Decentralized algorithms for evaluating centrality in complex networks

Katharina A. Lehmann, Michael Kaufmann

***Abstract*— Centrality indices are often used to analyze the functionality of nodes in a communication network. Up to date most analyses are done on static networks where some entity has global knowledge of the networks properties. To expand the scope of these analyzing methods to decentral networks we will propose here a general framework for decentral algorithms that calculate centrality indices. We will describe variants of the general algorithm to calculate four different centralities, with emphasis on the algorithm of the betweenness centrality. The betweenness centrality is the most complex measure and best suited for describing network communication based on shortest paths and predicting the congestion sensitivity of a network.**
The communication complexity of this latter algorithm is asymptotically optimal and the time complexity scales with the diameter of the network.
The calculated centrality index can be used to adapt the communication network to given constraints and changing demands such that the relevant properties like the diameter of the network or uniform distribution of energy consumption is optimized.

***Index Terms*— Centrality Indices, distributed algorithm, Self-diagnostic networks.**

I. Introduction

SINCE the first days of telecommunications the world has changed dramatically. After a long period where a letter would take days to be delivered the invention of cable and telephone made instant communication possible and achievable for many people. To grant this service a widespread network of cables had to be built that is being remodeled periodically. However, due to the tremendous costs it had never been possible to rebuild the whole hardwired system despite the fact that the demands on communication have changed drastically in the last decades.
In contrast to this wireless communication networks and fast changing peer-to-peer networks give the ability to choose the appropriate network architecture. Power efficient and stable networks are accomplishable if this architecture is dynamically changed, according to the demands of the user.
But up to date most protocols for peer-to-peer-networks just randomly choose the set of peers with which data is shared directly. Due to this, the resulting communication network has random features that hamper e.g. efficient searching algorithms. A first attempt to overcome this random character of the network has been proposed by Ng and Sia [3]: every participant of the peer-to-peer network holds a list of the kind of data packages he is interested in. A new participant chooses some entry point and then walks randomly through the existing network to find persons with similar interests. Then the participant chooses some of them for building up so called *quality links* and additionally some random links to guarantee the connectivity of the whole network. By this procedure every participant will be clustered into a subnet of persons with similar interests. A fast searching algorithm is now able to use the long random links in areas that are not likely to contain the required data and to switch to a broadcasting protocol in subclusters of persons with similar interests to the query.

The appropriate network architecture will also be extremely important in so called wireless multi-hop ad hoc communication networks. In classic mobile networks communication packets are routed over a centralized backbone system of base-stations while in multi-hop networks the mobile devices have the routing functionalities.
Each device is able to communicate directly with a part of the adjacent neighbors and provides routing services for communication between non-adjacent devices. There are two strict constraints that have to be considered: One is the limitation of resources such as energy or bandwidth, the second is the lack of any global information. To generate stable and efficient network architectures special algorithms have to be developed that are working decentrally and allow self-organization of the system. In [4] the following local organisation rule is proposed: Every new node will send a so called hello messages to its direct neighbors within a specified transmission radius to get connected to them. If the number of neighbors is too small within this radius it enhances its transmission power until a given minimal number k of neighbors is achieved. Every node receiving a hello message from a new node is forced to answer with a hello-reply message even if it already has the minimal number of neighbors. We will refer to this model in the following as *k-next-neighbors model*. The resulting network will be connected for reasonable high k with high probability and most devices will not have many more than k neighbors.

Both introduced decentral approaches generate quite functional networks but the main problem is that they tend to disfavour some nodes in a severe manner. In the first model commonly known entry points will have a higher probability of gaining too many links and in the *k-nearest-neighbors model* the special geometry of the network is disregarded. Since the network results in a grid like architecture routing on approximately shortest paths will affect inner nodes much more than nodes on the border of the network. That is, energy

The authors are with Parallel Computing, Wilhelm-Schickard-Institut für Informatik, Eberhard-Karls-Universität Tübingen, Sand 14, 72076 Tübingen, Germany. Email: lehmannk/mk@informatik.uni-tuebingen.de

consumption is not uniformly distributed but allows higher demands in some regions while relieving others unproportionally (s. Fig. 6(a)).

To overcome this situation the evaluation of the current network architecture is needed, followed by some appropriate adaption if necessary. A common way to determine the ability of a network to provide efficient communication is the calculation of centrality indices, first introduced in the social network analysis [5], [6]. One of the most prominent of such indices is the betweenness-centrality index that has become an important analysis tool in fields as diverse as epidemiology, design of router networks, load of communication networks and co-citation networks [7], [8], [9], [10], [11]. It measures which proportion of communications on shortest paths would be afflicted by a removal of a given node in the graph and was first proposed by Freeman [2]. It has been shown that this static feature of a network has a non-trivial inter-dependency with the load of the communicating nodes [10]. As a static feature of the network it gives a good approximation of the network's sensitivity to congestion. As such it can be used for the routing protocols to avoid paths with high congestion probability or to reconstruct the connections from individual nodes to optimize the network structure.

