Writing Technical Articles

The notes below apply to technical papers in computer science and electrical engineering, with emphasis on papers in systems and networks.

Read Strunk and White, *Elements of Style*. Again.

Give the paper to somebody else to read. If you can, find two people: one person familiar with the technical matter, another only generally familiar with the area.

Papers can be divided roughly into two categories, namely original research papers and survey papers. There are papers that combine the two elements, but most publication venues either only accept one or the other type or require the author to identify whether the paper should be evaluated as a research contribution or a survey paper. (Most research papers contain a "related work" section that can be considered a survey, but it is usually brief compared to the rest of the paper and only addresses a much narrower slice of the field.)

Research Papers

A good research paper has a clear statement of the problem the paper is addressing, the proposed solution(s), and results achieved. It describes clearly what has been done before on the problem, and what is new.

The goal of a paper is to describe novel technical results. There are four types of technical results:

1. An algorithm;
2. A system construct: such as hardware design, software system, protocol, etc.;

   One goal of the paper is to ensure that the next person who designs a system like yours doesn't make the same mistakes and takes advantage of some of your best solutions. So make sure that the hard problems (and their solutions) are discussed and the non-obvious mistakes (and how to avoid them) are discussed. (Craig Partridge)

3. A performance evaluation: obtained through analyses, simulation or measurements;
4. A theory: consisting of a collection of theorems.

A paper should focus on

- describing the results in sufficient details to establish their validity;
- identifying the novel aspects of the results, i.e., what new knowledge is reported and what makes it non-obvious;
- identifying the significance of the results: what improvements and impact do they suggest.

Paper Structure

- Typical outline of a paper is:
  - Abstract, typically not more than 100-150 words;
  - Introduction (brief!): introduce problem, outline solution; the statement of the problem should include a clear statement why the problem is important (or interesting).
  - Related Work (or before summary). Hint: In the case of a conference, make sure to cite the work of the PC co-chairs and as many other PC members as are remotely plausible, as well as from anything relevant from the previous two proceedings. In the case of a journal or magazine, cite anything relevant from last 2-3 years or so volumes.
  - Outline of the rest of the paper: "The remainder of the paper is organized as follows. In Section 2, we introduce ..Section 3 describes ... Finally, we describe future work in Section 5." [Note that Section is capitalized. Also, vary your expression between "section" being the subject of the sentence, as in "Section 2 discusses ..." and "In Section, we discuss ...".]
  - Body of paper
    - problem
    - approach, architecture
    - results

    The body should contain sufficient motivation, with at least one example scenario, preferably two, with illustrating figures, followed by a crisp generic problem statement model, i.e., functionality, particularly emphasizing "new" functionality. The paper may or may not include formalisms. General evaluations of your algorithm or architecture, e.g., material proving that the algorithm is O(log N), go here, not in the evaluation section.

    Architecture of proposed system(s) to achieve this model should be more generic than your own peculiar implementation. Always include at least one figure.

    Realization: contains actual implementation details when implementing architecture isn't totally straightforward. Mention briefly implementation language, platform, location, dependencies on other packages and minimum resource usage if pertinent.

    Evaluation: How does it really work in practice? Provide real or simulated performance metrics, end-user studies, mention external technology adoptors, if any, etc.

  - Related work, if not done at the beginning
  - Summary and Future Work
    - often repeats the main result
  - Acknowledgements
  - Bibliography

- Appendix (to be cut first if forced to):
  - detailed protocol descriptions
  - proofs with more than two lines
  - other low-level but important details

It is recommended that you write the approach and results sections first, which go together. Then problem section, if it is separate from the introduction. Then the conclusions, then the intro. Write the intro last since it glosses the conclusions in one of the last paragraphs. Finally, write the abstract. Last, give your paper a title.

Title

- Avoid all but the most readily understood abbreviations.
- Avoid common phrases like "novel", "performance evaluation" and "architecture", since almost every paper does a performance evaluation of some architecture and it better be novel. Unless somebody wants to see 10,000 Google results, nobody searches for these types of words.

  Use adjectives that describe the distinctive features of your work, e.g., reliable, scalable, high-performance, robust, low-complexity, or low-cost. (There are obviously exceptions, e.g., when the performance evaluation is the core of the paper. Even in that case, something more specific is preferable, as in "Delay measurements of X" or "The quality of service for FedEx deliveries".)

