UDP-Lite

Johannes Hamfler
KMI12
johannes.hamfler@hft-leipzig.de

ABSTRACT

In diesem Dokument wird das Leightweight User Datagram Protokoll (UDP-Lite) beschrieben, welches ähnlich UDP ist. Der Focus dieses Dokuments liegt in der Beschreibung der Vorteile, die UDP-Lite gegenüber UDP aufweisen kann. Des Weiteren wird das Protokoll in das ISO OSI-Referenzmodell eingeordnet und die Auswirkungen auf andere Schichten in diesem beschrieben. Dieses Dokument orientiert sich stark am RFC.

——————————— LaTeX *alternate bla*

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

Keywords

UDP-Lite

1. EINLEITUNG

UDP, welches im RFC 768 beschrieben ist wurde jahrelang als verbindungsloses Protokoll verwendet und ist weit verbreitet. Ach heute hat dieses große Bedeutung für Sprachdienste, Videokommunikation und Echtzeitübertragung. Der Vorteil des Protokolls gegenüber TCP liegt vor allem bei der Sprachkommunikation darin, dass verlorene und fehlerhafte Datenpakete nicht erneut übertragen werden, da diese nach wenigen Millisekunden schon nicht mehr von Bedeutung sind. UDP-Lite versucht das Problem der fehlerhaften Pakete, welche beim Empfänger gelöscht werden, zu mindern, indem die Option besteht fehlerhafte Pakete dennoch zu verwenden und an höhere Schichten weiterleiten zu können. Bei Sprachdiensten hätte dies den Vorteil, dass der in einer höheren Schicht angesiedelte Codec die korrekten Bits auf eine bestimmte weise verarbeitet, so dass diese nützlich für die Anwendung sind. Fehlerhafte Bits könnten für den Codec ebenfalls einen Nutzen darstellen, so dass UDP-Lite in diesem Zusammenhang einen Vorteil darstellen würde.

——————————— *proceedings* (18 × 23.5 cm [7" × 9.25"]) [1] `\alignauthor \balancecolumns`

In RFC 3828 findet sich ..

Da manche Codecs die Fähigkeit besitzen beschädigten Payload zu behandeln und nützliche Informationen aus diesem zu extrahieren, wurde bei UDP-Lite ein Paket in zwei Teile aufgegliedert werden. Ein Teil kann mit einem Fehlerkorrekturwert überprüft werden, um die Integrität der darin enthaltenen Daten zu sichern, ein anderer Teil kann ohne Prüfsumme vorhanden sein.

In dem Teil, in welchem eine Fehlerüberprüfung stattfinden soll, werden üblicherweise Steuerinformationen übertragen, welche unbedingt fehlerfrei vorhanden sein müssen, um die Parameter des Payloads beim Empfänger richtig interpretieren zu können. Sollte der Payload in diesem Teil beschädigt sein, wird das Paket beim Empfänger in der Transportschicht verworfen.

Der andere Teil des Payloads, welcher beim klassischen UDP üblicherweise Daten enthält, welche nicht zwingend neu übertragen werden müssen, kann ohne Fehlerkorrektur übertragen werden, damit die darüber liegenden Schichten auch beschädigte Daten bearbeiten können um aus diesen ebenfalls nützliche Informationen für eine Anwendung zu extrahieren. Da in diesem Teil nicht überprüft wird ob Fehler vorhanden sind, wird der Payload nicht für die Entscheidung der Weiterleitung an höhere Schichten verwendet.

Wird eine Prüfsumme über das gesamte Paket angewandt, so ist UDP-Lite semantisch identisch zu UDP.

Im RFC wurden Beobachtungen erläutert, welche hier kurz erwähnt werden.

Es wurden folgende Codecs als Beispiele genannt, welche mit UDP-Lite eine Verbesserung der decodierten Daten erreichen können: AMR speech codec [RFC-3267] Internet Low Bit Rate Codec [ILBRC] error resilient H.263+ [ITU-H.263] H.264 [ITU-H.264; H.264] MPEG-4 [ISO-14496] video codecs)

Des Weiteren ist es nützlich, wenn niedrigere Schichten beschädigte IP Pakete weiterleiten, wenn dies verlangt wird. Sollten Verbindungen sich ihrer Fehleranfälligkeit bewusst sein, so ist es möglich, dass eine physische Verbindung eine höhere Sicherheit für sensible Daten gewährleisten, was durch verschiedene Fehlerkorrekturverfahren erreicht werden kann.