The improvement of communication networks by evaluating the centrality and appropriate adaptation has been first proposed in [9] but the author did not provide any decentral algorithm with which the nodes could do the evaluation themself.

We want to propose here a general framework that can be used for the calculation of different centrality indices that are based on shortest paths and distances in the network. The emphasis of this paper is on the calculation of the betweenness-centrality index which gives most information on the structure of the network. We will show that the proposed algorithm for this latter centrality index is asymptotically optimal in its communication complexity and that its time complexity scales with the diameter of the network.

We will further introduce the concept of evolving networks. By a periodically evaluating and subsequently adaptation the network is globally optimized. We will provide a proof of concept that illustrates the flexibility of evolving networks.

The model is applicable to many networks first of all wireless multi-hop ad-hoc communication networks, but also router networks, peer-to-peer networks, upcoming e-government systems, and even biological networks.

The remainder of the paper is organized as follows: in Section 2 we define the setting of a wireless multi-hop system and describe a general framework for all centrality calculating algorithms. Section 3 gives the details of the algorithms for four classic centrality indices. We will further analyze their respective complexity and the decentrality, scalability and reliability of the algorithms. In Section 4 we will discuss how the evaluation of centrality can be used to optimize the global network architecture.

II. General Framework

A. The Model: Definitions and Settings

1) General terms: A *wireless multi-hop ad hoc communication network* consists of a set of communication devices that are connected by communication protocols that enable direct communication between devices within a given distance. In the following we will describe such a network in the following terms:

Let $G = (V, E)$ be a graph with undirected edges, without multi-edges. Every node $v \in V$ represents one device, every edge $e = (v, w) \in E$ represents a direct communication between nodes v and w. The *number n of nodes* is defined as the cardinality of V, the *number m of edges* is defined as the cardinality of E. A node w is a *neighbor* of node v if both are directly communicating with each other. We assume that every node maintains a list $N(v)$ of its neighbors and that the graph is always connected. The *degree $d(v)$* of node v is defined as the number of its *neighbors*.

A *path $p(s,t)$* from a *source node s* to a *target node t* is defined as a set of nodes $\{v_1, v_2, ..., v_k\}$ such that $(s, v_1), (v_k, t)$ and all (v_i, v_{i+1}) are directly connected. The *hop length $l(p(s,t))$* of a *path p* denotes the number of edges on it, that is $l(p(s,t)) = k + 1$.

The *distance $d(s,t)$* between two nodes s and t is defined as the minimum *hop length* of any *path* between s and t. Every *path* with minimum hop distance length will be called *shortest path* between s and t.

A node v is called *predecessor* of node t with respect to some *source node s* if v and t are neighbors and there is at least one *shortest path* between s and t running over v. Vice versa, t is called *successor* of node v. *Neighbors* that are neither *predecessors* nor *successors* with respect to a given *source node s* are called *neutral*. $P_s(t)$ denotes the set of all *predecessors* of node t with respect to *source node s*.

The *number of shortest paths* between s and t will be denoted by σ_{st}, with $\sigma_{ss} := 1$. The *number of shortest paths p between s and t that run over node v*, that is $v \in p(s,t)$, is denoted by $\sigma_{st}(v)$. The probability $p_{st}(v)$ of v lying on a randomly picked shortest path from s to t is given by $p_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$.

The *diameter of the graph D_G* is defined as the length of the longest shortest path between any two nodes of the graph.

The *transmission radius $r(v)$* of a node v is defined as the furthest geographical distance to any of its *neighbors*. Every node v has a unique identification string $ID(v)$.

2) Synchronization: We assume that the system can be synchronized. The basic idea is that each communication device has a *maximal possible transmission radius R*. Let T be the time needed for any message to travel R. Such, any message sent at *time t* will be received by all *neighbors* at the latest at time $t + T$. We further assume that any computation at any node can be done within a fixed time span T'. This assumption is based on the fact that the number of messages received at one time unit is restricted by the number of nodes in the system.

In summary, we assume that the receiving and computation of all current informations in the system is finished after some time span $T + T'$. With the introduction of a global clock it is now possible to synchronize the system. At a given starting time t_0 all nodes will send information to all neighbors. Every node will wait time interval T for messages from its *neighbors* and after that start computation on the received messages. The newly generated informations will be sent to all *neighbors* in time $t_0 + T + T'$ to be sure that every node has received all relevant messages and has finished every necessary computation. To simplify things the notion 'received (computed) in time t_i' will be a short cut for 'received (computed) in the time interval from time $t_0 + i(T + T')$ to time $t_0 + (i + 1)(T + t') - 1$'.

3) Analysis of algorithms: The *communication complexity* C of an algorithm is defined as the number of elementary messages that are sent throughout the algorithm (adapted from [15]). The *time complexity* T of an algorithm is defined as the number of time units required until the algorithm terminates [15]. The *package complexity* P of an algorithm is defined as the number of *packages* that have to be sent throughout the algorithm. A *package* is defined as some set of elemantary messages that can be sent together in one message.

