Project	**IEEE 802.16 Broadband Wireless Access Working Group**
Title	**Session #6 MAC and PHY Proposal Evaluation Scoring Results: Summary**
Date Submitted	**2000-03-08**
Source	Brian Petry 3Com 12230 World Trade Dr. San Diego, CA 92128 Voice: 858-674-8533 Fax: E-mail: brian_petry@3com.com
Re:	These results fulfill part of the "Development Plan for 802.16.1 Air Interface Standard" 802.16-99/05. The results are output from 802.16 session #5. They are the output from the session #5 score compilation committee. At session #6, 802.16 observers were also invited to submit score cards. The observer results are also contained in this document.
Abstract	This document contains *summary* statistics and scoring results. 802.16 members also have access to a report which contains all voters' score cards.
Purpose	802.16 members and observers should use this report to consider further improvements, mergers, etc. to the proposals.
Notice	This document has been prepared to assist the IEEE 802.16. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.
Release	The contributor acknowledges and accepts that this contribution may be made public by 802.16.
IEEE Patent Policy	The contributor is familiar with the IEEE Patent Policy, which is set forth in the IEEE-SA Standards Board Bylaws <http://standards.ieee.org/guides/bylaws> and includes the statement: "IEEE standards may include the known use of patent(s), including patent applications, if there is technical justification in the opinion of the standards-developing committee and provided the IEEE receives assurance from the patent holder that it will license applicants under reasonable terms and conditions for the purpose of implementing the standard."

Session #6 MAC and PHY Proposal Evaluation Scoring Results: Summary

Brian Petry
3Com

1 Introduction

Four proposals, 2 MAC an d 2 PHY, were scored in many different categories. The "evaluation criteria" for session #6 is for a proposal to achieve a average score of 7.0 or better in any category. A voter could abstain in any category of any proposal. Additionally, an "overall score" category was scored, which was not included in the evaluation criteria.

2 Mac Results

Both MAC proposals met the evaluation criteria. 26 ballots were processed.

2.1 References

For help interpreting these results, please refer to these proposals and evaluation categories.

Proposals:
802.16mc-00/09 http://grouper.ieee.org/groups/802/16/mac/contrib/802161mc-00_09.pdf: *Media Access Control Protocol Based on DOCSIS 1.1* (Glen Sater, Arun Arunachalam, George Stamatelos, Farid Elwailly, Jeff Foerster, Jung Yee, Scott Marin, Bill Myers, Leland Langston, Wayne Hunter, Phil Guillemette, Chet Shirali, Karl Stambaugh, George Fishel, Ray Sanders, Moshe Ran, Andrew Sundelin)

802.16mc-00/10 http://grouper.ieee.org/groups/802/16/mac/contrib/802161mc-00_10.pdf: *MAC Proposal for IEEE 802.16.1* (James F. Mollenauer, Ken Standwood, Jay Klein, Brian Petry, Carl Eklund, Juha Pihlaja, Kari Rintanen, Leonid Shousterman, Paolo Baldo)

Evaluation Categories (from 802.16m-99/03):

1 Meets system requirements

How well does the proposed MAC protocol meet the requirements described in the current version of the 802.16.1 Functional Requirements? (See Document IEEE 802.16s-99/00)

2 Mean access delay and variance

1. How effective are the mechanisms presented in controlling the delay and variance?

2. Does it seem possible for an operator to offer a bounded delay for a prescribed offered load?

3 Payload and bandwidth efficiency

1. How well does the overhead due to the proposed MAC PDU headers allow for efficient user data transfer over the air interface?
2. Is the proposed MAC protocol designed such that the MAC signaling is efficient in terms of not requiring excessive overhead?
3. How well does the proposed MAC protocol provide the mechanisms

for fair allocation and sharing of the bandwidth among users?

(Please include payload example.)