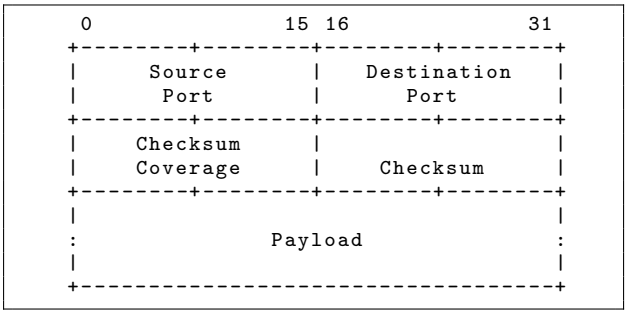
Außerdem sollte die Transport- und Vermittlungsschicht

[1] Two of these, the `\numberofauthors`

höher gelegene Applikationen nicht an ihrer Ausführung hindern, weil Pakete beschädigt sind. UDP eigent sich deshalb nur bedingt, da bei diesem die Prüfsumme gesetzt sein muss. Bei IP ist dies nicht der Fall.

2. PROTOKOLLÜBERSICHT *UDP-LITE*

Nachfolgend ist der UDP-Header abgebildet.



[1]

Dieser unterscheidet sich von dem UDP-Header in der Hinsicht, dass das Length-Feld mit einem Cecksum-Coverage-Feld ausgestattet wurde. Dieses ist dazu da, um die Länge anzugeben, bis wohin die Prüfsumme berechnet wird. Dies war möglich, da die Information über die Länge des Pakets aus IP-Paketen entnommen werden kann.

2.1 Beschreibung der Felder der PDU

Das Source- und Destination-Port-Feld sind dem von UDP gleich, wobei das Checksum-Coverage-Feld die Länge in Oketetten angibt, die von der Prüfsumme einbezogen werden. Hierbei wird ab dem ersten Oktett mit dem Zählen angefangen.

Der Header muss dabei immer mit einer Prüfsumme gesichert werden. Eine Prüfsumme von 0 bedeutet dabei, dass das gesamte Paket in die Prüfsumme einbezogen wird. Die Prüfsumme des UDP-Header muss 0 oder mindestens 8 sein, was bedeutet, dass die 8 Bytes des Headers beinhaltet sein müssen. Ein Paket mit einer Prüfsummenlänge zwischen 1 und 7 muss beim Empfänger verworfen werden, da dieses die Bedingung nicht erfüllt. Das berechnete Prüfsummenfeld muss einen Preuso-Header enthalten welcher auf IP basiert. UDP-Lite Pakete die eine gößere Prüfsummenlänge als der IP-Header annehmen kann müssen ebenfalls verworfen werden.

Da das Prüfsummenfeld ein 16-Bit-Komplement-Einerkomplement der Summe des Einerkomplements des Pseudo-Headers darstellt, muss das UDP-Lite-Paket im Payload ein vielfaches von 2Byte aufweisen. Notfalls wird dieses mit Nullen aufgefüllt. Die Informationen, aus welchen die Prüfsumme berechnet wird, werden dem IP-header entnommen. Bevor die Prüfsumme berechnet wird muss das Prüfsumenfeld jedoch auf 0 gesetzt werden. Sollte nach der Berechnung die Prüfsumme 0 ergeben, so werden 16 Einsen übertragen.

Da manche Anwendungen die UDP-Lite benutzen möglicherweise keine Fehlerbehandlung wünschen, kann hier die Prüfsummenlänge einfach auf 8 gesetzt werden um den Payload nicht mit einbeziehen zu müssen. Dadurch wird sichergestellt, dass die Steuerinformationen in jedem Fall korrekt ankommen.

2.2 Der Pseudo-Header

Der Pseudo-Header von UDP-Lite unterscheidet sich von dem in UDP insofern, dass der Wert des Längenfeldes nicht von UDP-Lite-Header genommen wird, sondern von Informationen aus den IP-Paketen. Dabei wird die Berechnung gleich wie bei TCP ausgeführt, was bedeutet, dass nicht nur der Payload, sondern auch der Header von UDP-Lite mit einbezogen wird. Dadurch, dass die Länge auf diese Weise berechnet, muss das Prüfsummenfeld bei 8 beginnen und ermöglicht so einen einfachen softwareseitigen Vergleich.

