Paper

Adaptation of the Kademila Routing for Tactical Networks

Tobias Ginzler[a] and Marek Amanowicz[b,c]

[a] Fraunhofer FKIE, Wachtberg, Germany
[b] Military Communication Institute, Zegrze, Poland
[c] Military University of Technology, Warsaw, Poland

Abstract—In this paper a modification of the widely used Kademlia peer-to-peer system to tactical networks is proposed. We first take a look at the available systems today to cover the range of possibilities peer-to-peer systems offer. We identify candidates for use in military networks. Then we compare two candidate systems in an environment with highly dynamic participants. The considered environment is focused on the special conditions in tactical networks. Then we give rationale for choosing Kademlia as a suitable system for tactical environments. Since Kademlia is not adapted to military networks, a modification to this system is proposed to adapt it to the special conditions encountered in this environment. We show that optimizations in the routing may lead to faster lookups by measuring the modified algorithm in a simulation of the target environment. We show also that the proposed modification can be used to extend the battery lifetime of mobile peer-to-peer nodes. Our results show that peer-to-peer systems can be used in military networks to increase their robustness. The modifications proposed to Kademlia adapt the system to the special challenges of military tactical networks.

Keywords—Kademlia, network enabled capabilities, peer-to-peer, wireless tactical military networks.

1. Introduction

Today peer-to-peer applications and protocols have gone far beyond the notorious file sharing. Applications like remote assistance search, distributed data storage or VoIP systems like Skype make use of peer-to-peer (P2P) systems. Starting with only a handful of protocols, an overwhelming variety of systems for quite every imaginable purpose has been developed. Peer-to-peer networks span the globe and consist of hundreds of thousands concurrent participants. Despite the success in civilian applications, no broad use in military applications is known yet. Especially the resilience of peer-to-peer networks is able to increase the availability of military communication infrastructure. Centralized networks have a single point of failure and facilitate effective adversary actions against the network. Peer-to-peer systems offer a distributed approach contrary to traditional server-centric architectures. We show that peer-to-peer systems exist which are able to work under difficult network conditions encountered in military network environments. Until now peer-to-peer networks have focused on wired infrastructure. In military environments, not only wired networks but also a large variety of wireless networks with mobile devices is used. An adaption of the broadly used Kademlia peer-to-peer system is proposed to adapt it to the military environment. The communication devices in the considered environment have to cope with limited CPU power, small bandwidths, high delay and many connection disruptions due to the nature of the wireless medium and their mobility. According to the network enabled capabilities (NEC) principle it is necessary to interconnect the closed legacy networks of today. An adapted peer-to-peer network available today may significantly improve the availability of the right information at the right place at the right time.

2. P2P Systems and Solutions

An overview of the state of the art of peer-to-peer systems is given in the following section. The scope of the overview of existing peer-to-peer systems is limited to the usage purpose of the system. It should offer a search functionality to find information elements which were previously stored in the network and a method to retrieve them. The network should be scalable so that thousands or even millions of participants can take part without a degradation of service. This respects the fact that in the NEC concept, sensors and systems may also be equipped with information technology resulting in a potentially huge number of network participants. The peer-to-peer system – more specific – the overlay network infrastructure employed by the system, should be resilient against network failures and the unexpected failure of participating nodes. We identify structured overlay systems to be the most promising as they have advantages regarding resilience against attackers and networks failures.

2.1. Early P2P Systems

Peer-to-peer systems which rely on a dedicated server [1] for search or login purposes have disadvantages. Such systems are a single point of failure. Load issues also render this approach unsuitable in a mobile environment. The so called unstructured overlay networks overcome the dependency on servers but searching for content is more difficult in such networks. In server based architectures searching and indexing is trivial and may be enriched with range queries or semantic search. A simple approach to find content in an overlay network without a server is to flood it with a search query [2]. This imposes heavy load on the network. If the search is limited to a fixed number of hops to increase scalability, the search may fail even if the content exists. The search is then considered *incomplete*. Super-peer networks are an alternative to flooding networks. All following approaches are considered to be *complete*, meaning the search succeeds if the content is available in the network or fails if it has not been stored.

2.2. Super-Peer Networks

Super-peer networks use a two-tier architecture. Long-lived or high-bandwidth nodes are declared as supernodes, while other nodes are declared as ordinary nodes. Super-peer networks form clusters of peers around supernodes (Fig. 1). The supernode answers search and storage requests on behalf of its connected peers. Every ordinary peer has to be connected to a supernode. The supernodes communicate by a dedicated protocol.

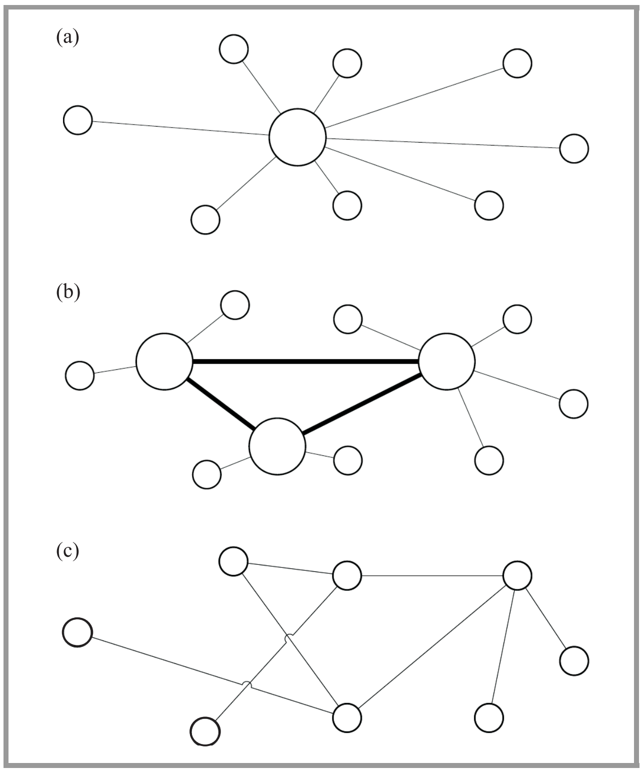

Fig. 1. Different schemes of information exchange: (a) client-server; (b) super-peer; (c) peer-to-peer.

