Infrastructure Aware Applications

Summary of the PhD Dissertation

by

Vilmos Bilicki

Supervisor:
Dr. Márk Jelasity

PhD School in Computer Science
Institute of Informatics
University of Szeged

Szeged 2010

Introduction: The success of the IP protocol stack and also the success of the Internet as a technology lies in its simplicity. Based on the classical hourglass model, a wide variety of first and second layer technologies are all connected to the IP and the intelligence is provided by the higher layers in most cases on the edge of the network. In theory, the details of the infrastructure are hidden from the applications and the application programmers so they can focus on the implementation of their own business logic and not be concerned about the different lower layers. The network devices in the core of the network are simple and stateless, their main task being fast packet forwarding, while logically near the customers the active devices may provide some basic services for the network operator. This paradigm is about 30 years old and since its introduction many things have changed. Almost all the services that have some connection with information exchange are now based or being based on the services provided by the IP network. The widespread use of overlay technologies and virtualization is also an important trend. This heterogeneous demand together with a wide variety of access layer technologies are the main facilitators of the paradigm shift we are now witnessing. The network is becoming more intelligent, while the applications and the underlying layers are becoming increasinlgy infrastructure aware. The main goal of this thesis is to provide an overview of this area through example applications. The thesis is divided into two parts: the first part deals with the network and provides an overview of the most important, but not well-known scalability issues of the intelligent services. The second part focuses on infrastructure aware applications through four examples. In the remaining part of the thesis summary, the following contribution points will be presented:

I/1 Network infrastructure

II/1 Hiding botnets

II/2 Low degree DHT in big churn

II/3 Scalable distributed storage

II/4 SIP compression

Part I - Network infrastructure: The contributions of the first part are related to the scalability issues of the network infrastructure concerning to the stateful services. First, we will briefly provide an overview of the layers of the network infrastructure defined in network engineering, with a basic description of the active devices used. Then an overview of the services requiring stateful handling of the traffic is introduced. After, the results of our measurements are presented.

I/1 Flows vs. stateful services: The IP network and the services offered on the top of the network are constantly evolving. First, let us take a brief look at the most important trends we should consider when we describe the infrastructure aware applications:

- The traffic generated by customers is doubling every four to five years (14PByte now, 42PByte in 2014)
- The widespread use of 3G as an access network (The IP traffic generated by the mobile users is expected to double every year until 2014)
- The growing popularity of the IPTV, Video-on-Demand services. By 2014 60% of the total IP traffic will probably video related.
- The widespread use of the P2P paradigm. Now video streaming makes up 7% of the total P2P traffic. New areas will emerge like content delivery.
- IPv6, Multicast. These technologies may serve as the basis for the widespread IPTV deployments.

We may conclude that the IP network is growing both in the terms of size and importance. In order to be able to understand what the *infrastructure aware* term means for an IP network some basic knowledge about the IP ecosystem is required. The IP network is not a monolithic entity, but it is built from many interacting intermediate-systems or end-systems. In the following we will provide a top-to-bottom overview of the current IP networks. The Internet is the network of the networks. It is built from more than ten thousand autonomous systems (AS) controlled by different legal entities. The interconnection among the ASes is done in peering points, where they exchange both the traffic and the routing information. The so-called backbone of the Internet is formed from these public or private peering point and the ASes. There is no central organization behind the Internet and there is no dedicated backbone. No one knows the exact topology of the Internet. As even the exact topology is unknown, the same is true for the traffic flows on the Internet: no one knows who the are in communicating parties on the Internet are in any given time frame. The intra AS networks may span the whole world, but these networks could also be a focused on a single location such as a campus. In most cases these networks are not ad-hoc, but they follow the well-known hierarchical engineering approach where the network is divided into the core, distribution and access layers. These layers have their own specific role in the network and the active device vendors design their device portfolios according to these three layers. In the following we will discuss the roles of these layers:

- The goal of the access layer is to provide the last mile for the end-systems, for an Internet Service Provider to the customers. A significant percentage of stateful services is deployed in this layer.
- In a larger region the islands of access layer networks are connected to each other through the distribution layer. Stateful services are less common in this layer.
- The goal of the core or the backbone is twofold: it should provide redundant data paths for the regional distribution layer islands, and the connection to remote networks (ASes) is also handled here. In most cases this layer is simple and has no stateful service.

The placement of the stateful services described here reflects the past best practice and it may change over time. One trend which could change this is the so-called future Internet where one basic idea is to foster a collaboration among the applications and the networks. The networks should understand the applications and they should provide context-based services (e.g. routing for Web services). Another step in this direction is the idea of active networks where significant intelligence could be placed on each device and these devices are open for third party applications. But let us now focus on the currently available and deployed stateful services. Stateful packet handling in our terminology means that the packets are treated based on some internal state maintained by the router. There are the following well-known services requiring state maintenance on the active devices (Routing, NAT, Netflow, Application recognition, Firewall). A common basic step of all stateful services is a lookup in the list, which is done for each incoming packet. In the case of routing this is called the IP lookup. The processing power needed for this lookup could be quite high. For example, in a distribution layer router with 48 interfaces each having a 1 GBit/s data transfer capability it means that in the worst case for small packets (e.g. 100 Byte long) the device should handle 64 packets every micro second. In other words, it has about 15 nanoseconds for each packet. The access speed of the currently available memory chips is in the range of 55 to 23 ns. In order to be able to find a given entry multiple lookups are needed. Depending on the data structure, the number of memory accesses for an exact match depends on the number of entries in the table (with the B-Tree data structure this is $O(log(n))$, where n is the number of entries). The state-of-the-art silver bullet for solving this issue is CAM (Content Addressable Memory), which is a hardware implementation of the associative array. It returns the address of the cell (or the contents associated with the cell) in one memory access cycle. For the longest matching lookup a special CAM called the Ternary CAM is used. The price and the power consumption of these memory banks is high. Due of this, even in the high-end routers the storage capacity of the TCAMs is around several hundred thousand entries. Also, there are various active device architectures. The low-end routers do the decision making with the help of CPU and conventional memory, while the middle- and high-end devices use TCAM for the lookup.

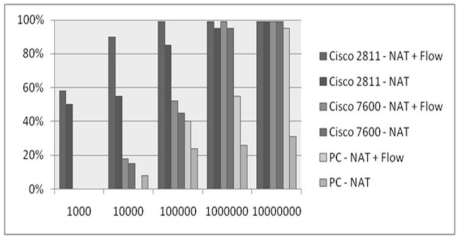


Figure 1: Packet loss

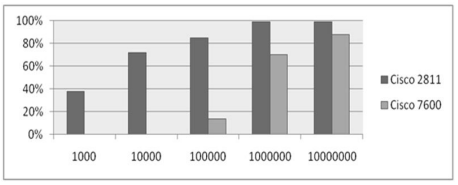


Figure 2: Netflow export loss

We conducted experiments with traditional PCs and active devices operating in different architectural layers with both unicast and multicast traffic in order to investigate the scalability issues caused by an increased flow number. Figure 1 shows the results of unicast measurements. We tested the scalability of the NAT and the NAT+Netflow stateful services on an access layer device (Cisco 2811), on a core layer device (Cisco 7600 + NPEG2) and on a PC. Figure 2 shows how the Netflow export accuracy is affected by an increase in the number of flows going trough the test device. With higher flow numbers the ratio of Netflow export packets rises. The results show, that depending on the capabilities of the device, there are thresholds where the given stateful service cannot process the traffic. Let us also mention here that we did not saturate the active devices (the traffic was less than 10 to 30% of the backbplane speed of the active device), but only offered a traffic volume built from 10 to 10M flows in the case of unicast measurements and 10 to 60K flows in the case of multicast measurements. From our measurements we may conclude that the number of flows going through an active device has a significant impact on the stateful services running on the device and this fact should be considered by the network engineers and application developers.

