Opportunities for Bandwidth Adaptation in Microsoft Office Documents

Eyal de Lara[†], Dan S. Wallach[‡], and Willy Zwaenepoel[‡]

[†] Department of Electrical and Computer Engineering
[‡] Department of Computer Science
Rice University
{delara,dwallach,willy}@cs.rice.edu

Abstract

Microsoft Office, the most popular office productivity suite, produces large documents that can result in long download latencies for platforms with limited bandwidth. To reduce latency and improve the user's experience, these documents need to be adapted for transmission on a limited-bandwidth network.

To identify opportunities for adaptation, we characterize documents created by three popular applications from the Microsoft Office suite: Word, PowerPoint, and Excel. Our study encompasses over 12,500 documents retrieved from 935 different Web sites.

Our main conclusions are: 1) Microsoft Office documents are large and require adaptation on bandwidth-limited clients; 2) embedded objects and images account for the majority of the data in these documents, with image types being the most popular non-text content, suggesting that adaptation efforts should focus on these elements; 3) compression considerably reduces the size of these documents; and 4) the internal structure of these documents (pages, slides, or sheets) can be used to download elements on demand and reduce user-perceived latency.

1 Introduction

Microsoft Office is the most popular productivity suite for creating documents. Its popularity derives, to some extent, from its ability to create *compound documents* that include data from more than one application. The potentially large size of these documents results in long download and upload latencies for mobile clients accessing the documents through bandwidth-limited links [3, 11, 22]. To reduce latency and improve the user's experience, compound documents and the applications that operate on them need to adapt to the available bandwidth.

To identify opportunities for adapting compound documents we need to understand their main characteristics. However, most studies of content types, especially those done on the Web [4, 21, 23, 24] have consistently ignored compound documents or treated them as opaque data streams, ignoring the rich internal structure that can be used to enhance bandwidth adaptation. In this paper we present an analysis of Office compound documents downloaded from the Web. We focus on those characteristics of Office documents that have implications for bandwidth-limited clients, and identify opportunities for adaptation. Although we report our findings with an emphasis on bandwidth-limited clients, we believe that these results will be useful for office suite designers and people interested in working with compound documents in general.

We undertook this study as part of our Puppeteer project, which uses component-based technology to adapt applications for different operating environments. Puppeteer is well suited for adapting compound documents that include data generated by several software components. By exposing the hierarchy of component data in the compound document, and making calls to the run-time APIs that the components expose, Puppeteer adapts applications without changing their source code. In contrast, traditional adaptation approaches have not been successful for applications that operate on compound documents mainly because the complex and proprietary nature of these applications thwarts source code modifications [9, 10] and the inclusion of several complex data types, usually embedded in a single file, makes system-based adaptation hard [12, 18, 20].

For this paper, we studied compound documents generated by three popular applications of the Office suite: Word, PowerPoint, and Excel. We chose to focus on Office applications based on four factors. First, Office is the most widely-used productivity suite. Moreover, a significant number of Microsoft Office documents are available on the Web, enabling us to gather the data for our experiments. Second, the Office file formats, although proprietary, are reasonably well documented. Third, the Office applications are highly integrated with each other and have published run-time APIs that can be used by Puppeteer to adapt the applications. Fourth, Office 2000 supports two native file formats: the proprietary OLE-based binary format and a new XML format. By using Office 2000 to convert old files to the new XML format, we can compare the tradeoffs of using a proprietary binary-based file format against a modern standards-based text format, both as intermediate formats suitable for document editing, and as publishing formats, suitable only for reading.

Although we concentrate exclusively on Office documents, we believe that our results apply to compound documents generated by other productivity suites. Since most of these suites support roughly the same features (embedding, images, etc), and document content is driven largely by user needs, it is likely that the main characteristics of documents produced by various productivity suites (*e.g.* distribution of document size, percentage that have images, number of pages, slides, etc.) would be similar.

