# Your Own Distributed System using Consensus Protocols

## Bellagio Casinó roulette multiplayer

Stefano Agostini agostinistefano1991@gmail.com

Salomé Paolo paolosalome@gmail.com

Valenti Alessandro alessandro.valenti1991@gmail.com

#### **ABSTRACT**

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings. It is an alternate style which produces a tighter-looking paper and was designed in response to concerns expressed, by authors, over page-budgets. It complements the document Author's (Alternate) Guide to Preparing ACM SIG Proceedings Using  $\text{LaTeX2}_{\epsilon}$  and BibTeX. This source file has been written with the intention of being compiled under  $\text{LaTeX2}_{\epsilon}$  and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file. A compiled PDF version is available on the web page to help you with the 'look and feel'.

#### 1. INTRODUZIONE

Le tecnologie Cloud, esplose in questi ultimi anni, hanno permesso di trasferire il peso della computazione e dell'approvvigionamento delle risorse informatiche verso internet. Perció la rete come strumento di sviluppo oggi offre scenari ampi nella direzione della distribuzione delle risorse e della computazione: la distribuzione, come nel caso che poniamo in esame, deve rappresentare un'avanguardia per lo sviluppo di applicazioni con caratteristiche di scalabilitá e di alta disponibilità per l'utente finale. Gli sviluppatori utilizzando i servizi cloud possono gestire in maniera semplificata l'immagazzinamento di informazioni e l'elaborazione di molti dati, concentrandosi principalmente sullo sviluppo della loro applicazione piuttosto che nell'utilizzo di un sistema ortodosso e dispendioso come avveniva nel passato. Aziende come Spotify, Netflix ed altre sfruttano il servizio cloud per rendere fruibile le loro applicazioni agli utenti di tutto il mondo, ottenendo molteplici consensi. Molte persone all'interno del mondo non sanno che queste applicazioni sono cosí efficienti proprio perché sfruttano una tecnologia cloud, che per quanto possa agevolare gli utenti e gli sviluppatori, presenta sempre qualche difficoltá da gestire:

- garantire l'accesso concorrente ad una vasta molteplicitá di utenti.
- 2. la possibilitá di connettersi da ogni angolo del mondo,

senza ledere la qualitá del servizio.

- la sincronizzazione degli eventi associati ad ogni applicativo.
- 4. la reperibilitá dei dati.

#### 1.1 Obiettivi

L'obiettivo preposto consiste nell'elaborare un gioco multiplayer che supporta molteplici entitá autonome che si contendono risorse condivise, l'aggiornamento in tempo reale di una qualche forma di stato condiviso, essere distribuito su molteplici nodi (eventualmente distribuiti geograficamente) e supportare la scalabilitá. E' proprio con l'obiettivo di rispettare tali punti cardine che abbiamo proceduto nello sviluppo della nostra applicazione distribuita sulla piattaforma Cloud: abbiamo operato nell'intenzione di venire incontro alle esigenze degli utenti, sempre più abituati ad avere interazioni costanti e ripetute con le applicazioni di uso comune. Il Bellagio Casinó é l'applicazione (accessibile mediante interfaccia Web) da noi proposta che consente agli utenti di giocare ad una versione semplificata di una roulette francese condividendo un unico tavolo di gioco. Tale applicazione permette di effettuare particolari puntate in base alla propria disponibilitá di credito.

#### 2. ARCHITETTURA

In questa sezione verrá illustrata a livello logico la nostra architettura descrivendo i servizi *Amazon* utilizzati e i moduli sviluppati. I componenti architetturali sono:

- un Cluster composto da istanze EC2, situato nella regione Francoforte.
- due Cluster composti da istanze EC2, situate nella regione Eire.
- Client composto da una applicazione Web.

i servizi AWS sfruttati dai componenti sono:

- SNS per la comunicazione inter-cluster.
- DynamoDB per la persistenza dei dati situato nella regione Eire.
- Redis per supportare la logica del gioco situato sia nella regione Eire che Francoforte.
- LoadBalancer per distribuire il carico.

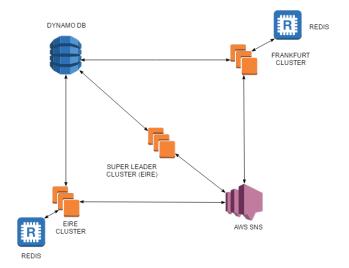


Figure 1: Architettura

La motivazione che ci ha spinto nell'adottare piú cluster su piú aree geografiche europee é stata la possibilitá di realizzare una scalabilitá di tipo geografico. Ció non toglie che tale realizzazione si possa estendere anche all'interno di altre regioni geografiche messe a disposizione da AWS. La motivazione che ci ha spinto a questa scelta é legata a fattori di latenza ma anche economici, facilitando lo sviluppo dell'applicazione e la fase di testing.