Super-peer networks are used in Skype and FastTrack-based networks. FastTrack is believed to use a controlled flooding algorithm among the supernodes to handle overlay network updates and search requests [3]. Flooding is done also by the 0.6 protocol version of Gnutella [4]. Still, flooding the supernodes is a comparatively inefficient method to search for content in an overlay network. Existing implementations show an increased robustness to churn compared to some unstructured flooding-based overlay networks [5]. Churn in context of peer-to-peer systems denotes the process of members joining the network and leaving it. Churn may either be caused by network failures or user behavior. The startup of nodes, or *bootstrapping*, is more difficult than in pure peer-to-peer systems, as nodes need to find a supernode first. Bogus clients can obtain supernode status by fraud and cause more damage to other clients than a normal peer in an decentralized network. Users may also try to prevent to be elected a supernode to save bandwidth and computational power, increasing the load on the remaining supernodes. Every decentralized system can be transformed into a supernode network by defining the supernode's cluster as a single member of the decentralized network [6].

2.3. Structured Overlay Networks

Today peer-to-peer networks can grow to impressive size [7]. Structured overlay networks were designed to support a very large number of participants. The largest existing overlay is based on the Kademlia structured overlay and is named KAD [8].

Structured networks use a key space where peers are placed in and searching for a node in the key space then follows a (structured) routing algorithm. Each peer carries a unique identifier, defining its position in the key space. Kademlia is based on a structured peer-to-peer overlay network [9]. In Kademlia an XOR metric is introduced to define a distance between two nodes. The XOR distance is the bitwise exclusive OR on the peers' identifiers interpreted as an integer. The other important metrics used by other protocols are the prefix-based metric used by Tapestry [10], the ring metric of Chord [11] and the combined prefix/proximity metric of Pastry [12].

The routing scheme is similar for all structured peer-to-peer systems. The overlay routing is responsible for finding nodes according to their identifier (key) in the overlay. This is called *key based routing* (KBR). The routing algorithms differ but they share the principle of approaching the destination key in every routing hop and terminating at the closest node.

A simple store and retrieve functionality can be supplied by a distributed hash table (DHT) on top of the KBR. The DHT facilitates to store information into the overlay and retrieve information from the network. The application programming interface (API) is similar to a standard hash table. The idea is to attach a key to every piece of information which has to be stored in the overlay. The key is often derived by hashing the representation of the information.

The information is then routed and stored at the r nodes with the identifiers closest to the key of the information, with r as a redundancy parameter. At these locations, the information can also be found by other nodes. Any node looking for the information calculates the key from it and uses the key based routing for finding the node the data is stored at (DHT GET). The DHT is described as an integral part of Kademlia, but it is possible to deploy a DHT on top of every overlay network with key based routing.

As an example of a structured peer-to-peer routing scheme we take a closer look on Kademlia. The routing table of Kademlia is a binary tree (Fig. 2). Each leaf contains a list

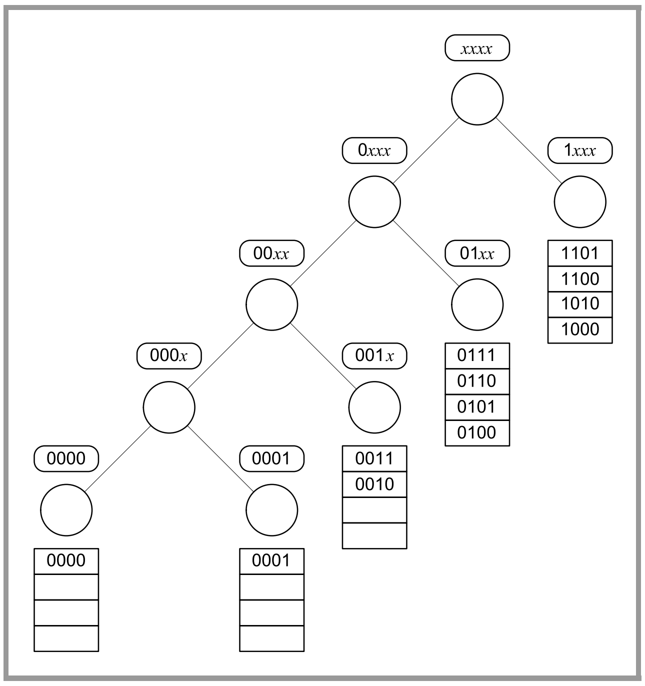

Fig. 2. Kademlia routing table.

of nodes, the so called *buckets*. A bucket holds a fixed number (k) of references to reach other nodes. As the network may contain up to 2^b nodes, the routing table size has to be limited. In Kademlia the memory requirement for the routing table is $O(k \cdot b)$, with b as the number of bits of an identifier. A node carries a tag which defines which identifiers are contained in its subtree. In the figure, b and k are assumed to be 4, the standard key length of Kademlia is 160. A tag of $1xxx$ means that the highest value bit is 1 for the whole subtree and the other bits are unknown. The right subtree of the root carries this tag. The local node identifier is assumed to be 0000 in the depicted tree.

The two subtrees below the root separate the identifier space in two halves: one subtree contains references to nodes closer to the local node than half of the maximum distance (the left side) and the other one contains references to nodes further away (the right side). The rightmost bucket holds k references to nodes which differ in their most significant bit from the identifier of the local node. The left subtree is constructed recursively with increasingly matching prefix length. This enables the local node to store more references to closer nodes than to nodes which are far away. The leftmost bucket contains only the reference to the local node, its sibling bucket may hold exactly one node which differs only in the least significant bit. It is less and less likely that the buckets to the left are filled the farther left they are. That is due to the equal distribution of the identifiers.