Thesis 1 *The performance of stateful services implemented in different layers depends heavily on the number of unicast or multicast flows.*

It is evident that the P2P ecosystem is responsible for a significant percentage of the total IP traffic. However, the flow level distribution of this traffic share is not well understood and has not been analyzed. The different P2P approaches use different peering strategies and also the selection of the underlying communication services. Due to this heterogenity, it is not trivial to treat the P2P ecosystem as one class of applications from the viewpoint of the number of flows generated. This issue arises in the literature where it was shown in a client-based analysis of the traffic, when the number of concurrent flows for a WWW is in the range of several tens or at most one hundred, the number of concurrent flows for P2P applications is one magnitude higher. In a protocol-based analysis it turned out that the Bittorrent users are responsible for this phenomenon as over 24% of the them have over 100 parallel connections. We may conclude that some of the P2P applications generate a significant flow volume. Currently the ISP Friendly P2P research community focuses on the traffic volume and the localization of the traffic. The number of flows as an additive metric for optimization has not yet been considered. We saw in Thesis 1 that the number of flows generated by the applications should also be considered both by the traffic engineers and the application developers.

Thesis 2 *The ISP friendly P2P service should take into account the number of flows generated by the overlay too.*

Some interesting questions we would like to answer in the next parts are:

1. As both the Internet and the P2P Botnet on the top of the Internet are distributed in nature, is it possible to detect a sophisticated botnet from a single point (single AS)?

2. As we have seen, the degree of the nodes (the number of flows generated) forming the P2P overlay is important. Is it possible to build a low degree overlay which is stable even in the case of significant churn?

3. We saw that with the all-IP solution we use the same protocol stack on the top of different access technologies. In the case of 3G the bandwidth is a scarce resource and compression seems to be a feasible solution for the new protocol. So can the current compression algorithms be applied without modification?

4. If we consider a localized, but extremely distributed system with strict performance criteria (file share), a question arises of how we could achieve robust but effective consistency for a file store.

Own contribution The results shown in this thesis group are all the results of the author. The results related to the multicast traffic of this contribution point were published in research paper [2].

Part II - Infrastructure aware applications: In this part we would like to answer the questions arising from the previous part.

Issue: As both the Internet and the P2P Botnet on the top of the Internet are distributed in nature, is it possible to detect a sophisticated botnet from a single point (single AS)?

II/1 Hiding botnets: If we want to identify and filter P2P botnet traffic in an automated way, we can focus on roughly three kinds of activity: propagation, attacks by the botnet, and overlay traffic, which involves repairing failed overlay links, adding new nodes to the overlay, and spreading and searching commands of the botmaster. We argue that the most promising approach is to focus on overlay traffic. It has a small volume, but it is arguably the most regular and reliable traffic any P2P botnet generates, since the overlay network has to be constantly repaired, and bots need regular information about commands of the botmaster, updates, and so on. The basic motivation of our work is that we think that the *structure* of P2P networks is a very promising, although difficult, target to try to detect. The structure is completely insensitive to actual flow characteristics: Nodes can mimic other protocols or they can behave randomly, but the traffic dispersion graph they generate must still reveal the overlay network they are organized in. Evidently, this structure will have to be correlated to malicious behaviour as well. It has already been argued that it is very difficult to detect P2P traffic using packet or flow classification methods alone. Most of the key characteristics of P2P traffic lie in the network defined by flows, called the traffic dispersion graph (TDG). By building and analyzing TDGs of locally observable flow data after the classification phase, it is possible to extract important additional clues about the organization of an application and, for example, label them as P2P. The TDG is defined on top of a set of flows S that have the usual format <srcIP, srcPort, dstIP, dstPort, protocol>. The TDG is the directed graph $G(V, E)$ where V, the set of vertices, contains the set of IPs in S, and E, the set of edges, contains the edges (a, b) such that there is a flow in S with srcIP $= a$ and dstIP $= b$.

Since we would like to challenge the feasibility of local methods for detecting botnets, in order to be convincing we need to be generous to the local methods that are available for traffic classification. Therefore we assume that *traffic that belongs to a given P2P botnet can be isolated in an unlabelled way*. That is, we assume that there are methods available to group a set of flows together that belong to the P2P botnet, but that we cannot determine whether the identified class of traffic is, in fact, botnet traffic. Note that this is already an extremely strong assumption. We will argue that even with this assumption, it is very difficult to identify the traffic as P2P traffic generated by a large P2P network if a P2P botnet applies certain P2P techniques. To show this, we will examine the TDG-based P2P traffic identification approach.

As a P2P protocol model for the botnet, we use an ordered ring with exponential long-range links, a simplified version of the Chord topology: we have N nodes with IDs $0, 1, \ldots, N-1$. Node i is connected to nodes $i-1 \pmod N$ and $i+1 \pmod N$ to form the ring. In addition, node i is connected to nodes $i+2^j \pmod N$ for $j = 1, 2, \ldots, (\log_2 N) - 1$, which are the long-range links.

It is important to make a distinction between the overlay and the flows that exist in the overlay. Two nodes a and b are connected in the overlay if a "knows about" b. This, however, does not imply that a will ever actually send a message to b. For example, a might remember b simply in order to increase robustness in the case of a

failure. On the other hand, a node a might send a message to b even though b is not the neighbour of a in the overlay (that is, for the overlay to function properly, a does not need to remember b after sending it a message). For example, in Kademlia if a wants to find the node of ID x then a will actually make contact with all nodes on the route to x. This is why Storm bots generate so many messages locally as part of the overlay traffic.

In short, we want to model the flows and not the overlay *per se*, so our model refers to the *flows* we can potentially observe. In the actual overlay there would probably be links to the 2nd, 3rd, etc, neighbours in the ring as well that are learned from direct neighbours.

In the following we describe two fairly straightforward techniques that future botnets could use to hide their traffic. The key point is that, using these techniques, the functionality of the overlay can be preserved while using far fewer links and traversing fewer routers:

- **Clusters for Sharing Long Range Links** - In the ring every node has two neighbours that it actually communicates with at any given time, but it has $\log N$ long-range links, all of which are frequently used for communication to achieve as few as $O(\log N)$ hops in overlay routing (where N is the network size). Evidently, the ring would be sufficient for communication, but then sending a message from a node to another random node would require $O(N)$ hops in expectation.

