Comparing Push- and Pull-Based Broadcasting

Or: Would “Microsoft Watches” Profit from a Transmitter?

Alexander Hall1,* and Hanjo Täubig2,**

1 Computer Engineering and Networks Laboratory, ETH Zürich
CH-8092 Zürich, Switzerland
hall@tik.ee.ethz.ch
2 Department of Computer Science, TU München
D-85748 Garching b. München, Germany
taeubig@in.tum.de

Abstract. The first main goal of this paper is to present Sketch-it!, a framework aiming to facilitate development and experimental evaluation of new scheduling algorithms. It comprises many helpful data-structures, a graphical interface with several components and a library with implementations of selected scheduling algorithms. Every scheduling problem covered by the classification-scheme originally proposed by Graham et al. [22] can easily be integrated into the framework.

One of the more recent enhancements of this scheme, the so called broadcast scheduling problem, was chosen for an extensive case study of Sketch-it!, yielding very interesting experimental results that represent the second main contribution of this paper. In broadcast scheduling many clients listen to a high bandwidth channel on which a server can transmit documents of a given set. Over time the clients request certain documents. In the pull-based setting each client has access to a slow bandwidth channel whereon it notifies the server about its requests. In the push-based setting no such channel exists. Instead it is assumed that requests for certain documents arrive randomly with probabilities known to the server.

The goal in both settings is to generate broadcast schedules for these documents which minimize the average time a client has to wait until a request is answered.

We conduct experiments with several algorithms on generated data. We distinguish scenarios for which a slow feedback channel is very advantageous, and others where its benefits are negligible, answering the question posed in the title.

1 Introduction

During the past 40 years scheduling problems have received a lot of research interest, a huge theoretical background was developed. For books on the topic

* Supported by the joint Berlin/Zürich graduate program CGC, financed by ETH Zürich and German Research Foundation (DFG).
** Supported by DFG grant Ma 870/6-1 (SPP 1126: Algorithmics of large and complex Networks) and by DFG grant Ma 870/5-1 (Leibniz Award Ernst W. Mayr).
see for instance \[13, 14, 15\]. While working on scheduling problems it would often be convenient to have a tool with which an algorithm could be implemented and tested quickly. “Playing” with the algorithm can help gaining intuition of how it works, or on the other hand potentially speed up the finding of counter-examples and bad cases. Quite often it is also meaningful to get hints on the performance of an algorithm by having a quick glance at its empirical behavior. Then again some heuristics can only be evaluated by conducting such experiments. But besides for testing new algorithms, a tool which is able to animate the progress of an algorithm could also prove very helpful for presentations or in teaching topics of scheduling theory.

These points stimulated the development of Sketch-it!, a framework for simulation of scheduling algorithms. To maximize the applicability, its design was closely linked to the $\alpha/\beta/\gamma$-classification-scheme, originally proposed by Graham et al. \[22\]. Basically all problems covered by this scheme can be tackled with the help of the framework.

In this paper we give a short overview of the framework, in order to introduce it to a broad audience. Furthermore we present experimental results in the broadcast scheduling domain, which were obtained with the help of Sketch-it!. The motivation for this is partly to demonstrate the usability of the tool, but mainly we believe that the results are of interest in their own. In the next section we motivate and define broadcast scheduling.

**Motivation and Problem Statement of Broadcast Scheduling**

Due to the increasing availability of infrastructure that supports high-bandwidth broadcast and due to the growth of information-centric applications, broadcast scheduling is gaining practical importance \[4\]. The general setting of the broadcast scheduling problem is that (possibly many) clients request documents (e.g. web pages) from a server, and the server answers these requests via a high-bandwidth channel to which all clients are connected. If several clients have requested the same document, a single broadcast of this document satisfies all their requests simultaneously. One wants to determine a broadcast schedule that optimizes some objective function, usually the average response time (the time a client has to wait on average until her request is satisfied; in the scheduling literature, this is also called the average flow time).

There are two principally different settings: on the one hand on-demand or pull-based broadcasts and on the other hand push-based broadcasts.

In the pull-based setting each client has access to a low-bandwidth feedback channel, e.g. a modem connection, whereon it notifies the server about its requests. Two examples of such systems are @Home network \[24\] and DirecPC \[18\], which provide Internet access via cable television and via satellite, respectively.

In the push-based setting no feedback channel exists. The server tries to anticipate user behavior from the previously observed popularities of individual documents. A classical example are Teletext systems, where the user can select a page on her remote control and then has to wait until it appears in the periodic broadcast. A more recent application is the SPOT technology announced