During a key lookup it is tried to cut the distance to the destination key at least by half. For doing so the source node XORs the own node identifier with the key to look up. The bucket with the longest shared prefix tag is selected and α nodes are picked from the bucket and the routing request is forwarded to them. Kademlia is able to parallelize its routing requests. The degree of parallelization is α and can be chosen freely. The method of picking a node from the bucket is not specified by the authors of Kademlia. A common implementation is to take the closest node to the destination key. This minimizes the hop count to the final destination. If the routing tables are reasonably filled and identifiers are equally distributed the routing algorithm terminates in $O(\log(n))$. The contacted node sends back the n_{tell} closest nodes of the requested key to the sender and adds the sender if the routing table is not already full. In a pathologic case with no lookup traffic for a long time, a stabilization interval t_{stab} is used to ping nodes from the routing table.

The Chord system is another popular peer-to-peer overlay. Chord and Kademlia share the way how node identifiers are generated. The main difference is the structure of the key space. In Chord, every node is positioned in a ring according to its identifier. The identifier next to another identifier in the ring has a numerically higher identifier, featuring a wraparound at 0. The routing table – or *finger table* – of a node contains a reference to the next node in the ring. This node is called the successor of the node. The finger table contains b references to the successor of the identifiers $(n+2^i)-1, i = 0..b-1$. As in Kademlia this leads to a good knowledge of the node about its near nodes and less knowledge about far nodes. Routing can be done in a matter of binary search in the ring and runs in $O(\log(n))$ overlay hops. The distance to the destination node can be cut at least by half each routing hop if the routing table is correct.

If nodes join or leave the system without notice the Chord routing table gets outdated. A stabilization algorithm is used to repair the finger table and the successor reference. The stabilization uses a reference to the predecessor of a node. In a periodic manner the local node requests the predecessor from its successor. If the local node is not the predecessor, the successor reference is adapted to the node that is returned as a predecessor. The requested node may also adjust its predecessor reference. To be more resilient against node failures, a node may keep up to n_{succ} successor candidates in a list which are tried one after another if the first entry fails.

To stabilize the finger table, periodic search requests for the identifiers in the table are done and the found node replaces the finger reference. The stabilization uses bandwidth which may not be available for search purposes. The setting how often successor stabilization (t_{succ}) and finger table stabilization (t_{finger}) is done, is an important performance parameter.

The difference of Kademlia to other structured overlay networks is the symmetry of the XOR metric. That follows directly from the symmetry of the XOR operation. Peer A has the same distance to B as B has to A. This allows peers to learn about close nodes from incoming routing requests. It reduces the traffic necessary to maintain the overlay network. This feature makes it more promising for use in disadvantaged networks.

2.4. Performance Analysis Approaches

To analyze the performance of peer-to-peer systems, a peer-to-peer system is often simulated to isolate the influences of the underlying network and the user behavior. Experiments in real networks also exist, but for some peer-to-peer systems no widely used networks exist (e.g. Pastry). If available, often the DHT function of the system is used as a test application. The approaches to measuring a DHT's performance differ. Mostly the correctness of the algorithm is shown. Measurements in Pastry [13] were conducted as a simulation within a single Java VM, so node interaction breaks down to Java object invocation. The network model used was derived from [14]. The same network model is used in CAN's performance analysis in [15], but in contrast the node interaction has a fixed delay. Some publications [16], [17] deal with comparing different algorithms in a similar environment with different link delays, making the results somewhat comparable.
In [18] not only the algorithms, but also implementations of Chord, Pastry, Kademlia and Bamboo are measured. The analysis took place in an internet environment emulated by Linux Traffic Control. In [19] the Kademlia network is crawled and the behavior of network nodes is described. The decisive influence of its implementation on the content retrieval delays is shown in [20]. Performance measurement in peer-to-peer systems is challenging because a large number of nodes have to be set up and measured in a controlled manner. The conducted measurements show different approaches to this issue. A balance between simple setup with a precise measurement and realistic network behavior with a sufficient number of nodes has to be found.
We examine one study in further detail. In [17] the routing of Chord, Tapestry, Kademlia, Kelips and OneHop are evaluated. As Kelips uses large routing tables in size of $O(\sqrt{n})$ and Chord and OneHop are not well suited for networks with high churn rate the comparison breaks down to a comparison of Tapestry and Kademlia. The authors also identified the most important parameters and gave recommendations for the parameter values. Kademlia is able to invest bandwidth either in neighborhood consolidation or lookup correctness. The original authors of Kademlia propose a consolidation interval of one hour. The authors of the performance analysis decided to measure the system with a stabilization interval from 4 to 19 minutes. The stabilization interval of 19 minutes resulted in best behavior in terms of routing correctness and delay performance. Although identified as only a minor effective parameter by the authors, it would be interesting to investigate the effect of a consolidation interval longer than 19 minutes.
As Tapestry and Kademlia show similar success in simulations while Kademlia has the ability to learn new contacts through incoming routing requests and is also able to parallelize its requests it is considered the more promising overlay for even more difficult environments as considered in the prior analysis. The Chord overlay was not included in the comparison, so an analysis was done to compare Chord and Kademlia.