  There is a middle ground: we can reduce the number of long-range links to a constant number, and still have relatively efficient routing: $O(\log^2 N)$ hops or even $O(\log N)$ hops. However, let us keep in mind that we are interested in the flows and not the overlay. In fact, we can modify our model to have a *single* long-range flow per node, and still have $O(\log^2 N)$ hops for routing messages in expectation. The trick is to create clusters of $\log N$ consecutive nodes in the ring, and allow each node to actually use only one of its long-range links. Routing proceeds as usual. But when a node decides to send a message over a long-range link, it first has to locate the node in its cluster that is allowed to use that link and send the message to that node along the ring. Note that nodes that are in the same cluster can rely on an identical set of long-range links since clusters can be interpreted as replicas of a node in an overlay of size $N/\log N$.

  Next, we state without proof that a much simpler stochastic approach in which we have no clustering at all, but where each node can use only one random long-range link results in a similar routing complexity in expectation. Here a node has to look at its $\log N$ neighbours in the ring and pick the best long-range link that is allowed by some of these neighbours.

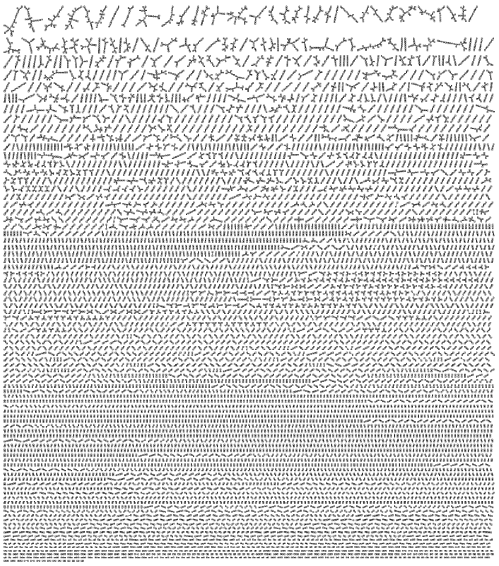
  Therefore, from a flow viewpoint all nodes now have two ring flows (one in and one out) and two long-range flows on average (one out and one in on average).

- **Locality** - We can also optimize the ring by attempting to assign IDs to nodes in such a way that the resulting ring has links which intersect the smallest possible number of routers. Several algorithms are known for achieving such optimized topologies that could be adapted to this application.

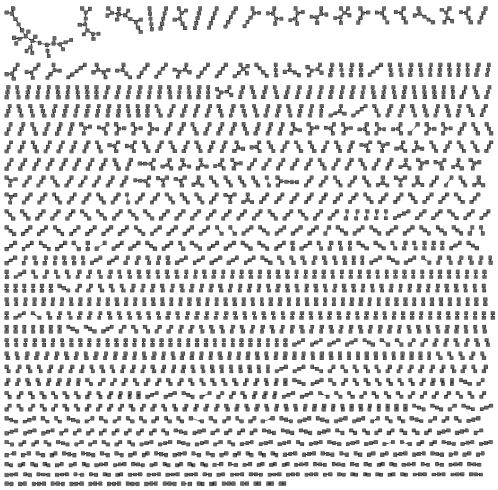
To examine the partial TDGs as seen locally from several points of the Internet, we (i) created a static AS-level model of the Internet topology and routing (we applied the results from CAIDA project), (ii) mapped the overlay network (100,000 nodes) to this AS-level topology, (iii) and we analyzed the local TDGs that are defined for each AS by the set of traversing flows in our model. A visualization of the TDG associated with the most central AS is shown in Figure 3. Here we may conclude that with the help of localization and clustering that the supercomponent present in the case of random mapping(Figure 3c) disappears (Figure 3a) even with an AS with largest beetweeness value. The information available at the less central ASes is significantly less (Figure 3b). The maximal node degree we observed in every TDG that we generated is no more than 4.

Overall, then, we may conclude that when localization and clustering are applied, the overlay network traffic is almost completely hidden. A non-trivial proportion of the traffic can be seen only at the most central ASes, but even there, what is visible is predominantly unstructured.

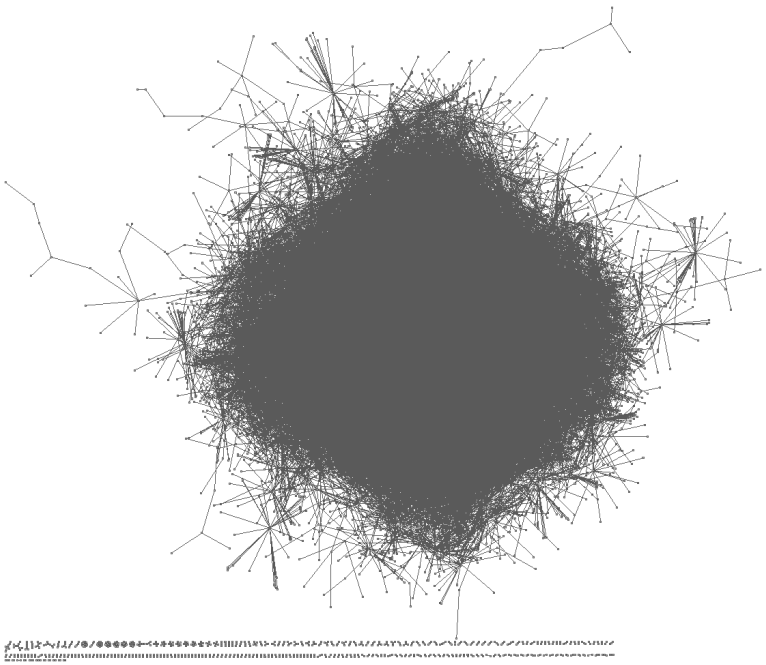
Thesis 3 *It is possible to build P2P botnet which could not be detected with the help of the TDG method from a single point (AS).*



(a) AS174 (betweenness 48,904,554), localized, clustered



(b) AS3491 (betweenness 4,460,142), localized, clustered



(c) AS3491 (betweenness 4,460,142), random

Figure 3: Visualizations of TDGs at various ASes. AS174 had maximal betweenness in the dataset.

Thesis 4 *With the help of localization and link clustering, the P2P botnet can hide from a single point TDG-based monitoring.*

The statistics we collected were the following: the number of nodes, the number of edges, the number of weakly connected components, the size of largest weakly connected component, the average node degree (where we count both incoming and outgoing connections) and finally, a metric called InO. InO is the proportion of nodes that have both incoming and outgoing connections. The first observation we made was that the most efficient factor for hiding the overlay traffic is via clustering. Recall that the main effect of clustering is to reduce the flows each node participates in from $O(\log N)$ to 4 on average. The effect of localization is significant as well, but it is less dramatic overall.

Thesis 5 *Localization has a minor effect on botnet hiding and link clusterisation has a significant effect on the visibility of the P2P botnet.*

There is one exception: the largest connected component, where localisation results in a value that is two orders of magnitude smaller than that for the two most central ASes. We should emphasize that results presented here are based on the assumption, that within one AS transit traffic, traces can be aggregated and treated in a unified way. Although not impossible, this is a rather strong assumption, especially for the most interesting ASes with high betweenness centrality, which handle enormous volumes of transit traffic. In practice, the information visible locally could be even more fragmented.