B. Centrality indices based on shortest paths and distances

The following centrality measurements are based on shortest paths and therefore meaningful in communication networks:

- **Closeness Centrality** [12]

$$C_C(v) = \frac{1}{\sum\limits_{t \in V} d(t, v)} \tag{1}$$

- **Graph Centrality** [13]

$$C_G(v) = \frac{1}{\max\limits_{t \in V} d(t, v)} \tag{2}$$

- **Stress Centrality** [14]

$$C_S(v) = \sum_{s \in V} \sum_{\substack{t \in V, \\ s \neq t}} \sigma_{st}(v) \tag{3}$$

- **Betweenness Centrality** [2]

$$C_B(v) = \sum_{s \in V} \sum_{\substack{t \in V, \\ s \neq t}} p_{st}(v) \tag{4}$$

Centrality indices are designed such that the highest value indicates the most central node. An example graph with these centrality indices can be seen in Fig. 1

The interpretation of the meaning of this centrality indices is based on two assumptions. First, most routing protocols try to establish communications on shortest paths. Second, the hop distance is a (scaled) approximation for the real length of the path between two communicating nodes. Under this assumptions, the **Closeness Centrality** gives an indication how long it would take the node to communicate in a serial manner to all other nodes in the network. The **Graph Centrality** indicates how long the parallel communication to all other nodes would take at most. The **Stress Centrality** is measuring on how many shortest paths a node lies.

The **Betweenness Centrality** is the most complex measure. It sums over all probabilities $p_{st}(v)$. Such, it measures the expected routing service demands of node v if every node was communicating with every other simultaneously. This can be interpreted as an approximative congestion sensitivity of v.

C. Two phase model

The general framework for algorithms calculating these centralities consists of two phases. In the first phase, called **Count Phase**, the *distance* or *number of shortest paths* is calculated, respectively. In the second phase, called **Report Phase**, the according value is reported back to all other nodes. We will first describe the two phases and then show that both phases can be interlaced with each other. The general framework is sketched in Fig. II-C.

In the following, messages that carry new and relevant data will be called *relevant* messages. A message that contains data a node already has received earlier is called a *backfiring* message.

1) Count Phase: To describe the calculation of the number of shortest paths between any *source node* s and *target node* t we first have to state the following Lemmata.

Lemma *1:* A node v is on the shortest path from a *source node* s to a *target node* t if and only if
$d(s, t) = d(s, v) + d(v, t)$.

This follows directly from the definitions of *shortest paths* and *distance*.

Lemma *2:* The *number of shortest paths* σ_{st} from s to t is the sum of shortest paths from s to all predecessors p of t:

$$\sigma_{st} = \sum_{p \in P_s(t)} \sigma_{sp} \tag{5}$$

Lemma 2 follows from Lemma 1. We use the synchronization of the network as assumed above. All nodes start with the counting at the same global time t_0. At this starting time an initial *number of shortest paths (NOSP) message* is generated and sent to all neighbors. Every *NOSP message* contains the $ID(s)$ of the *source node* s and an integer number that denotes the number of shortest paths from s to the currently sending node. This number will be called the *weight* w of the message. The *weight* of the initial message will be 'one' since the *number of shortest paths* from a node to itself is defined to be one.

Based on Lemma 2, every node v will sum the weight of all messages from direct *predecessors* with respect to s. This sum is the *number of shortest paths* σ_{sv} from s to v and will be stored such that it can be retrieved by the $ID(s)$ of the *source node*. This newly computed *number of shortest paths* will be broadcasted with $ID(s)$ to all its neighbors in the next time unit. This includes also the broadcasting to the predecessors. Thereby, the message will be a *backfiring* message for some of the *neighbors*. To decide whether a message came from a *predecessor* and therefore is *relevant* or a *backfiring message* every node checks for all messages whether the $ID(s)$ already has an entry in their *number of shortest paths* table. If so, it ignores the according message and will not broadcast it further. By definition of the synchronization it is clear that in time t_k all nodes t with distance k will receive their first message

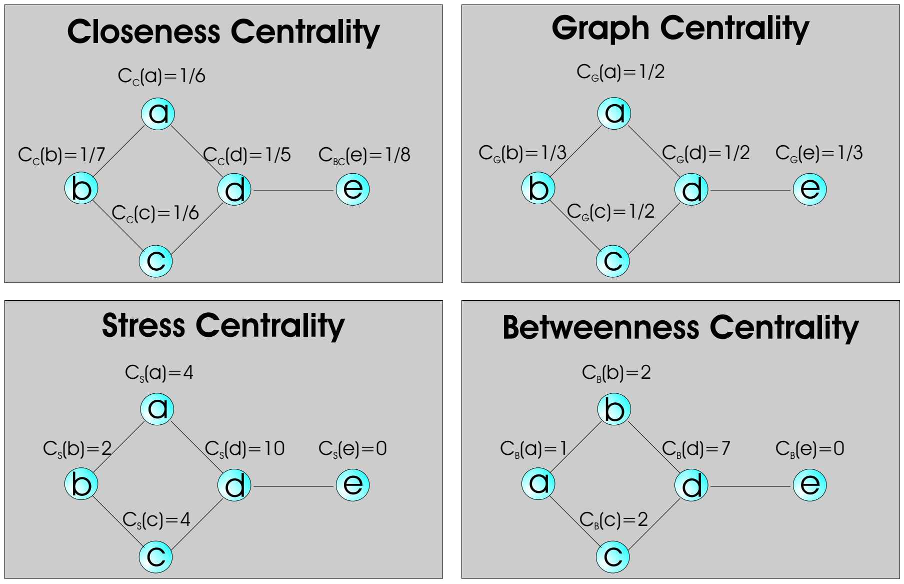


Fig. 1. The figure shows the four different centrality indices that are based on shortest paths and therefore are important for communication networks