3. Comparison of Chord and Kademlia

Chord and Kademlia are compared in a simulated tactical environment to find out, which system is more suitable in a military network. Kademlia has been identified as a possible candidate in the previous section. We take the churn-optimized parameter settings from [17] as a starting point. Then we analyze the behavior of the two overlays in the presence of network errors and compare the results. We use the Chord and Kademlia implementations of the OverSim framework described in [21]. OverSim runs inside the OMNet++ network simulator [22]. The simulation includes a network and delay model as well as a model of the behavior of the nodes themselves.
The network model consists of wireless terminals equipped with IEEE 802.11b wireless LAN infrastructure mode and a fixed transmission capacity of 2 Mbit/s. There are 32 nodes per access point, forming an isolated collision domain. Each access point is attached to an IP router with a 100 Mbit/s Ethernet link. The router has a fixed delay line to every other router. The tactical network model (Fig. 3) has 4 access points and 4 routers. We used the INET extension of OverSim to simulate the full network stack from overlay down to physical layer. This model respects the increased availability of commercial of the shelf (COTS) hardware for military purposes and the tendency to use broadband radio equipment. The availability of a backbone network is anticipated as well.

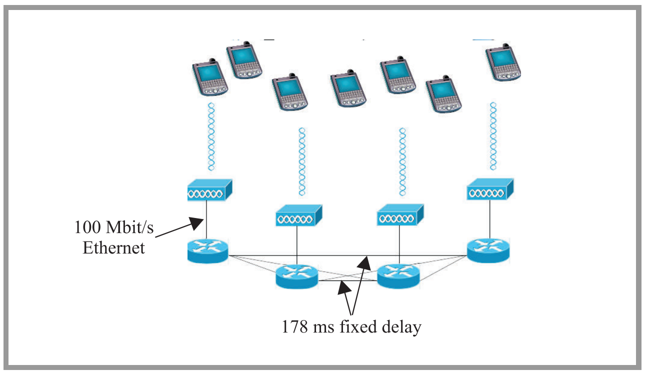

Fig. 3. The tactical network model.

Before a packet is sent, a packet error is applied according to a Bernoulli experiment with variable error probability. If the experiment yields 1, the message is tagged with an error bit. The packet is sent and network resources are consumed. The receiver silently discards the message if the error bit was set.

The delay between the routers is set to a fixed value of 178 ms. It is the mean value from the “King” data set [23], a collection of delays between servers in the internet. The delay on the wireless link is determined by the data link access and media access layer of the WLAN.

The simulated network contains 128 nodes. A tactical network may have connectivity to other tactical networks or even networks of strategic scale. The node count may rise up to thousands of peers. In the simulation the number of peers was limited by the used simulation framework and the peer and network model, not by the P2P systems. All overlay nodes are equal in capabilities and connectivity. Nodes are evenly distributed around the access point and do not move.

The nodes are dynamic in their behavior. This means that new nodes arrive and nodes leave the network over time. This behavior is called *churn* and reflects user fluctuation, either due to network failures or user behavior. Churn can be described by the arrival process of new nodes and their lifetimes. Different churn models are described in [24], [25] and [26]. Tactical users fluctuate more than the typical P2P user, reflecting roaming and network failures within the tactical domain. In [27] a distribution is proposed to model the lifetime process of a P2P system user in the internet. The lifetime of user connections follows the Weibull distribution with a mean of 164 minutes and a median of only 16 minutes. It shows a preference for short-lived connections. This is an effect which is assumed to be present in tactical networks as well. As no tactical peer-to-peer systems are known to the authors, the behavior of its users has to be estimated. Our model reflects this fact by assuming a similar Weibull distribution with the same shape but different mean lifetime. We introduce use two churn models: *normal churn* with 163 minutes mean lifetime and *intense churn* with an even shorter mean lifetime of 60 minutes.

The most important parameters of Chord and Kademlia in a churn intense environment were isolated by Li *et al.*

We took the “best” parameter set of Chord from their publication [17] to optimize for a high success ratio. The churn intense scenario in this publication is modeled as a Poisson arrival process with a mean of 1 hour. Due to the fact that we use a different churn model as described above, a different network underlay, message sizes and a reduced node count of 128, the resulting traffic production was 10 byte $s^{-1}node^{-1}$. Experiments with different parameter settings for n_{succ}, t_{finger}, t_{succ} showed that the initial parameter set already resulted in good success ratios in the tactical environment.

The newly derived parameters with the highest success ratio are shown in Table 1. The parameters for Kademlia were found by matching the traffic rate for our environment with Chord’s traffic rate while maximizing the success ratio. Especially the stabilization interval could be chosen longer, as Kademlia needs stabilization only if not enough routing traffic is present. The packet error rate was varied to measure the influence on the performance of the two overlays. In every simulation run the error rate for all nodes was equal. We varied the packet error rate (PER) in steps from 0.001 to 1.

Table 1
The overlay parameters used for comparison of Chord and Kademlia in the tactical model

Parameters	Chord	Parameters	Kademlia
n_{succ}	8	n_{tell}	8
t_{finger}	120 s	k	8
t_{succ}	20 s	α	3
b	2	t_{stab}	1000

We measure the *delivery ratio* of the overlay routing process. This is the ratio of terminating routing request per total number of routing requests. The higher the ratio the more reliable the routing is. Another method to measure the correctness of the routing is to measure the *success ratio*, that is to take the ratio of successful routing by the amount of total routing request. A successful routing terminates at the node closest to a given search key. A terminating request does not necessarily find the node closest to the requested key. Successful routing requests are measured by issuing a search request on a key which is identical to a node identifier in the network. As this approach introduces a priori knowledge about the existence of certain nodes, a higher success ratio than expected is measured. A correct measurement of the success ratio would require complex distance comparison, slowing down the simulation. For these reasons we preferred to use random keys and the delivery ratio to test the lookup correctness. It has to be noticed that the success ratio of a DHT exceeds the delivery ratio of the routing by far. As the DHT may use the r closest nodes storage locations, the success ratio of the DHT mainly depends on this parameter.

