Path Searching with Transit Nodes in Fast Changing Telecommunications Networks

Robert Mertens, Joerg Stachowiak, Timo Steffens

Fraunhofer IAIS, Schloss Birlinghoven, 53754 Sankt Augustin

Contact: robert.mertens@iais.fraunhofer.de

Abstract: Transit nodes are a concept that has proven to speed up pathfinding in roadnetworks considerably. The main idea of transit nodes is to precompute paths for selected routes and use these as shortcuts during the search. While this method can be employed for road networks that remain static over a period of time, it can not be directly applied to complex fast changing networks such as those used for telecommunication since precomputed shortcuts can become invalid at any time. In this paper, an approach is presented that is based on transit nodes but extends the concept to accommodate for the special requirements of fast changing networks.

1 Introduction

For telecommunications products such as VDSL or ADSL2+, a customer's line needs to be connected to the IP-infrastructure of a telecommunications provider. Such a connection can be regarded as a path in the provider's telecommunications network. When customers inquire about the availability of a product for a line, a path search is conducted on the provider's network in to order check whether the product is available using the existing infrastructure. The result of this search indicates where connections have to be rewired and which components of the Second Mile (i. e. the Gigabit-Ethernet network) have to be reconfigured. Customers can conduct such an inquiry online or it can be started by a salesperson during a sales talk. In both scenarios, time is a critical factor. If customers have to wait too long for an answer, a sales talk might become difficult or the customer might be distracted when conducting an online inquiry. Especially in large telecommincations networks that contain a huge number of components, classical search algorithms [Dij59] do not face up to the challenge of finding a path within the required time limit. Informed search algorithms can not be used due to the lack of appropriate heuristics in the domain. This paper shows how the problem can be tackled by adopting the concept of transit nodes [BFM06] that was originally conceived for pathfinding in road networks and by adapting it for the special qualities of telecommunications networks. The remainder of the paper is organized as follows: The next section introduces the concept of transit nodes and describes how it can be employed in the telecommunications domain in general. Section 3 shows which properties of fast changing networks conflict with the use of transit nodes and how the resulting problems can be solved by a heuristic repair mechanism. The

paper concludes with a comparison of the algorithm against a classical search algorithm and brief discussion of experiences gathered with the approach presented in this paper.

2 Transit Nodes in Telecommunications Networks

The concept of transit nodes for accelerating pathfinding originates in the application domain of road networks. In the TRANSIT algorithm presented in [BFM06], a number of transit nodes are computed in a preprocessing step and stored in a database. These transit nodes (e.g. higway entraces/exits) allow for travelling long distances to other regions. The basic idea behind the algorithm is that once a network N_2 of few transit nodes that are well distributed over a network $N_2 \subset N_1$ exists, a pathfinding task can be diveded into three subasks t_1, t_2 and t_3. t_1 computes a path to the transit node tn_1 that is nearest to the starting point, t_2 computes a path from the target node to the transit node tn_2 that is nearest to the target node and t_3 computes a path from tn_1 to tn_2. In the approach presented in [BFM06] all required distances (start node to tn_1, tn_1 to tn_2, and tn_2 to the final target node) are precomputed and stored in a database.

