# Automatic Transport Selection and Resource Allocation for Resilient Communication in Decentralised Networks

Matthias Wachs Fabian Oehlmann Christian Grothoff
Free Secure Network Systems Group
Technische Universität München
Email: {wachs,oehlmann,grothoff}@in.tum.de

***Abstract*—Making communication more resilient is a main focus for modern decentralised networks. A current development to increase connectivity between participants and to be resilient against service degradation attempts is to support different communication protocols, and to switch between these protocols in case degradation or censorship are detected. Supporting multiple protocols with different properties and having to share resources for communication with multiple partners creates new challenges with respect to protocol selection and resource allocation to optimally satisfy the applications' requirements for communication.**

**This paper presents a novel approach for automatic transport selection and resource allocation with a focus on decentralised networks. Our goal is to evaluate the communication mechanisms available for each communication partner and then allocate resources in line with the requirements of the applications.**

**We begin by detailing the overall requirements for an algorithm for transport selection and resource allocation, and then compare three different solutions using (1) a heuristic, (2) linear optimisation, and (3) machine learning. To show the suitability and the specific benefits of each approach, we evaluate their performance with respect to usability, scalability and quality of the solution found in relation to application requirements.**

## I. Introduction

Reliable connectivity and reasonable goodput with low latency are important for the success of peer-to-peer (P2P) networks. However, the contemporary Internet rarely provides unrestricted communication: various parties try to restrict or shape traffic for political, economic or technical reasons [2].

Modern P2P networks focus on antagonising the restrictions built into the Internet infrastructure by obfuscating information flows and sending traffic via routes or mechanisms that are not easily restricted. Leading P2P designs have started to support multiple "pluggable" transport mechanisms, resulting in architectures that can leave a compromised or degraded medium of communication and switch to a different communication mechanism. Being able to choose between different communication mechanisms creates new challenges as peers typically communicate with multiple other peers at the same time, and allocating resources for one peer may impact the quality of the connection to the others. Thus, transport selection for pluggable transports in P2P networks must consider not only performance and availability of the respective transport mechanisms, but also resource constraints and application-specific preferences.

This paper presents solutions to the challenge of transport selection and resource allocation with a focus on the specific requirements of censorship-resistant decentralised networks. For security reasons, we require that peers do not make their resource allocation decisions based on unreliable information from other peers; to minimise information leakage, peers also do not exchange information about their resource limitations, neighbour set or even the results of the resource allocation process.

We present a formalisation of the problem and three different approaches to solve the problem and evaluate these approaches with respect to usability of the approach and quality of the resulting solutions. We compare three solutions: a greedy heuristic, an algorithm based on linear constraint optimisation problem, and a method using reinforcement learning where agents learn to allocate resources while maximising social welfare.

## II. The transport selection and resource allocation problem

Existing P2P networks like I2P [6], GNUnet [7], SpovNet [3] and Tor [1] enable the use of multiple transport mechanisms, but so far they use rather simplistic processes like heuristics to decide which mechanism to use. This is problematic, as this decision can clearly have a significant impact on the quality of communication an application can provide.

In a P2P network, each peer communicates with a set of communication partners. The network defines a set of *transports* (e.g. TCP and UDP), peers can use to communicate with each other, but not every peer must support every transport. Transports provide *addresses* indicating how to connect to a peer. When a peer supports multiple transports, multiple addresses may be available to connect to this peer and even a single transport can provide multiple addresses (e.g. IPv4 and IPv6). Addresses can be located in different *network scopes* (e.g. LAN and WAN) with different *resource restrictions*. Resources available on a system have to be distributed among communication partners. To not cannibalise other applications running on the same system, *quotas* may restrict the amount of resources available to the application. Different addresses may have different *properties* (e.g. delay and loss rate) based on

978-1-4799-6201-3/14/$31.00 

the transport and the network scope of the address. Therefore metrics are required to compare and select the "best" address. Applications using the transport underlay to communicate with other peers have to specify *preferences* (e.g. low latency and goodput) to express which properties are important for good performance and which peers are important to communicate with. Applications should also be enabled to provide positive or negative *feedback* to indicate how satisfied they are with regard to the current performance.