## 2.1 Cluster EC2

All'interno di questa sezione andiamo a trattare nel dettaglio quali sono le funzionalitá e le componenti dei Cluster EC2.

I Cluster istanziati sono soggetti ad una gerarchia dettata dalla realizzazione di uno stato condivisibile, che corrisponde alle fasi di gioco dell'applicazione. Pertanto possiamo suddividere tali cluster in base alla loro mansione :

- SuperLeader Cluster: identifica il Cluster per la gestione dello stato di gioco, e per le azioni di registrazione e di accesso sito in Irlanda.
- Frankfurt/Eire Cluster: identifica uno dei Cluster di gioco sito sia in Irlanda che a Francoforte.

All'interno di ogni Cluster per mantenere lo stato corrente del gioco, abbiamo adottato Raft come protocollo del consenso. Abbiamo utilizzato 3 istanze EC2 per Cluster, in quanto é il numero minimo di nodi necessario per raggiungere il consenso mediante tale protocollo.

## 3. IMPLEMENTAZIONE

### 3.1 Metodologie di sviluppo

L'intero progetto é stato realizzato seguendo la metodologia di sviluppo *Scrum*, la quale prevede di dividere il progetto in blocchi rapidi di lavoro(Sprint), alla fine di ciascuno dei quali si deve creare un incremento del software. L'intera stesura del progetto, dalla progettazione all'implementazione, é stata quindi suddivisa in sprint con scadenza molto stretta assegnati ai singoli componenti del team. Per la suddivisione dei task sono stati effettuati incontri giornalieri, mentre per

la sincronizzazione e per il controllo di versione é stato utilizzato il servizio di *Google Drive*. Il periodo di sviluppo, é stato caratterizzato da incontri giornalieri (Daily Scrum) incentrati sulla tematica del giorno, con la presenza del team al completo. A supporto dello sviluppo é stato previsto, fin dal primo istante, un ulteriore incontro prima della partenza di ogni nuovo sprint.La metodologia utilizzata si é dimostrata efficiente poiché ha permesso di progettare e realizzare un architettura ben congegnata in un arco temporale ridotto.

## 3.2 Tecnologie utilizzate

Il progetto é stato interamente realizzato utilizzando il framework Node.js, che consente di sfruttare il linguaggio JavaScript, tipicamente utilizzato nella client-side, anche per la scrittura di applicazioni server-side. La caratteristica principale di Node.js risiede nella possibilità di accedere alle risorse del sistema operativo in modalità event-driven evitando il modello basato su processi o thread concorrenti utilizzato dai classici web server. Per facilitare l'uso di Node.js é stato utilizzato il framework Express.js, per la realizzazione semplificata del middleware HTTP. L'applicazione client é costituita da una singola pagina HTML con contenuti Javascript ed elaborata attraverso l'uso della libreria JQuery rendendo l'interfaccia dinamica e consistente e di facile utilizzo.

#### **3.3** Raft

Rafté un protocollo del consenso in alternativa a Paxos che offre la possibilità di condividere uno stato comune tra i vari nodi del Cluster. Questi si suddividono in leader, candidate e follower. Viene raggiunto il consenso attraverso l'elezione di un leader, la cui mansione principale é quella di propagare il log dei cambiamenti di stato ai propri follower. Inoltre per ribadire la propria leadership deve informare periodicamente i propri follower attraverso dei brevi messaggi. Ogni follower deve ricevere questi messaggi all'interno di un arco temporale fissato. Se ció non dovesse avvenire verrá richiesta una nuova elezione di leader; colui che indice la nuova elezione cambia il proprio ruolo in candidate. Per far sí che Raft tolleri K guasti occorrono almeno 2K+1 nodi all'interno del Cluster.

Per l'implementazione di tale protocollo é stato adottata la libreria Skiff. Implementa la persistenza dello stato attraverso un database in-memory di tipo MemDown il quale utilizza un'interfaccia LevelUp. Per quanto concerne la comunicazione del log e dell'heartbeat utilizza server TCP, in ascolto su ogni nodo, attraverso chiamate di tipo RPC.