2.3 Die Anwendungsschnittstelle

Die Anwendungsschnittstelle stellt die gleichen Funktionen wie bei UDP zur Verfügung. Des Weiteren sollte eine Möglichkeit der sendenden Anwendung bestehen, den Prüfsummenlängenwert an UDP-Lite zu übertragen. Zumindest sollte jedoch die Möglichkeit für eine empfangende Anwendung bestehen, die Weiterleitung von Paketen mit Prüfsummenlängen kleiner als eine Festgelegte zu blockieren.

Im RFC wird empfohlen, dass UDP-Lite standardmäßig das Verhalten von UDP imitieren sollte, indem das Prüfsummenlängenfeld der Länge des UDP-Lite-Pakets entsprechen soll, um das gesamte Paket verifizieren zu können. Über einen expliziten Aufruf, einem sogenannten System Call, beim Sender sollen Anwendungen die fehlertolerant sind UDP-Lite ihre Fehlertoleranz mitteilen. Eine empfangende Anwendung die ebenfalls eine teilweise angewandte Prüfsumme nutzen wollen, sollte dies ebenfalls über einen einen solchen Aufruf kund geben.

Da im Internet Pfade variieren und verschiedene Eigenschaften aufweisen, können keine pauschalen Aussagen über die Fehlerschemas einer Verbindung gemacht werden. Deshalb sollten Anwendungen die UDP-Lite nutzen keine Annahmen der Fehler in einem UDP-Lite-Paket machen, solange der Bereich nicht in die Berechnung der Prüfsumme mit einbezogen wurde. Anwendungen sollten deshalb, wenn nötig ihre eigenen Fehlerprüfmechnismen nutzen.

2.4 Die IP-Schnittstelle

Wie bei UDP muss auch UDP-Lite den Pseudo-Header der IP-Implementierung erhalten, welcher die IP-Adresse und Protokollfelder des IP-Headers beinhaltet, sowie die Länge des IP-Payloads welche aus dem Längenfeld der IP-Pakete entnommen werden kann. Der Sender darf dabei nicht den IP-Payload mit Padding-Bytes auffüllen, da die Länge des UDP-Lite-Pakets daraus entnommen werden soll.

2.5 IP-Jumbo-PDUs

Da das Prüfsummenlängenfeld Werte bis 65535 annehmen kann, können genaue Prüfsummenlängen benutzt werden. Dies ist bei Jumbo-PDUs (Jumbogrammen) nicht der Fall. Es kann entweder das gesamte Paket mit der Prüfsummenlänge von 0 oder alle Oktette bis zum 65535ten Oktett einschließen.

2.6 Betrachtung der niedrigeren Schichten

Frames die UDP-Lite-Pakete enthalten dürfen von niedrigeren Schichten nicht verworfen werden, da die Fehlerbehandlung in einer höheren Schicht erfolgt. Eine Ausnahme wäre der Fall, wenn ein Fehler im sensiblen Datenbereich existiert. Das Prüfsummenlängenfeld könnte für Verbindungen, welche partielle Fehlererkennung ermöglichen dafür benutzt werden, niedrigeren Schichten mitzuteilen, in welchen Bereichen Fehler auftreten dürfen. Da der sensible Teil des UDP-

Lite-Pakets zwischen dem ersten Oktet des IP-Headers und dem letzten Oktett liegt, welcher vom Prüfsummenlängenfeld bekannt gegeben wird, kann der sensible Teil in der gleichen Weise behandelt werden, wie ein UDP-Paket.

Da Verbindungen, welche keine partielle Fehlererkennung ermöglichen, in einem Fehlerfall das Paket verwerfen müssen, wird das UDP-Lite-Paket in gleicher Weise wie ein UDP-Paket behandelt.

Somit lässt sich bei UDP-Lite sagen, dass dieses Protokoll nur eine Verbesserung erwirken kann, wenn in Schicht 2 des OSI-Referenzmodells die Partielle Prüfsumme und die Prüfsummenlänge von UDP-Lite genutzt wird. Dies würde seine Wirkung jedoch erst in Fehleranfälligen Umgebungen entfalten.

