REAL-TIME COMPUTATION OF POINT-TO-MULTIPOINT ROUTES IN OPTICAL NETWORKS FOR DIGITAL TELEVISION

Roman Messmer
ORF Austrian Broadcasting Corporation
Broadcast Production Systems
A-1136 Vienna, Austria
roman.meszmer@orf.at

Jörg Keller
FernUniversität in Hagen
Dept. of Mathematics and Computer Science
58084 Hagen, Germany
joerg.keller@fernuni-hagen.de

ABSTRACT
Multimedia networks in television studios are on the move from expensive special equipment to complete digitization into mainstream data networks. Differences are real-time requirements and large bandwidths of single connections. We present an algorithm to compute routes for a point-to-multipoint connection within an optoelectronic switched television studio network in real-time after a connection request. This is achieved by combining offline and online algorithms. We validate the algorithm with realistic and synthetic example networks and connection requests.

KEY WORDS
point-to-multipoint routing, digital television network, online algorithm

1 Introduction

Television studios have achieved high levels of digitization. But still, connecting the output of a microphone in a studio to several receivers in the same building during a live transmission still requires expensive equipment and massive cabling. The consequent move to using digital broadband data networks is hindered by several factors. First, while the bandwidth requirement of an audio signal is small, the bandwidth requirement of a high definition video signal can nearly consume the complete bandwidth of a link. Second, typical data transmission protocols are not suitable for the strict real-time requirements of audio and video connections, e.g. with respect to jitter.

A major Austrian television corporation has conducted a multi-year study in the course of which it developed the hardware for a digital optoelectronic switched network for a television studio building. All links are optical, switches and connection to signal producers or consumers are electronic. The network forms a connected, undirected graph, possibly with cycles. A major part in this undertaking was the algorithm to process requests for new connections, where a connection has one source and several receivers or sinks. The algorithm should be able to accept and process requests in real-time, and to provide for real-time reaction in the case of a link failure.

In the present paper, we will concentrate on the routing and leave out the fault-tolerance aspect. Thus, the problem to be solved is to compute a tree with the source as the root and the sinks as the leaves, using links only in such way that their capacity is not overbooked in view of already existing requests [1]. The computation and the distribution within the network is to be done in real time. Also, the utilization of the links shall be high in practical settings.

In principle, heuristics for steiner trees could be used, but as the network may have hundreds of nodes, the runtime would be prohibitive for real-time behavior. Our solution is a mix of an offline algorithm to solve an all-pair-shortest path problem, which is periodically run, and a greedy online algorithm that builds upon this pre-computed information and therefore is very fast. Although it does not guarantee a minimum spanning tree, all of our examples achieved optimal or near optimal solutions.

The remainder of this paper is organized as follows. Section 2 elaborates the situation of digital television studios and the routing problem to be solved. In Sect. 3, we present our combination of online and offline algorithms and prove their correctness. The experimental results are reported in Sect. 4. In Sect. 5,we discuss related work. Finally, Sect. 6 gives a conclusion and an outlook to future work.

2 Background Information

The demands of real-time video (jitter, bandwidth, reliability) often are hard to meet using data networks. For example, the expectations concerning the reliability of transmission lines are in the area of 99.9999%, and cannot be satisfied with standard IT equipment as Ethernet lines or the like [2]. Problems that crop up are network congestion, limited available bandwidth and poor response times. However, not only network-related problems but also problems generally inherent with IT, like short life span of equipment, online system upgrades, virus attacks and deficient software stability for mission-critical applications are to be addressed when planning to create a so-called “Broadcast IT system”. Traditionally the audio and video network system (AV system) of an HDTV station is a copper or fibre based point-to-point connection network. Traditional devices for distributing AV signals are digital AV routers, which consist mainly of signal distribution amplifiers, reclockers and switching hardware. Its main advantages are

its well-known end-to-end signal delay time as well as its simple configuration and installation.