Le motivazioni che ci hanno condotto a questa scelta sono da addurre alla semplicità di esecuzione e di condivisione del log tra i vari nodi del Cluster, la cui propagazione avviene in tempi irrisori. L'utilizzo di Skyff con l'architettura dei Cluster da noi ideata consente di raggiungere una certa tolleranza a guasti di vario genere sulle varie istanze EC2. In caso di un guasto ad una macchina, se questa risulta essere un leader quello che avviene é una nuova elezione, con conseguente riallineamento dei nodi del Cluster allo stato corrente del gioco. Nel caso di un guasto nei confronti di un follower bisogna monitorare il numero di nodi presenti nel Cluster, come precedentemente illustrato.

## 3.4 Comunicazione

Per quanto riguarda la comunicazione, vogliamo porre l'attenzione verso due tipologie presenti all'interno del nostro elaborato:

- Comunicazione *intra-Cluster*: sfruttiamo le chiamate *RPC*, come descritto nel paragrafo precedente.
- Comunicazione inter-Cluster: Sfruttiamo il servizio AWS di SNS.

Amazon SNS é un servizio di notifiche push rapido, flessibile e completamente gestito che ti consente di inviare messaggi individuali o collettivi a un numero elevato di destinatari. Grazie a questa soluzione, inviare notifiche push a utenti di dispositivi mobili, destinatari di posta o addirittura altri servizi distribuiti é semplice e conveniente. Per comunicare lo stato del gioco a tutti nodi dei vari Cluster sono stati creati 5 Topic per permettere lo scambio di informazioni e di cambiamento di stato, ció verrá chiarificato successivamente.

La motivazione per cui é stato scelto questo servizio sono le seguenti:

- SNS permette la realizzazione di un servizio di tipo Publish/Subscribe.
- Sfrutta endpoint di tipo http per la ricezione di notifiche.
- Permette la comunicazione tra piú aree geografiche.
- Garantisce un certo grado di riservatezza.

Generalmente SNS viene utilizzato in abbinamento con SQS, peró nel nostro caso non avevamo una necessitá di permanenza e di ordinamento dei messaggi.

#### 3.5 Persistenza

Poiché la nostra applicazione é un gioco multiplayer e avendo necessitá di gestire la registrazione e l'accesso di piú giocatori, ci occorre un servizio di persistenza dati. A tal proposito la persistenza é stata realizzata sfruttando due tipi di servizio:

- Redis: é un servizio di cache distribuito di Amazon AWS.
- DynamoDB: é un database NoSQL distribuito di Amazon AWS.

Nel dettaglio *Redis* viene sfruttato dai Cluster di gioco per poter immagazzinare le ultime informazioni relative alle puntate degli utenti, rendendo tali dati accessibili per l'elaborazione delle vincite da parte del sistema. Alla fine di questa operazione il sistema provvederá all'eliminazione di tali dati per permettere l'aggiornamento degli stessi coerentemente con la mano del gioco.

Per quanto concerne DynamoDB sono state create due tabelle

- una per conservare i dati relativi agli utenti (credenziali di accesso e credito).
- una per conservare le puntate elaborate dal sistema, precedentemente contrassegnate come vincenti o perdenti.

Quando un'applicazione effettua delle scritture su una tabella in DynamoDB i dati impiegano del tempo per propagarsi in ogni locazione di memoria della regione AWS corrente. I dati saranno consistenti in tutte le locazioni di memoria in un periodo lungo  $\sim 1$  sec o anche meno. Dato che la nostra

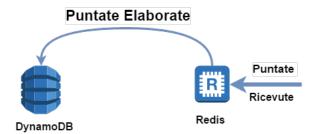


Figure 2: Processo scrittura delle puntate con interazione DynamoDB e Redis

applicazione non effettua chiamate al Database frequentemente, la scelta dell'*eventual consistency* ci é sembrata la piú opportuna anche in termini di costi.

Abbiamo sfruttato la chiave di partizionamento che offre DynamoDB per collocare tramite funzione hash le puntate appartenenti alla stessa mano nella stessa partizione; cosí come la chiave di ordinamento per catalogare nella stessa partizione le puntate appartenenti ad utenti differenti. Su consiglio delle linee guida fornite da DynamoDB abbiamo scelto di utilizzare come chiave di partizionamento la mano del gioco poiché offre un range di valori piú ampio rispetto a quello offerto dalla chiave username.