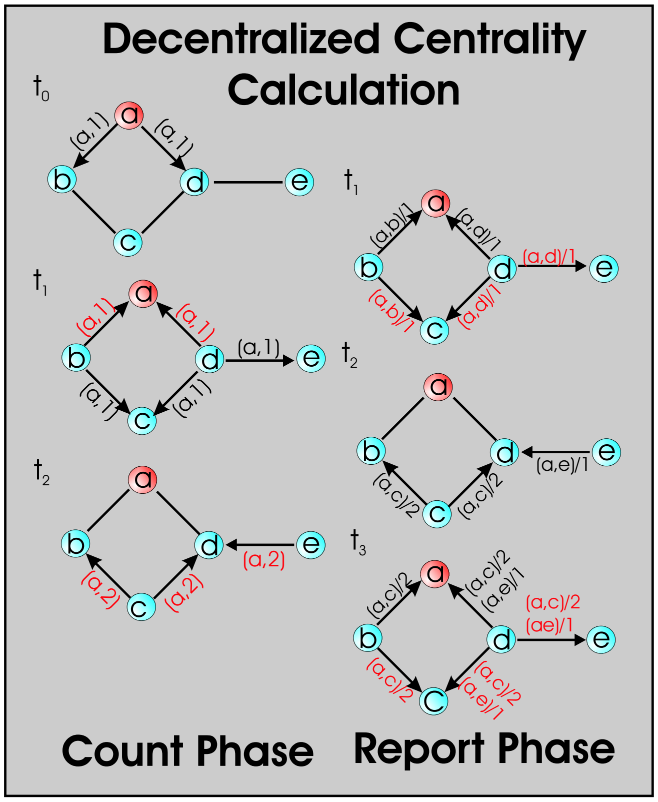


Fig. 2. The figure shows the general algorithm for calculating centralities in a decentralized manner at the example of calculating the number of shortest paths from node a to all other nodes. On the left side of the figure the Count Phase is shown, on the right the Report Phase. Red messages indicate *backfiring* messages that will be ignored by the receiving node. In time t_0 node a is sending its *NOSP message* to its neighbors b and d. b broadcasts it to a (ignored) and c in t_1. Additionally b sends a *report* message $(a,b)/1$ to a and c because it knows its *number of shortest paths* to a. This *report* message will be ignored by c since it is not yet in the Report Phase regarding *source node* a. Also node d will broadcast the *NOSP message* to a (ignored), c and e. Additionally, it sends the *report* message $(a,c)/1$ to all its neighbors. In time t_2 every node knows its *number of shortest paths* to a and nodes e and c start reporting it back. The last *report* message will reach node a at time t_4 and the algorithms terminates.

from *source node* s and are able to calculate their *number of shortest paths* to s.

With this observation the *hop distance* between any two nodes can be determined by storing the time t_i at which the first message with $ID(s)$ has been received. It follows that no *weight* is needed in a *NOSP message* if only the distance is calculated.

It is important to note that the *NOSP messages* from different *source nodes* do not interfere with each other. Thus, all *number of shortest paths* and *distances* can be calculated simultaneously. Moreover, every node can collect all relevant *NOSP messages* and send them to all its neighbors in packages. The elementary *NOSP messages* might then be coded as pairs of ($ID(s)$,*weight*).

2) Report Phase: As observed above every node t with *distance* k to any other *source node* s will know its *distance* and *number of shortest paths* in time t_k. To report the according value back every node t will create a *report message* with an ID-pair $(ID(t), ID(s))$ and as a *weight* the *distance* $d(s,t)$ or *number of shortest paths* σ_{st}, respectively. This *report message* is sent to all *neighbors* of t in time t_{k+1}.

A node v receiving a *report message* in time t_i will perform the appropriate computation to calculate its centrality and forward the message without changing it to all neighbors in time t_{i+1}. To avoid the forwarding of *backfiring* messages each node has to build up an array of size n^2. In this is stored whether the *report message* with a certain ID-pair has been received already. Every *distance* or *number of shortest paths* is uniquely identified by the *source* and *target* node and will only be reported once. It follows, that every *relevant report message* will have a different ID-pair. Since all pairs of nodes will report some value back an array of size n^2 is necessary and sufficient.
The detailed computations for the four different centralities introduced above will be described in more detail in the next section.