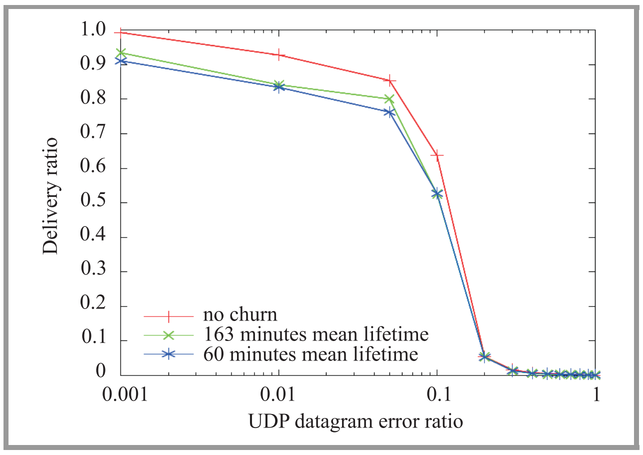

Fig. 4. Delivery ratio of Chord.

The results of the comparison between Chord and Kademlia are shown in Figs. 4 and 5. As a comparison three different levels of churn: no churn, normal churn and intense churn are depicted. The delivery ratio of both over-

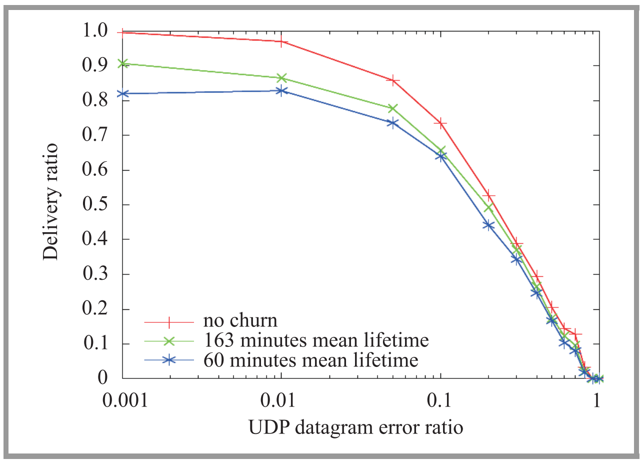

Fig. 5. Delivery ratio of Kademlia.

lay types declines with increased packet error rate. Chord achieved higher delivery ratios if a low packet error rate is present. As packet error rate increases, Kademlia shows better performance.

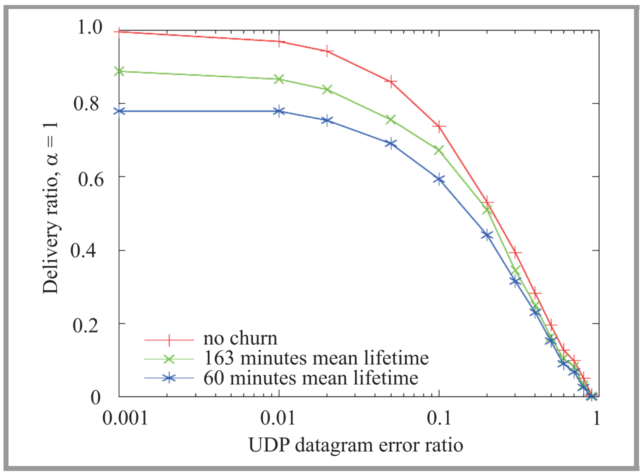

Fig. 6. Delivery ratio of Kademlia with $\alpha = 1$.

This provides rationale to prefer Kademlia in environments with high churn and high packet error rates. Chord is not able to use parallel lookups. To isolate the effect of parallel routing requests the measurement is repeated with the Kademlia parallelization parameter $\alpha = 1$, effectively disabling parallel lookups (Fig. 6). Still Kademlia is more stable when high packet error ratios are encountered. The reason for better performance in the presence of high packet error ratios is to be found in the rigidity of Chord where node failures have a more serious impact on the correctness of the routing table than in Kademlia.

4. Modification of Kademlia

We propose a change to the Kademlia routing mechanism to improve its performance in the presence of network errors and high latency. The modified Kademlia is able to incorporate signaling from lower layers or applications [28]. Our aim is to make the routing more adaptive to the underlying network structure. As the concept of proximity routing [29] requires additional messages, the proposed concept does not. It incorporates information from the routing or application layer, which can be delivered without cost in terms of additional traffic. The guarantees of the Kademlia routing, especially the completeness and the complexity properties remain untouched. A node running a modified version integrates seamlessly in running networks without modification.

Our approach is not to modify any existing routing parameters but to use a different method of choosing contacts. Although the method is also applicable to other peer-to-peer systems, the scope of this paper is limited to Kademlia. Cross-layer information is integrated into the routing decisions. The additional information is called preference value or simply the preference of a link or node.

The Kademlia routing described in Subsection 2.3 is changed in the way the sender of a lookup request selects nodes to contact. The original algorithm first selects the appropriate bucket (Fig. 7) and puts the contained routing entries into a list L of candidates. The first α candidates are then contacted and the lookup request is forwarded to them. In some situations if a bucket is very sparsely filled, entries from adjacent buckets may be used. In the modified version every contact is now augmented with a preference value. We introduce a weight factor w, which determines the influence of the preference. A factor of 1 means the next hops are determined according to the preference value and L is sorted according to the preference values. A weight of 0 represents the unmodified algorithm. Intermediate values of the weight affect the order in a continuous manner. The new sorting order is defined by:

$$m_d = weight\frac{pref_{s,d}length(L)}{max\_pref} + (1 - weight)pos_d,$$

where: s denotes the local source node making the routing decision and d a remote destination node. The original position of d the in L is pos_d. The list L is then reordered in descending order according to m_d.

The effect is shown in Fig. 7, a different node of the same bucket is preferred over a closer one. As the original version minimizes the hop count by always choosing the closest nodes the modification increases the hop count.