- If you need inspiration for a paper title, you can consult the *Automatic Systems Research Topic or Paper Title Generator*.

Authors

- The IEEE policies (Section 6.4.1) state the following about authorship:

The IEEE affirms that authorship credit must be reserved for individuals who have met each of the following conditions: 1) made a significant intellectual contribution to the theoretical development, system or experimental design, prototype development, and/or the analysis and interpretation of data associated with the work contained in the manuscript, 2) contributed to drafting the article or reviewing and/or revising it for intellectual content, and 3) approved the final version of the manuscript, including references.

Abstract

- The abstract must **not** contain references, as it may be used without the main article. It is acceptable, although not common, to identify work by author, abbreviation or RFC number. (For example, "Our algorithm is based upon the work by Smith and Wesson.")
- Avoid use of "in this paper" in the abstract. What other paper would you be talking about here?
- Avoid general motivation in the abstract. You do not have to justify the importance of the Internet or explain what QoS is.

- Highlight not just the problem, but also the principal results. Many people read abstracts and then decide whether to bother with the rest of the paper.
- Since the abstract will be used by search engines, be sure that terms that identify your work are found there. In particular, the name of any protocol or system developed and the general area ("quality of service", "protocol verification", "service creation environment") should be contained in the abstract.
- Avoid equations and math. Exceptions: Your paper proposes $E = m c^2$.

Introduction

- Avoid stock and cliche phrases such as "recent advances in XYZ" or anything alluding to the growth of the Internet.
- Be sure that the introduction lets the reader know what this paper is about, not just how important your general area of research is. Readers won't stick with you for three pages to find out what you are talking about.
- The introduction must motivate your work by pinpointing the problem you are addressing and then give an overview of your approach and/or contributions (and perhaps even a general description of your results). In this way, the intro sets up my expectations for the rest of your paper -- it provides the context, and a preview.
- Repeating the abstract in the introduction is a waste of space.
- Example bad introduction:

  Here at the institute for computer research, me and my colleagues have created the SUPERGP system and have applied it to several toy problems. We had previously fumbled with earlier versions of SUPERGPSYSTEM for a while. This system allows the programmer to easily try lots of parameters, and problems, but incorporates a special constraint system for parameter settings and LISP S-expression parenthesis counting.

  The search space of GP is large and many things we are thinking about putting into the supergpsystem will make this space much more colorful.

- A pretty good introduction, drawn from Eric Siegel's class:

  Many new domains for genetic programming require evolved programs to be executed for longer amounts of time. For example, it is beneficial to give evolved programs direct access to low-level data arrays, as in some approaches to signal processing \cite{teller5}, and protein segment classification \cite{handley,koza6}. This type of system automatically performs more problem-specific engineering than a system that accesses highly preprocessed data. However, evolved programs may require more time to execute, since they are solving a harder task.

  Previous or obvious approach:
  (Note that you can also have a related work section that gives more details about previous work.)) One way to control the execution time of evolved programs is to impose an absolute time limit. However, this is too constraining if some test cases require more processing time than others. To use computation time efficiently, evolved programs must take extra time when it is necessary to perform well, but also spend less time whenever possible.

Approach/solution/contribution:
The first sentence of a paragraph like this should say what the contribution is. Also gloss the results.

In this chapter, we introduce a method that gives evolved programs the incentive to strategically allocate computation time among fitness cases. Specifically, with an *aggregate computation time ceiling* imposed over a series of fitness cases, evolved programs dynamically choose when to stop processing each fitness case. We present experiments that show that programs evolved using this form of fitness take less time per test case on average, with minimal damage to domain performance. We also discuss the implications of such a time constraint, as well as its differences from other approaches to {\it multiobjective problems}. The dynamic use of resources other than computation time, e.g., memory or fuel, may also result from placing an aggregate limit over a series of fitness cases.

Overview:
The following section surveys related work in both optimizing the execution time of evolved programs and evolution over Turing-complete representations. Next we introduce the game Tetris as a test problem. This is followed by a description of the aggregate computation time ceiling, and its application to Tetris in particular. We then present experimental results, discuss other current efforts with Tetris, and end with conclusions and future work.