Conclusions: We analyzed the detectability of DHT-based P2P botnets from a single AS based on TDG graphs. We created a novel DHT approach based on local clusters. It turned out that localization itself is not enough for hiding botnets, but combined with our local cluster approach botnets could not be detected from a single point on the Internet.

Own contribution: Thesis 5 was the work of the author, while the rest of the theses are joint results. The results of this contribution point were published in research paper [6].

Issue: As we saw, the degree of the nodes forming the P2P overlay is important both from the viewpoint of network scalability (Thesis 1, Thesis 2) and botnet hiding (Thesis 3). It is an interesting question of whether it is possible to build a low degree overlay which is stable even in the case of significant churn.

II/2 Small degree DHT with large churn: In general, P2P clients typically have a large number of neighbours due to maintenance traffic, and regular application traffic such as a search. There are only a few notable exceptions, such as Symphony and Viceroy, which are overlay networks of constant degree. It is still an open question of whether it is possible to create overlay networks of a very small constant maximal degree that are efficient and scalable. Research activity concerning Symphony or Viceroy has not yet focused on the lower end of maximal node degree, potentially as small as 3 or 4. In fact, even negative results are known that indicate the inherent lack of scalability of constant degree networks. In the following we will show that it is possible to build a Symphony-inspired overlay network of a very small constant degree, and with the application of a number of simple techniques, this overlay network can be made scalable and robust as well.

We applied the following techniques for reducing the number of routing hops in small constant degree networks:

- Lookahead - Greedy routing can be augmented by a lookahead procedure where nodes store the addresses of the neighbours of their neighbours locally as well, up to a certain distance.

- Degree balancing - To enforce a strict small upper bound on node degree, nodes that are already of maximal degree have to reject new incoming long-range links.

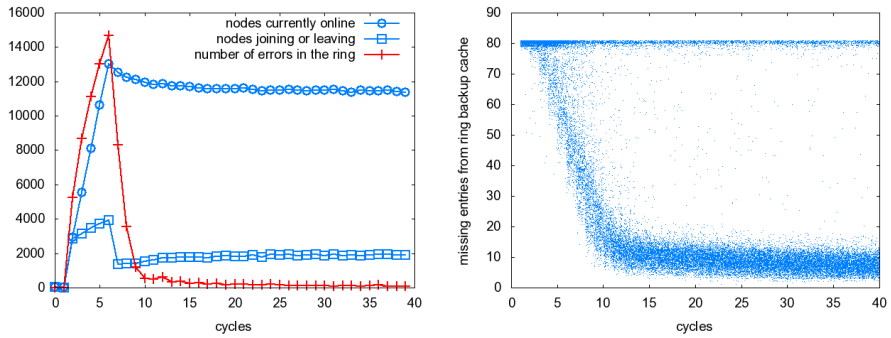


Figure 4: The evolution of the topology and the backup links for $N = 2^{14}$. In the figure on the right points belong to individual nodes and they have been randomly shifted so as to visualise the density.

- Stratification - Since each node has only a small constant number of long-range links (1 or 2 in our case), many hops will follow the ring. It is therefore important that neighbouring nodes in the ring have different long-range links. We propose a stratified sampling technique that involves dividing the long range links into a logarithmic number of intervals $[e^i/N, e^{i+1}/N]$ $(i = 0, \ldots, [\ln N] - 1)$.

- Short-link avoidance - Interestingly, if the average route is long, then it might be beneficial to exclude long-range links that are too short. We introduce a parameter m: the number of shortest intervals that should be excluded when selecting long-range links. For example, for $m = 2$, the first possible interval will be $[e^2/N, e^3/N]$.

In order to study the routing performance (hop count) of the static network, we applied simulation and we found that routing in networks of a very small constant degree is feasible if the given techniques are applied. We found that the most effective technique is lookahead, based on locally available information in the neighbourhood of the nodes. In addition, degree balancing is very important. Additional methods such as short-link avoidance and stratification also provide a 5-10% improvement, depending on the parameters. With these techniques we can route in around 30 hops in a network of size $N = 2^{20} \approx 1,000,000$ with a maximal node degree of only 4.

P2P networks are dynamic so fault tolerance is a critical issue; and the use of backup links can make Symphony scalable. We showed that Symphony-like topologies can be made scalable by adding only $O(\log N)$ backup links for all the links, and under moderate failure rates even $O(\log \log N)$ will suffice.

Thesis 6 *The Symphony topology is scalable; that is,* $\lim_{h\to\infty} p(h, q) > 0$*, if (i) all the links have* $O(\log N)$ *backup links, or if (ii) all the links have* $O(\log \log N)$ *backup links and* $q \leq e^{-2} \approx 0.135$.

This is rather counter-intuitive given that routing still takes $O(\log^2 N)$ steps. It is also promising, because these results suggest that collecting good quality backup links can dramatically improve scalability at a low cost.

We performed the experiments using the PeerSim event-based simulator. Our goal was to design a protocol to construct and maintain a Symphony topology in a fault tolerant way, with backup links described previously. To this end, we applied T-Man; a generic protocol for constructing a wide range of overlay topologies and we extended it with the above-mentioned methods.

The scenario we experimented with involves node churn. Applying appropriate models of churn is of crucial importance from a methodological point of view. Researchers have often applied an exponential distribution to model uptime distribution, which corresponds to a failure probability independent of uptime. Measurements of a wide range of P2P networks suggest that a Weibull distribution of uptime is more realistic, with a shape parameter around $k = 0.5$.

Figure 4 illustrates the speed at which the ring topology was formed despite continuous churn. The improvement caused by the backup links (80 links per node) for the ring links can also be seen. Moreover, most nodes

collect good quality backups, but some of them seem to have no useable backups at all; these nodes have a very short session time and spend very little time in the network.

We also found that despite heavy churn, which results in a constant fluctuation of the ring neighbours, the effective degree was small and the network scaled well. Short-link avoidance improved the hop count by a large margin.

Thesis 7 *It is possible to build a low degree DHT which is robust even in the case of large churn.*

Thesis 8 *The T-Man based DHT exetended with the four methods(Lookahead, Degree balancing, Stratification and Short-link avoidance) is stable in the case of significant churn (Weibull k=0.5)*

Conclusions: We demonstrated the feasibility of P2P systems where nodes communicate only with a very limited number of peers during their lifetime. We empirically analyzed known as well as new techniques from the point of view of improving the search performance in a Symphony-like overlay network. Then we presented theoretical results which indicate that if we add $O(\log N)$ (or, in a certain parameter range, $O(\log \log N)$) backup links to all the links (where N is the network size), then a constant degree network becomes fault tolerant even in the limit of infinite network size, while its effective degree remains constant (that is, the network can remain in stealth mode) since the backup links are not used for communication unless they become regular links replacing a failed link. This result is counter intuitive because routing in Symphony requires $O(\log^2 N)$ hops on average. We also provided event-based simulation results over dynamic and realistic scenarios with a proof-of-principle implementation of a constant degree network, complete with gossip-based protocols for joining and maintenance. Our results have at least two implications. First, they are a strong indication that P2P botnets need to be taken seriously by the P2P community. In our previous part we showed that stealth mode P2P networks are practically invisible for state-of-the art methods for P2P network detection. Current botnets do not exploit P2P technology to its full potential, and by the time they learn how to do that, they will be very difficult to detect and remove. The second implication is not related to malware. Due of the effects of the number of flows (e.g. several P2P protocolls) it is important to know how the small degree P2P networks could be made reliable.