A P2P network using a multi-transport approach should automatically select the "best" transport available for each communication partner, and continuously evaluate the performance of the chosen transport. The transport selection operation performed by each participant in the network has to (1) decide on a set of peers with which it will maintain connectivity, (2) choose a single address for each peer from the set of available addresses, and (3) allocate a certain amount of the resources while satisfying the resource constraints. Due to peers joining and leaving the network, inputs to the problem may change frequently. We expect peers to join and leave the network at frequencies in the range of seconds, whereas address properties may change within milliseconds. Whenever inputs change, the transport selection process may want to adjust the solution. The output of the address selection and resource allocation process is the set of addresses to use to communicate with other peers, containing a single address for each remote peer together with the resources assigned to each address.

### A. Objectives for transport selection

In addition to considering application preferences and address properties, the selection algorithm should consider additional high-level objectives that transcend the preferences of an individual application. To provide a useful communication between participants, a minimum amount of resources is required for each connection. Thus, if an address is selected at all, at least a certain minimum amount of resources has to be assigned (**Usability**). Communicating with a larger number of participants increases the resilience of the P2P network. Therefore, the result should distribute resources over a range of peers instead of preferring communication with a tiny number of peers (**Diversity**). Resources should be allocated to peers according to their relative importance in the communication as expressed by the applications' preferences. So if a peer is valuable, it should get more resources assigned than a peer that does not contribute (**Relativity**). Available resources should be fully allocated to allow participants to achieve maximum utilisation when communicating (**Utilisation**). Transports with high overhead should be avoided to minimise useless resource consumption and maximise application performance (**Austerity**). Allocations should exhibit some stability to minimise transport initialisation overheads and provide predictable performance to applications (**Stability**). We assume that the P2P framework assigns specific weights for the sub-objectives to evaluate the overall quality of the peer selection, address selection and resource allocation.

### B. Scope

For the approach presented in this paper, we assume that only a single address per (selected) peer is to be determined, based on the idea that each connection creates inherent overheads (handshake, socket, buffers) and that establishing multiple parallel connections is thus inherently wasteful. This restriction is not a fundamental limitation since our solutions can be easily extended to permit this behaviour. We further assume that the process of deciding which peers to communicate with is partially covered by the application, and the transport selection algorithm then decides on the address to use; the address must be initially provisioned with some minimum amount of resources, but in later rounds the transport selection algorithm may terminate the connection with the peer.

## III. Design

A component for automatic transport selection and resource allocation has to interact with both the underlying transport infrastructure and the applications using this transport infrastructure to communicate. We will now sketch three different solutions to find an "optimal" set of addresses and resource allocation with respect to the inputs provided by the transport underlay and higher layer applications and the user defined resource constraints. Each solver satisfies the requirements of this problem and has distinct advantages and disadvantages. A detailed analysis about the design and implementation of the solvers can be found in our technical report [7].

### A. The heuristic solver

The first approach is a fast heuristic based on the idea to distribute resources roughly proportional to the importance a communication partner has for the high layer applications. The heuristic solver views the different network scopes as *buckets* of bandwidth and distributes the bandwidth in each bucket to peers in *relation* to how important this peer is for the applications.

The heuristic selects the "best" address available for a peer by comparing the *performance* properties; the focus here is on latency, but the stability of the choice is also considered. To ensure *usable* connections, the heuristic activates an address only if a minimum amount of bandwidth for all active addresses in this scope can be provided. Resources in the respective scope are distributed among the selected addresses by first assigning every address a minimum amount of bandwidth to ensure *diversity* of connections and then distributing the remaining bandwidth among all addresses *relative* to the preferences the higher layer applications have specified (with respect to bandwidth) for peers. If not enough resources can be provided to maintain a connection, the heuristic may tell the underlay to disconnect from a peer.