Body of Paper

Hints and common mistakes

Bibliography

- Avoid use of *et al.* in a bibliography unless list is very long (five or more authors). The author subsumed into *et al.* may be your advisor or the reviewer... Note punctuation of *et al.*.
- If writing about networks or multimedia, use the network bibliography. All entries not found there should be sent to me. A listing of frequently-used references for networks is available.
- Internet drafts must be marked ``work in progress". Make sure that they have been replaced by newer versions or RFCs. Any Internet Draft reference older than six months should automatically be suspicious since Internet Drafts expire after that time period.
- Book citations include publication years, but no ISBN number.
- It is now acceptable to include URLs to material, but it is probably bad form to include a URL pointing to the author's web page for papers published in IEEE and ACM publications, given the copyright situation. Use it for software and other non-library material. Avoid long URLs; it may be sufficient to point to the general page and let the reader find the material. General URLs are also less likely to change.
- Leave a space between first names and last name, i.e., "J. P. Doe", not "J.P.Doe".

Acknowledgements

- Acknowledge your funding sources. Some sources have specific wording requirements and may prefer that the grant number is listed. The NSF requires text like "This work was supported by the National Science Foundation under grant EIA NN-NNNNN."
- Generally, anonymous reviewers don't get acknowledged, unless they really provided an exceptional level of feedback or insight. Rather than "We thank X for helping us with Y", you might vary this as "X helped with Y.".

Reporting Numerical Results and Simulations

In all but extended abstracts, numerical results and simulations should be reported in enough detail that the reader can duplicate the results. This should include all parameters used, indications of the number of samples that contributed to the analysis and any initial conditions, if relevant.

When presenting simulation results, provide insight into the statistical confidence. If at all possible, provide confidence intervals. If there's a "strange" behavior in the graph (e.g., a dip, peak or change in slope), this behavior either needs to be explained or reasons must be given why this is simply due to statistical aberration. In the latter case, gathering more samples is probably advised.

Figures should be chosen wisely. You can never lay out the whole parameter space, so provide insight into which parameters are significant over what range and which ones are less important. It's not very entertaining to present lots of flat or linear lines.

The description of the graph should not just repeat the graphically obvious such as "the delay rises with the load", but explain, for example, how this increase relates to the load increase. Is it linear? Does it follow some well-known other system behaviors such as standard queueing systems?

LaTeX Considerations

- There's no need to enclose numbers in $$ (math mode).
- Use `\cite{a,b,c}`, not `\cite{a} \cite{b} \cite{c}`.
- Use the `\usepackage{times}` option for LaTeX2e - it comes out much nicer on printers with different resolutions. Plus, compared to cmr, it probably squeezes an extra 10% of text out of your conference allotment.
- Multi-letter subscripts are set in roman, not italics. For example,
- `x_{\mathrm max}`
- For uniformity, use the LaTeX2e graphics set, not the earlier psfigure set:
- `\usepackage{graphics}`
- `...`
- `\begin{figure}`
- `\resizebox{!}{0.5\textheight}{\includegraphics{foo.eps}}`
- `\caption{Some figure}`
- `\label{fig:figure}`
- `\end{figure}`

Things to Avoid

Too much motivational material
: Three reasons are enough -- and they should be described very briefly.

Describing the obvious parts of the result
: "Obvious" is defined as any result that a graduate of our program would suggest as a solution if you pose the problem that the result solves.

Describing unnecessary details
: A detail is unnecessary, if its omission will not harm the reader's ability to understand the important novel aspects of the result.

Spelling errors
: With the availability of spell checkers, there is no reason to have spelling errors in a manuscript. If you as the author didn't take the time to spell-check your paper, why should the editor or reviewer take the time to read it or trust that your diligence in technical matters is any higher than your diligence in presentation? Note, however, that spell checkers don't catch all common errors, in particular word duplication ("the the"). If in doubt, consult a dictionary such as the (on line) Merriam Webster.

Text in Arial:
: Arial and other sans-serif fonts are fine for slides and posters, but are harder to read in continuous text. Use Times Roman or similar serif fonts. Unusual fonts are less likely to be available at the recipient and may cause printing or display problems.

Guidelines for Experimental Papers

"Guidelines for Experimental Papers" set forth for researchers submitting articles to the journal, *Machine Learning*.