The main disadvantages over networked media, i.e. connections over network lines and switches, are topological inflexibility and a huge expenditure of AV lines for connecting the different rooms via central distribution rooms at one line per signal. In the case of an extended TV-station campus, there are also length limitations for signal lines (at dimensions above 450 feet per line) to be considered, which leads to the need for a lot of compensation equipment like frame stores, line amplifiers, equalizers and so on. Today there is no future-proof way of getting around IT switching technology when thinking of creating new extensive AV infrastructure, where flexibility in terms of using high performance glass fibre links for connections and a short setup-time are demanded.

In many cases there is a need for distributing signals or signal groups from one single signal source to several consuming devices. In connection with fibre optics switching there is no more need of using a cost-intensive central switching infrastructure. Signals are routed between the source nodes and destination nodes over free fibre capacity. What is left are several control instances (semi-automatic and/or manual) to send signal switch commands as needed. Nodes also can be physically connected and disconnected as the production situation affords it.

Consider for instance a camera signal in a remote studio combined with one or two microphone and intercom signals. From the mentioned control room a switching command is sent over the network to a number of switches to put up a connection path from the source device to several consuming devices as there are monitoring equipment as video monitors and loudspeakers connected to the studio, recording devices as media servers, continuity systems and transmission lines in the network. Many of these connections are different concerning their signal path, so a path calculation is necessary after every switching command.

Depending on the size of the network this can be an intensive computing operation, as the network is not bus-based, and hence multicast and broadcast facilities are not present [3]. The routing algorithm for example has to concern a limit of connection path length as there is an additionally hardware-depending signal delay in the area of micro- to milliseconds per hop, mainly due to conversions between optics and electronics and vice versa. The underlying problem is called a limited steiner tree problem which is known to be NP-hard [4, 5]. There are a number of approximation algorithms to solve this kind of problem.

Additionally there is a need of immediate reaction after a switching control command. Known algorithms — most of them based on a variant of Prim's algorithm which finds a minimum spanning tree for a connected weighted graph — have to go through the whole network to build a multicast tree and therefore have runtimes more than linear in the size of the graph. This leads to a latency problem which excludes to calculate point-to-multipoint routes in real-time for more than few dozen nodes.

3 Routing Algorithm

To achieve real-time behavior, the proposed algorithm is divided into two parts: an offline part and an online part. Cf. [6] for this technique, although in a different setting. The first part, the offline part, has the goal of constructing a routing table for each node. Using a distributed algorithm [7], the routing table is computed from each router locally by accessing a common database which presents the network topology information at the nodes' boot time. A regular database check is proposed to guarantee long-term coherence between database and switch routing information. After processing a connection request, the common database and the local routing information are updated automatically.

Using the Floyd-Warshall algorithm the distances and paths between all nodes are computed in the offline part, i.e. we solve an all-pairs-shortest-path problem. Note that in principle we could restrict to computing shortest paths from all sources to all sinks, but this would not make a real difference. Every hop between routing nodes is assigned a weight of 1, as the major delay occurs during the opto-electronic conversion in routing nodes. Therefore, the physical distances between the nodes do not play a role in our scenario.

After finishing the first part every router has knowledge about the complete network, e.g. the number of nodes and their connections.

The second part starts with an inquiry for a point-to-multipoint path calculation. Let $G = (V, E)$ be the graph representation of the network, V the set of all nodes and E the set of all edges, $M = \{v_1, \ldots, v_m\}$ the set of destination nodes, and s the source node. The online part of the algorithm proceeds as follows.

1. Retrieve all distances $d(s, v_1), \ldots, d(s, v_m)$, i.e. the distances of all destinations from the source, from the routing table, together with the corresponding paths.

2. Sort the distance values in ascending order. From now on we will assume $v_1, \ldots, v_m$ to represent this order, i.e. v_1 has the shortest distance from s, and v_m has the largest distance from s.

3. Start the Multipoint path MPATH with $MPATH =$path(s, v_m).

4. Let i be a loop index counting down from $m = |M|$ to 2, to add node v_{i-1} to the multicast tree.

   (a) If $dist(v_i, v_{i-1}) < dist(v_{i-1}, s)$
   then add path(v_i, v_{i-1}) to $MPATH$
   else add path(s, v_{i-1}) to $MPATH$. Information about the path length between v_i to v_{i-1} is retrieved from the local routing table.