4 Simplicity of implementation/low complexity

How well does the proposed MAC protocol allow for an implementation that is simple and generic enough that it is likely to be accepted by industry?

5 Scalability

Does the MAC protocol support a broad range of operational bandwidths and number of connections across all services?

6 Service Support Flexibility

1. How completely does the MAC protocol support the services mentioned in the 802.16.1 Functional Requirements?
2. How well does the MAC protocol support additional services?

7 Robustness

1. Is the MAC protocol able to recover from events such as unexpected shut down or loss of link?
2. How well does the MAC Layer react in the face of errors arising from the Physical Layer?

8 Security

How well does the MAC protocol provide security mechanisms to meet the 802.16.1 Functional Requirements?

9 Maturity

Does the proposed MAC protocol have data to demonstrate its ability to operate in an actual system that is representative of the BWA networks targeted for 802.16.1?

10 Sign-on process

1. How well does the MAC protocol resolve initial two way ranging?
2. How automatic is the sign-on process?

11 Adequacy of management functions

How well does the MAC protocol provide link management functions for subscribers' timing, power, and frequency?

12 Convergence with existing protocols

How simple is it to adapt the proposed

MAC protocol to well-known LAN and WAN protocols?

13 Ability to work with physical layer variations, e.g., duplexing, constellation, etc.
How independent is the proposed MAC protocol of the PHY protocol?

2.2 802.16 Members

```
Miniumum Scores
                  1    2    3    4    5    6    7    8    9   10   11   12   13   Avg
09: Sater     : 3.0  3.0  3.0  3.0  2.0  2.0  3.0  3.0  3.0  3.0  3.0  3.0  1.0  2.7
10: Mollenauer: 1.0  1.0  1.0  0.0  1.0  0.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.5

Maximum Scores
                  1    2    3    4    5    6    7    8    9   10   11   12   13   Avg
09: Sater     :10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10: Mollenauer:10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

Standard Deviation
                  1    2    3    4    5    6    7    8    9   10   11   12   13   Avg
09: Sater     : 2.0  2.5  2.5  2.3  2.6  2.7  2.5  2.1  2.3  2.3  2.5  2.3  2.7  2.4
10: Mollenauer: 2.6  3.0  2.9  3.1  2.9  3.1  3.4  2.5  3.2  3.4  3.4  3.1  3.0  3.0

Average Scores
                  1    2    3    4    5    6    7    8    9   10   11   12   13   Avg
09: Sater     : 7.9  7.4  7.3  7.7  7.4  7.3  7.6  8.3  7.8  7.9  7.7  7.9  7.3  7.7
10: Mollenauer: 7.3  6.7  7.1  6.2  6.7  6.9  6.3  7.1  5.9  6.6  6.5  6.6  6.9  6.7

Average of the "overall" category
09: Sater     :  7.9
10: Mollenauer:  6.6
```

2.3 802.16 Observers

No ballots received

3 PHY Results

Both PHY proposals met the evaluation criteria. 26 ballots were processed.

3.1 References

For help interpreting these results, please refer to these proposals and evaluation criteria.

Proposals:
802.16.1pc-00/14 (2000-02-25) *PHY layer proposal for BWA* Jay Klein, Lars Lindh, Carl Eklund, Petri Bergholm, Naftali Chayat

802.16.1pc-00/13 (2000-02-25) *Physical Layer Proposal for the 802.16 Air Interface Specification* Jeff Foerster, Arun V. Arunachalam, George Stamatelos, Farid Elwailly, Jung Yee, Phil Guillemette, Moshe Ran, Wayne Hunter, Leland Langston, William Myers, Scott Marin, George Fishel, Ray W. Sanders, Karl Stambaugh, Glen Sater, Chet Shirali

Evaluation Criteria (from 802.16p-99/03):

1 Meets system requirements

How well does the proposed MAC protocol meet the requirements described in the current version of the 802.16.1 Functional Requirements? (See Document IEEE 802.16s-99/00)

2 Spectrum efficiency

Defined in terms of single sector capacity assuming all available spectrum is being utilized (either in terms of Gbps/Available Spectrum or in terms of Mbps/MHz)

3 Simplicity of implementation

How well does the proposed PHY allow for simple implementation or how does it leverage on existing technologies?