1. Papers that introduce a new learning "setting" or type of application should justify the relevance and importance of this setting, for example, based on its utility in applications, its appropriateness as a model of human or animal learning, or its importance in addressing fundamental questions in machine learning.
2. Papers describing a new algorithm should be clear, precise, and written in a way that allows the reader to compare the algorithm to other algorithms. For example, most learning algorithms can be viewed as optimizing (at least approximately) some measure of performance. A good way to describe a new algorithm is to make this performance measure explicit. Another useful way of describing an algorithm is to define the space of hypotheses that it searches when optimizing the performance measure.
3. Papers introducing a new algorithm should conduct experiments comparing it to state-of-the-art algorithms for the same or similar problems. Where possible, performance should also be compared against an absolute standard of ideal performance. Performance should also be compared against a naive standard (e.g., random guessing, guessing the most common class, etc.) as well. Unusual performance criteria should be carefully defined and justified.
4. All experiments must include measures of uncertainty of the conclusions. These typically take the form of confidence intervals, statistical tests, or

estimates of standard error. Proper experimental methodology should be employed. For example, if "test sets" are used to measure generalization performance, no information from the test set should be available to the learning process.

5. Descriptions of the software and data sufficient to replicate the experiments must be included in the paper. Once the paper has appeared in Machine Learning, authors are strongly urged to make the data used in experiments available to other scientists wishing to replicate the experiments. An excellent way to achieve this is to deposit the data sets at the Irvine Repository of Machine Learning Databases. Another good option is to add your data sets to the DELVE benchmark collection at the University of Toronto. For proprietary data sets, authors are encouraged to develop synthetic data sets having the same statistical properties. These synthetic data sets can then be made freely available.
6. Conclusions drawn from a series of experimental runs should be clearly stated. Graphical display of experimental data can be very effective. Supporting tables of exact numerical results from experiments should be provided in an appendix.
7. Limitations of the algorithm should be described in detail. Interesting cases where an algorithm fails are important in clarifying the range of applicability of an algorithm.

Other Hints and Notes

From Bill Stewart (Slashdot, May 7, 2006), edited

- Write like a newspaper reporter, not a grad student.
- Your objective is clear communication to the reader, not beauty or eruditeness or narration of your discoveries and reasoning process. Don't waste their time, or at least don't waste it up front.
- Hit the important conclusions in the first few sentences so your reader will read them. If you'd like to wrap up with them at the end of your memo, that's fine too, in case anybody's still reading by then, but conclusions come first.
- If you're trying to express something complex, simplify your writing so it doesn't get in the way. For something simple, 10th grade language structures will do, but if it's really hairy stuff, back down to 8th grade or so.
- Think about what your audience knows and doesn't know, and what they want and don't want. Express things in terms of what they know and want, not what you know.

From MarkusQ, Slashdot, May 7, 2006

- **Top down design** Starting with an outline and working out the details is the normal way of tackling an engineering problem.
- **Checking your facts** Engineers should be used to checking anything that is even remotely doubtful before committing to it. So should writers.
- **Failure mode analysis** For each sentence ask yourself, could it be misread? How? What is the best way to fix it?

- **Dependency analysis** Are the ideas presented in an order that assures that each point can be understood on the basis of the readers assumed knowledge and the information provided by preceding points?
- **Optimization** Are there any unnecessary parts? Does the structure require the reader to remember to many details at once, before linking them?
- **Structured testing** If you read what you have written assuming only the knowledge that the reader can be expected to have, does each part work the way you intended? If you read it aloud, does it sound the way you intended?

The Conference Review Process

It is hard to generalize the review process for conferences, but most reputable conferences operate according to these basic rules:

1. The paper is submitted to the technical program chair(s). Many current conferences require electronic submission, in either PostScript or PDF formats, occasionally in Word.
2. The technical program chair assigns the paper to one or more technical program committee members, hopefully experts in their field. The identity of this TPC member is kept secret.
3. The TPC member usually provides a review, but may also be asked to find between one and three reviewers who are not members of the TPC. They may be colleagues of the reviewer at the same institution, his or her graduate students or somebody listed in the references. The graduate student reviews can be quite helpful, since these reviewers often provide more detailed criticism rather than blanket dismissal. Any good conference will strive to provide at least three reviews, however, since conferences operate under tight deadlines and not all reviewers deliver as promised, it is not uncommon that you receive only two reviews.
4. In some conferences, there is an on-line discussion of papers among the reviewers for a particular paper. Usually, a lead TPC member drives the discussion and then recommends the paper for acceptance, rejection or discussion at the TPC meeting.
5. The technical program chair then collects the reviews and sorts the papers according to their average review scores.
6. The TPC (or, rather, the subset that can make the meeting), then meets in person or by phone conference. Usually, the bottom third and the top third are rejected and accepted, respectively, without (much) further discussion. The papers discussed are those in the middle of the range, or where a TPC member feels strongly that the paper ended up in the wrong bin, or where the review scores differ significantly. Papers that only received two reviews are also often discussed, maybe with a quick review by one of the TPC members as additional background. The rigor of the TPC meeting depends on the size and reputation of the conference. In some workshops and conferences, the TPC chairs may well make the final decision themselves, without involving the whole TPC.