   (b) (optional) If destination nodes already are covered by the existing path, they are no longer considered for the calculation.

(c) A variable which calculates the number of hops is increased when the If-case has been taken, i.e. if the shortest path to the next destination is shorter than to the source. The variable is reset to zero otherwise. If the maximum hop count is reached, the next destination node must be connected to the source. So an unacceptable delay time due to the accumulation of hop time is avoided. This way the routing propagation is limited.

5. As there is the possibility of loops in the constructed graph $MPATH$, a general spanning tree algorithm is used to eliminate them.

The online algorithm is a greedy algorithm as it only compares destination nodes in a certain order. It differs from previous approaches by not considering cheapest paths first, but starting with the farthest node. This has the advantage that the destination with the largest distance does not get an additional handicap, i.e. increased hop count, by being treated last. Also, by taking the farthest destination first, other destinations with smaller distance may already be covered on that path, if destinations are close to each other. In contrast to the trivial solution that only takes the paths from the source to all destinations, our algorithm leads to a better utilization, i.e. lower bandwidth requirements, of the network, as we trade additional hop count for reduced link capacity by connecting destinations directly. In contrast to heuristics for minimum spanning trees or ones using depth first search [8], that have to consider all nodes in the graph G, our algorithm only considers precomputed paths, so that the last step only works on a graph G' consisting of the source and the m destinations, and at most m edges, where each edge corresponds to a precomputed path in graph G.

The correctness of the algorithm is straightforward: in steps 1 to 4, a graph is constructed that contains the source, all destinations, and paths in between. Therefore it is connected. This graph either is a tree with s as its root. If it contains a loop, this is removed in the last step, so that a tree remains.

Let $m = |M|$ be the number of destination nodes. Only the online part of the algorithm is considered when analyzing the algorithm complexity. Retrieving the distances in algorithm step 1 takes $O(m)$ time. Sorting the distance values in step 2 takes time $O(m \cdot \log m)$. Step 3 takes time $O(1)$. Step 4 can be executed in $O(m)$ time without optional check for already visited nodes. Step 5 can be solved in time $O(m \log m)$ by Prim's algorithm with Fibonacci heaps. So the total time complexity of the algorithm also results in $O(m \cdot \log m)$, where m typically is much smaller than $|V|$.

4 Experimental Results

The Java implementation of our algorithm running on a standard notebook computer was validated by a large number of routing tests which proved the efficiency and fitness for AV-network purposes. As a first example, the German academic network backbone (DFN/X-WIN) was examined as a potential candidate for multimedia mass transmissions. We chose node 36(SAA) as the source, and destination nodes 1(KIE), 3(GRE), 19(ZIB) and 25(LEI). The routing results for two different propagation limits (infinite and 13 hops) are presented in Figs. 1 and fig. 2.

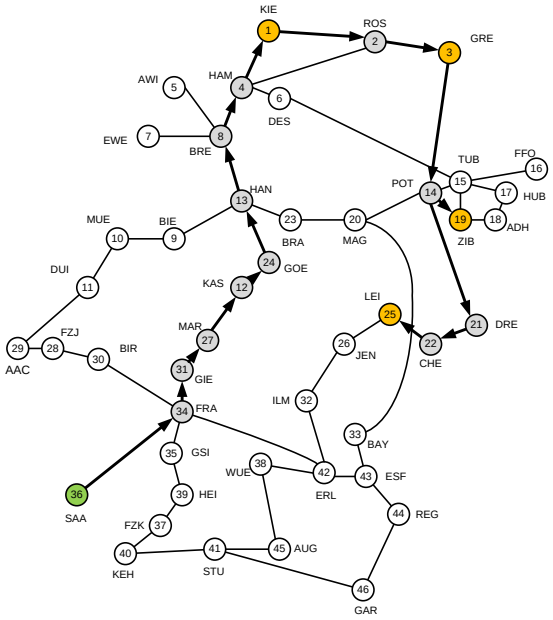

Figure 1. German Academic Network backbone, without propagation limit

The difference of the routing propagation can be recognized easily. As the propagation limit (max. hop count) is set to 13 in Fig. 2 the last destination node cannot be reached by connection to the previous destination node. The graph branches at node 34(FRA) and takes a shortest path from the source to 25(LEI). The algorithm runtime then was approximately 5500 microseconds, the original runtime without limitation was about 7000 microseconds on the mentioned notebook computer.