## 3.6 Implementazione Clusters

Come illustrato precedentemente i Cluster da noi sviluppati possono essere suddivisi funzionalmente in due gruppi distinti: SuperLeader e Frankfurt/Eire Cluster. Tutti i Cluster definiti sono configurati, come precedentemente indicato nella sezione Architettura, con un numero fissato pari a 3 di istanze EC2. La scelta di tale numero permette la gestione di un solo guasto (come descritto in Raft) e aumentando il numero di istanze il risultato sarebbe invariato con un grado di tolleranza ai guasti maggiore. Un'altra motivazione che ci ha spinto a tale scelta é di natura economica. In ogni caso l'aggiunta o meno di altre istanze all'interno dei nostri Cluster puó essere effettuata senza alcun problema, a patto che vengano identificate all'interno di una VPC attraverso un tag univoco. Il sistema é configurato in modo che ogni istanza all'avvio sappia identificare esattamente quali sono le macchine appartenenti al proprio gruppo, permettendole di sfruttare il protocollo Raft per comunicare con tutte le entitá del proprio gruppo. Tutto ció sinteticamente rappresenta un servizio di Discovery Service atto a configurare dinamicamente i Cluster Raft.

#### 3.6.1 Stati del Sistema

Poniamo ora l'attenzione verso gli Stati del Sistema. I due tipi di stato che si alternano sono:

- *Play*: identifica le fasi di gioco dell'applicazione in cui l'utente puó effettuare la puntata desiderata.
- Compute: identifica la fase di estrazione del numero e il calcolo delle puntate vincenti.

Nella definizione di questi stati viene allegato l'istante di tempo in cui viene generato. Si ha un cambiamento di stato quando entrambi i Cluster di gioco comunicano la terminazione dello stato corrente, come illustrato nella figura 3.

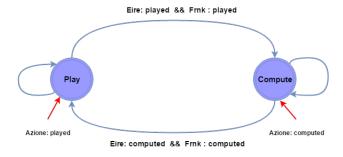


Figure 3: Rappresentazione degli Stati del sistema

## 3.6.2 SuperLeader

Il SuperLeader é il Cluster adibito alla definizione dello stato di gioco e all'accesso e alla registrazione degli utenti.

Come primo passo definiamo la gerarchia del Cluster che si compone di un leader e di due peer (follower). Tale definizione é derivata dai ruoli che impone *Raft.* Il leader, cosí come i due peer, si compone di due processi :

- processo Raft (padre) che si occupa del consenso per mezzo di server TCP e sfrutta SNS per pubblicare i cambi di stato.
- processo HTTP (figlio) per ricevere le notifiche SNS e le richieste da parte degli utenti.

Nel dettaglio il processo *Raft* sfrutta la libreria *Skiff* per mantenere il consenso su un determinato stato di gioco attraverso la propagazione del log e mediante chiamate *RPC*.

Lo stato Compute é lo stato iniziale di esecuzione del leader, questo viene propagato attraverso il servizio SNS a tutti i Cluster di gioco. Il leader prima di effettuare un cambio di stato attende la ricezione del completamento della fase precedente da parte di Entrambi vengono propagati verso i Cluster di gioco attraverso il servizio SNS in maniera ordinata.

DyanamoDB 3Topic SNS

#### 3.6.3 Frankfurt/Eire Cluster

Redis DyanamoDB 1

## 3.7 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 serif¹ typeface, but that is handled by the document class file. Take care with the use of² 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 \( \mathbb{L}T\_{EX}\) \( User's \) \( Guide[?]. \)

## 3.8 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.

#### 3.8.1 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  $\alpha$  to  $\omega$ , available in LaTeX[?]; 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).

#### 3.8.2 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 L<sup>A</sup>TEX; 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 **dis-playmath** 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.

#### 3.9 Citations

Citations to articles [?, ?, ?, ?], conference proceedings [?] or books [?, ?] 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 [?]. 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 *Late Year's Guide*[?].

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.

## 3.10 Tables

<sup>&</sup>lt;sup>1</sup>A third footnote, here. Let's make this a rather short one to see how it looks.

<sup>&</sup>lt;sup>2</sup>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
$\pi$	1 in 5	Common in math
\$	4 in 5	Used in business
$\Psi_1^2$	1 in 40,000	Unexplained usage

Figure 4: A sample black and white graphic.

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 ETEX 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.

## 3.11 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 files to be displayable with LATEX. If you work with pdfLATEX, use files in the .pdf format. Note that most modern TEX system will convert .eps to .pdf for you on the fly. More details on each of these is found in the *Author's Guide*.

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!

#### 3.12 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

Figure 5: A sample black and white graphic that has been resized with the includegraphics command.

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:

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[?] 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 \defs recompilation will be, to say the least, problematic.

## 4. 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.

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 6: A sample black and white graphic that needs to span two columns of text.

Figure 7: A sample black and white graphic that has been resized with the includegraphics command.

#### 5. 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.

#### **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 TFX 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 \additionalauthors 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.