Other References

- Bartleby has dictionaries, grammars, an encyclopedia, and *Columbia Guide To Standard American English*
- The Free Dictionary, also an online dictionary
- Berkeley Information Systems and Technology Publications style guide
- Guide to Grammar and Style, by J. Lynch
- alt.usage.english FAQ, addressing common grammar questions
- Writing Tips
- Key to common comments made on your papers
- Drafting the Paper in an Academic Style
- Religious Studies style sheet
- Cisco style guide
- Oded Goldreich wrote an essay entitled "How not to write a paper", with recommendations on style and organization.
- Don Knuth has online the TeX source of a book on "Mathematical Writing" (also useful for Computer Science).
- *The structure of paper/report in Systems*, by Michalis Faloutsos, U.C. Riverside
- The Elements of Style. William Strunk Jr. and E.B. White. Macmillan Publishing Co., New York, 1979.

  This is an **amazing** little book that you can read in a few hours. Your writing style will never be the same afterwards. This $8 book is the best investment you can ever make.

- "A Guide to Writing as an Engineer"
- *Spring into Technical Writing for Engineers and Scientists*
- Bugs in Writing. Lyn Dupre', (2nd ed.)

  This is a great book that expands on Strunk&White. It has more examples, and was written by an author who edited numerous technical publications, many of which were in computer science.

- The Chicago Manual of Style, Univ. of Chicago Press.

  This is the bible for American academic style. It's long and heavy, but has everything you ever want to know about style. When in doubt, or if you get conflicting stylistic advice, following The Chicago Manual of Style is your best choice.

- A Handbook for Scholars by Mary Claire van Leunen; Alfred Knopf, Publisher.

  This is another useful book written for publishing (computer) scientists.

- The UIST Guide for Authors is geared towards a specific conference, but the general process and guidelines are similar to many other conferences.
- *The Science of Scientific Writing*, George D. Gopen and Judith A. Swan, In *American Scientist*, Vol. 78, No. 6 (Nov-Dec, 1990), pp. 550-558.

This is a useful article that teaches scientists how to write single sentences and paragraphs.

- *The Mayfield Handbook of Technical and Scientific Writing*, Perelman, Paradis and Barrett, Mayfield, 1998.

  It is an extensive resource explaining how to write papers, reports, memoranda and Ph.D. thesis, how to make high-performance slides and oral presentations, how to avoid common pitfalls and mistakes in English, etc., with many examples of "good" and "bad" practices.

- Roy Levin and David D. Redell, An evaluation of the ninth SOSP submissions -or- How (and how not) to write a good systems paper, *ACM SIGOPS Operating Systems Review* **17**(3):35-40 (July, 1983).
- Alan Snyder, How to get your paper accepted at OOPSLA, *OOPSLA '91 Proceedings*, pp. 359-363.
- Mark Allman, A Referee's Plea, 2001
- Ralph Johnson *et al*, How to get a paper accepted at OOPSLA, *Panel at OOPSLA'93*, pp 429-436.
- Craig Partridge, How to Increase the Chances Your Paper is Accepted at ACM SIGCOMML.

  Generally useful advice that also applies to other networking conferences.

