,n)\in\mathcal{L}$
- *starting value of total delay,* $td_{in}(0) = 1$

We now show a single iteration of the model that will provide a better insight to the reader about the model. For the given topology, using (16) and (17) gives:

$$B(1) = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 \\ 0.5 & 0 & 0 \end{bmatrix}$$

Substituting $td_{in}(0)=1$ in (19) gives $g_{in}(1)$. When $g_{in}(1)$ is substituted in (18), resulting matrix $\mathcal{G}(1)$ is:

$$\mathcal{G}(1) = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 \\ 0.5 & 0 & 0 \end{bmatrix}$$

Equations (22) to (25) can be utilized to find the vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$.

$$\gamma(1) = \begin{bmatrix} 8 & 6 & 6 \end{bmatrix}$$

$$\eta(1) = \begin{bmatrix} 18 & 17 & 9 \end{bmatrix}$$

Evaluation of (26) to (28) and (31) gives:

$$Ld(1) = \begin{bmatrix} 0 & 1.1417 & 1.1417 \\ 0.7803 & 0 & 0 \\ 0.2828 & 0 & 0 \end{bmatrix}$$

Td is estimated by using (32).

$$Td(1) = \begin{bmatrix} 0 & 1.1298 & 1.1049 \\ 1.1563 & 0 & 0 \\ 1.1315 & 0 & 0 \end{bmatrix}$$

Substituting Td(1) in (36) gives the value of goodness factor for the next iteration.

$$\mathcal{G}(2) = \begin{bmatrix} 0 & 0.4944 & 0.5056 \\ 0.4610 & 0 & 0 \\ 0.4601 & 0 & 0 \end{bmatrix}$$

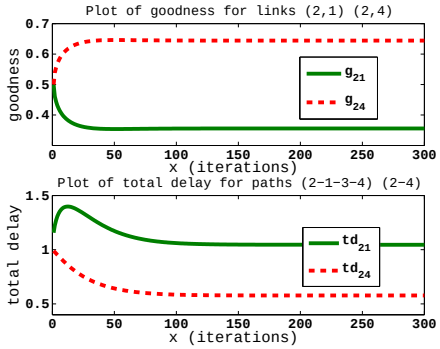

Fig. 3. Plots of goodness and delay

One can keep on iterating the above mentioned steps to arrive at the steady state values of the goodness,

$$\mathcal{G}(\infty) = \begin{bmatrix} 0 & 0.4657 & 0.5343 \\ 0.3558 & 0 & 0 \\ 0.1861 & 0 & 0 \end{bmatrix}$$

From Figure 3 we see that the goodness for the link $(2,1)$ is less than that for $(2,4)$ as the total delay for the packets to reach destination from 2 by traversing the link $(2,4)$ is significantly less than following the path $(2 \rightarrow 1 \rightarrow 3 \rightarrow 4)$. This phenomenon is discussed in more detail in the next example where we show that the probability that a packet follows loops is small.

If node 1 has some data to send, then it can utilize two paths to route its data packets i.e. one path is $(1 \rightarrow 2 \rightarrow 4)$ and the second one is $(1 \rightarrow 3 \rightarrow 4)$. From the pre-assigned data generation rates of node 2 and 3, the goodness of the link $(1,3)$ is expected to be more than $(1,2)$.

The results of our formal model in Figure 4-A verify the above mentioned intuitive logic. The goodness for link $(1,3)$ stabilizes at approximately 0.534 while of link $(1,2)$ stabilizes at 0.466. So more data is sent on the link $(1,3)$ in order to achieve better performance.

Now we take new parameters for another simulation.