2.7 Kompatibilität mit UDP

Da UDP und UDP-Lite eine ähnliche Syntax und Semantik hat, können Anwendungen UDP-Lite anstatt UDP nutzen mit der Eigenschaft das ganze Paket mit einer Fehlerkorrektur zu sichern. Des Weiteren sind durch die Ähnlichkeit nur geringe Änderungen an Anwendungen vorzunehmen, um UDP-Lite zu nutzen.

UDP-Lite hat ein eigene IP-Protokoll-Identifikation (136), welche es einem Empfänger erlaubt ob UDP-Lite oder UDP genutzt wird. Ein Empfänger, welcher das UDP-Lite-Protokoll nicht kennt wird ein ICMP-Paket mit einer Fehlernachricht zurück senden. Damit kann festgestellt werden, ob anderen Systemen UDP-Lite bekannt ist. Diese Nachrichten können folgende sein:

- ICMP "Protocol Unreachable"
- ICMPv6 "Payload Type Unknown"

Ein Problem würde bei der Verwendung einer gleichen UDP-Identifikation enstehen, da ein UDP-Lite-Payload mit einer partiellen Prüfsumme von UDP-Anwendungen verworfen wird und UDP-Pakete, welche nur teilweise den IP-Payload füllen, können nicht an UDP-Lite-Anwendungen weitergeleitet werden. Das Problem dabei wäre die Nicht-Benachrichtigung des Senders, welches durch folgende Maßnahmen laut dem RFC eingedämmt werden könnte:

- Explizite Nutzung der Signalisierung innerhalb des Payloads ohne die partielle Prüfsumme zu verwenden, um dem Sender das Erkennen der UDP-Lite-Unterstützung zu ermöglichen
- Nutzung eines anderen Protokolls zur Signalisierung, wie zum Beispiel SIP, damit erkannt werden kann, ob der Empfänger UDP-Lite nutzen kann

Da jedoch UDP-Lite eine eigene Identifikation besitzt, müssen diese Varianten nicht benutzt werden.

2.8 Sicherheitsbetrachtungen

- `\section` [2] **document**

[2]This is the second footnote. It starts a series of three footnotes that add nothing informational, but just give an idea of how footnotes work and look. It is a wordy one, just so you see how a longish one plays out.

2.9 Type Changes and *Special* Characters

We have already seen several typeface changes in this sample. You can indicate italicized words or phrases in your text with the command `\textit`; emboldening with the command `\textbf` and typewriter-style (for instance, for computer code) with `\texttt`. But remember, you do not have to indicate typestyle changes when such changes are part of the *structural* elements of your article; for instance, the heading of this subsection will be in a sans serif[3] typeface, but that is handled by the document class file. Take care with the use of[4] the curly braces in typeface changes; they mark the beginning and end of the text that is to be in the different typeface.

You can use whatever symbols, accented characters, or non-English characters you need anywhere in your document; you can find a complete list of what is available in the *LaTeX User's Guide*[6].

2.10 Math Equations

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

2.10.1 Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin. . .\end` construction or with the short form `$. . .$`. You can use any of the symbols and structures, from α to ω, available in LaTeX[6]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\to\infty} x = 0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

2.10.2 Display Equations

A numbered display equation – one set off by vertical space from the text and centered horizontally – is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\to\infty} x = 0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty} x_i = \int_0^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

[3]A third footnote, here. Let's make this a rather short one to see how it looks.

[4]A fourth, and last, footnote.

Table 1: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

2.11 Citations

Citations to articles [2, 4, 3, 5], conference proceedings [4] or books [7, 6] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the `.tex` file [6]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide*[6].

This article shows only the plainest form of the citation command, using `\cite`. This is what is stipulated in the SIGS style specifications. No other citation format is endorsed or supported.

2.12 Tables

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material is found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

2.13 Figures

Like tables, figures cannot be split across pages; the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures, use the environment **figure** to enclose the figure and its caption.

This sample document contains examples of **.eps** and **.ps** files to be displayable with LaTeX. More details on each of these is found in the *Author's Guide*.

Figure 1: A sample black and white graphic (.eps format).

Figure 2: A sample black and white graphic (.eps format) that has been resized with the `epsfig` command.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. and don't forget to end the environment with figure*, not figure!

Note that either **.ps** or **.eps** formats are used; use the `\epsfig` or `\psfig` commands as appropriate for the different file types.