- What kinds of papers does USENIX publish?
- Alan Jay Smith, *The Task of the Referee*, *IEEE Computer* **23**(4):65-71 (April 1990).
- Grammar, Punctuation, and Capitalization, NASA SP-7084
- Stylewriter software

Talks

- "The Short Talk" (Charles Van Loan)
- "Pointers on giving a talk" (D. Messerschmitt)
- *Tips for Preparing Delivering Scientific Talks and Using Visual Aids* (ONR)

Miscellaneous

- International Standard Paper Sizes

Common Bugs in Writing

1. Avoid use of passive tense if at all possible. Example: "In each reservation request message, a refresh interval used by the sender is included." reads better and shorter as "Each ... message includes ..."
2. Use strong verbs instead of lots of nouns and simple terms rather than fancy-sounding ones. Examples:

verbose, weak verbs, bad	**short, strong, good**
make assumption	assume
is a function of	depends on
is an illustration	illustrates, shows
is a requirement	requires, need to
utilizes	uses

3. Check for missing articles, particularly if your native tongue doesn't have them. Roughly, concepts and classes of things don't, most everything else more specific does. ("Routers route packets. The router architecture we consider uses small rodents.") Don't use articles in front of proper nouns and names ("Internet Explorer is a popular web browser. The current version number is 5.0. Bill Gates did not write Internet Explorer.") [NEED POINTER HERE]
4. Each sentence in a paragraph must have some logical connection to the previous one. For example, it may describe an exception ("but", "however"), describe a causality ("thus", "therefore", "because of this"), indicate two facets of an argument ("on the one hand", "on the other hand"), enumerate sub-cases ("first", "secondly") or indicate a temporal relationship ("then", "afterwards"). If there are no such hints, check if your sentences are indeed part of the same thought. A new thought should get its own paragraph, but still clearly needs some logical connection to the paragraphs that preceded it.
5. Protocol abbreviations typically do not take an article, even if the expanded version does. For example, "The Transmission Control Protocol delivers a byte stream" but "TCP delivers a byte stream", since it an abstract term. ("The TCP design has been successful." is correct since the article refers to the design, not TCP.)

   Note that abbreviation for organizations do take a definite article, as in "The IETF standardized TCP.".

   Since the "P" in TCP, UDP and similar abbreviations already stands for "protocol", saying the "the TCP protocol" is redundant, albeit common. (LCD, Liquid Crystal Display, is another common case where many are tempted to incorrectly write LCD display. Indeed, Google references 2,060,000 instances of that usage.)

6. Use consistent tense (present, usually, unless reporting results achieved in earlier papers).

7. **None**: None can take either singular or plural verbs, depending on the intended meaning (or taste). Both *none of these mistakes are common* and *none of these mistakes is common* are correct, although other sources only lists the singular and The Tongue untied makes finer distinctions based on whether it refers to a unit or a measure.
8. Use hyphens for concatenated words: "end-to-end architecture", "real-time operating system" (but "the computer may analyze the results in real time"), "per-flow queueing", "flow-enabled", "back-to-back", ...

   In general, hyphens are used

   - adding prefixes that would result in double vowels (except for co-, de-, pre-, pro-), e.g., supra-auditory;
   - all-: all-around, all-embracing;
   - half-: half-asleep, half-dollar (but halfhearted, halfway);
   - quasi-: quasi-public
   - self-: self-conscious, self-seeking (but selfhood, selfless)
   - to distinguish from a solid homograph, e.g., re-act vs. react, re-pose vs. repose, re-sign vs. resign, re-solve vs. resolve, re-lease vs. release
   - A compound adjective made up of an adjective and a noun in combination should usually be hyphenated. (WiT, p. 230) Examples: cold-storage vault, hot-air heating, short-term loan, real-time operating system, application-specific integrated circuit, Internet-based.
   - words ending in -like when the preceding word ends in 'l', e.g., shell-like
9. Don't overuse dashes for separation, as they interrupt the flow of words. Dashes may be appropriate where you want to contrast thoughts very strongly or the dash part is a surprise of some sort. Think of it as a very long pause when speaking. In many cases, a comma-separated phrase works better. If you do use a dash, make sure it's not a hyphen (- in LaTeX), but an em-dash (--- in LaTeX).
10. Avoid *scare quotes*, as they indicate that the writer is distancing himself from the term.
11. Numbers ten or less are spelled out: "It consists of three fields", not "3 fields".
12. Use until instead of the colloquial *till*.
13. Use. *Eq. 7*, not *Equation (7)*, unless you need to fill empty pages.
14. *Optimal* can't be improved - *more optimally* should be *better* or maybe *more nearly optimal*.
15. Avoid in-line enumeration like: "Packets can be (a) lost, (b) stolen, (c) get wet." The enumeration only interrupts the flow of thought.
16. Avoid itemization (bullets), as they take up extra space and make the paper read like PowerPoint slides. Bullets can be used effectively for emphasis of key points. If you want to describe components or algorithms, often the description environment works better, as it highlights the term, providing a low-level section delineation.
17. Instead of "Reference [1] shows" or "[1] shows", use "Smith [1] showed" or "Smith and Jones [1] showed" or "Smith *et al* [1] showed" (if more than two authors). "et al" is generally used for papers with more than two authors. (Note that "et al" makes the subject plural, so it is "Smith et al [1] show" not "shows".) Or, alternatively, "the foobar protocol [1] is an example ...". This