Own contribution: Thesis 8 is the result of the applicant, while the rest is the result of shared work. The results of this contribution point were published in research paper [5].

Issue: We saw that with the all-IP solution we use the same protocol stack on the top of different access technologies. In the case of 3G, the bandwidth is a scarce resource and compression seems to be a feasible solution for the new protocol. So can the current compression algorithms be applied without modification?

II/3 SIP compression: Mobile system providers have invested a huge amount of money into their systems. To a get return of their investment they have to provide acceptable services for customers. The customers would like to use these services independently of their location and hardware framework. There is a need for a communication protocol to check the available and required services and their parameters. Session Initiation Protocol (SIP) has been chosen by 3GPP for this purpose. Now it is clear, that today SIP is one of the most important Internet protocols in 3G mobile core system. However, it is highly redundant, because it is an extendable ASCII-based communication protocol. Recently, concerns have been raised in 3GPP that session setups in the all-IP network were too lengthy due to excessive signaling over the radio link. The delays were estimated to be as long as 10 seconds . To overcome this problem, they agreed to use some kind of signaling compression along with UDP/IP header compression to shorten the transmitted messages. 3GPP then raised the issue with IETF. After evaluating several proposals, the Robust Header Compressing Group (ROHC) came up with the idea of a universal decompressor, and defined a communication layer, called Signaling Compression (SigComp) for this functionality. IETF has left many issues open or implementation specific, such as the negotiation of algorithms and parameters (like buffer sizes). It is also not yet known how to integrate SigComp with the SIP protocol, which is the main subject to compression.

We did not find any articles in the literature which deals with SIP compression. So our task was first to analyze the SIP protocol and compression theory to find feasible approaches for the compression. We developed a demonstration system which connects two SIP user agents to each other and ensures the compression and decompression of the messages between them.

We present three approaches that are based on each other and indicate the level of SIP specificity. We applied the following measurements in order to evaluate the different algorithms: we measured the efficiency of the compression with the help of a publicly available SIP flow collection, and the time needed for compression and decompression was estimated with the help of CPU cycles needed to execute a given algorithm on UDVM or in the case of compression, directly on the CPU.

- In our first approach we treated the messages as simple text-based messages, which was why we could not write efficient compression algorithms. First we tested the LZ77 algorithm. Since the dictionary contained only a few keywords as well as all the 256 ASCII characters, the compression ratio was over 100%. In the second test the Huffman algorithm was used. The tree was built based on the frequency of occurrence of the characters in the message and then the message was compressed character by character. Since the tree was built dynamically, we had to include information about the tree that is needed to decompress the message. This solution did not work because the compressed message was longer than the original. The compression ratio was over 150%, but for shorter messages the ratio was actually over 250%. This terrible (un)compression ratio clearly shows that the well-known compression algorithms cannot be used "as they are" in the case of data transfer with the SigComp layer.

  Thesis 9 *The classical compression algorithms without modification are not suitable for SIP compression.*

- Afterwards we used dictionary-based algorithms, because we had preliminary information about the message that can be incorporated into the compression algorithms as probability values. We found a dictionary with SIP instructions ordered in terms of decreasing probability values. We know that numbers and normal characters are more probable than special characters, and groups of symbols may appear more than once (run-length encoding). We constructed several compression algorithms (mainly differing in symbol coding) starting from the well-known algorithms (LZ77, Arithmetic, Huffman). These algorithms were modified so that they could use our dictionary efficiently. Moreover, we made all of them adaptive. We used a kind of *Move to Front* algorithm to change the symbol probabilities. When a symbol occurred, we decreased its position in the dictionary. We used the dictionary both on the encoder and decoder side, improving the efficiency of the compression. The dictionary is of great importance in the case of the LZ77 algorithm, thus we used the revised dictionary mentioned above. It resulted in some additional enhancements so that the length was represented only by 1 byte, while in the first test the length was 2 bytes. However, the real improvement was that we did not encode the short matches (1, 2, or 3 long) by LZ77, but these were forwarded un-coded. We introduced a special sign to distinguish the encoded and the un-coded parts. The efficiency of the modified algorithm was still not as good as any of the Huffman or SubExponential compressors, but it was more efficient than the methods implemented as our initial test. The compression ratio was between 55% and 75%. Recurring parts in the message to be compressed can further increase the efficiency, and a SIP message is one of these kind of messages. The efficiency of the Deflate encoder rose by 10-12%.

  Thesis 10 *The modified Deflate (run length encoding + context modeling) provides the best compression ratio.*

  We measured the time needed for compression/decompression and based on this measuerement we found that the modified Deflate (run length encoding + context modeling) is the best SIP when the call setup delay is important. Still, the modified LZ77 was the fastest on the decompressor side.

Thesis 11 *The modified Deflate (run length encoding + context modeling) is the best SIP when call setup delay is important.*

- The entropy of a message flow is better than the entropy of a message, so we can achieve better compression ratios when compressing message flows. Better compression results could be attained by using previous messages (dynamic compression) because the similarity of the messages could be high (e.g. the address of the sender and the receiver is the same) and there are algorithms which can benefit from this property. This approach looks promising so it should be investigated further.

Conclusions: Based on our experiments, we found that there is no optimal solution for compression/decompression. Each compression algorithm can be good or bad depending on the criteria specified by the application. We should add that the compression ratio is normally considered to be the most important: But in the case of data transfer using the SigComp layer, there are special requirements and, as a consequence, the speed and the memory usage are more important than the accessible compression ratio.

The modified Deflate (run length encoding + context modeling) algorithm provides the best SIP call setup transfer time (1.62 s). However, this is the slowest solution on the compressor side and with many concurrent sessions (SIP proxy) this can cause a bottleneck. The modified LZ77 algorithm is the fastest on decompressor side. Context modelling (i.e. prefix-free encoders) is the fastest on the compressor side. We note that the algorithms are dictionary-based, which is optimal for messages with no special symbols. With help of adaptive methods we can extend this optimality to messages with special symbols as well.

We achieved compression ratios below 50% and decreased the total delay of SIP call setup by 1.5 seconds. With dynamic compression (compressing message flows), the compression ratio could be under 30% and the transmission delay about 1 s. The experiments showed that SIP messages can be efficiently compressed. In the future we would like to study dynamic compression and see how efficiently SIP can be integrated into the SigComp layer.

We think that our tests justified the philosophy of the protocol architecture; that is, let us allow the communicating parties to select the best method for themselves according to their requirements.

Own contribution: The results seen in this thesis group are the result of the author. The results of this contribution point were published in research paper [4].