- *transmission + propagation delay,* p_{in} *= 0.1*
- *ratio of size of bee packet to data packet,* $\alpha = 0.01$

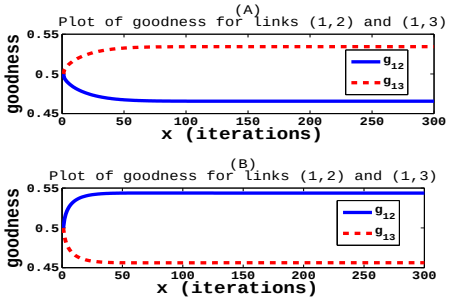

Fig. 4. goodness for links (1,2) and (1,3)

- *bee generation rates of nodes,* $\boldsymbol{\beta} = [2\ 2\ 2\ 0]$
- *data generation rates of nodes,* $\boldsymbol{\xi} = [5\ 2\ 2\ 0]$
- *service rate of link,* $(1,3)$ $S_{13} = 8$
- *service rate of links,* $S_{in} = 15 \ \ \forall \ (i,n) \in \mathcal{L} \setminus (1,3)$
- *starting value of total delay,* $td_{in}(0) = 1$

For these parameters we can see that service rate is less for the queues of the link $(1,3)$ as compared to the link $(1,2)$. As both the nodes 2 and 3 have the same data traffic rates, delay experienced via path $1 \rightarrow 3 \rightarrow 4$ should be more than $1 \rightarrow 2 \rightarrow 4$ because the service rate of node 3 is smaller as compared to the service rate of other nodes. As a result the goodness of the link $(1,3)$ is lesser then the link $(1,2)$. One can clearly see from Figure 4-B that the goodness of both paths, estimated by our formal model, is as expected.

B. Example 2

Now we verify our model on a well known topology, SimpleNet, shown in Figure 5. We compare the results obtained from our model with the results obtained from extensive simulations in OMNeT++.

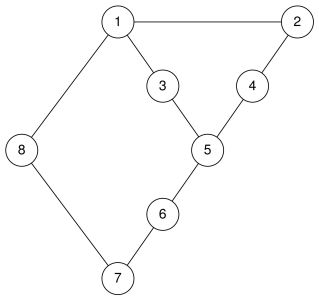

Fig. 5. SimpleNet

Following parameters are used to obtain the results of formal model:

- *transmission + propagation delay,* p_{in} *= 0.1*
- *ratio of size of bee packet to data packet,* $\alpha = 0.01$
- *bee generation rates of nodes,* $\boldsymbol{\beta} = [3\ 2\ 2\ 2\ 3\ 2\ 0\ 2]$
- *data generation rates of nodes,* $\boldsymbol{\xi} = [2\ 2\ 2\ 2\ 2\ 2\ 0\ 2]$
- *service rate of links,* $S_{in} = 34 \ \ \forall \ \ (i,n) \in \mathcal{L}$
- *starting value of total delay,* $td_{in}(0) = 1$
- *destination node,* $\mathcal{D} = 7$

The Figures 6 and 7 compare the delays of the paths from node 1 to node 7 via node 3 and from node 1 to node 7 via node 8 respectively. One can see that the values of the delay estimated by our formal model are approximately the same as obtained from the network simulator.

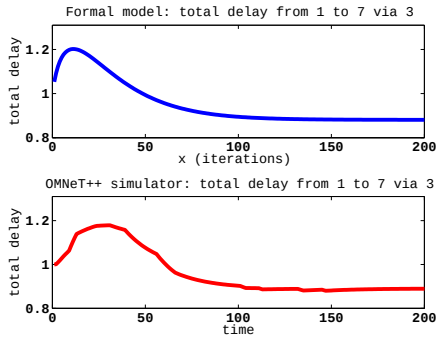

Fig. 6. Total delay experienced by a packet to reach node 7 from 1 via 3

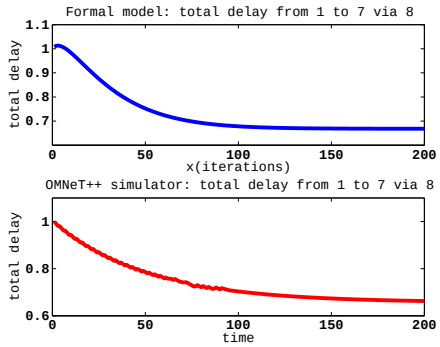

Fig. 7. Total delay experienced by a packet to reach node 7 from 1 via 8

In Figure 8 the goodness of the link $(1,3)$ is graphically depicted. Again the results obtained from our formal model are validated by the OMNeT++ simulations. In Figure 9 we