### B. The linear optimisation solver

To combine address selection and resources allocation in an integrated approach, the linear optimisation solver views the problem as a mathematical optimisation problem. In linear optimisation, problems are defined using a (linear) objective

function to be maximised under a set of objectives, formulated as linear (in)equations [5]. The address selection and resource allocation problem becomes a mixed integer linear problem (MILP) because a binary output is required to indicate if an address was selected (1) or not (0).

To formulate the problem as a MILP, one has to carefully formulate a set of linear constraints which ensure that the solution satisfies resource constraints. In our formulation, we distinguish between *feasibility constraints* ensuring that the solution is valid in the given domain and *optimality constraints* driving the solution towards the system objectives. As feasibility constraints we define constraints enforcing *diversity* (maintain a minimum number of connections), *usability* (minimum resources for active addresses), *scope* (one address per peer), *finite solution* (prevent unbounded solutions, quotas must be finite). To obtain a solution optimal with respect to the objectives defined in II-A, we add optimality constraints optimising for *utilisation* (use resources available in network scopes), *austerity* (prefer transports with smaller overhead), *diversity* (establish connections with a larger number of peers), and *relativity* (distribute according to application preferences). The solver's running time can be reduced by exploiting the fact that the Simplex algorithm used to solve the problem can re-use an existing solution if only the coefficients in the problem changed. As an output, the optimisation algorithm provides for each address a binary variable that models the selection of the address, and another with the amount of inbound and outbound bandwidth that was assigned.

### C. The machine learning solver

The machine learning solver uses reinforcement learning [4] to learn *good* address selection and bandwidth allocation strategies. The reinforcement learning (RIL) solver uses an autonomous *agent* per requested peer performing *actions* to learn or exploit the allocation strategy. To perform actions, the agent can increase and decrease bandwidth assigned to an address, switch to a different address or decide to do nothing. Based on the impact actions have with respect to the objectives defined in II-A and feedback received from applications, the agent receives a *reward* indicating if previously taken actions have improved the allocation or not. Based on this reward, the agent updates his allocation strategy. To achieve a global optimal solution for all peers, the solver uses a social welfare algorithm to achieve good allocations for all peers. Contrary to previous solvers, this approach also supports *over-allocation* of the available resources. However, *allocated* resources might then ultimately not be *used* as applications may not generate enough traffic to fully utilise the allocations. Thus, over-allocation can be useful even if *over-utilisation* creates significant penalties.

## IV. Implementation

To ensure the applicability of our proposed design for transport selection and resource allocation in practice, and to validate the design, scalability and performance of all proposed solution approaches, we implemented and experimentally compared the three solvers. The source code and the evaluation tools are available on our website[1].

We implemented the algorithms to execute independently from the transport underlay and the higher-layer applications. Specifically, the solvers run as separate (operating system) process and both the transport underlay as well as the applications can interact with the solver component in a non-blocking way. This ensures that the rest of the P2P framework can operate independently from our component without having to worry about blocking operations or shared resources. This also facilitates the integration of the implementations into different P2P applications. The MILP solver relies on the GNU Linear Programming Kit (GLPK)[2], a free software package intended for solving linear optimisation problems.

## V. Evaluation

To evaluate the proposed solvers, we used the production code in a simulation environment to evaluate our proposed design under controlled circumstances.

To evaluate the scalability of the solvers, we measured the running time and memory consumption of each respective solver by incrementally adding peers and addresses to the problem. Each time a new peer and addresses are added, our benchmarking tool requests the solver to find an allocation. In addition to changing the problem size by adding new peers and addresses, the tool requests *incremental* solutions after updating properties and preferences for peers and addresses already existing in the problem. Incremental solutions are particularly interesting for the linear optimisation solver since the solver can re-use an existing solution of the problem whenever the problem size does not change. The results of the solver scalability evaluation are presented in Section V-A.

To evaluate the quality of the solutions provided by the solvers, we analysed the quality of the solutions provided by the three different solver approaches. We designed multiple scenarios that peers might experience including fluctuating address properties and application preferences and evaluated the quality of the address selection and resource allocations produced by the solvers. To ensure that the heuristics were not somehow tuned specifically to the evaluation scenarios, we did not perform performance tuning of the solvers on the scenarios used for the evaluation. The results of the solver quality evaluation are presented in Section V-B.

