1 Introduction

- What is a Darknet, effects, features for its users
- Arising problems from its characteristics
- name the analyzing methods (and 1 sentence explanation)
  - analytic
  - simulation
  - emulation
  - test bed (and/or measurement in real world networks)
- Advantages of event based large scale simulation and in general analysability of darknets

2 Peer-to-peer Networks and Darknets

2.1 Peer-to-peer (P2P) Networks

In this section a short history of the internet leading to the development and spreading of peer-to-peer network architecture is given. The key concepts and properties of peer-to-peer systems are explained. It should provide a basic understanding of differences and advantages of the peer-to-peer network design.

2.1.1 From classic client-server architecture to distributed networks

The Internet emerged from several military and sience research networks, with ARPANET beeing the most commonly known. It was planed and designed as a decentralized telecommunication network resiliant to outages. Although it is resiliant on a network level, most its services have a centralized structure. On failure not the whole network will fail, but a service, as a website like www.tu-darmstadt.de or its email system, can come unavailable easily.

This originates from the client-server architecture used by most protocols and services. The clients tries to find a single server to send its request to. The server processes the request and sends the response back. If the server fails after the client has sent its request and before a response can be send, the request fails too. There are several methods in protocol and network design to prevent such failures.

The basic approach is to devide the responsibilities in different independent domains. For example an email server is not responsible for all the emails in the world but only for those of one (or more) domains. Note that such a domain does not necessarily mean a domain name in the DNS system but can be an arbitrary domain like a company, an university or just a group of people.

The next step in reducing the responsibility is to reduce what a single component is responsible for. The most important approaches are the avoidance of single points of failure, and the introduction of redundancy and load balancing, which distributes the requests over different servers. However, all approaches on this level scale very badly and therefore achieving fault tolerance or dealing with a large amount of clients or requests is expensive on this level.

Taking the splitting of responsibility to the maximum, every entity is responsible for everything. This is basicaly the idea behind the peer-to-peer architecture in which every client is simultaniously a server. Naturally not every client an be responsible for everything, but each of them is participating in the service the network provides, serving a fraction of it while for each fraction there are multiple, redundant “servers”. This both reduces the load of every single server and simulataniusly adds redundancy. Since there is commonly no distinction between clients and servers in a P2P network, a participant is called a node or peer.

2.1.2 The rise of P2P systems and its usage today

With the Domain Name System (DNS) and the Simple Mail Transfer Protocol (SMTP) some parts of the peer-to-peer design are already in use since the early days of the internet. More than that, P2P ideas were researched, developed and used in smaller scale for a long time. But it took until the late ’90 and file sharing networks like Napster to popularize the P2P concept.

TP2P networks experienced a huge increase in participation and usage when sharing copyright protected material over them started, and gave them an infamous image. Under this the term P2P suffered for quite a time. Just in the past few years several larger companys began to utilize peer-to-peer systems, e.g. for delivering larger sets of data to a huge amount of customers. This improved the acceptance and the image of P2P technology.

Today P2P systems are more and more used as a resiliant and scalable basis for communication, content delivery, and distributed storage. These benefits are both used in public and in unclear twighlight or even illegality.

2.1.3 What a peer-to-peer overlay is

The Internet connects many devices worldwide over different, commonly multiple connections. In this global network nodes can take part in “virtual” networks over the top of the underlying network. In this overlay network, nodes are connected by logical links, each of which having a corresponding path in the underlying network. Therefor the P2P network graph is a subset of the underlying networks topology graph. TODO:Bild zu Overlays, ggf wikipedia? lizen?

Commonly a seperate addressing scheme for communication in such a network is used. In most cases a node is identified by just a sequence of bits, its ID. It is usually represented as a number in decimal or hexadecimal form. All possible IDs together are called the address or ID space and its size varies from network to network but is constant within one network.

Both these properties together form a more or less idependant network put on top of the underlying one. This is called a peer-to-peer overlay.

2.2 Darknets: privacy preserving P2P overlays

In the last section we gave a brief overview of the key aspects of peer-to-peer systems and their envolvement over time. As any tool and technology, P2P systems have advantages and disadvantages and can be used for good or evil. Now we will discuss how the demand for privacy led to a more specialized class of peer-to-peer networks, the darknets.