While telecommunications networks have a number of properties in common with road networks, they also differ at some points. The most important difference is that, once a path has been taken (i. e. once a customer line is realized using the pins, twin wires, ethernet ports and so on), its elements can not be used for a subsequent search since the elements on that path are already in use. In analogy to the road network scenario this would mean that once A had travelled the highway from Munich to Hamburg, B could no longer use that highway. For the original concept of transit nodes, this means that once a transit node (or any other node for that matter) has been used, it is no longer available. As a consequence, a new transit node would have to be computed in order to replace the transit nodes used in a successful run of the algorithm for each successful run. In order to avoid this, the approach presented in this paper uses a two-level representation of the network (cf. [SS05] for hierarchical levels in road maps). The first level represents all parts of the network the path is actually constructed of (wires, pins, ports etc.). The second level represents the network's topology. Second level nodes stand for entities that aggregate first level entities (cable $\vdash$ wires, ethernet-linecard $\vdash$ ethernet-ports, distribution box $\vdash$ pins etc.). Thus when a route is taken through a second level entity, only the corresponding first level entities used for that route become unusable for later pathfinding. The second level entities in the topological network representation and all unused first level entities aggregated in them remain usable for later searches. Hence, the entities on the topological level are used as transit nodes. In this model, the transit nodes are no longer part of the network itself (as in the hierarchical model proposed in [SS05]), in other words N_2 is no longer a subnetwork of N_1. Algorithms to determine the network's topolgy like the one described in [AS98]) are not needed as the network's topology is already explicitly modeled in the database for maintanance purposes.

In order to use the nodes of N_2 as transit nodes, a mapping between nodes in N_1 and N_2 is established. This mapping connects all wires with the cable they are part of, all pins with the distribution box they are part of and so on. The cost for travelling from one transit

node to another one ist precomputed using paths in the first level network N_1. This cost and the path used for the computation are both stored in the database for each precomputed path. This way, a connection between transit nodes is stored with a number of paths that can be used to realize that connection instead of just one path as in the approach described in [BFM06]. Whether an alternative path is stored in the database depends on a threshold applied to its cost-function.

Due to changes in the network that are caused by construction work (e. g. to extend the network or by reconfigurations) and the fact that an entity of N_1 can be used for one customer line only, N_1 changes at a very fast rate. Consequently not every change in the network can be propagated to the representation of the respective transit nodes in realtime. Therefore the precomputed paths stored as connections between transit nodes in the database can not be guaranteed to be valid when retrieved. In order to cope with this problem, a special heuristics has been devised. This heuristics and the technique used to map N_2 entitites to N_1 entities is described in detail in the next section. The fact that a transit node is stored with a number of paths leads to a very high amount of transit paths. Given the size of a country wide telecommunications network in countries the size of Germany, storing this data would consume too much space in the database. In order to counter this effect, transit paths are not stored for every connection between entities in N_2. Instead only neigboring N_2-nodes are used for this purposes. The final transit routes used for realizing a connection are constructed on the fly as described in the next section.

3 Path Validation

As described in the last section, the mapping $f : N_2 \rightarrow N_1$ of connections between transit nodes (nodes in the topological network N_2) to actual paths in N_1 is not a bijective function since one transit node can be mapped to an arbitrary number of connections between nodes in N_1. Because nodes in N_1 become unusable once they are part of a customer line, there can even be constellations in which no connection between any two entities in N_1 can be found that are part of two entities in N_2 that can in principle be connected. While the information about distances between nodes in N_2 is updated regularly, it is not guaranteed that these updates happen in time. In order to compensate for the resulting effects, paths are validated on the fly for every search. Additionally, a fallback mechanism is executed in case no precomputed path can be validated. Since paths in N_2 can potentially be realized by a number of paths in N_1, every search also constructs a concrete path in N_1 if possible. The overall validation process is visualized in fig. 1. It consists of a number of subtasks that are described in the following paragraphs.

Topological ressource search uses lookups in the transit nodes table in the database. Since the table on which the lookup is executed can not be guaranteed to be totally up to date, it stores a number of alternative paths from a node in N_2 to another node in N_2 and can thus deliver a whole set of possible paths in N_1 for a query. As mentioned before, these transit paths do not cover all possible connections but only those between neigboring nodes in N_2. Therefore, this search searches for transit routes in N_2 for which known paths exist in N_1 using an uninformed search that terminates once a certain threshold for the cost function