3) Interlacing of the Phases: All *NOSP messages* are independent from each other as are the *report messages* from each other. Of course, a *report message* can only be sent if the according *distance* or *number of shortest paths* is already calculated. But beside this dependency a node can be in different phases with respect to different *source nodes*. That is, a node v may at the same time be waiting for some *NOSP message* from some node s, sending the *report message* for some node s' and having finished the **Report Phase** for another node s'' two time units ago.

4) Analysis: We will first analyze the *time complexity* of the general algorithm and then determine the *communication* and *package complexity*.
Let $D(v)$ denote the furthest distance of any node t to v. Node v will receive its last *NOSP message* at time $t_{D(v)}$. It will send its last *report message* at time $t_{D(v)+1}$ and it will receive its last *report message* at time $t_{2\cdot D(v)+1}$. It is important to observe that until this time the node has received at least one *report message* every two time units. The reason for this is that there is some node in every distance d from 1 to k and that the time until the according *report message* gets back to v is twice the distance. With this observation, every node is able to determine the end of the calculation and afterwards react to the calculated value.
In a global view the algorithm will stop after $2 \cdot D_G + 1$ time units. That is, the *time complexity T* of any algorithm in this framework is $O(D_G)$.
It is very important to state that the *diameter* D_G of a graph is more depending on the geographical area that the networks spans than on the number of nodes in it. We assume that every node has an average transmission radius that depends very little on the number of nodes in the system. In this case, the expected diameter of the graph will be the diameter of the geographical area divided by the average transmission radius. Such, the *time complexity T* is not scaling with the number of nodes but approximately scaling with the root of the spanned geographical area. It is obvious that the *time complexity* cannot further be diminished.
To calculate the *communication complexity* we make the following observation. In the **Count Phase** every node sends one *NOSP message* over every edge. In the **Report Phase** every node sends $n-1$ *report messages* back to the source. Thus, the *communication complexity* C is $O(n^2 \cdot m)$. This is optimal since every node needs to know the *distance* or *number of shortest paths* to every other node to calculate the centrality index it is interested in.
To calculate the *package complexity* we observe that one node cannot receive and send more than n^2+n different elementary messages in one time unit. This implies that the number of packages sent over one edge per time unit is bounded by a constant. The *package complexity P* is therefore given by $O(D_G \cdot m)$.

III. Decentralized Calculation of Centrality

In the following we will sketch the algorithms for the four different centralities and analyse their respective complexity. In the following, a *report message* that contributes to the centrality index of a node v is called a *contributing* message.

A. Closeness Centrality

The *distance* between all nodes is calculated in the **Count Phase** as described above. A *report message* is contributing to the **Closeness Centrality** of node v if it contains its ID. Every node v will compute its Closeness Centrality by adding up all weights from *contributing* messages. *Contributing* message will not be broadcasted since no other node beside v is interested in the *weight* of the message. After having received the last *contributing* message the Closeness Centrality is determined by taking the inverse of the sum.

B. Graph Centrality

The *distance* between all nodes is calculated in the **Count Phase** as described above. Every node v will store the currently maximal *weight* of all received *contributing* messages. After it has received its last *contributing* message the stored value is inversed and hence the Graph Centrality of the node.

C. Stress Centrality

To calculate **Stress Centrality** *distance* and *number of shortest paths* have to be computed in the **Count Phase**. The *report messages* hold as *weight* the *distance* from ID-pair $(ID(s), ID(t))$. Before describing the **Stress Centrality** algorithm a further lemma is needed.

Lemma *3:* The number of shortest path $\sigma_{st}(v)$ from a *source node* s to a *target node* t running over v is given by $\sigma_{st}(v) = \sigma_{sv} \cdot \sigma_{vt}$.

Proof: The proof is given by an inductional argument. For a direct successor Lemma 3 and Lemma 2 coincide regarding that the *number of shortest paths* between v and t is one if v and t are *neighbors*. Assume that the lemma is true for all nodes up to distance k from v. Let t be a node in distance $k+1$. Accordingly, the number of shortest path from any *source node* s to t is given by

$$\begin{aligned}\sigma_{st}(v) &= \sum_{p \in P_s(t)} \sigma_{sp}(v) && (6)\\ &= \sum_{p \in P_s(t)} \sigma_{sv} \cdot \sigma_{vp} && (7)\\ &= \sigma_{sv} \sum_{p \in P_s(t)} \sigma_{vp} && (8)\end{aligned}$$

Equ. (6) is given by lemma 1, equ.(7) uses the assumption. Regarding that $\sum_{p \in P_s(t)} \sigma_{vp}$ is by lemma 1 the *number of shortest paths* σ_{vt} from v to t lemma 3 is proved. ■

To calculate its **Stress Centrality** a node v has to decide whether a given *report message* with ID-pair $(ID(s), ID(t))$ is contributing to its centrality or not. According to lemma 1 this can be easily determined by adding up the *distances* of s to v and v to t and compare it to the *weight* of the message. If the *weight* equals the sum the message is *contributing* otherwise the message will just be broadcasted to all neighbors.