2.2.1 Consequences of decentralisation

As discussed in section 2.1 the essence of P2P systems are their decentralisation. Not a single server but virtually every pariticpant of the system is responsible for serving requests which results in a high failure resiliance. While this is a valuable property for its users, it can be problematic if trying to stop the service of a network. Single nodes clearly can be filtered or blocked by their ISP or a firewall. Though, this does not prevents the unauthorized distribution of copyright protected material or repressed communication as there will be still many nodes left.

2.2.2 The demand for privacy preservation

The distribution of objectonable information can therefor not be effectivly prevented but the operator of a node could still be prosecuted, if their membership and identity is revealed. In the used file sharing networks the communication between two nodes is indeed encrypted and not readable by outstanders, but in order to exchange infromation the nodes have to connect to each other. Therefor their identity, in the internet commonly the used IP address at a given time, is revealed to each other.

This is, not limited to file sharing networks. Contrary to content delivery networks, any regime undesired communication can be somewhat pushishable. This applys to mancy cases, for example a whisle blower in a missbehaving company or regime critics under a dictatorship. For an free society the freedom of speach and the ability to inform oneself is important. Therefor it is essential to be able to communicate whithout surveilliance and obstraction.

2.2.3 Key concepts of darknets

For those oppressed environments a more privacy preserving class of peer-to-peer networks has evolved, the darknets. In the following section the main differences to classical P2P systems are briefly noted while they get explained in detail in chapter 3.

Exchanging any kind of information on the internet leads to the necessaty of directly connecting of at least two participants. As explained, this leads to revealing the identity of those in the underlying network to each other. To overcome the impact of this privacy relevant information disclosure, connections are only established between mutually known and trusted parties.

Additionally to communicating with kown and trusted peers only, no information about which peers a node is connected to are passed. On compromisation only the identiy of directly connected peers are affected. Anyway, no node can be held responsible for contact to other nodes, since not even participants know to which peers a node is connected to.

But since only communicating with trusted peers would end in a very limited reachability, communication between not directly connected nodes has to be forwarded on some way. To do so the identity in the underlying network and information about the topology, who is connected to whom, have to be concealed.

To achieve this, forwarded messages are modified to originate from the forwarding node itself. Returning answers are modified accordingly and are passed back to the source of the original request. TODO:grafik die das erläutert? As this is done on every node, the identity of the origin of a message is preserved and no information about the topology is reveald.

2.3 Darknet characteristics and resulting challenges

Section 2.1 explained the methodical differences of darknets to classical P2P networks. Now follows, althoug already briefly touched, a discussion of the arising characteristics and the thereby resulting challenges for the practical use of darknets.

2.3.1 Hidden Topology

In summary the membership of a node in a darknet is only known to its trusted peers. The other way, all a node knows about the topology is the list of its online peers. If the destination node of a message is not contained in this list, it has, at least in the basic darknet model, no clue where to send it to.

An alike situation can be found in a wireless sensor network (WSN), where the topology of the network is changing and unclear to the single nodes. It is caused by the limited rage of the nodes and their constrains in energy consumption. To overcome this problem, WSNs often use the information they can get from forwarded messages: the rough direction the source node of a message. This is prevented in darknets by the hop-by-hop anonymity.

2.3.2 Difficulties for routing ...

This high rate of protection of privacy relevant information comes with numerous difficulties in designing and evaluating darknets that should be simultaneously resilient and scalable. When sending a message to a node that is not a connected neighbor, a routing decision has to be made. In conventional networks the next node can be chosen on the basis of topology information about the network, e.g. in form of a classical routing table or structured overlays in P2P systems.

Because any topology information about the network is confidential, they are not distributed. Therefore the topology of a darknet can in general not be used to make routing decisions. This holds as well for meta topology information such as the origin of a message. From this the approximate direction a node lies within could be estimated.

2.3.3 ... and evaluation

Comparable difficulties arise for measuring any kind of quality while evaluating darknets. For example it is important to be able to compare the path a message travels along to the best possible path from its source to its destination, however a quality of a path is measured. This contradicts so fundamentaly the protection of privacy, that it is virtual impossible to observer and analyze a real world darknet without disrupting its actual usecase. So other then other networks, both “normal” and P2P networks, some kind of extra study environment is required to develop and improve darknets.

3 Metrics and evaluation in networks

3.1 Metrics for routing benchmarks

As touched before, to develop or improve a network, metrices to measure its quality and performance are

- The simplest metric is the path length, or hop count, the amount of hops a packet has to be forwarded on until it reaches its destination. It is an important factor of delays in communication and also affects the bandwidth between nodes. The shorter the chosen path is, the faster the communication is and the less the network has to be utilized. The path length, and its average and maximum in a network, are the most commonly used metrics to compare routing.