Issue: If we think about a localized but extremly distributed system with strict performance criteria (file share) a question might arise of how we can achieve robust, but effective consistency in the case of a file store.

II/4 Scalable storage: It is not unusual to find more than 100 GBytes of storage capacity, over 500 MBytes of RAM and two GHz or more CPU clock frequency in a desktop PC. It seems that these parameters are constantly increasing. A typical installation of an operating system and the software required does not consume more than ten to fifteen GBytes. The rest of the storage space is unused. A typical medium-sized company has more than 20 PCs. A university or research lab usually has over two hundred PCs. In this case the storage capacity that is wasted may be several TBytes in size. So it would great if we could utilize this untapped storage capacity. In order to solve the above-mentioned problem we decided to design and implement LanStore with the following design assumptions:

- It is highly distributed without central server functionality.
- It has low server load. We would like to utilize the storage capacity of desktop machines; these machines are used when our software runs in background.
- It is optimized for LAN. The use of multicast and a special UDP-based protocol is acceptable.

- It is self-organizing and self-tuning. We used a multicast-based vote solution to implement the so-called 'Group Intelligence' in order to handle computer laboratory churn.
- It is a file-based solution. For effective caching we chose file-based storage instead of a block-based one.
- It has campus, research laboratory-type file system usage. Also, file write collisions are rare.
- It has an optimal storage consumption failure survival ratio. As a first approach we selected Reed-Solomon encoding for data redundancy.

To utilize the storage capacity of the member nodes in a uniform way, we divided the files into equal fragments. In this way every storage node has the same number of stored data fragments. We would like to collect the free space from PCs in computer laboratories, classrooms, and so on.

There is a high probability that one or more machines will be rebooted or turned off. We measured the churn in the laboratory and found that the mean number of the online Windows workstations was always above the critical level of 50%, but there was a significant fluctuation (over 10%) so an algorithm which provides failure protection for the stored data is necessary. We opted to use forward error correcting codes (FECs) for error correcting. With the help of these algorithms we created n data fragments for m original data fragments. This means that we could reconstruct n failing data fragments. The consistency of the metainformation and the state of the workflows running on the system was provided by a voting algorithm. If there was a critical number of working data nodes, the remaining nodes could be reconstructed. Our solution was transaction-based. At the end of a transaction a vote was taken and any changes were written to a permanent storage unit when the majority of nodes agreed on the next common state. If there was no majority acceptance of the new state, the transaction rolled back. After the changes were written into a permanent storage, a second vote was taken of the result. If there was a successful majority vote the whole task was marked as fulfilled; if there was no successful majority result the first and the second transactions rolled back. Here we will not go into the details, but highlight only the most innovative parts of the framework.

Data redundancy module: The task of this module is to provide the necessary data redundancy for error correction. Several approaches are available in the literature. The most popular one is that of data mirroring. This is an easy-to-use and implementable technique with low processing overheads, but we pay the price on the storage consumption side. The creation of data parity blocks is another popular way, but apart from its optimal storage consumption this technique can correct only one error at a time. This is a big drawback. In our case a special class of forward error correcting codes (FECs), called erasure codes provides the best solution. Since we can detect failing data, we only have erasure errors. In the case of FECs one can select the required redundancy level and the algorithm generates the necessary error correcting data blocks for the existing data blocks. If a data block fails, it can be calculated from the remaining data and error correcting blocks. Here there are two types of FECs: codes with guaranteed error correcting capabilities and codes which have an error correcting capability with a given probability. We opted for the first code family because of its guaranteed error correcting capability. The price, however, is the processing overheads which depend on the selected error correction capability. This is one or two magnitudes higher than that for the second case. We chose a special case of the Bose-Chaudhuri codes called the Reed-Solomon code. The basic theory for this is quite straightforward: we have n data blocks and we need m data blocks to correct fewer than m erasure errors. To produce m data blocks we require a special equation system where every partial matrix is invertible. To produce such an equation system, the Reed-Solomon approach makes use of the Vandermonde matrix. The Galois field is used as the space where the operations are performed. With this solution we replace the complex calculation-intensive operations by lookup tables. Here we use the Luigi Rizzo implementation of the Reed-Solomon code. The module divides the processed files into 64 KByte long stripes and calculates redundancy data for these slices. These stripes form the basic unit of the versioning system.

CPU Clock Frequency (GHz)	N	K	Throughput (MBit/s)
1	64	32	40
2	64	32	80
3	64	32	120
3	200	100	38.4

Table 1: Reed solomon performance

Our Paxos implementation: Functionality is provided by three abstractions: Leader, Consensus algorithm and Learner. From a high-level point of view the system works as follows. The clients send instructions to a leader. This leader carries out a three-phase transaction on the participating nodes and sends the results to the client. We implemented the Paxos algorithm using several optimizations to achieve better response times. For the system to progress we need the majority of nodes to be live. It may happen that in a fluctuating system, the majority of nodes are always present, but are constantly changing. For example, the prepare request is received by node A, then node A restarts and node B finishes its restarting process. So node B will only receive an accept request. The classic Paxos algorithm recommends rejecting this message. But with this solution it can happen that we have to replay the whole propose/accept procedure. Instead of this we suggest the following. If a node receives an accept request without previously receiving a propose request it shall answer this request. If it disagrees with the value suggested by the accept request it shall handle the accept request as a propose request; if it agrees with the received value then it shall handle the accept request as a propose and accept request. With this modification we did not change the durability of the algorithm, but in some cases we reduced the required number of message exchanges from six to two.

Evaluation: First we evaluated the performance of the Paxos implementation with the help of simulations. It performed as it was expected. With higher churn ratio the progress of the transactions was slover, but the consistency was guarantied.

As a second step we measured the raw encoding capacity of Reed-Solomon encoding. The results are shown in Table 1. We may conclude that the currently used processors produce a useable throughput for 64/32 (64 nodes, and out of these 32 contain error-correcting information).

Thesis 12 *The Reed Solomon encoding-based error correction method is a feasible solution for file encoding with current CPUs and the churn level we measured in the laboratory.*

The above-mentioned measurements only provide a picture of the raw coding capacity of a typical PC. Although this process is the most time-consuming part of the whole transaction, the remaining task could add significant delays. To be able to compare our solution with already existing systems, we tested our framework in different scenarios. One common method of file system testing is the Andrew benchmark, which was created to measure the efficiency of the Andrew file system. It measures the time needed for specific tasks (create file, etc). Among these popular tasks, the size of the manipulated files is important. The distribution of file sizes of the UNIX file system could be modelled as a Pareto distribution with parameters a=1.05 and k=3800. It is known that the Windows file-system file length distribution could be modelled with the help of a lognormal distribution and a tail with a two-step lognormal distribution. As a simple, but appropriate solution we chose the Pareto distribution to model the file size distribution of user homes. We created an application which generates files with the length of a Pareto distribution and the depth of its directory path had a linear distribution. Each character inside the files was generated with a linear random distribution. We uploaded and downloaded the generated file/directory set with the help of the GUI. We used the Windows SMB file share as a comparison partner. A test network was set up with 10 PCs, each having P4 3 Ghz processors, 1 GByte of RAM and a 100 MBit/s network adapter connected via a HP4108 switch functioning as server nodes and a similar PC as a client node. The redundancy ratio was set to 7/3, so for every seven original data items three error correction items were

	Lanstore		Windows file share	
	Delay	Throughput	Delay	Throughput
a	353	-	5.3	-
b	116	-	3.8	-
c	213	0,02	3.5	1,25
d	53	0,08	6.1	0,7
e	262	4.02	144	7,32
f	240	4.39	104	8,5

Table 2: Results

generated. The following tasks were measured on the LanStore and on a Windows share which was one of the server nodes:

1. The delay of directory creation (a), and deletion (b) in seconds, with 615 randomly generated directories, with depth and the name space of a random linear distribution. We executed this task on LanStore and on a Windows share system.