keeps the reader from having to flip back to the references, as they'll recognize many citations by either author name or project name. No need to refer to RFC numbers in the text (except in RFCs and Internet Drafts). Exception for very low-level presentation: "RFC822-style addresses".

18. Use normal capitalization in captions ("This is a caption", not "This is a Caption").
19. All headings must be capitalized consistently, either in heading style, capitalizing words, or sentence style, across all levels of headings. Generally, captions for figures and tables are best left in sentence style.
20. Parentheses or brackets are always surrounded by a space: "The experiment(Fig. 7)shows" is wrong; "The experiment (Fig. 7) shows" is right.
21. Avoid excessive parenthesized remarks as they make the text hard to read; fold into the main sentence. Check whether the publication allows footnotes - some magazines frown upon them. More than two footnotes per page or a handful per paper is a bad sign. You probably should have applied to law school instead.
22. The material should make just as much sense without the footnotes. If the reader constantly has to look at footnotes, they are likely to lose their original place in the text. As a matter of taste, I find URLs better placed in the references rather than as a footnote, as the reader will know that the footnote is just a reference, not material important for understanding the text.
23. There is no space between the text and the superscript for the footnote. I.e., in LaTeX, it's `text\footnote{}` rather than `text \footnote{}`.
24. Check that abbreviations are always explained before use. Exceptions, when addressed to the appropriate networking audience: ATM, BGP, ftp, HTTP, IP, IPv6, RSVP, TCP, UDP, RTP, RIP, OSPF, BGP, SS7. Be particularly aware of the net-head, bell-head perspective. Even basic terms like PSTN and POTS aren't taught to CS students... For other audiences, even terms like ATM are worth expanding, as your reader might wonder why ATM has anything to do with cells rather than little green pieces of paper.
25. Never start a sentence with "and". (There are exceptions to this rule, but these are best left to English majors.)
26. Don't use colons (:) in mid-sentence. For example, "This is possible because: somebody said so" is wrong - the part before the colon must be a complete sentence.
27. Don't start sentences with "That's because".
28. In formal writing, contractions like *don't*, *doesn't*, *won't* or it's are generally avoided.
29. Be careful not to confuse `its` with `it's` (it is).
30. Vary expressions of comparison: "Flying is faster than driving" is much better than "Flying has the advantage of being faster" or "The advantage of flying is that it is faster.".
31. Don't use slash-constructs such as "time/money". This is acceptable for slides, but in formal prose, such expressions should be expanded into "time or money" or "time and money", depending on the meaning intended.
32. Avoid cliches like "recent advances in ...".
33. Don't use symbols like "+" (for "and"), "%" (for "fraction" or "percentage") or "->" (for "follows" or "implies") in prose, outside of equations. These are only acceptable in slides.

34. Avoid capitalization of terms. Your paper is not the U.S. Constitution or Declaration of Independence. Technical terms are in lower-case, although some people use upper case when explaining an acronym, as in "Asynchronous Transfer Mode (ATM)".
35. Expand all acronyms on first use, except acronyms that every reader is expected to know. (In a research paper on TCP, expanding TCP is probably not needed - somebody who doesn't know what TCP stands for isn't likely to appreciate the rest of the paper, either.)
36. Each paragraph should have a lead sentence summarizing its content. If this doesn't work naturally, the paragraph is probably too short. Try reading just the first lines of each paragraph - the paper should still make sense. For example,

    There are two service models, integrated and differentiated service. Integrated service follows the German approach that anything that isn't explicitly allowed is verboten. It strictly regulates traffic, but also makes the trains run on time. Differentiated service follows the Animal Farm appraoch, where some traffic is more equal than others. It seems simpler, until one has to worry about proletariat traffic dressing up as the aristocracy.