- The overhead measures how much resources have to be used to transmit the actual information. It can be measured as the overhead messages ratio or the overhead bandwidth ratio and is a grade for the efficiency of a system.

- In one simulation, we will inspect the count of failed return paths. Since a response is sent back the path it came from, it has a fatal impact if a node on that path fails. It can show a upper bound of reliability of a system if it relies on this method.

- TODO:more?

3.2 Evaluation methods for networks

For evaluating distributed systems like P2P overlays, and darknets in particular, there are four different approaches.

TODO:...mehr

3.2.1 Full blown test environment or mathematical model evalution?

The original software can be tested, almost or completely unmodified, in a testbed. This is used to test the functionality itself but scales very badly for multiple nodes and even more for larger networks, as it is time and resource intensive and therefor expensive to setup multiple nodes. In particular for evaluating darknets, information gathering methods have to be added since this is against their normal usecase.

A completely different approach is to build a analyical model and derive formulas for the relevant values from it. This is quite flexible for varying parameters and very scalable and fast. But on the other hand the derivation of formulas can be challenging and small changes in the model or algorithm can render most of the work inappropriate. With analytical models upper and/or lower bounds can be estimated, but real world performance can depend on protocol details and other factors hard do model.

3.2.2 Inbetween: emulation and simulation

By removing all irrelevant and independent parts of the original software and run many instances of such slimed clients a network can be emulated. Main benefit is that no new software has to be written that can be faulty or not sound to the original one. While very little new implementation effort is required, in common the client still has unnecessary components e.g. for network communication. Certainly, this is not applicable to every software, as they may suffer from inflexible software design where the routing algorithms components are not easily seperable from the rest.

This consummes eventually much more resources in time and memory than a simulation, whereby the abstract, relevant behavior is implemented in a simulation environment. Large networks and complex algorithms can be simulated. Implementation is straight forward and can be easier then building a proper formula. However, the algorithms must soundly be implemented and simulation can take much computation time and memory depending on the model and the network to be simulated.

3.2.3 Applicability on darknets

In principle all of the discussed evaluation methods are applicatable, each with a different effort needed. The most work has to be done befor and during using a test environment with a not or only minimaly changed client. Either the properties of privacy and anonimity has to be abandoned, if a existing network is extended to collect statistics, or a expensive testbed has to be build. The same holds eased for an emulation. An analytical model, as the other extrema, suffers from of either a lack of realism or a hughe amount in building up a proper and sound model.

While concentrating solely on the problems of routing in a darknet, a simulation based approach is the best method. Relativly easy implementation of routing algorithms, the high flexibility and scalabilty compared with a good accuracy and soundness, make this a powerfull way of researching and developing darknets.

3.3 Survey of previous darknets

4 Model Description

4.1 Why we chose the simulation based approach

As explained in Section 3.2, different methods to evaluate darknet systems exist, each suitable for different usecases. To analyse a routing algorythm, its performance, strengths, and weaknesses, larger scale networks are neccessary. For this, especially while dealing with changeing algorythms and parameters, the simulation approch is most appropriate since it has the best tradeoff of flexibility and scalabiliy. However, this method is up to now underrepresented. To change this we decided to develop an easy to use and extend model and implement it in an wideley adopted network simulation framework as OMNet++. TODO:how to prove this statement? This gives the posibility to easily estimate the behavior of a routing algorythm in a darknet while maintain comparability of the results.

4.2 Node based with fixed neighbor set and add-hoc anonymity (static)

Virtually every P2P client supports a basic set of capabilities. These are connecting to other clients, storing and recieving of information and some kind of searching. Depending on the network searching for files or other nodes can be different or the same. Naturally these reqests can or even have to be responded to.

So on an abstract level, a P2P node has to be able to connect to other nodes, make requests and send an response to a received request. The basic model used in this work does not distinguesh between the different request types since in most cases it makes no difference for routing algorithms. This is the same for darknet nodes. Only the peers a darknet node connects to or accepts connections and messages from of course are limited.

In practice transfering the real world trust relation to the darknet and the distribution of the current addresses of onces peers are two of the hardest tasks. There are some fundamentaly different posibilities to solve this depending on the overall design. But since these tasks are better analyzable than routing algorythms our model excludes these topics and concentrates on routing makeing it easyer to evaluate this seperate problem.