We used a desktop PC with an Intel Xeon W3520 quad core CPU with at 2.67Ghz and 24 GiB of memory running Ubuntu 14.04-AMD64 for all of the measurements. GLPK version 4.54 was used to solve the linear optimisation problems.

### A. Solver scalability evaluation

The scalability of the different solvers largely depends on the number of peers, addresses and network scopes. Asking the MILP solver to produce optimal solutions for larger problems can be problematic — even for problems with just a few

[1] https://gnunet.org/svn/gnunet/src/ats/
[2] https://www.gnu.org/software/glpk/

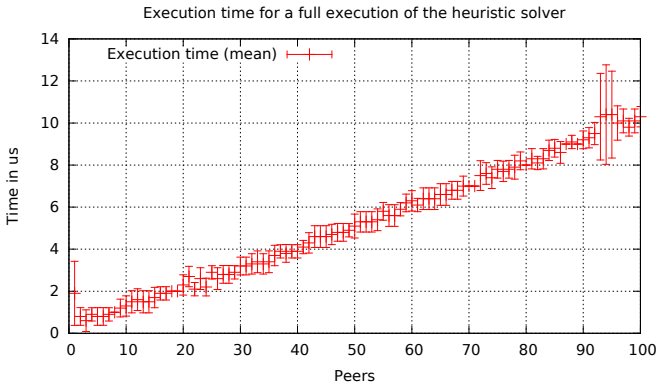


Fig. 1. Execution time for the heuristic solver in relation to the number of peers.

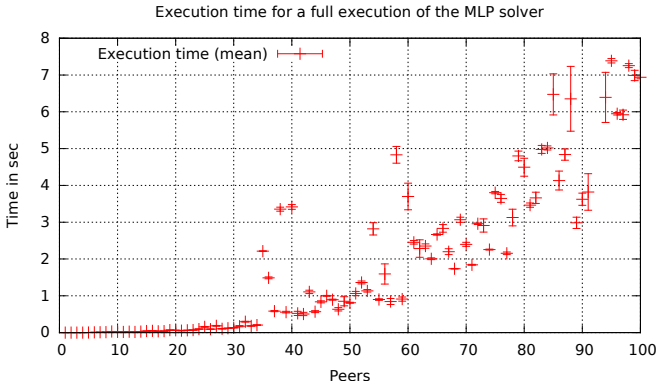


Fig. 2. Execution time for the MILP solver to solve the problem from scratch in relation to the number of peers.

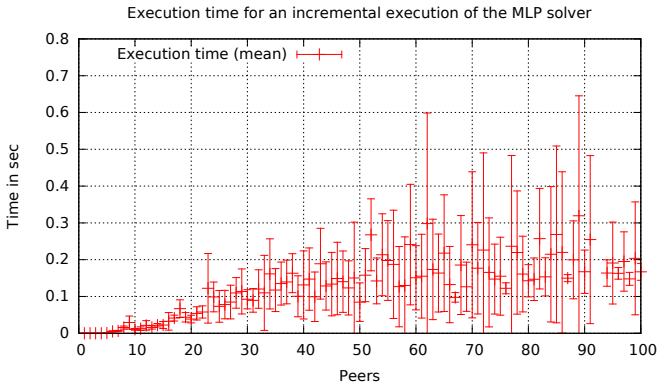


Fig. 3. Execution time for the MILP solver to incrementally solve problem in relation to the number of peers.

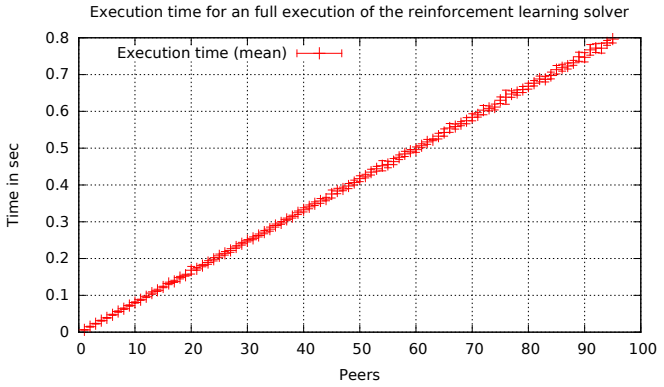


Fig. 4. Execution time for the reinforcement learning solver in relation to the number of peers.