We test three methods to generate a preference value. The first method is to take the channel delay between the local node and the next hop i of L as preference value. After a normalization step the delay is used as preference. We call this modification *modification 1*. The second method is to take the bit error rate between the local node and the remote node $BER_{s,d}$ as a preference value, it is called *modification 2*. *Modification 3* only takes a value defined by the remote node into consideration. The node may set a low value to attract routing traffic or a high value to avoid it.

We use a simplified model of a tactical environment to be able to simulate more nodes. The simplified network model contains 1024 nodes. For modification 1 and 2 all nodes

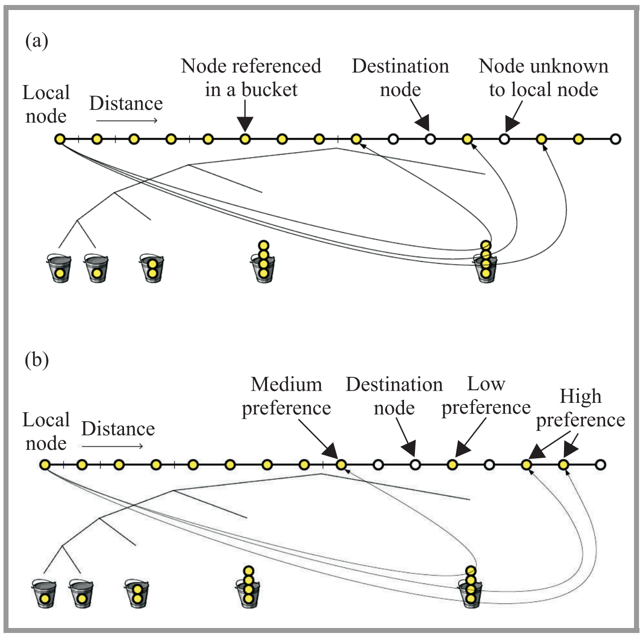

Fig. 7. Next peer selection: (a) original version; (b) changed lookup.

are equal, modification 3 introduces two different types of nodes.

The simulation time is 5 hours including a warm up time of 1 hour. Every run was repeated 10 times with different random number seeds. All nodes in the network feature a UDP network stack. Before a packet is sent to another node an error model and a delay model are applied. Before a packet is sent, a bit error may be applied according to a variable error probability. No churn is used in the simplified model. The nodes are placed equally distributed. The link delay between two nodes increases linearly with the Euclidean distance of the nodes. The maximum delay is about 7 s, this is below the message timeout value of 10 s. This simulates the effects of lower layer protocols such as multi hop propagation in a simplified manner. Lost messages get detected by the overlay 10 s after they have been send. In Fig. 8 the results of modification 1 and 2 are shown. The measurement with weight set to 0 is included as a reference to the unchanged Kademlia routing algorithm in the simplified network model. The figure depicts the time it takes to perform a DHT GET request with modification 1 and modification 2. The DHT GET latency is the duration it takes to retrieve a previously stored value from the DHT. The weight was modified to analyze the effect of the preference values to the routing. The absolute numbers of sender traffic and success ratio are of lesser importance as they are mainly dependent on the parameter settings of the overlay. It is possible to increase the success ratio for example by an increase of parallel lookups. The absolute values for the DHT latency are mainly dependent on the network model and the link delays. We focus on the relative change of the values when we modify the routing. In Fig. 8a the effects of both modifications on the DHT GET latency is shown. The preference for faster connections in modification 1 does not seem to pay off in terms of latency. The reason for the low impact of modification 1 is how Kademlia sends parallel lookups during the routing process. Since Kademlia tries to hold α lookup request in flight, the fastest response is immediately processed and the routing continues with the sending of another routing request. The parallelization elevates the effects of preferring fast nodes as the probability is high that 1 of α nodes reacts. Timeouts dominate the influence on latency. Timeouts occur if a node has failed. Sending slots are blocked for the duration of the timeout.

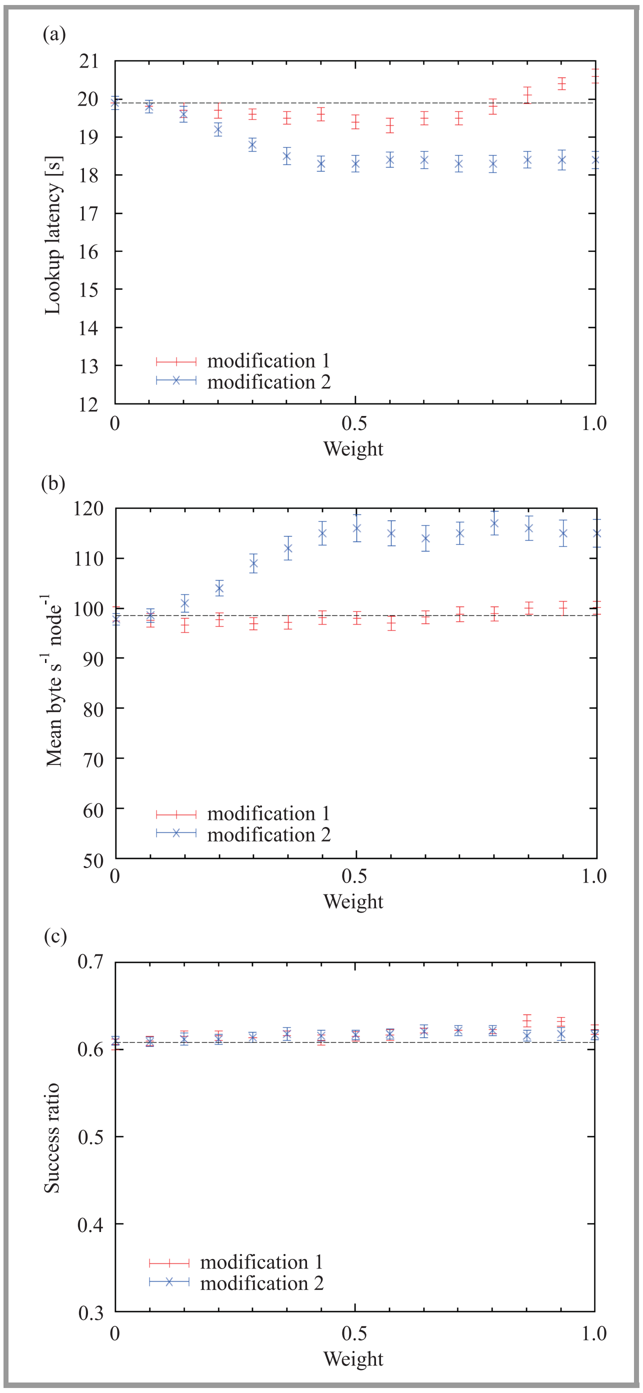

Fig. 8. Effects of modification 1 and 2: (a) DHT GET latency; (b) overlay traffic sent; (c) DHT success ratio.