Additionally a darknet client has to be abled to forward reqeuests to other nodes. In darknets the hop-anonymity is typically achieved by modifying forwarded messages as to come from one self. Responses to forwarded requests have to be modified and forwarded respectively to the origin the reqeust was received from.

This simple model is static, all nodes are always online and a failing node is not possible. It is very simple and scalable because only a minimal overhead is needed. But it is not very realisitc since nodes are not online all the time and can fail on one way or another.

4.3 Churn model extension with bootstrapping and offline detection (dynamic)

The earlier implemented static model of all nodes being online during the complete simulation run is very easy and gives already chance to investigate large parts of a darknets behavior. However, it is not very realistic.

To take into account the posibilitie of nodes shutting down, having connection problems or failing otherwise, the model is extended by a churn based lifetime model. It gives the nodes a probabilistic lifetime, so not all nodes are online in the begining, but they bootstrap into the network. They have to establish a connection to their peers.

Nodes online/offline state can change during the simulation, so the peers of this node have to be informed. As this simulator concentrates only on the routing algorythm we are not concerned how a node would detect a change of state of one of its peers. The simulator notifies all peers of a node of the change. In the real world something like an TCP-like acknowledgment method would be used. TODO:change code: model it by offline/online notification from simulation environment

4.4 (??)Extension of the model by example

5 Implementation

As statet above, network simulation is an both effective and efficient method to evaluate network performance, but an underrepresented one for darknets. In this chapter we describe the used discrete event based simulation framework OMNeT++, its components and their purpose, the implementation of our model as a basic darknet node and its extension of possibly offline nodes.

5.1 The simulation library and framework OMNeT++

5.1.1 Network simulators and why we cose OMNeT++

In chapter 2.5 we stated that the benefit in using simulation for routing evaluation is the reduced time and memory usage compared to emulation or a testbed. This would comes in exchange for a huge implementation effort, since a simulator, the client model and a network is needed. Fortunately simulatiors and frameworks exist already.

Besides some general purpose simulatiors and many specialized ones, there are even some for network analysation and evaluation. Widely used open source products are ns2/ns3 and OMNeT++. Additionally to saving the implementation of a simulator, using a commonly adopted framework gains a better comparability and to other results and respectability of the results by other researchers.

We decided against ns3 as it is way more complex and too overloaded for our purpose, aiming at a high scalability. The large functionality adds many potetial source of errors as it is harder to understand and implement a correct and sound simulation. Instead we chose OMNeT++ for its extendability, modular design and wide varity of already implemented telecomunication protocols shipped with the INET package for example.

5.1.2 What is OMNeT++

OMNeT++ is an discrete event based simulation library and framework written in C++. It is modular and extensible and primarily used to simulate networks not only limited to telecomunication networks. Models, the basic entity, are written in C++ and can be assebled to larger models. Those compound models, up to the whole network, are described with a higher level language, the NEtwork Description language (NED).

Models, as well as the parts of a compound model, communicate with each other via messages. Messages are handed between input and output ports, in terms of OMNeT++ called gates, of models. A model can send messages on a gate and is informed if a messages arives on one. Messages are also used internaly of a model to trigger scheduled events. Gates can be connected to gates of the composing models directly so messages can be passed “up” and “down” within a compound model. Models on the same layer can be connected via channels, which can have corresponding properties to indicate the characteristics of a communication channel like the bandwith, the delay or even an error rate.

With the NED a network of modules and interconnecting channels can be composed. Along with the models of such a network, the whole simulation is configured via an .ini file.

The simulation is processing a sequence of scheduled events which can call actions in the models, which can then trigger scheduling more events. The composition of a network to be simulated and the simulation process is described in more detail in the following.

5.1.3 The simulation process

The models code is compiled together with the simulatior. The simulator is then executed and a network is build according to the .ini file and the specified .ned modules.

After the initialisation and configuration phase of the nodes is done, the main simulation phase begins. In this phase the actual simulation process takes place. Previously scheduled events are processed in the specified order and trigger modules to exchange messages. This phase ends when no more messages are scheduled or on exceeding a maximum simulation time and is followed by the finalisation phase.

In the finalisation phase the most important task is to collect and eventually process measurements and statistics. Beside this, some modules need a proper clean up what is can be done in this phase.

5.1.4 Measurement recording

OMNeT++ provides two ways of recodring statistics. The traditional way was to calculate the values inside the modules and output them in in the finalisation phase in special vector or scalar objects. Because this required recompilation on even the simplest changes, like for example to record the minimum instead of the maximum of a value, recording by signals was introduced in OMNeT++ 4.1.