Following lemma 3, a node v will calculate for all *contributing* messages with ID-pair $(ID(s), ID(t))$ the product of $\sigma_{sv} \cdot \sigma_{vt}$ and sum them up. This is possible because the *number of shortest paths* is symmetric. That is, $\sigma_{sv} = \sigma_{vs}$ and these values are stored in the *NOPS* table of v.
After a node has received the last *report message* the correct value of its **Stress Centrality** is obtained.

D. Betweenness Centrality

The Betweenness Centrality is the most interesting centrality index for a communication network. It is superior to the Stress Centrality because it indicates the *congestion sensitivity* of a node where the Stress Centrality just absolutely counts the number of shortest paths the node lies on.
The algorithm is nearly the same as the one for Stress Centrality but it needs additionally the report of the *number of shortest paths* between any node pair (s, t) (referred to as *NOSP weight*). The definition of *contributing* is the same as in **Stress Centrality** algorithm. For every *contributing report message* with ID-pair $(ID(s), ID(t))$ it will calculate the appropriate product of $\sigma_{sv} \cdot \sigma_{vt}$ and divide it by the *NOSP weight* of the message. The sum of all these fractions gives the Betweenness Centrality.
Fig. 3 and Fig. 4 describes the pseudocode for this algorithm.

```
Count Phase for calculating Betweenness Centrality at node v
\\ initialize first message with ownID and initial numberOfShortest-
Paths 1
  new message($ID_v$, 1);
\\ send message to all neighbors in time $t_0$
while any message has been received in time $t_i$ do
  forall messages do
        if if $ID_s$ of received message has been stored already in time $t_i$ do ignore
    else do
      • store $ID_s$ and sum numberOfShortestPaths over all messages with same $ID_s$
      • store time and numberOfShortestPaths together with $ID_s$
      • new message($ID_s$, numberOfShortestPaths)
      • add new message to listOfMessagesToSent
    od
  od
  sent listOfMessagesToSent to all neighbors at time $t_{i+1}$
od
```

Fig. 3. This pseudocode describes the Count Phase of a Betweenness Centrality calculating algorithm.

```
Report Phase for calculating Betweenness Centrality at node v
forall $\sigma_{sv}$ between v and s known at time $t_i$ generate
report message with ($ID_s$,$ID_v$) and weight $\sigma_{sv}$
\\ send report messages to all neighbors in time $t_{i+1}$
while any report message has been received in time $t_i$ do
  forall messages do
    if ID-pair ($ID_s$,$ID_t$) has already been registered do ignore
    else do
      • multiplicate $\sigma_{sv}$ with $\sigma_{vt}$, divide by weight of message and add value to betweenness-centrality memory cell
      • add report message to listOfMessagesToSent
    od
  od
  sent listOfMessagesToSent to all neighbors at time $t_{i+1}$
od
```

Fig. 4. Pseudocode for the preparing and handling of report messages

E. Decentrality, Scalability and Reliability

In the following we want to discuss the decentrality, scalability and reliability features of the general framework.

1) **Decentrality**
It can be easily seen by the above given pseudocodes that all operations only require the list of neighbors which with direct communication is possible. Since these lists can be provided by algorithms with local decision rules like in Glauche et al. [4], the algorithm is totally decentral.
2) **Scalability**
As mentioned above the proposed algorithm scales in its *time complexity* with the diameter of the graph. As outlined before the diameter of a graph is normally depending on the area the network spans and not as much on the number of nodes in this area. It follows that the time needed for calculation might be constant if the area is restricted. The demands on memory scale with n^2. This can be reduced to a linear dependency without enhancing the *communication complexity* by more than

a factor. [1]

3) **Reliability under failure**
Centrality indices are measures for static graphs. If the computation time is so long that a big part of the network has meanwhile changed the results will not be very meaningful. However, we want to show here that the algorithms always terminates and will give meaningful results if the change of the network has not been too dramatically.
The termination of any of the algorithms is given by the fact that every node will forward every *NOSP* and *report message* just once.
The quality of centrality indices after some change of the network is depending on the architecture of the network. The architecture of a wireless ad-hoc network is grid like due to the restricted transmission radius. By this the *number of shortest paths* between any nodes is very high and the failure of some portion of the graph likely will not change the *distance* between them. This implies that the calculated values for **Closeness Centrality** and the **Graph Centrality** of the remaining nodes will resemble the values in the changed network. This result can be further improved if failing nodes are able to broadcast a *failure notice* with all known distances to other nodes. With this every other node is able to update its centrality index by simply subtracting the according values.
The situation is more complicated with **Stress** and **Betweenness Centrality**. With every failure the *number of shortest paths* between any node pair is changing in a complex way. This makes any updating of the current calculated values very complicated. Still, all remaining nodes will calculate the correct centrality with respect to the moment were the Count Phase began. It is reasonable to assume that the **Betweenness Centrality** depends mainly on the geographical position a node has in the spanned network area. If it is on a border of that area the likeliness is high that the index is low. If it is an inner node the index will be high. With this observation it follows that the absolute centrality index may vary due to minor changes in the network but still the calculated value will reflect the position of a node in the network and an adaptation according to the current value is reasonable even in cases of failure.