4 CPE cost optimization

How does the proposed PHY affect CPE cost?

5 Spectrum resource flexibility

Flexibility in the use of the frequency band (i.e., minimum frequency band required to operate and migration capabilities)

6 System diversity flexibility

How flexible is the proposed PHY to any other system variations and future technology improvements or new services?

7 Protocol Interfacing complexity

Interaction with other layers of the protocol, specifically MAC and NMS

8 Implication on other network interfaces

Intrinsic transport efficiency of telecomm and datacomm services

9 Reference system gain*

Sector coverage performance for a typical BWA deployment scenario (supply, reference system gain)

*In order to compare between PHY proposals, we define the reference system gain (RSG) as the output power of the transmitter minus the receiver threshold at a given working point, including back-off required for proper transmission. We will assume a 0 dBW transmitter (prior to back-off), and an ideal LNA (0 dB NF). Include BER working points of both 10-6 and 10-10 (post-coding).

10 Robustness to interference

Resistance to intra-system interference (i.e., frequency re-use) and external interference cause by other systems

11 Robustness to channel impairments

3.2 802.16 Members

```
Miniumum Scores
                  1    2    3    4    5    6    7    8    9   10   11   Avg
13: Foerster  : 2.0  2.0  3.0  1.0  1.0  2.0  1.0  3.0  2.0  2.0  3.0  2.0
14: Klein     : 1.0  1.0  0.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  0.9

Maximum Scores
                  1    2    3    4    5    6    7    8    9   10   11   Avg
13: Foerster  :10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
14: Klein     :10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0

Standard Deviation
                  1    2    3    4    5    6    7    8    9   10   11   Avg
13: Foerster  : 2.5  2.5  2.5  2.6  2.5  2.8  2.7  2.4  2.6  2.6  2.5  2.6
14: Klein     : 2.4  3.0  3.2  3.1  2.6  3.0  2.6  2.5  2.6  2.8  2.9  2.8

Average Scores
                  1    2    3    4    5    6    7    8    9   10   11   Avg
13: Foerster  : 7.6  7.5  8.0  7.6  7.3  7.2  7.6  7.7  7.7  7.4  7.4  7.5
14: Klein     : 7.5  7.1  5.9  6.9  7.7  7.1  7.1  7.5  7.1  7.1  7.0  7.1

Average of the "overall" category
13: Foerster  :  7.8
14: Klein     :  6.6
```