dozen peers and addresses the solver can take gigabytes of memory and hours of running time to prove the optimality of the solution. We thus allowed the MILP solver to terminate with an approximate solution of guaranteed quality (within 2.5% of the optimal solution). We also used a 10 s timeout; however, that timeout was then never reached in practice. With these restrictions, memory consumption for all approaches was relatively small (at the order of a few megabytes).

Figures 1, 2, 3 and 4 show the execution time in relation to the number of peers in the problem. Each peer always provides ten different addresses, equally distributed over five different network scopes. Properties of new addresses are initialised with random values. To evaluate the performance to solve an updated problem, the properties of 10% of all addresses currently in the problem are updated.

### B. Solver quality evaluation

To evaluate quality of the solvers we used three different scenarios, modelled to represent the behaviour of a file sharing application, a telephony application, and the case where both file-sharing and telephony execute together. To evaluate the quality of the solutions provided by the different solvers the simulator collects information about the selected addresses and allocated bandwidth as well as the current properties and preferences as specified by the scenario generator. These inputs are then used to evaluate the quality of the allocation using a goal function similar to the objective function of the MILP solver. This goal function includes the utility of the current allocation, if bandwidth is assigned according to preferences specified by the applications (relativity), if connections to a larger number of peers is established (diversity) and if the addresses were selected according to the properties and preferences for these properties. In addition, the goal function includes a penalty if resource constraints are violated to penalise the reinforcement learning (RIL) solver for over-allocation.

To evaluate quality of the solvers we used three different scenarios, with two different durations (to observe the effect of learning). We run the simulations for 10 seconds in the *short* variant and 20 seconds in the *long* variant. We use two network scopes: scope $n_0$ with a large amount of bandwidth available, and scope $n_1$ providing only half the bandwidth. We have two neighbours $p_0$ and $p_1$, each with one address in each of the network scopes.

In the *throughout* scenario, we emulate an application trying to achieve a high throughput to a one peer, and not caring about other peers. The application wants to maximise throughput to peer $p_0$, and has no concern for latency at all. We generate delay values for the addresses in $n_1$ to provide better latency properties than for the addresses in scope $n_0$, which provides more bandwidth but worse delay properties. After this setup we than begin to issue preferences in regular intervals of 500 ms with respect to bandwidth for $p_0$ with linearly increasing values and for $p_1$ with a constant low value to indicate our disinterest in this peer.

In the *latency* scenarios, we emulate an application requiring low latency values to communicate with both $p_0$ and $p_1$. We generate delay values for the addresses of neighbour $p_0$ and $p_1$ located in $n_0$ with random values between 20 and 25 ms and better delay values for addresses in scope $n_1$ with delay values between 10 and 15 ms for $p_0$ and 1 and 30 ms for $p_1$. We then begin to issue preferences for both peers with respect to latency, linear increasing values for $p_0$ and sinusoidal for $p_1$.

In the *mixed* scenario, we simulate two applications issuing conflicting preferences. For the addresses located in $n_0$, we create delay properties for both addresses with values between 20 and 25 ms, whereas for addresses in scope $n_1$, we create delay values between 10 and 15 ms for $p_0$ and values linear increasing between 1 and 30 ms for $p_1$. The values for $p_1$'s address in $n_1$ were particularly chosen to make the solver switch to $p_1$'s address in scope $n_0$. The application begins to generate preferences for $p_0$ to prefer maximise throughput and for peer $p_1$ to maximise delay.

Table I gives the results for the goal function for the different scenarios. The values are normalised in relation to the quality of the solutions produced by the MILP solver. The results show that learning is effective as the RIL solver's solution improves for longer runs, and it outperforms the heuristic for most scenarios.