IV. Discussion

We have shown in this paper that the decentral calculation of diverse centrality indices is possible. The general framework has been shown to work optimally with respect to all analyzed complexity measurements.
We want to discuss here how the proposed algorithms give rise to optimization of the network. We introduce here a model of evolution of the networks that consists of evaluating and adaptation phases. We assume that some lower level protocol will generate the network architecture whenever a node is lost or a new node is attached to the network. This architecture might not be optimal. As described above the algorithms for a peer-to-peer network in [3] and wireless multi-hop ad hoc networks in [4] disfavour some of the nodes. The latter will produce a mesh or grid like architecture where inner nodes are more probable to route a communication than are nodes on the border of the network. This can be seen in Fig. 6(a). A periodically evaluation of for example the **Betweenness Centrality** will reveal these differences. With an appropriate local rule each node decides whether and how it will change its *list of neighbors*. This gives rise to an emergent behaviour that optimizes the global network architecture. The evolution of a network as a cyclic process of evaluation and adaptation is sketched in Fig. 5. As a proof of concept we made some simulations on a network with 1000 nodes. The nodes were identically indepently distributed in a (10,000 x 10,000) 2D-area and connected according to the *k-next-neighbors rule* with 10 nearest neighbors. One example can be seen in Fig. 6(a). We conducted 10 evaluation steps with the following adaption rule.

[1]We will be happy to give the details of this improvement on request.

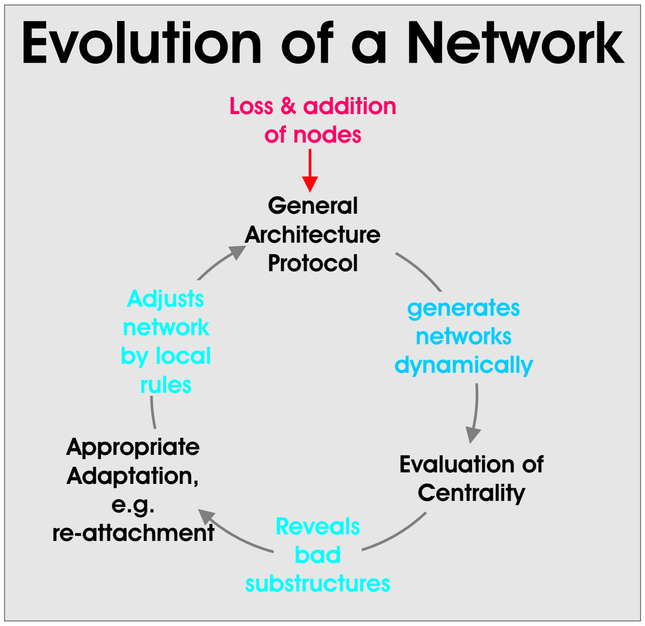
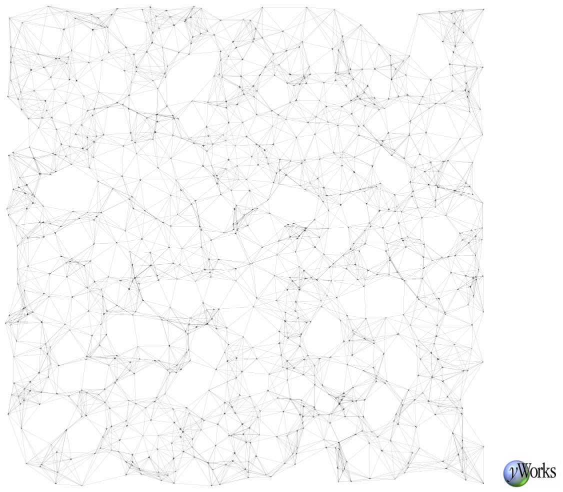

Evolution of a Network
Loss & addition of nodes
General Architecture Protocol
generates networks dynamically
Evaluation of Centrality
Reveals bad substructures
Appropriate Adaptation, e.g. re-attachment
Adjusts network by local rules

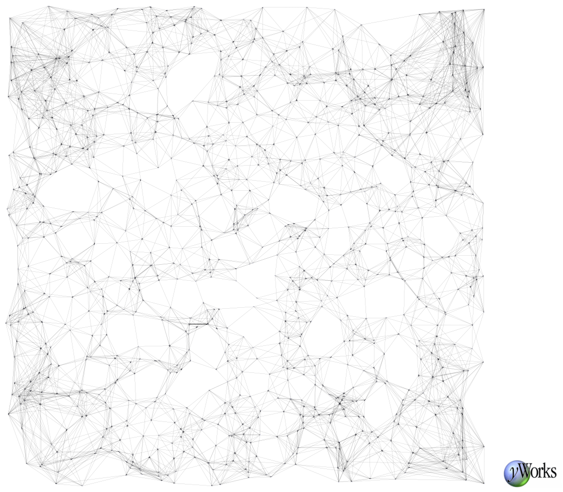
Fig. 5. The diagram shows an evolving network in which a basic architecture algorithm generates networks. The network is changing whenever nodes are lost or added to it. All nodes periodically evaluate one or more centrality indices. After evaluation they adapt their direct neighborhood according to some local rule. With an appropriate adaptation rule nodes with an too high load will be relieved, the diameter of the network decreased and the communication stabilized.