Another evaluation considers routing with clustered destinations. Based on the last network example we assume three destination clusters in Fig. 3, a northern cluster (AWI, EWE, HAM, DES, BRE), an eastern cluster (POT, TUB, FFO, HUB, ZIB, ADH) as well as a southern cluster (FZK, KEH, STU, AUG). The clusters represent German cities with relatively short distances within the clusters and a potential need of common infrastructure. In this example the propagation limit was set to a value higher than the number of nodes. The result is very close to an optimum routing. Setting the limit to a value smaller than the length of the remotest node the algorithm calculates different routes. In

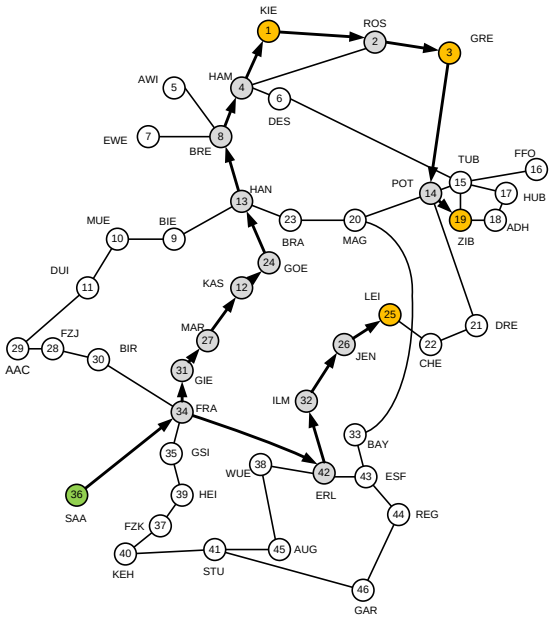

Figure 2. German Academic Network backbone, with propagation limit

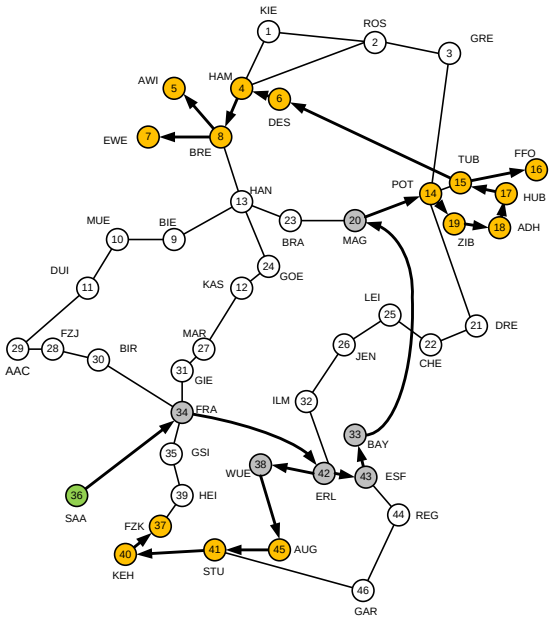

Figure 3. German Academic Network backbone with clustered destinations

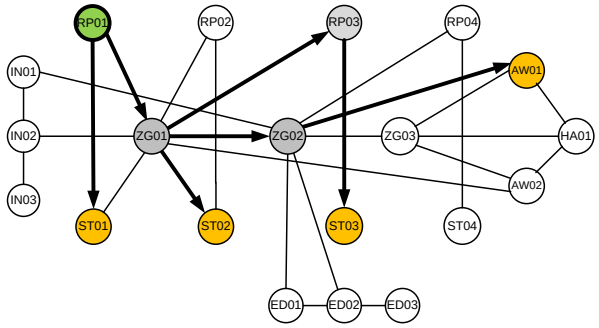

Figure 4. TV station AV-network sample 1

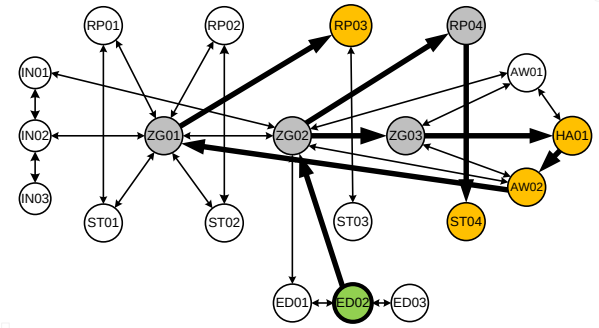

Figure 5. TV station AV-network sample 2

most cases a second vertical branch (FRA, GIE,. . . , HAN) is used to transport the signal to the remote northern cluster to serve those destination nodes with a smaller number of hops and to meet the given maximum delay.

A further example shows an example of a fibre-optic-wiring of a typical small television station. Figures 4 and 5 show routing paths for different sources and destinations which were computed by our algorithm.

In Fig. 4 node RP01 catches the routing command and calculates a multicast tree as shown in thick lines. Destination nodes are ST01, ST02, ST03 and AW01. This routing path is optimal, as there exists no shorter possible path. Fig. 5 uses ED02 as source node, RP03, HA01, AW02, RP04 and ST04 as destination nodes. The algorithm took about 7700 microseconds to calculate the graph.

We also created a larger, synthetic network with 100 nodes. Its structure is a regular grid network. The innermost node was chosen as source node, 14 destination nodes in different combinations and locations were set up. The result in Fig. 6 depicts a quite reasonable routing at a calculation time around only 10 milliseconds.

5 Related work

There exist a number of point-to-multipoint algorithms developed for special purposes.