We downloaded over 12,500 documents, comprising over 4 GB of data, from 935 different sites. Our main results are:

1. Office documents are large, with average sizes of 196 KB, 891 KB, and 115 KB for Word, PowerPoint and Excel respectively. Their large sizes suggest a need for adaptation in low bandwidth situations.
2. Office documents are component rich. 18.19% of Word documents and 46.38% of PowerPoint documents have at least one embedded component. Images were the most common component type.
3. In large documents, images and components account for the majority of the data, suggesting that they should be the main target of the adaptation effort.
4. For small documents, the XML format produces much larger documents than OLE. For large documents, there is little difference.
5. Compression considerably reduces the size of documents in both formats. Moreover, once compressed there is no significant difference in the sizes of the two file formats.
6. XML formats are easier to parse and manipulate than the OLE-binary formats.

The rest of this document is organized as follows. Section 2 provides some background on compound documents and their enabling technology. We also discuss relevant characteristics of the three Office applications that we use in this study. Section 3 describes the documents we used in our experiments. Section 4 presents our experimental results. Section 5 discusses the relevance of our findings to other productivity suites. Finally, section 6 discusses our conclusions.

2 Background

To its user, a compound document appears to be a single unit of information, but in fact it can contain elements created by different applications. A compound document could, for instance, consist of a spreadsheet and several images embedded into a text document.

In the general case, every data type in a compound document (spreadsheet, text, image, sound, etc.) is created and managed by a different application. The different applications used to create the document can be thought of as *software components* that provide services that are invoked to create, edit, and display the compound document.

In the remainder of this section we review the technologies used by Office to enable compound documents. We start with an overview of COM, OLE, and Automation. We then talk about the two native file formats supported by Office. Finally, we present a taxonomy of components found in Office applications.

2.1 COM, OLE and Automation

Office compound documents are based on the Component Object Model (COM) [5] and the Object Linking and Embedding (OLE) [6] standards, which govern the interactions between the various software components used to create compound documents.

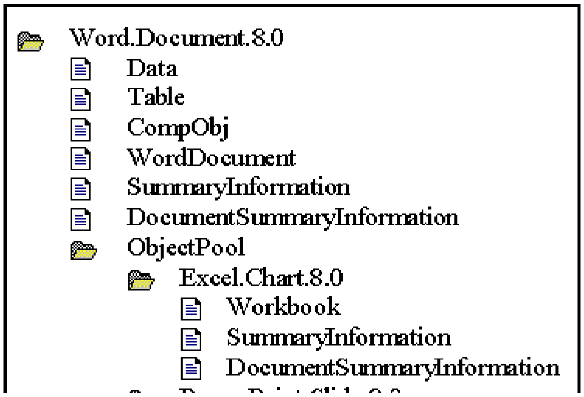
COM enables software components to export well-defined interfaces and interact with one another. In COM, software components implement their services as one or more COM objects. Every object implements one or more interfaces, each of which exports a number of methods. COM components communicate by invoking these methods.

OLE is a set of standard COM interfaces that enable users to create compound document by *linking* and *embedding* objects (components) into container applications, hence the name OLE.

Automation is an OLE technology, which enables third party applications to remotely control Office applications. Puppeteer adapts applications, to a large extent, by invoking Automation interfaces to modify application behavior when executing on bandwidth limited platforms. For example, using Automation interfaces, Puppeteer can adapt a large PowerPoint presentation by loading only a couple of slides, instead of the full presentation, before returning control to the user. While the user works on these slides, Puppeteer loads the remaining slides in the background, and as new slides become available, it instructs PowerPoint to append them to the presentation.

2.2 File formats

Office 2000 supports two native file formats: the traditional OLE-based binary format (hereafter, "OLE archive") and a new XML-based format. The OLE archives [13, 14, 15] rely on the OLE Structured Storage Interface (SSI) to provide a unified view of the compound document in a single file. SSI implements an abstraction similar to a file system



- Word.Document.8.0
  - Data
  - Table
  - CompObj
  - WordDocument
  - SummaryInformation
  - DocumentSummaryInformation
  - ObjectPool
    - Excel.Chart.8.0
      - Workbook
      - SummaryInformation
      - DocumentSummaryInformation