2. The delay of file upload (c) and download (d) in seconds and the throughput in MByte/second with 200 randomly generated files with a Pareto size distribution (a=1.05, k= 3800) and with a random hierarchy. The aggregate size of these files was 4.08 Mbyte.

From the results listed in Table 2 we may conclude that for small files our system is about two magnitudes slower than the currently used network file systems. The reasons for this lie in the distributed nature of our system. In the current implementation every operation is handled in separate transactions and after every transaction a vote is taken of the success or failure of the transaction. As we have seen, with small files or with administrative tasks like a directory tree manipulation, these overheads can take a longer time to finish than the whole file upload. We can correct this behaviour by batch processing the operations. When we upload a directory we can then assign a transaction for the whole process instead of managing every single operation as a transaction. To test the framework as a video archive, we had to measure with the help of different file size distributions. Video files are in most cases larger than normal files, so we used the value of 3,800,000 for k. With this value we generated 75 files with an aggregated size of 1.03 GBytes and the directory hierarchy was randomly generated. The test bench was the same as in the previous measurement. The results we got for file upload and file download are shown in rows (e) and (f) of Table 2. We can see that with larger files our solution provides a delay and throughput comparable to traditional network file systems. With batch processing this result can be further improved. With a stable environment we can achieve a higher throughput than tradition file systems by sending the error correcting data fragments only when they are needed. The data storage efficiency was measured as the ratio of the size of stored files and each file data size stored. A record size in our database was about 35 bytes, which is not comparable to the stored data quantity. We may conclude that the data storage efficiency really only depends on the error correcting rate applied.

Thesis 13 *The Paxos implementation with the optimized handling of non-requested messages is stable in a laboratory environment with a significant churn rate.*

Conclusions: We presented a solution for a cheap, reliable, high performance LAN-based distributed storage. The basic idea is not new, but we could not find a system which is optimized for such circumstances. The measurement results demonstrate the viability of this solution even with current desktop computing capabilities. We think that in the near future, with increasing processor capacity, similar solutions will be widely adopted.

Own contribution The key results of this contribution except the churn estimation in the laboratory environment are the results of the work of the author and they were published in research papers [1] and [3].

Conclusions

The chief goal of this work was to highlight the importance of infrastructure awareness in the area of application development. The dissertation is based on two parts, and 5 thesis groups containing 13 theses.

In the first part we showed that while the network is moving in the direction of the context-based treatment of network flows, it is not well known that the number of flows provides serious scalability issues. We demonstrated that even with the currently available simple stateful services and high-end HW architectures there is a serious scalability weakness. In parallel with this we said that the number of flows produced by some P2P applications provide a significant fraction of the total number of parallel flows produced by end-users. Despite this fact, in the today's ISP friendly P2P systems concentrate only on the volume of the traffic and not on the composition of the traffic. These results were presented as two theses:

- Thesis 1: The performance of stateful services in the distribution and core layers depends heavily on the number of unicast or multicast flows.
- Thesis 2: The ISP friendly P2P system must take into account the number of flows generated by the overlay too.

The results shown in this thesis group are all the results of the author. The results related to the multicast traffic of this contribution point were published in research paper [2].

In the second part we provided four examples of infrastructure aware applications in four separate studies. The detectability of the P2P botnets is a serious question as they are becoming even more sophisticated. We showed that with a low-degree DHT-based approach one can construct botnets which are not detectable from a single point of the Internet. The results of this part are presented in three theses:

- Thesis 3: It is possible to build P2P botnet which could not be detected with the help of the TDG method from a single point (AS).
- Thesis 4: With the help of localization and link clustering the P2P botnet can hide from a single point TDG-based monitoring.
- Thesis 5: Localization has a minor effect on the hiding capability of the P2P botnet, while link clusterization has significant effects on the visibility of the P2P botnet.

The results of this contribution point were published in research paper [6]. The results connected to the Thesis 5 are the results of the author. The rest are the results of shared work.

Next, we studied the issue of low-degree DHT construction and we showed that it is possible to build scalable, robust low degree DHTs with the help of link grouping and gossip-based information exchange. The results of this part are presented in three theses:

- Thesis 6:The Symphony topology is scalable; that is, $\lim_{h\to\infty} p(h,q) > 0$, if (i) all the links have $O(\log N)$ backup links, or if (ii) all the links have $O(\log\log N)$ backup links and $q \leq e^{-2} \approx 0.135$.
- Thesis 7: It is possible to build a low degree DHT that is robust even in the case of large churn.

- Thesis 8: The T-Man based DHT exetended with the four methods (Lookahead, Degree balancing, Stratification and Short-link avoidance) is stable in the case of significant churn (Weibull k=0.5)

The results of this contribution point were published in research papers [5]. The results connected to Thesis 8 are the results of the author. The rest are the results of shared work.

SIP compression is not related to the P2P overlay and the number of flows directly, but it is related to the 3G links and the capabilities of the handheld devices. We showed that the classical compression algorithms are not suitable for the SIP compression task. We modified several well-known algorithms and found that some of these algorithms achieve the best compression ratios, while others are better from the point of view decoder delay. The results of this part are given in three theses:

- Thesis 9:The classical compression algorithms without modification are not applicable for SIP compression.
- Thesis 10:The modified Deflate (run length encoding + context modeling) algorithm is the best SIP when the call setup delay is important.
- Thesis 11:The modified LZ77 algorithm is the fastest on the decompressor side. Context modelling (i.e. prefix-free encoders) is the fastest on the compressor side.

The results shown in this thesis group are all the results of the author and were published in research paper [4].

Another example of infrastructure awareness is the distributed storage system optimized for local communication and churn measured in the laboratories. Here we applied the classical Paxos algorithm with simplifications in order to achieve consistency even in the case of a constantly changing set of nodes. We showed that with Reed Solomon error correction and the modified Paxos algorithm one can build a reliable high performance local storage. The results of this part are given in two theses:

- Thesis 12:The Reed Solomon encoding-based error correction method is feassible soulution for file encoding with today's CPU power and the churn level we measured in the laboratory.
- Thesis 13: The Paxos implementation with the optimized handling of non-requested messages is stable in a laboratory environment with the measured churn rate.

The results shown in this thesis group except the churn estimation are all the results of the author and were published in research paper in research papers [1] and [3].