37. `$i$th`, not `$i-th$`.
38. Units are always in roman font, never *italics* or LaTeX math mode. Units are set off by one (thin) space from the number. In LaTeX, use ~ to avoid splitting number and units across two lines. `\;` or `\,` produces a thin space.
39. For readability, powers of a 1,000 are divided by commas.
40. Use "kb/s" or "Mb/s", not "kbps" or "Mbps" - the latter are not scientific units. Be careful to distinguish "Mb" (Megabit) and "MB" (Megabytes), in particular "kb" (1,000 bits) and "KB" (1,024 bytes).
41. It's always kHz (lower-case k), not KHz or KHZ. Units and Measurements, Taligent style guide
42. Use "ms", not "msec", for milliseconds.
43. Use "0.5" instead of ".5", i.e., do not omit the zero in front of the decimal point. (*Words into Type* recommends that "for quantities less than one, a zero should be set before the decimal point except for quantities that never exceed one.")
44. Avoid "etc."; use "for example", "such as", "among others" or, better yet, try to give a complete list (unless citing, for example, a list of products known to be incomplete), even if abstract. See also Strunk and White:

    Etc.: Not to be used of persons. Equivalent to **and the rest, and so forth**, and hence not to be used if one of these would be insufficient, that is, if the reader would be left in doubt as to any important particulars. Least open to objection when it represents the last terms of a list already given in full, or immaterial words at the end of a quotation. At the end of a list introduced by **such as**, **for example**, or any similar expression, etc. is incorrect.

45. If you say, "for example" or "like", do not follow this with "etc.". Thus, it's "fruit like apples, bananas and oranges". The "like" and "for example" already indicate that there are more such items.

46. Avoid bulleted lists of one-sentence paragraphs. They make your paper look like a slide presentation and interfere with smooth reading.
47. Avoid excessive use of "i.e.". Vary your expression: "such as", "this means that", "because", ... "I.e." is not the universal conjunction!
48. Remember that "i.e." and "e.g." are *always* followed by a comma.
49. Do not use ampersands (&) or slash-abbreviations (such as s/w or h/w) in formal writing; they are acceptable for slides.
50. "respectively" is preceded by a comma, as in "The light bulbs lasted 10 and 100 days, respectively."
51. "Therefore" and "thus" are usually followed by a comma, as in "Therefore, our idea should not be implemented."
52. Never use "related work**s**" unless you are talking about works of art. It's "related work".
53. Similarly, "code**s**" refer to encryption keys, not multiple programs. You would say "I modified multiple programs", not "multiple codes".
54. Use "in Figure 1" instead of "following figure" since figures may get moved during the publication or typesetting process. Don't assume that the LaTeX figure stays where you put it.
55. Text columns in tables are left-aligned, numeric columns are aligned on the decimal or right-aligned.
56. Section, Figure and Table are capitalized, as in "As discussed in Section 3". Figure can be abbreviated as Fig., but the others are not usually abbreviated, but that's a matter of taste - just be consistent.
57. Section titles are *not* followed by a period.
58. In LaTeX, tie the figure number to the reference, so that it doesn't get broken across two lines:
59. `Fig.~\ref{fig:arch}`
60. Do not use GIF images for figures, as GIFs produce horrible print quality and are huge. Export into PostScript. At that stage, you'll learn to "appreciate" Microsoft products. `xfig` and `gnuplot` generally produce PostScript that can be included without difficulties.
61. Only use line graphs when you are trying to show a functional or causal relationship between variables. When showing different experiments, for example, use bar graphs or scatter plots.
62. Figures show, depict, indicate, illustrate. Avoid "(refer to Fig. 17)". Often, it is enough to simply put the figure reference in parenthesis: "Packet droppers (Fig. 17) have a pipe to the bit bucket, which is emptied every night."
63. If you quote something literally, enclose it in quotation marks or show it indented and in smaller type ("block quote"). A mere citation is not sufficient as it does not tell the reader whether you simply derived your material from the cited source or copied it verbatim.
64. Technical report citations must have the name of the organization such as the university or company. Conferences must cite the location.
65. Do not refer to colors in graphs. Most people will print the paper on a monochrome (black and white) printer and will have no idea what you are talking about. Make sure that graph lines are easily distinguishable when printing on a monochrome printer.
66. Do not forget to acknowledge your funding support. If you do forget, you may not have any to acknowledge in the future.

67. Check your references to make sure they are up to date. For example, Internet Drafts might have been replaced by RFCs and technical reports or workshop papers by conference or journal papers.