With this method the modules just have to emit a signal of a declared type with an optional value. How these singals are recorded and processed can controled via the simulation configuration.

5.1.5 INET: The communication networks simulation package

Since OMNeT++ is a general purpose network simulation framework not limited to a specific type of network, it contains no concrete network models. These are provided by additional packages. For communication networks, like the internet, this is the INET package. It contains models for a great varity of components the internet and similar networks consist off. Several wireless and wired protocoals are covered and reach from ethernet up to application protocols like HTTP.

For the darknet simulation we use the INET package to provide modular and realistic underlaying protocols. Depending on the level of realism to be achieved, the underlaying topology can be built as a single switch network or a whole internet like network with mulitple routers.

5.2 Implementaion of the basic model

As in any P2P Overlay an addressing scheme is needed for darknets as well. We chose a simple string representation for the nodes identification, the nodeID. Other addressing schemes can be mapped to strings if needed.

The DarknetBaseNode class implements the general behaviour of a darknet node and the basic interaction with the simulation framework components. It is responsible for sending and handling recieved messages. Selfmessages, used in OMNeT++ for notification and scheduling events are detected and dispatched to an overwritable method. This way, subclasses can easily use and extend this notification system. A simple implementation of sending messages to other nodes is implemented, including setting the source of the message to the node, calling the method that determines the nodes the message is send to and then sending the message to all these nodes. In the easiest case, a deriaved class has only to implement the method that chooses the next nodes to gain message sending functionality.

If no routing descision is needed, or possible, a simplier methoden sendDirectMessage is provided. It sends the message directly to the specified node. This is a huge simplification of the real world and only used in special cases like connecting an offline node that can obviously not in the connected list. The simplification at this point is valid, since in the underlying model of the simulation we concentrate only on the routing algorithm and the problem of how to contact a node whose underlay address (in the internet the IPv4 address for example) is unknown, is ignored.

The DarknetBaseNode class also basically manages the list of known trusted peers and the list of connections to online peers. The first lists limits to which other nodes connections are allowed. The second one keeps track of those who are online, that is who are available to forwarded messages to, along with the address in the underlaying network.

The second key property of darknets, the anonymized forwarding of messages for other nodes, is simply implemented as well. The source nodeID of messages that have to be forwarded are set to the nodes one. The nodeID from which the message was received is saved along with the request ID. Responses contain the requests ID they are the response for. On forwarding a message, the time-to-life counter (short: TTL) contained in each message is decremented by one. Thereby the network is not so easily overloaded and each request is garantueed to be terminating, but eventually failing for not reaching its destination.

Routing the response back the request came from can be called backrouting. The request ID can be used, while backrouting, to determin to which node a response of a request is to be send to. This whole mechanism is only a simplistic implementation of the corresponding darknet characteristic and can be extended or overwritten with a more advanced algorithm by derived classes.

As explained in the model description, we simplified the different reqeust and response types in different P2P networks possible to only one common request and an accoring response type. If further differentiation of the message types is needed, they can be implemented by extending the message class. How a node acts up on reception of such a message type can be defined in the handleIncomingMessage method.

5.3 Trust relation and network generation

Unlike as in a standard P2P overlay and modeled by existing P2P simulation frameworks as OverSim, a darknet node connects only to specific nodes. It is not possible to specify the (maximum) amount of nodes in the network and simply spawn or delete them on demand. Every node has to have a set of its trusted peers from the beginning.

To handle even very large social graphs of several thousands of nodes we developed a python script to generate the network composition in a .ned file and the node and simulation configuration in an .ini file. The script takes the network graph as file, starting with the nodeID and followed by all its trusted peers seperated by a blank. A directed symetric graph is assumed, in which each directed edge between two nodes has a reversly directed edge specifying them mutiality of trust between those nodes. However this is not enforced and complexer scenarios with not necessarily mutual trust relations can be modeled.

The network is set up with just a single router all nodes are connected to. To simulate complexer underlay network topologies, the script can be modified accordingly. Thereby it would be easily possible to simulate for example a split of a subset of the network by just putting some nodes on a different router and removing the channel between both routers during simulation.

5.4 Churn based lifetime model

The so far implemented static model lacks the capability of modeling nodes that go online or offline during the simulation. The model with a node lifetime adds this feature. In the implementation, a way has to be built in, that nodes can detect or get informed a changed state of one of their peers.