3.3 802.16 Observers

No ballots received

4 Appendix: Score Compilation Program Code

The following is the program used to process voters' score cards, calculate statics and display results. It was written by Brian Petry and is included here with a "public domain" copyright (which is a slightly-modified version of the well-known University of California copyright used for its variants of UNIX):

```
#!/usr/local/bin/perl

# Copyright (c) 2000. 3Com Corporation.  All Rights Reserved.
#
# Redistribution and use in source form only, with or without modification,
# is permitted provided that the following conditions are met:
#
# 1. Redistribution of the code must retain the above copyright notice, this
#    list of conditions and the following disclaimer.
# 2. All advertising materials mentioning features or use of this software
#    must display the following acknowledgement:  This product includes
#    software developed by 3Com Corporation.
# 3. The name, 3Com Corporation, may not be used to endorse or promote
#    products derived from this software without specific prior written
#    permission.
#
# This software is provided by 3Com "as is" and any express or implied
# warranties, including, but not limited to, the implied warranties of
# merchantability and fitness for a particular purpose are disclaimed.
# In no event shall 3Com be liable for any direct, indirect, incidental,
# special, exemplary, or consequential damages (including, but not limited
# to, procurment of substitute goods or services; loss of use, data, or
# profits; or business interruption) however caused and on any theory of
# liability, whether in contract, strict liability, or tort (including
# negligence or otherwise) arising in any way out of the use of this software,
```

```
# even if advised of the possibility of such damage.
############

#
# Process a bunch of csv (comma separated version) files that are output from
# Micro$oft Excel spreadsheets (saveas->csv).  Each spreadsheet is a scorecard
# Each row is a proposal and each column is a category.  Each csv file is
# specified on the command line.
#
# This script handles two different spreadsheets: a PHY and a MAC.  They are
# different in the number of proposals and score categories.  But the script
# must be invoked on a batch of on type of spreadsheets (you can't mix PHY and
# MAC spreadsheets in one run).
#
# An "extra" category that is not averaged in with the others is an "overall
# score."
#
# The spreadsheet also allows abstentions in any proposal/category cell, which
# is identified by a "-1".  These abstention scores are not included in
# statistics.
#

#
# usage:
# perl score.pl file1.csv file2.csv ...
#

# some constants
$ignorehead = 6;         # number of lines to ignore in the header
$nmaccategories = 13;    # number of MAC score categories
$nphycategories = 11;    # number of PHY score categories
$nmacprops = 2;          # number of MAC proposals
$nphyprops = 2;          # number of PHY proposals
$pass = 7;               # avg score needed to pass

# declare some arrays
my @avgscores;           # average scores
my @sumscores;           # summed scores (used for average and std. deviation
my @minscores;           # minimum scores
my @maxscores;           # maximum scores
my @devscores;           # standard deviation

# print a header: categories
sub header {
   print "              ";
   for ($cat = 0; $cat < $ncategories; $cat++) {
      printf "%4d ", $cat+1;
   }
   if (scalar(@_) > 0) {
      if (@_[0] == 1) {
         print "  Avg";
      } elsif (@_[0] == 2) {
         print " Ovral";
      }
   }
   print "\n";
}

if (scalar(@ARGV) == 0) {
   print "usage: perl score.pl file1 file2 ...\n";
   exit;
}

# initialize minimum scores
for ($prop = 0; $prop < 99; $prop++) {
   for ($cat = 0; $cat < 99; $cat++) {
      $minscores[$prop][$cat] = 99;
   }
}

# initialize maximum scores
for ($prop = 0; $prop < 99; $prop++) {
   for ($cat = 0; $cat < 99; $cat++) {
      $maxscores[$prop][$cat] = -1;
   }
}
```

```
# For each csv file specified on the command line
$voters = 0;             # accumulator: count of voters (csv files)
$phy = 0;
foreach $vfilename (@ARGV) {

   $voters++;

   open(vfile, $vfilename) or die "Can't open file $vfilename: $!\n";

   # ignore first $ignorehead lines of file
   $x = 0;
   while ($x < $ignorehead) {
      $_ = <vfile>;
      if (/PHY/) {
         $phy = 1;
      }
      $x++;