If all slots are blocked the routing stalls until the first timeout occurs. Because less timeouts occur, modification 2 (which prefers low BER links) performs better in terms of reduced latency. This observation is in contrast to existing implementations as the RTT is often used to approximate the reliability of a link. In environments with high BER, preferring low BER links is better than to use faster links. Modification 2 facilitates a trade-off between delay and the amount of transmission capacity needed to maintain the peer-to-peer overlay.

Figure 8b shows the mean number of bytes sent per node per second for a fixed weight. Modification 1 does not change the amount of sender traffic as it has a too little effect on the routing decisions. Modification 2 increases the bytes sent. This is because the original Kademlia routing decision is optimized for low overlay hop count. Any change to the routing decision will increase the hop count. The higher hop count leads to an increase of the traffic of modification 2. At a weight of 0.5 the modification offers minimum latency, at this weight it causes an increase of 17% in overlay routing traffic. Our results show the possibility to exchange reduced latency for an increase in sender traffic.

The success ratio of the DHT lookups is shown in Fig. 8c. A DHT lookup was counted as successful if the value could be retrieved without timeout from 1 of the redundant storage locations ($r = 3$). The success ratio stayed constant or increased slightly with increased weight. The absolute rate of success is less important as it is always possible to trade bandwidth for increased success ratio by parameter changes. The important observation is the success ratio does not decrease as an effect of the modifications.

To test modification 3 we set every 10th node to the least preferred value. This simulates a node with limited battery capacity. As sending consumes scarce battery power, we measured the accumulated number of bytes sent out by the tagged nodes over the whole simulation duration. The results can be seen in Fig. 9. The original amount of data sent out is shown as crosses, the simulation run with modification 3 is shown in an x-shape. The amount of bytes per node is not equal for all nodes even in the unmodified scenario. Nodes join the network successively, so node 0 is the first and 255 the last. The effect is not visible if churn is applied. In Kademlia long lived nodes are preferred over newly arrived nodes if a bucket is full. This bucket eviction policy leads to the fact that node 0 features the highest traffic and node 255 the least. The low battery nodes can lower their amount of sending by up to 25% with modification 3, while still taking part in the overlay with no disadvantages. The accumulated traffic of all nodes over the whole simulation time remains nearly unchanged at 151.11 MB versus 153.38 MB with modification 3, also the success ratio remains nearly unaffected.

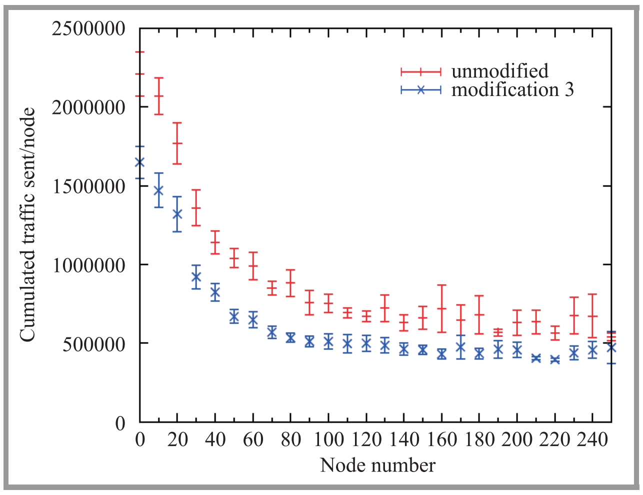

Fig. 9. Traffic shaping by modification 3.

5. Summary

We analyzed different peer-to-peer systems for their suitability in an error-prone military network. Chord and Kademlia were identified as candidates to be suitable in such networks. The candidates were compared in a simulated tactical environment. The environment features wireless and wired networks and a faithful media access simulation. We showed that Kademlia offers a higher delivery ratio than Chord in the presence of churn and high packet error rates. Then we introduced a change to Kademlia's routing algorithm to include cross-layer information. We gave three examples how to use the extension. Error rates of links are reported through the new interface (modification 2). It was shown that it is possible to reduce the number of timeouts and thereby decrease the latency of the peer-to-peer system. A cross-layer signaling of the link latency did not improve the performance of Kademlia in the considered environment because the parallelization of routing requests in Kademlia elevates the effects. Nodes which need to save battery power can use extension 3 to reduce their contribution to the overlay network. Low battery nodes remain full members of the network and suffer no disadvantages but they send significantly less traffic. The overlay can cope with a considerable percentage of disadvantaged nodes with no limitations. Our modified client may join a Kademlia network without interfering with existing clients and overlay networks. The presented results show a possibility to increase the availability of information in tactical networks. Future steps include the analysis of the peer-to-peer system with a tailored publish/subscribe capability.

References

[1] Dr. Scholl. Opennap, 2000 [Online]. Available: http://opennap.sourceforge.net/

[2] The gnutella protocol specification v0.4 [Online]. Available: http://www.stanford.edu/class/cs244b/gnutella_protocol_0.4

[3] J. Liang, R. Kumar, and K. Ross, "The kazaa overlay: a measurement study", in *Proc. 19th IEEE Ann. Comput. Commun. Worksh.*, Bonita Springs, USA, 2004.