Here in thesis groups we summarize the publications:

	Contribution - short title	Theses	Publications
I/1	Flows vs. stateful services	Thesis 1, Thesis 2	[2]
II/1	Hiding botnets	Thesis 3, Thesis 4, Thesis 5	[6]
II/2	Low degree DHT with large churn	Thesis 6, Thesis 7, Thesis 7	[5]
II/3	SIP compression	Thesis 9, Thesis 10, Thesis 11	[4]
II/4	Scalable storage	Thesis 12, Thesis 13	[1] [3]

Table 3: Thesis contributions and supporting publications

Acknowledgements

The writing of this thesis took me a lot of time and effort. This was not only my time, but the time of my family. First, I would like to thank to my wife Andrea and my children Vilmos, Máté and András for their patience and support. I would also like to thank my father, mother and brother for all the support I have received from them. My father-in-law and mother-in-law helped our familiy a lot. Also without the long brainstorming meetings I conducted with my supervisor Márk Jelasity and his keen insight into things, this work could not have been done. I also received encouragement from the leader of our department, Tibor Gyimóthy, who gave me useful suggestions and enough free time to be able to concentrate on this study. Next, I would like to express my gratitude to Márta Fidrich, with whom I discussed many aspects of my research. The former and current members of the Wireless Laboratory who also helped me a lot include: György Roszik, Péter Bagrij, Gábor Sey, Zoltán Sógor, József Dombi Dániel, Miklós Kasza, Vilmos Szűcs, Róbert Béládi, Ádám Végh and Gábor Molnár. I would like to thank them too.

My thanks also go to David P. Curley who zealously reviewed this work from a lingustic point of view.

I believe that things in life do not happen by chance, but are the result of the will of a higher entity. I am happy that He helped me set and achieve my goals.

Vilmos Bilicki, April 2010.

References

[1] Vilmos Bilicki. Lanstore: a highly distributed reliable file storage system. In *.NET Technologies2005 conference proceedings(.NET'05)*, volume 3, pages 47–57, Plzen, Czeck Republic, 2005. University of West Bohemia. http://wscg.zcu.cz/ROTOR/Journal/Archive/2005$_{V}ol_{3}.pdfpage = 59$.

[2] Vilmos Bilicki. Testing and verifying an ipv6 based multicast network. In *ICCGI '06: Proceedings of the International Multi-Conference on Computing in the Global Information Technology*, pages 3–13, Washington, DC, USA, 2006. IEEE Computer Society.

[3] Vilmos Bilicki and Jożsef Dombi. Building a general framework for the consistency management of distributed applications. In *.NET Technologies 2006 conference proceedings(.NET'06)*, volume 3, pages 55–63, Plzen, Czeck Republic, 2006. University of West Bohemia. http://dotnet.zcu.cz/NET$_{2}006/Papers_{2}006/!Proceedings_{F}ull_{P}apers_{2}006.pdf$.

[4] Manta Fidrich, Vilmos Bilicki, Zoltan Sogor, and Gabor Sey. Sip compression. *Periodica Polytechnica, Electrical Engineering.*, 47(1-2):37–56, 2003. http://www.inf.u-szeged.hu/ bilickiv/research/CSCS2002.ps.

[5] Márk Jelasity and Vilmos Bilicki. Scalable p2p overlays of very small constant degree: An emerging security threat. In *SSS '09: Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 399–412, Berlin, Heidelberg, 2009. Springer-Verlag.

[6] Márk Jelasity and Vilmos Bilicki. Towards automated detection of peer-to-peer botnets: On the limits of local approaches. In *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09)*, pages 1–8. USENIX, 2009. http://www.usenix.org/events/leet09/tech/.

Coauthor's declaration

I hereby certify that I am familiar with the thesis of the applicant *Mr. Vilmos Bilicki* entitled *Infrastructure aware applications*. Regarding our joint results referred to in this thesis and published in

- M. Fidrich, V. Bilicki, Z. Sógor, G. Sey.: SIP compression, Periodica Polytechnica Electrical Engineering, 2003 47/1-2

The applicant's contribution was prominent in obtaining the results in article.

Szeged, 12 April 2010

Fidrich Márta

Marta Fidrich

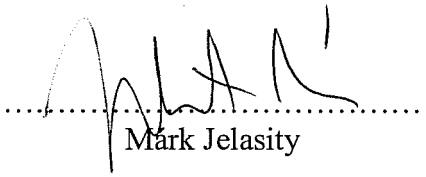
Coauthor's declaration

I hereby certify that I am familiar with the thesis of the applicant *Mr. Vilmos Bilicki* entitled *Infrastructure aware applications*. Regarding our joint results referred to in this thesis and published in:

- Márk Jelasity and Vilmos Bilicki. Scalable p2p overlays of very small constant degree: An emerging security threat. In SSS '09: Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems, pages 399–412, Berlin, Heidelberg, 2009. Springer-Verlag.
- Márk Jelasity and Vilmos Bilicki. Towards automated detection of peer-to-peer botnets: On the limits of local approaches. In 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09), pages 1–8. USENIX, 2009. http://www.usenix.org/events/leet09/tech/.

The results described in theses 5 and 8 are the results of the applicant. The rest of the results connected to the articles (Thesis 3,4,6,7) are the results of the collaborative efforts.

Szeged, 8 April 2010



Márk Jelasity

Coauthor's declaration

I hereby certify that I am familiar with the thesis of the applicant *Mr. Vilmos Bilicki* entitled *Infrastructure aware applications*. Regarding our joint results referred to in this thesis and published in

- • Vilmos Bilicki and József Dombi. Building a general framework for the consistency management of distributed applications. In .NET Technologies 2006 conference proceedings(.NET'06), volume 3, pages 55–63, Plzen, Czeck Republic, 2006. University of West Bohemia. http://dotnet.zcu.cz/NET_2006/Papers_2006/!Proceedings_Full_Papers_2006.pdf.

The results describing the stability and churn in a laboratory are based on my contributions. The rest of the results are based on the applicant's contribution.

Szeged, 8 April 2010

Dombi József Dániel

Coauthor's declaration

I hereby certify that I am familiar with the thesis of the applicant *Mr. Vilmos Bilicki* entitled *Infrastructure aware applications*. Regarding our joint results referred to in this thesis and published in

– • M. Fidrich, V. Bilicki, Z. Sógor, G. Sey.: SIP compression, Periodica Polytechnica Electrical Engineering, 2003 47/1-2

The applicant's contribution was prominent in obtaining the results in article.

Szeged, 8 April 2010

.....

Zoltán Sógor

Coauthor's declaration

I hereby certify that I am familiar with the thesis of the applicant *Mr. Vilmos Bilicki* entitled *Infrastructure aware applications*. Regarding our joint results referred to in this thesis and published in

- M. Fidrich, V. Bilicki, Z. Sógor, G. Sey.: SIP compression, Periodica Polytechnica Electrical Engineering, 2003 47/1-2

The applicant's contribution was prominent in obtaining the results in article.

Budapest, 8 April 2010

.....

Gabor Sey