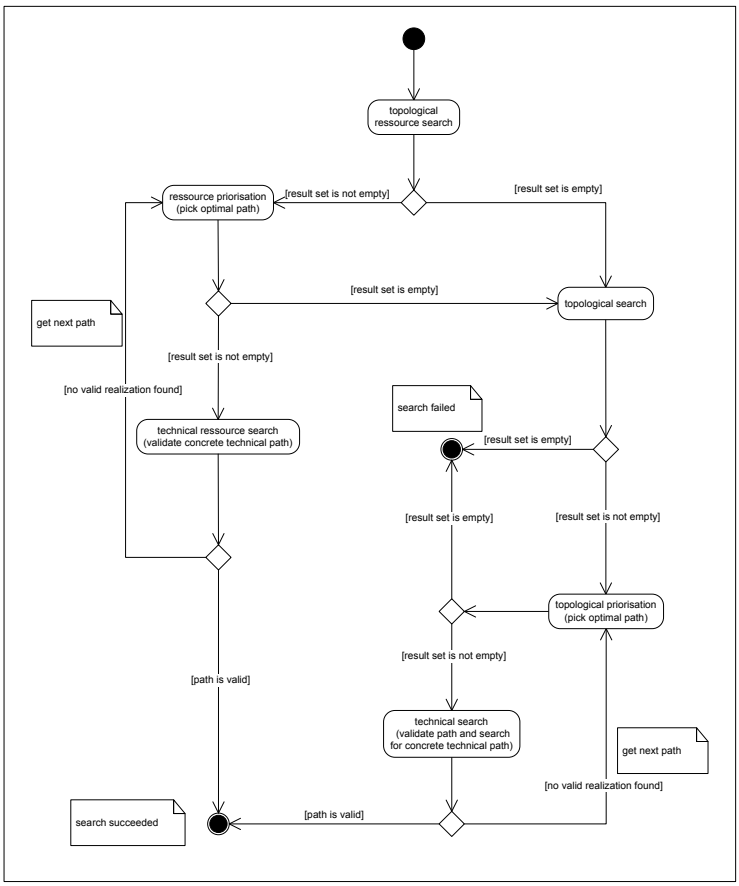

Figure 1: Control flow of the routing algorithm.

is exceeded. Note that not all of these paths are guaranteed to be still usable as the update process of the lookup table can not react to all changes in the network instantaneously. Hence they have to be validated in a subsequent step.

Topological search operates on the topological network N_2. It collects a set of paths in N_2 from $a \in N_2$ to $b \in N_2$. The difference between topological search and topological ressource search is that topological search does not rely on values precomputed in N_1. Instead, the cost function used for searching uses default values that can be determined by analysing data like cable length that is attached to the aggregate entities in N_2. Topological search is implemented as an uninformed search that computes all paths with a cost function value less than a predefined threshold.

Topological priorisation analyses a result set delivered by topological search. It decides which path in N_2 is preferred based on a number of rules that can differ from the cost function used in topological search (this is why all possible paths are retrieved in topolog-

ical search). An example of such a rule is that certain cable materials are preferred or that certain node types are not used.

Technical search validates a path on the technical level. I.e. it searches for unused wires (N_1 entity) in a cable (N_2 entity) or unused pins (N_1 entity) in a distribution box (N_2 entity) to realize a path consisiting of N_2 entities (as generated by topological priorisation). A path is valid if and only if unused N_1 entities can be found to realize the N_2-path passed by topological resssource search.

Technical ressource search is analogous to technical search in that it validates a path in N_1. It differs from technical ressource search in that it analyses precomputed paths in N_1 that are stored in the database.

Ressource priorisation works in a way similar to topological priorisation in that it analyses a result set delivered by a preceding search run. It decides which path in N_2 is preferred based on the same cost function as used in topological priorisation.

The optimal execution path of the algorithm depicted in fig. 1 is topological ressource search followed by ressource priorisation followed by a successful technical ressource search (roughly on the left hand side of fig. 1). The right hand path in fig. 1 is executed as a fallback strategy. Both, ressource priorisation and technical priorisation are used to cope with problems constituded by the fast paced changes of the network.