We decided to implement an acknowledge mechanism. Basically a acknowledge message (short: ACK) is send back on receieving a message. If such an ACK is not received within a certain amount of time, the message can be resent up to a selectable number of times. If no ACK is received after that, the peer is no longer considered the be online and connected. No further messages are send to this node.

If a node comes online, it connects to its peers. This incomming connection establishment request can be used to trigger such a request in reverse direction by a node itself. As these reqeusts do not rely on the list of connected peers, this can be used all the time. This process is applicable as well for rejoining nodes during the main simulation phase.

5.5 two simple example routing models

As a demonstration of the simplicity of extending the model and implement a darknet routing algorithm, we implemented two different, yet very simple routing algorithms applicatable to darknets. The first one is random walk routing in which a message travels on a randomly chosen path through a graph. In the second one, called flooding, a message is forwarded to all connected peers a node has. Both perform miserable compared to todays commonly used routing algorithms, but are both simple to implement and to follow, and are well researched.

5.5.1 The random walk algorithm

A “random walk” over a graph is a randomly chosen path between two vertices. These paths are commonly far away from optimum and there are basically no further constrains, even loops could be allowed. In an decentralized environment, these paths can of course not be selected beforehand, but every node just pics the next node from its available peers randomly. Basically this is all needed to implement this routing, the findNextHop method simply returns one randomly choosen entry of the “connected” list. Depending on the size of the network these paths will not hit the destination since the message exceedes its time to life.

A simple optimisation, which would virtually allways be used in pratice, comes at no cost: a simple loop prevention on the backrouting of the response. The node only saves the source nodeID of a request, the first time it sees this reqeust. Although, if requiered for the soundness of a model, this can still be implemented, depending on the conrete constrains of the model.

Another extension to the simplest random walk model was choosen to be investigated: the effect of multiple started requests on the average legth of found paths. The initial requesting node sends multiple requests which could all take different paths. Only the first of those requests ariving at the destination node is responded to. For this the nodes has to save the ID of reqeuests it already served.

5.5.2 Floody routing

The second implemented algorithm is flooding in which a message is forwarded to every available peer in the “connected” list. This is done just once for the first time a message is recieved by a node. This way, flooding can allways find the shortest path between two nodes. The disadvantage is a heavy load on the network since a message would be forwarded to every node in the network at least once, most nodes even would get the message multiple times from many of their peers.

To limit the impact on the networks load, a much smaller TTL is used compared to randomwalk, and therefor only nearby nodes will be reachable. Additionally the high degre of concurrency of events during the first flooding simulation was problematic since way to many messages were waiting for procession. To resolve this problem, we increased the time delay and its variance between requests.

6 Simulation and Evaluation

After implementing the built model into OMNeT++ modules the simulator is compiled. The easiest and prefered way is to compile all models into one binary together with the simulator code. Then, only the module composition and network desription from ".ned" files and the simulation configuration is needed to run a simulation.

6.1 Simulation environment and evaluation

We deployed the simulator and the needed configuration files on a simulaiton server. The machine has 24 cores, 128GB of RAM and runs a 64bit linux 2.6 kernel. Although it is possible to build multiprocessing simulations with OMNeT++, it adds a lot of synchronisation overhead. As our primal setup connects all nodes with a single router, there is no possibility to split the network in usefull partitions that can be simulated individually and only synchronize few exchanged messages. Therefor we decided to use only one processor core per simulation and just run multiple simulation simulataniously.

To evaluate the simulations all possible outputs by the modules were recorded with the described OMNeT++ signaling recording mechanism. They get processed, eg minimum, maximum etc are computed, by OMNeT++, and are written to a simulation specific recording file. From there, they are further processed by us with basic unix command line tools like "grep", "cut" and so forth.

6.2 Used metrices (nicht wirklich hier; eher in implementation)

- (average/max) path length
- sent message count
- faild routings / requests OR droped packages

6.3 Q: Scalability of the model/framework

Used RAM; RAM RAM and moar RAM (sprich: metriken die nicht in darknetzen anfallen also nur fuer simulation relevant sind kurz erklaeren)

6.4 Q: Impact of fanout degre at randomwalk on found pathlength

comparison to flooding which finds shortest path

6.5 Q: Probability of path failure on return for churn model

7 Conclution and Future Work

- Everything is epic, but.. ;)
- Extend OMNet model to take the underling network into account
  - real path length; relevant if protocol tries to take it into account but difficult in a darknet environment
- Implement tested darknets and compare to their tests
- Implement untested/new nets and improve routing parameters
  - or even decide if algorythm is pratical usefull or not