2.14 Theorem-like Constructs

Other common constructs that may occur in your article are the forms for logical constructs like theorems, axioms, corollaries and proofs. There are two forms, one produced by the command `\newtheorem` and the other by the command `\newdef`; perhaps the clearest and easiest way to distinguish them is to compare the two in the output of this sample document:

This uses the **theorem** environment, created by the `\newtheorem` command:

THEOREM 1. *Let f be continuous on $[a, b]$. If G is an antiderivative for f on $[a, b]$, then*

$$\int_a^b f(t)dt = G(b) - G(a).$$

The other uses the **definition** environment, created by the `\newdef` command:

Definition 1. If z is irrational, then by e^z we mean the unique number which has logarithm z:

$$\log e^z = z$$

Two lists of constructs that use one of these forms is given in the *Author's Guidelines*.

There is one other similar construct environment, which is already set up for you; i.e. you must *not* use a `\newdef` command to create it: the **proof** environment. Here is a example of its use:

Figure 4: A sample black and white graphic (.ps format) that has been resized with the `psfig` command.

Table 2: Some Typical Commands

Command	A Number	Comments
`\alignauthor`	100	Author alignment
`\numberofauthors`	200	Author enumeration
`\table`	300	For tables
`\table*`	400	For wider tables

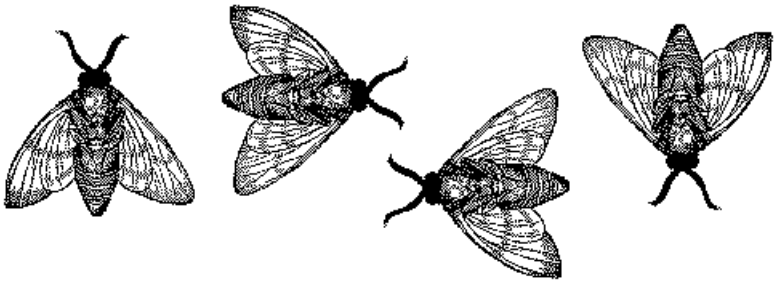


Figure 3: A sample black and white graphic (.eps format) that needs to span two columns of text.

PROOF. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty}\frac{f(x)}{g(x)}=L.$$

Then

$$l=\lim_{x\to c}f(x)=\lim_{x\to c}\left[gx\cdot\frac{f(x)}{g(x)}\right]=\lim_{x\to c}g(x)\cdot\lim_{x\to c}\frac{f(x)}{g(x)}=0\cdot L=0,$$

which contradicts our assumption that $l\neq 0$. □

Complete rules about using these environments and using the two different creation commands are in the *Author's Guide*; please consult it for more detailed instructions. If you need to use another construct, not listed therein, which you want to have the same formatting as the Theorem or the Definition[7] shown above, use the `\newtheorem` or the `\newdef` command, respectively, to create it.

A *Caveat* for the TEX Expert

Because you have just been given permission to use the `\newdef` command to create a new form, you might think you can use TEX's `\def` to create a new command: *Please refrain from doing this!* Remember that your LATEX source code is primarily intended to create camera-ready copy, but may be converted to other forms – e.g. HTML. If you inadvertently omit some or all of the `\def`s recompilation will be, to say the least, problematic.

3. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LATEX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

4. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

5. REFERENCES

[1] *RFC 3828.*

[2] M. Bowman, S. K. Debray, and L. L. Peterson. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, 15(5):795–825, November 1993.

[3] J. Braams. Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, 12(2):291–301, June 1991.

[4] M. Clark. Post congress tristesse. In *TeX90 Conference Proceedings*, pages 84–89. TeX Users Group, March 1991.

[5] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, 15(5):745–770, November 1993.

[6] L. Lamport. *LaTeX User's Guide and Document Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[7] S. Salas and E. Hille. *Calculus: One and Several Variable.* John Wiley and Sons, New York, 1978.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TEX Expert

A.3 Conclusions

A.4 Acknowledgments

A.5 Additional Authors

This section is inserted by LaTeX; you do not insert it. You just add the names and information in the `\addition-alauthors` command at the start of the document.

A.6 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.

B. MORE HELP FOR THE HARDY

The sig-alternate.cls file itself is chock-full of succinct and helpful comments. If you consider yourself a moderately experienced to expert user of LaTeX, you may find reading it useful but please remember not to change it.