## VI. Discussion

Our heuristic can compute the address selection and the resource allocation very fast, which is beneficial given frequent changes in the problem due to peers joining and leaving and updated address properties and application preferences; however, the greedy nature limits the quality of the solution and the heuristic does not benefit from the requirement to solve the problem repeatedly.

Treating the address selection and resource allocation process as an optimisation problem and using optimisation techniques has the advantage that the solution found is always an optimal solution and objectives can be weighted according to the application's needs by adapting coefficients within the objective function. However, having to formulate the problem as MILP requires a careful design to formulate all constraints and the object function as linear equations. Requiring the output to contain binary variables makes solving the problem significantly more expensive since solving a mixed integer problem is NP-hard, while for linear programming polynomial time algorithms exist. MILP solver running time can be reduced by exploiting the fact that the Simplex algorithm can reuse an existing solution if only the coefficients in the problem changed but not the size of the problem itself (peers joining and leaving, addresses being added or removed). Furthermore, Simplex typically produces feasible but suboptimal solutions quickly; thus it is important to bound CPU time with timeouts or reduce CPU consumption by allowing the MILP solver to terminate with an approximate solution of guaranteed quality.

TABLE I
Normalised quality of the solutions produced by the solvers.

| Scenario | Heuristic | MILP | RIL |
|---|---|---|---|
| throughput short | 0.905 | 1.00 | 0.513 |
| throughput long | 0.949 | 1.00 | 0.690 |
| latency short | 0.510 | 1.00 | 0.692 |
| latency long | 0.506 | 1.00 | 0.803 |
| mixed short | 0.547 | 1.00 | 0.367 |
| mixed long | 0.552 | 1.00 | 0.969 |

An a-priori definition of the objectives for address selection and resource allocation is a difficult task, especially as the requirements of applications may change over time. Furthermore, some of the constraints that were formulated are rarely "hard" constraints — an application that sometimes slightly overshoots bandwidth targets might be more desirable than an application that sticks to constraints and fails to deliver performance when it is critical. These challenges can be addressed using reinforcement learning which may predict future developments. In particular, a learning algorithm has the chance to adapt to the current observed utilisation behaviour of the application and can adjust its allocations accordingly. This can then reduce the amount of allocated but unused resources. However, reinforcement learning takes time for the adaptation, and thus naturally performs worse if evaluated under the same goal function as the MILP.

## VII. Conclusion

Based on an analysis of the challenges arising from the support of multiple transports under resource constraints, this paper presented three methods for transport selection and resource allocation for decentralised P2P networks supporting multiple transport protocols. We demonstrate that both reinforcement learning and constraint solving methods can deliver significant performance benefits over ad hoc heuristics.

## Acknowledgments

This work was funded by Deutsche Forschungsgemeinschaft (DFG) under ENP GR 3688/1-1.

## References

[1] Appelbaum, J., and Mathewson, N. Pluggable Transport Specification. https://gitweb.torproject.org/torspec.git/blob/HEAD:/pt-spec.txt, January 2014. accessed: 2014-04-25.

[2] BEREC. A view of traffic management and other practices resulting in restrictions to the open Internet in Europe. http://ec.europa.eu/digital-agenda/sites/digital-agenda/files/Traffic%20Management%20Investigation%20BEREC_2.pdf, January 2011. accessed: 2014-04-25.

[3] Bless, R., Hübsch, C., Mies, S., and Waldhorst, O. The Underlay Abstraction in the Spontaneous Virtual Networks (SpoVNet) Architecture. In *Proc. 4th EuroNGI Conf. on Next Generation Internet Networks (NGI 2008)* (Apr. 2008), pp. 115–122.

[4] Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research (JAIR) 4* (1996), 237–285.

[5] Karp, R. M. Reducibility among combinatorial problems. *Complexity of Computer Computations* (1972), 85–103.

[6] The I2P Project. Transport overview - transport selection. http://geti2p.net/en/docs/transport, December 2013. accessed: 2014-04-25.

[7] Wachs, M. *A Secure Communication Infrastructure for Decentralized Networking Applications*. PhD thesis, Technische Universität München, Garching bei München, under submission.