Adaptation rule
If the **Betweenness Centrality** of a node is smaller than a given minimum it connects to the next furthest node. Thus it increases its transmission radius. If the **Betweenness Centrality** of a node is higher than a given maximum it will remove the connection to the neighbor with the highest centrality index.

The minimum threshold was set to 1000, the maximum to
50 000.



(a) Network before evolution



(c) Network after evolution

Fig. 6. The figure shows on the left a communication network of 1000 nodes. The nodes are identically independently distributed on a (10,000x10,000) grid. Each node is connected to its next 10 neighbors, symbolized by an edge between the nodes. The network on the right has evolved from the left network after 10 simulation steps. In each simulation step the **Betweenness Centrality** of all nodes has been calculated. Subsequently, every node with a Betweenness Centrality index less than 1500 enhanced its transmission radius to connect to the next furthest. Every node with a Betweenness Centrality index of more than 50 000 cut the connection to the neighbor with the highest Betweenness Centrality. The result is a network in which nodes on the border spread their connections deep into the network. The inner part of the network is partly sparser than before. The evolved network has a lower diameter and lower average distance. Whereas in the left network some nodes have Betweenness Centralities of up to 105 000, in the right network the maximal index is 49 000.

After 10 simulation steps the *diameter of the graph* is reduced from 26 to 22, the average *hop distance* from 10.66 to 10.07. The *maximal transmission radius* has increased from 1262 to 1776, the *average transmission radius* from 654 to 814. The maximal **Betweenness Centrality** is more than halved from 105 000 to 49 000.

Of course, every graph will reduce its diameter if the average transmission radius is enhanced. The important observation here is that the nodes with high load in routing will maintain or decrease their transmission radius. The global decrease of the diameter is therefore conducted by increasing the transmission radius of the less loaded nodes.

This single simulation shown here is exemplary. Still, finding the correct minimal and maximal **Betweenness Centralities** according to the number of nodes is not trivial as is neither determining an appropriate adaptation rule.

We want to emphasize that the main goal of our paper is to show that a decentral computation of centralities is possible in an efficient way. The above given example is therefore just an illustration of what we think can be done with these centralities. Nonetheless, the given adaptation rule might be a good starting point for real applications: It encourages the nodes to enhance their transmission radius if they are not likely to suffer high routing demands from others. Simultaneously, highly recommended router points are relieved. Thereby, every node contributes similar amounts of energy to the global system, either by maintaining a higher transmission radius or by an expanded routing service.

References

[1] U. Brandes, *"Faster Evaluation of Shortest-Path Based Centrality Indices,"* J. Math. Soc., vol. 25(2), pp. 163-177, 2001

[2] L.C. Freeman, *"A set of measures of centrality based on betweenness," Sociometry*, vol **40**, pp. 35-41, 1977

[3] C.H. Ng, K.C. Sia, C.H. Chan, I. King, *"Peer Clustering and Firework Query Model in the Peer-to-Peer Network,"* Dep. Comp. Science & Eng., Chinese University of HongKong, HongKong, China, Tech. Rep.,2002

[4] I. Glauche, W. Krause, R. Sollacher, M. Greiner *Continuum percolation of wireless ad hoc communication networks* Physica A, **325 (3-4)**, 577-600 (2003)

[5] S. Wassermann, K. Faust *Social Network Analysis: Methods and Applications* Cambridge University Press, 1994

[6] J. Scott *Social Network Analysis: A handbook* Sage Publications, London, 2003 (2nd ed.)

[7] M. E. J. Newman, M. Girvan *Finding and evaluating community structure in networks* preprint cond-mat/0308217

[8] M.E.J. Newman *Who is the best connected scientist? A Study of scientific coauthorship networks* Phys. Rev. E **64**, (2001)

[9] V. Krebs *The social Life of Routers* The Internet Protocol Journal **3(4)**,14-25 (2000)

[10] P. Holme *Congestion and centrality in traffic flow on complex networks* Advances in Complex Systems **6**, 163-176 (2003)

[11] Klovdahl, A.S., Graviss, E.A., Yaganehdoost, A., Ross, M.W., Wanger, A., Adams, G. and Musser, J.M. *Networks and Tuberculosis: An Undetected Community Outbreak Involving Public Places* Social Science and Medicine, **52(5)**, 681-694, 2001.

[12] G. Sabidussi *The centrality index of a graph* Psychometrika **31**, 581-603 (1966)

[13] P. Hage, F. Harary *Eccentricity and Centrality in Networks* Social Networks **17**, 57-63 (1995)

[14] A. SHIMBEL *Structural parameters of communication networks* Bull. Math. Biophys.**15**, 501-507 (1953)
[15] ROBERT G. GALLAGER *Distributed Minimum Hop Algorithms* Technical Report LIDS -P- 1175, MIT, (1982)