Kodialam et al. [9] present a multicast tree selection heuristic to solve the point-to-multipoint routing problem (PMPRP). The heuristic is based on deferred loading of links. It allows routing optimization when part of the network links are critical and heavily loaded. Prerequisites are link-state and capacity information about the network. The algorithm extracts all available facts to predict possible future link demands which are considered for path routing calculations. Like our algorithm it is amenable for parallel and distributed implementation.

A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area was surveyed by Jia [1]. He uses a fully distributed heuristic algorithm with a possible dynamic membership change of multicast groups. It concerns two edge related parameters, distance and cost, separately.

Chen [10] presents two techniques for fast computation of constrained shortest paths. A reduction of discretization errors lead to faster algorithms.

Galiasso [11] presents a hybrid evolutionary algorithm for solving the PMPRP. A heuristic and a Steiner tree algorithm are used to find near-optimal solutions to the PMPRP. His method uses single split paths (a maximum of two) to optimize the bandwidth and accommodates the option of multiple paths by request.

Christensen [12] also presents a solution to the point-to-multipoint Routing problem based on Steiner trees and genetic algorithms.

Kostic [13] works on high bandwidth data dissemination using an overlay mesh and an algorithm called Bullet, which is a scalable and distributed algorithm that enables nodes spread across the network to self-organize into an overlay mesh.

Routing restoration algorithms which efficiently compute new routes that avoid failed links are developed in [14]. The task is to select restoration paths which retain nearly optimal performance while minimizing the disruption to the traffic flow.

6 Conclusions

We have presented an algorithm to compute a spanning tree in order to route point-to-multipoint connections in an optoelectronic multimedia AV-network for a television studio. The algorithm obeys the real-time requirements by splitting into an offline part and a greedy online part. The computed spanning trees are minimal or close to minimal in all examples considered. Our algorithm differs from other approaches by applying a seldomly seen combination of offline and online parts, by considering destinations in the order of their distance from the source, and by considering the farthest destination first.