[4] T. Klingberg and R. Manfredi, Gnutella 0.6, 2002 [Online]. Available: http://www.rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[5] S. Guha and N. Daswani, "An experimental study of the skype peer-to-peer voIP system", Techn. Rep., Cornell University, Dec. 2005.

[6] B. Yang and H. Garcia-Molina, "Designing a super-peer network", Techn. Rep. 2002-13, Stanford University, 2002.

[7] M. Steiner, E. W. Biersack, and T. En Najjary, "Actively monitoring peers in KAD", in *6th Int. Worksh. Peer-to-Peer Sys. IPTPS'07*, Bellevue, USA, 2007.

[8] *Why Kad Lookup Fails*, H. Schulzrinne, K. Aberer, and A. Datta, Eds. in *Proc. IEEE 9th Int. Conf. P2P Comput. 2009*, Seattle, Washington, USA, 2009.

[9] P. Maymounkov and D. Mazieres, "Kademlia: a peer-to-peer information system based on the XOR metric", in *Int. Worksh. Peer-to-Peer Sys. IPTPS, LNCS*, vol. 1, Cambridge MA, USA, 2002.

[10] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment", *IEEE J. Selec. Areas Commun.*, vol. 22, no. 1, pp. 41–53, 2004.

[11] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications", in *Proc. 1st ACM SIGCOMM*, San Francisco, USA, 2001, pp. 149–160.

[12] A. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems", in *Proc. IFIP/ACM Int. Conf. Distr. Sys. Platforms (Middleware)*, Heidelberg, Germany, 2001, pp. 329–350.

[13] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: the design of a large-scale event notification infrastructure", in *Proc. 3rd Int. COST264 Worksh. Networked Group Communication NGC'2001* , *Lecture Notes in Computer Science*, J. Crowcroft and M. Hofmann, Eds., vol. 2233. London: Springer, 2001, pp. 30–43.

[14] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork", in *Proc. IEEE INFOCOM'96*, San Francisco, USA, 1996, pp. 594–602.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network", Techn. Rep. TR-00-010, International Computer Science Institute, Berkeley, USA, Oct. 2000.

[16] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlays", in *Proc. IEEE INFOCOM 2003*, San Francisco, USA, 2003.

[17] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil, "A performance versus cost framework for evaluating DHT design tradeoffs under churn", in *Proc. IEEE INFOCOM 2005*, Miami, USA, 2005, pp. 225–236.

[18] D. Kato and T. Kamiya, "Evaluating DHT implementations in complex environments by network emulator" in *Proc. Int. Worksh. Peer-to-Peer Sys. IPTPS'07*, Bellevue, USA, 2007.

[19] M. Steiner, T. En-Najjary, and E. W. Biersack, "Exploiting KAD: possible uses and misuses", *Comput. Commun. Rev.*, vol. 37, no. 5, pp. 65–70, 2007.

[20] M. Steiner, D. Carra, and E. W. Biersack, "Faster content access in KAD" in *Peer-to-Peer Computing*, K. Wehrle, W. Kellerer, S. K. Singhal, and R. Steinmetz, Eds. New York: IEEE Comput. Society, 2008, pp. 195–204.

[21] I. Baumgart, B. Heep, and S. Krause, "OverSim: a flexible overlay network simulation framework", in *Proc. 10th IEEE Glob. Internet Symp. GI'07 in conj. with IEEE INFOCOM 2007*, Anchorage, USA, 2007, pp. 79–84.

[22] A. Varga, "OMNeT++ discrete event simulation system", 2009 [Online]. Available: http://www.omnetpp.org

[23] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: estimating latency between arbitrary internet end hosts", in *Proc. 2nd ACM SIGCOMM Internet Measur. Worksh. 2002*, Marseille, France, 2002.

[24] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling heterogeneous user churn and local resilience of unstructured p2p networks", in *Proc. IEEE Int. Conf. Netw. Prot. ICNP'06*, Washington, USA, 2006, pp. 32–41.

[25] R. Brunner, *A Performance Evaluation of the Kad-Protocol*. Corporate Communications Department Institut Eurécom France, 2006.

[26] K. C. Almeroth and M. H. Ammar, "Multicast group behavior in the internet's multicast backbone (mbone)", *IEEE Commun. Mag.*, vol. 35, pp. 224–229, 1997.

[27] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks", in *Proc. 6th ACM SIGCOMM Conf. Internet Measur. IMC'06*, J. M. Almeida, V. A. F. Almeida, and P. Barford, Eds., Rio de Janeiro, Brazil, 2006, pp. 189–202.

[28] T. Ginzler and M. Amanowicz, "Simulation of the impact of packet errors on the kademlia peer-to-peer routing", in *RTO Informa. Sys. Technol. Panel Symp. IST–092/RSY–022*, NATO, 2010.

[29] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in distributed hash tables", in *Proc. Int. Worksh. Future Direct. Distrib. Comput. FuDiCo*, O. Babaoglu, K. Birman, and K. Marzullo, Eds., Bertinoro (Forli), Italy, 2002, pp. 52–55.

Tobias Ginzler graduated as a computer scientist (Dipl.-Inf.) in 2006 at the University of Bonn, Germany. He then joined the research facility in Wachtberg which is now known as Fraunhofer FKIE. His topics of research at the department of communication systems include multicast key management, tactical messaging, service oriented architecture and peer-to-peer systems. In parallel he is a Ph.D. candidate at the Military University of Technology, Warsaw.
e-mail: tobias.ginzler@fkie.fraunhofer.de
Fraunhofer FKIE
Neuenahrer Str. 20
53343 Wachtberg, Germany

Marek Amanowicz – for biography, see this issue, p. 53.