Contributions to Mechanisms for Adaptive Use of Mobile Network Resources — Errata

Olivier Mehani*

July 25, 2012

Contents

1 Introduction

It never fails. As soon as something is published, I start find blatant mistakes and errors in there that completely escaped me before. This is of course true for my Ph.D thesis [1] too. This document is an attempt to set the record straight, at least for all the mistakes that have been identified so far.

2 Chapter 2

Footnote 12, page 44 "powers of ten are used for bytes, while powers of two are used for bits" Corrected in the text

There, this confused even me, though I was aware of the problem. This is the other way round:

- powers of *two* for bytes; when Iperf prints "41.8 GBytes," it means "41.8 GiB;"
- powers of *ten* for bits; when Iperf prints "35.9 Gbits," this is exactly what it means, "35.9 Gb."

The code for this conversion, in `src/stdio.c` is actually quite confusing, and reported in Listing 1 for easier perusal.

To avoid any further confusion, a patch to implement IEC 80000-13:2008 unit multiples for powers of two [3] in Iperf is now available at http://git.mytestbed.net/?p=iperf.git;a=commitdiff;h=9f23526f641fc84ee7b2cac537e48b9fc15559bc.

*olivier@mehani.name

3 Chapter 3

Table 3.1, page 59 "Fitting with only P_{wa} yields much smaller and less varying RSEs than for $P_{wd,u}$"

While this statement is true, this does not give much guarantee on the absolute value of the fitting. Trying to fit the same data on *both* P_{wa} and P_{wd} gives seemingly better resuts. However, some other hidden factors seem to also interact here (the Wi-Fi consumption is found to be decreasing with the throughput, which is counter intuitive and should be re-validated), and the data from [2] unfortunately does not allow us to discriminate the cases. Heavily culling the test cases seems to improve the fit, but reduces it's generality.

In short: do not use data from this table.

4 Chapter 4

Footnote 1, page 77 "a more CPU efficient method based on a Newtown search" Corrected in the text

This is of course a Newton search. Newtown here is a suburb of Sydney.[1]

5 Conclusion

These are all the errors I found in the document so far. Small ones such as typos have already been corrected in the text, as indicated next to the erroneous statement. However, bigger mistakes were left verbatim in order no to change the contents too much, and this errata must be taken into account.

Please let me know if you find any other mistake, I'd be glad to address them here.

References

[1] O. Mehani. "Contributions to Mechanisms for Adaptive Use of Mobile Network Resources". PhD thesis. Mines ParisTech / University of New South Wales, Dec. 2011. URL: http://olivier.mehani.name/publications/2011mehani_adaptive_use_mobile_networks_phd.pdf.

[2] H. Petander. "Energy-aware Network Selection Using Traffic Estimation". In: *MICNET 2009, 1st ACM workshop on Mobile Internet through Cellular Networks*. Ed. by S. Lu and H. Li. ACM SIGMOBILE. Beijing, China: ACM, Sept. 2009, pp. 55–60. ISBN: 978-1-60558-753-0. DOI: 10.1145/1614255.1614268. URL: http://www.nicta.com.au/research/research_publications/show?id=2159.

[3] *Quantities and Units — Part 13: Information Science and Technology*. ISO/IEC 80000-13:2008. ISO/TC12 WG12, IEC/TC25, Apr. 2008. URL: http://www.iso.org/iso/catalogue_detail?csnumber=31898.

[1] https://en.wikipedia.org/wiki/Newtown,_New_South_Wales

Listing 1: Iperf 2.0.5's code to convert numbers to multiples units of bytes and bits (from `src/stdio.c`). The former (upper case multiples) are divided by 1024s, while the latter (lower case) are by 1000s.

```
/* ------------------------------------------------------------------- 
 * byte_snprintf
 *
 * Given a number in bytes and a format, converts the number and
 * prints it out with a bits or bytes label.
 *   B, K, M, G, A for Byte, Kbyte, Mbyte, Gbyte, adaptive byte
 *   b, k, m, g, a for bit,  Kbit,  Mbit,  Gbit,  adaptive bit
 * adaptive picks the "best" one based on the number.
 * outString should be at least 11 chars long
 * (4 digits + space + 5 chars max + null)
 * ------------------------------------------------------------------- */

void byte_snprintf( char* outString, int inLen,
                    double inNum, char inFormat ) {
    int conv;
    const char* suffix;
    const char* format;

    /* convert to bits for [bkmga] */
    if ( ! isupper( (int)inFormat ) ) {
        inNum *= 8;
    }

    switch ( toupper(inFormat) ) {
        case 'B': conv = kConv_Unit; break;
        case 'K': conv = kConv_Kilo; break;
        case 'M': conv = kConv_Mega; break;
        case 'G': conv = kConv_Giga; break;

        default:
        case 'A': {
                double tmpNum = inNum;
                conv = kConv_Unit;

                if ( isupper((int)inFormat) ) {
                    while ( tmpNum >= 1024.0  &&  conv <= kConv_Giga ) {
                        tmpNum /= 1024.0;
                        conv++;
                    }
                } else {
                    while ( tmpNum >= 1000.0  &&  conv <= kConv_Giga ) {
                        tmpNum /= 1000.0;
                        conv++;
                    }
                }
                break;
            }
    }

    if ( ! isupper ((int)inFormat) ) {
        inNum *= kConversionForBits[ conv ];
        suffix = kLabel_bit[conv];
    } else {
        inNum *= kConversion [conv];
        suffix = kLabel_Byte[ conv ];
    }

    /* print such that we always fit in 4 places */
    if ( inNum < 9.995 ) {          /* 9.995 would be rounded to 10.0 */
        format = "%4.2f %s";        /* #.## */
    } else if ( inNum < 99.95 ) {   /* 99.95 would be rounded to 100 */
        format = "%4.1f %s";        /* ##.# */
    } else if ( inNum < 999.5 ) {   /* 999.5 would be rounded to 1000 */
        format = "%4.0f %s";        /*  ### */
    } else {                        /* 1000-1024 fits in 4 places
                                     * If not using Adaptive sizes then
                                     * this code will not control spaces*/
        format = "%4.0f %s";        /* #### */
    }
    snprintf( outString, inLen, format, inNum, suffix );
} /* end byte_snprintf */
```