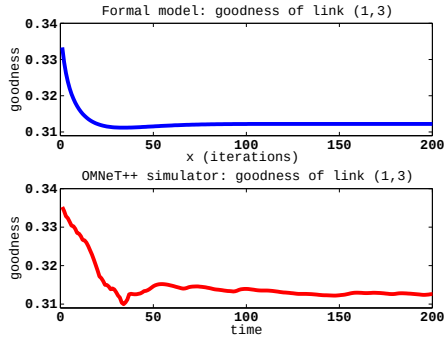

Fig. 8. Goodness of link $(1,3)$

compare the goodness of link $(1,8)$. The steady state value of goodness in both cases is 0.41. The goodness of link $(1,2)$ can be found by subtracting the sum of goodnesses of links $(1,3)$ and $(1,8)$ from 1 i.e $(1-(0.41+0.31))=0.28$.

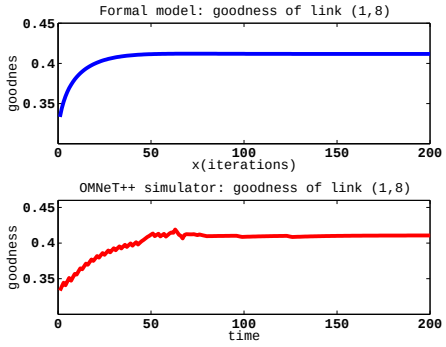

Fig. 9. Goodness of link $(1,8)$

Finally, we show that the probability that a packet will enter into loops is significantly small in *BeeHive*. In this

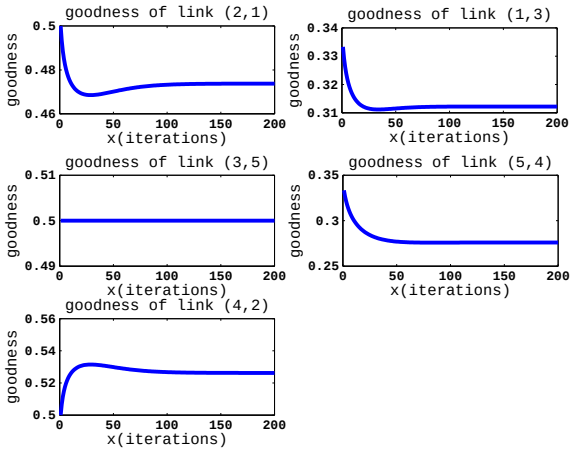

Fig. 10.

topology we consider the loop $(2\rightarrow 1\rightarrow 3\rightarrow 5\rightarrow 4\rightarrow 2)$ and find out the probability that a packet originating at node 2 and traveling towards the destination node 7 will follow this cyclic path. From Figure 10 we can find the steady state goodness of all these links. Multiplying the steady state values of goodness gives the probability that a packet will move in this loop. The resulting probability comes out to be 0.01063 which is significantly small and it shows that only 1% packets can follow cyclic paths.

Finally we compare the cumulative throughput of the system obtained by the formal mathematical model and the OMNeT++ simulation. In Figure 11 one can again see that the results from the model match with the results of the simulation.

VI. Conclusions

In this paper we have introduced a formal framework for analyzing a well known Nature-inspired routing protocol, *BeeHive*. We have used probabilistic recursive functions for analyzing online stochastic packet switching behavior of the algorithm. The queuing delays experienced due to the

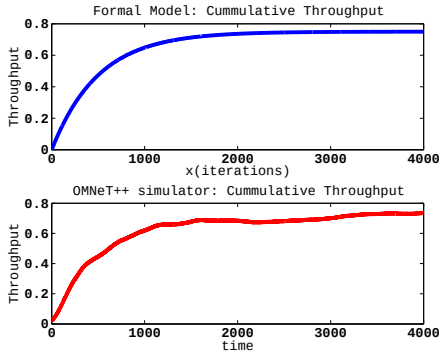

Fig. 11. Throughput

congestion have been analyzed using the formal concepts of (M/M/1) queuing theory. The formal model is simple yet general enough to capture the relevant features of *BeeHive* protocol.