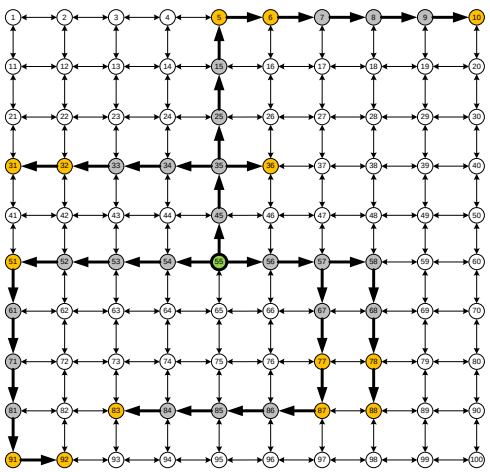

Figure 6. Synthetic demo network

Future work involves improvements of the offline part. Currently, we create the routing tables from scratch again and again, although typically only small changes occur such as a link being removed because its capacity is fully consumed or because it has a failure. Here, a differential algorithm may help.

Furthermore, by computing alternative routes directly after establishing a connection [15], fault-tolerance could be integrated into the current proposal in a straightforward manner.

References

[1] X. Jia et al., A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks, *IEEE/ACM Transactions on Networking*, 6(6), Dec 1998, 828–837.

[2] A. Kovalick, *Video Systems in an IT Environment — The Essentials of Professional Networked Media*, (St. Louis: Focal Press, 2006).

[3] M. Gerla et al., Multicast protocols for high-speed, wormhole-routing local area networks, *Proc. SIGCOMM '96: Conference on Applications, technologies, architectures, and protocols for computer communications*, Stanford, USA, 1996, 184–193.

[4] V. P. Kompella et al., Multicast routing for multimedia communication, *IEEE/ACM Transactions on Networking*, 1(3), June 1993, 286–292.

[5] C. A. S. Oliveira et al., Optimization problems in multicast tree construction, AT&T Labs Research Technical Report TD-6DTLXR, June 2005.

[6] S. Sichalou, L. Georgiadis, Algorithms for precomputing constrained widest paths and multicast trees, *IEEE/ACM Transactions on Networking*, 13(5), Oct 2005, 1174–1187.

[7] G. Franzl, Path selection methods with multiple constraints in service-guaranteed WDM networks, *IEEE/ACM Transactions on Networking*, 12(1), Feb 2004, 59–72.

[8] Z. Li, J. J. Garcia-Luna-Aceves, Finding multi-constrained feasible paths by using depth-first-search, *Wireless Networks*, 13(3), June 2007, 323–334.

[9] M. Kodialam et al., Online multicast routing with bandwidth guarantees: a new approach using multicast network flow, *IEEE/ACM Transactions on Networking*, 11(4), Aug 2003, 676–686.

[10] S. Chen et al., Two techniques for fast computation of constrained shortest paths, *IEEE/ACM Transactions on Networking*, 16(1), Feb 2008, 105–115.

[11] P. Galiasso, R. L. Wainwright, A hybrid genetic algorithm for the point to multipoint routing problem with single split paths, *Proc. 2001 ACM Symposium on Applied Computing (SAC '01)*, Las Vegas, USA, 2001, 327–332.

[12] H. L. Christensen et al., A hybrid algorithm for the point to multipoint routing problem, *Proc. 1997 ACM Symposium on Applied Computing (SAC '97)*, San Jose, USA, 1997, 263–268.

[13] D. Kosti et al., Bullet: high bandwidth data dissemination using an overlay mesh, *Proc. 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, Bolton Landing, USA, 2003, 282–297.

[14] D. Applegate et al., Coping with network failures: routing strategies for optimal demand oblivious restoration, *Proc. Joint Int.l Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '04/Performance '04)*, New York, USA, 2004, 270–281.

[15] P. K. Gummadi, An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks, *IEEE/ACM Transactions on Networking*, 11(1), Feb 2003, 81–94.