4 Evaluation

In order to measure the performance of the algorithm presented in this paper, the algorithm has been compared to two versions of a standard uniformed search algorithm on a test set of ca. 1300 search queries. The results of this comparison are depicted in fig. 2. The lowest curve in the figure ('Ressource Router') plots the computing time required by the algorithm. The other two curves ('Raw Router' and 'Version 2.210') plot the computing times of two different versions of a standard uninformed search algorithm for the same queries. In 'Version 2.210' database queries have been modified to speed up the algorithm. These modified database queries are not used in 'Ressource Router'. 'Ressource Router' is still significantly faster than the other two algorithms. The curve showing the computing times of 'Ressource Router' can be regarded as consisting of three different segments (0-23%, 23-97% and 97-100%). The computing times in these segments can be interpreted as follows: In the first segment, transit paths can be validated for the complete customer line. In the second segment, parts of the customer line can be instantiated using transit paths and other parts require topological search. In the third segment, no paths can be instantiated. Therefore the algorithm has to conduct an exhaustive search which makes it as slow as the two comparison algorithms. These results show that the algorithm works considerably faster than a standard uninformed algorithm in case a route can be found. Given the requirements imposed by marketing considerations, the fact that a positive result is found significantly faster than with a standard algorithm, makes the approach a full success. Search times in negative cases are less relevant, because in these cases it does not matter whether the customer loses interest.

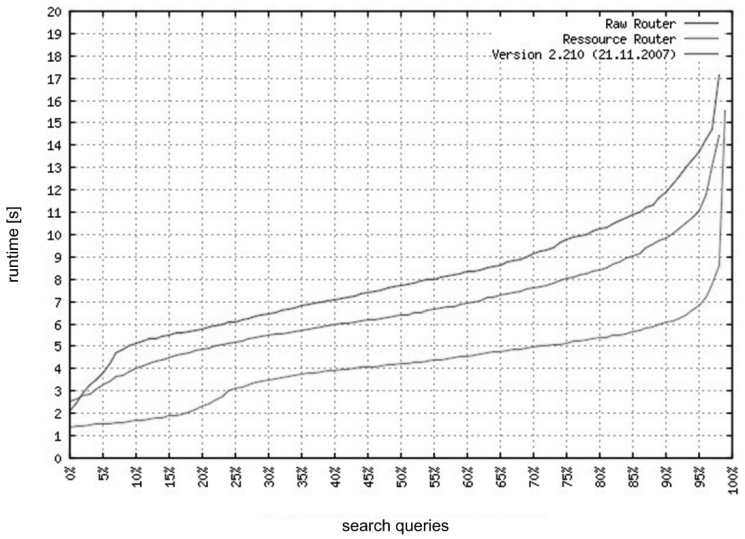

Figure 2: Comparison of the algorithm with two reference algorithms.

5 Conclusion and Further Work

This paper presented an approach that adapts the concept of transit nodes originally designed for computing paths in road networks successfully to telecommunications networks. In order to accommodate for the special properties of the application domains, both the concept of transit routes and a heuristics for on the fly path validation have been devised. The algortihm has shown to successfully meet the requirements imposed by the application domain. Future work will be conducted in the direction of evaluating whether direct updates for changes in N_1 can be propagated instantaneously to the transit paths stored in the database and for which kind of updates this is feasible.

References

[AS98] B. Awerbuch and Y. Shavitt. Topology Aggregation for Directed Graphs. In *Third IEEE Symposium on Computers and Communications (ISCC)*, pages 47–52. IEEE, 1998.

[BFM06] H. Bast, S. Funke, and D. Matijevic. TRANSIT: Ultrafast Shortest-Path Queries with Linear-Time Preprocessing. In C. Demetrescu, A. Goldberg, and D. Johnson, editors, *9th DIMACS Implementation Challenge — Shortest Path*. DIMACS, 2006.

[Dij59] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[SS05] P. Sanders and D. Schultes. Highway hierarchies hasten exact shortest path queries. In *13th European Symposium on Algorithms (ESA'05)*, pages 568–579, 2005.