We have also developed an empirical verification framework in OMNeT++ to validate the correctness of our formal model. We validated our formal model on two topologies and compared its results with the results obtained from the extensive simulations. The estimated performance values of the model have only a small deviation from the real values measured in the network simulator.

In future we want to extend the framework to model the formation of regions and zones. *BeeHive* utilized these relevant concepts for scalability. Once these relevant concepts are formally modeled then we can test our framework on large topologies like Japanese NTTNeT. Our extended model and its validation on large topologies will be the subject of the forthcoming publications.

APPENDIX

Theorem 1:
The solution of the equation

$$\gamma_i \quad = \quad \beta_i + \textstyle\sum_{m\in\mathcal{N}_i} \gamma_m \mathfrak{b}_{mi} \qquad i=1,2,\cdots,|\mathcal{A}| \wedge i \neq \mathcal{D} \qquad (38)$$

is given by,

$$\boldsymbol{\gamma} = \left(I - B^t\right)^{-1} \boldsymbol{\beta}$$

Proof:

$$\gamma_i \quad = \quad \beta_i + \textstyle\sum_{m\in\mathcal{N}_i} \gamma_m \mathfrak{b}_{mi} \qquad i=1,2,\cdots,|\mathcal{A}| \wedge i \neq \mathcal{D} \qquad (39)$$

Substituting $i = 1, 2, \cdots, |\mathcal{A}| \wedge i \neq \mathcal{D}$ in (39) and using from (17) that $\mathfrak{b}_{in} = 0$ when $i = n \vee (i,n) \notin \mathcal{L}$

$$\begin{array}{rcccccc}
\gamma_1 & = & \beta_1 + \gamma_1 \times 0 + \gamma_2 \mathfrak{b}_{21} + \cdots + \gamma_{|\mathcal{A}|} \mathfrak{b}_{|\mathcal{A}|1} \\
\gamma_2 & = & \beta_2 + \gamma_1 \mathfrak{b}_{12} + \gamma_2 \times 0 + \cdots + \gamma_{|\mathcal{A}|} \mathfrak{b}_{|\mathcal{A}|2} \\
:: & & :: & :: & :: & :: & :: \\
:: & & :: & :: & :: & :: & :: \\
\gamma_{|\mathcal{A}|} & = & \beta_{|\mathcal{A}|} + \gamma_1 \mathfrak{b}_{1|\mathcal{A}|} + \gamma_2 \mathfrak{b}_{2|\mathcal{A}|} + \cdots + \gamma_{|\mathcal{A}|} \times 0
\end{array}$$

On the R.H.S of all the equations given above there is always a term that becomes zero. Moreover, $\gamma_{\mathcal{D}}$ is also zero. So there are $(|\mathcal{A}|-1)$ equations each having $(|\mathcal{A}|-1)$ terms involving γ_{in}.

Rearranging above equations we get:

$$\begin{array}{rcccccc}
\beta_1 & = & \gamma_1 - \gamma_2 \mathfrak{b}_{21} - \gamma_3 \mathfrak{b}_{31} - \cdots - \gamma_{|\mathcal{A}|} \mathfrak{b}_{|\mathcal{A}|1} \\
\beta_2 & = & -\gamma_1 \mathfrak{b}_{12} + \gamma_2 - \gamma_3 \mathfrak{b}_{32} - \cdots - \gamma_{|\mathcal{A}|} \mathfrak{b}_{|\mathcal{A}|2} \\
:: & & :: & :: & :: & :: & :: \\
:: & & :: & :: & :: & :: & :: \\
\beta_{|\mathcal{A}|} & = & -\gamma_1 \mathfrak{b}_{1|\mathcal{A}|} - \gamma_2 \mathfrak{b}_{2|\mathcal{A}|} - \gamma_3 \mathfrak{b}_{3|\mathcal{A}|} - \cdots + \gamma_{|\mathcal{A}|}
\end{array}$$

Representing the above set of equations in matrix form, we obtain:

$$\begin{bmatrix} \beta_1 \\ \beta_2 \\ :: \\ :: \\ \beta_{|\mathcal{A}|} \end{bmatrix} = \begin{bmatrix} 1 & -\mathfrak{b}_{21} & -\mathfrak{b}_{31} & \cdots & -\mathfrak{b}_{|\mathcal{A}|1} \\ -\mathfrak{b}_{12} & 1 & -\mathfrak{b}_{32} & \cdots & -\mathfrak{b}_{|\mathcal{A}|2} \\ :: & :: & :: & \cdots & :: \\ :: & :: & :: & \cdots & :: \\ -\mathfrak{b}_{1|\mathcal{A}|} & -\mathfrak{b}_{2|\mathcal{A}|} & -\mathfrak{b}_{3|\mathcal{A}|} & \cdots & 1 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ :: \\ :: \\ \gamma_{|\mathcal{A}|} \end{bmatrix}$$

Using the symbols of these matrices defined in section IV we obtain:

$$\boldsymbol{\beta} = \left(I - B^t\right)\boldsymbol{\gamma} \qquad (40)$$

Pre-multiplying both sides of (40) by $\left(I - B^t\right)^{-1}$ we finally obtain our required solution.

$$\boldsymbol{\gamma} = \left(I - B^t\right)^{-1}\boldsymbol{\beta}$$

REFERENCES

[1] H. F. Wedde and M. Farooq, "A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks." *Journal of Systems Architecture*, vol. 52, no. 8-9, pp. 461–484, 2006.

[2] T. Seeley, *The Wisdom of the Hive*. London: Harvard University Press, 1995.

[3] H. F. Wedde and M. Farooq, *Intelligent Network Traffic Engineering through Bee-inspired Natural Protocol Engineering*. Natural Computing Series, Springer Verlag, in press.

[4] G. D. Caro and M. Dorigo, "Antnet: Distributed stigmergetic control for communications networks," *Journal of Artificial Intelligence Research (JAIR)*, vol. 9, pp. 317–365, 1998.

[5] H. Wedde, M. Farooq, and Y. Zhang, "BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior," in *Ant Colony Optimization and Swarm Intelligence, LNCS 3172*. Springer Verlag, Sept 2004, pp. 83–94.

[6] S. Liang, A. Zincir-Heywood, and M. Heywood, "Intelligent packets for dynamic network routing using distributed genetic algorithm," in *Proceedings of Genetic and Evolutionary Computation Conference*. GECCO, July 2002.

[7] H. Wedde and M. Farooq, "A performance evaluation framework for nature inspired routing algorithms," in *Applications of Evolutionary Computing, LNCS 3449*. Springer Verlag, March 2005, pp. 136–146.

[8] N. Bean and A. Costa, "An analytic modelling approach for network routing algorithms that use "ant-like" mobile agents." *Computer Networks*, vol. 49, no. 2, pp. 243–268, 2005.

[9] A. Varga, *OMNeT++: Discrete event simulation system: User manual*.

[10] J. Mehdi, *Stochastic Processes*. John Wiley Publication, 1983.

[11] P. W. Jones and P. Smith, *Stochastic Processes: An Introduction*. NewYork, USA: Oxford University Press Inc., 2001.

[12] S. Ross, *Stochastic Processes*. John Wiley Publication, 1983.

[13] H. Taha, *Operations Research*. John-Wiley & Sons, 1982.

[14] K. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley - Interscience Publication, 2002.

[15] F. Kelly, S. Zachary, and I. Zeidins, *Stochastic Networks Theorey and Applications*. Oxford Science Publications, 1996.