   }

   # if we are processing PHY score cards
   if ($phy) {
       $nprops = $nphyprops;
       $ncategories = $nphycategories;
   } else {
       $nprops = $nmacprops;
       $ncategories = $nmaccategories;
   }

   # read in the scores
   for ($prop = 0; $prop < $nprops; $prop++) {
      @row = split /,/, <vfile>; # convert input row to an array

      # delete the proposal name and save it
      @cname[ $prop] = shift @row;

      # delete any extra elements after saving the overall score
      if ($phy) {
          @overall[ $prop] = @row[ $ncategories+2];
      } else {
          @overall[ $prop] = @row[ $ncategories];
      }
       while (scalar(@row) > $ncategories) {
          pop @row;
      }

      @scores[ $prop] = [ @row ]; # store the row in a 2-dimensional array

      # for each category for a proposal, store the scores in arrays and
      # accumulate statistics
      for ($cat = 0; $cat < $ncategories; $cat++) {
         # save minimum score
         $thisscore = $scores[ $prop][ $cat];

         # if a score is negative (e.g., -1), it is an abstention
         if ($thisscore < 0) {
            $thisscore = -1;
         }

         # validate that score is within range
         if ($thisscore > 10) {
            $thisscore = 10;
            $scores[ $prop][ $cat] = 10;
         }

         # save minimum score (check abstention)
         if ($thisscore != -1 && $thisscore < $minscores[ $prop][ $cat]) {
             $minscores[ $prop][ $cat] = $thisscore;
         }

         # save maximum score (check abstention)
         if ($thisscore > $maxscores[ $prop][ $cat]) {
             $maxscores[ $prop][ $cat] = $thisscore;
         }

         # if score is not an abstention, accumulate statistics
         if ($thisscore != -1) {
             # count non-abstentions
```

```
            $numscores[ $prop][ $cat] ++;

            # accumulate total for mean and std dev
            $sumscores[ $prop][ $cat]  += $thisscore;

            # accumulate sum-of-squares for standard deviation
            $devscores[ $prop][ $cat]  += $thisscore * $thisscore;
         }
      }

      # Accumulate average of the overall score
      if (@overall[ $prop]  != -1) {
          $sumoverall[ $prop]  += @overall[ $prop];
          $numoverall[ $prop] ++;
      }
   }

   # find the voter's name
   while (<vfile>) {
      if (/[ Ff] amily/) {
          $family = (split /,/)[ 0];
      }
      if (/[ Gg] iven/) {
          $given = (split /,/)[ 0];
      }
   }

   # print, in abbreviated form, the voter's scorecard
   print "$given $family: \n";
   header 2;
   for ($prop = 0; $prop < $nprops; $prop++) {
      $avg = 0;
      $num = 0;
      printf "%-14s:", @cname[ $prop];
      for ($cat = 0; $cat < $ncategories; $cat++) {

         # if abstention
         if ($scores[ $prop][ $cat]  == -1) {
            print " ABST";
         } else {
             $avg += $scores[ $prop][ $cat];
            $num++;
             printf "%4.1f ", $scores[ $prop][ $cat];
         }
      }

      # print the overall score
      if ($overall[ $prop]  == -1) {
         print " ABST";
      } else {
          printf "%4.1f ", $overall[ $prop];
      }

      #printf "%4.1f\n", $avg/$num;
      print "\n";
   }
   print "\n";

   close(vfile);
}

# calculate the average scores (needed for standard deviation also)
for ($prop = 0; $prop < $nprops; $prop++) {
   for ($cat = 0; $cat < $ncategories; $cat++) {
      # if everyone abstained in this category of this proposal
      if ($numscores[ $prop][ $cat]  == 0) {
          $avgscores[ $prop][ $cat]  = -1;
      } else {
          $avgscores[ $prop][ $cat]  = $sumscores[ $prop][ $cat] /$numscores[ $prop][ $cat];
      }
   }
}

print "--------------------------RESULTS---------------------------------\n\n";

# print the minimum scores
print "$voters ballots processed\n\n";
```

```
print "Miniumum Scores\n";
header 1;
for ($prop = 0; $prop < $nprops; $prop++) {
    printf "%-14s:", @cname[ $prop];
    $avg = 0;
    $cnt = 0;
    for ($cat = 0; $cat < $ncategories; $cat++) {
        if ($minscores[ $prop][ $cat]  != 99) {
            $avg += $minscores[ $prop][ $cat];
            printf "%4.1f ", $minscores[ $prop][ $cat];
           $cnt++;
       } else {
           print "ABST ";
       }
    }
    printf "%4.1f\n", $avg/$cnt;
}

# print the maximum scores
print "\nMaximum Scores\n";
header 1;
for ($prop = 0; $prop < $nprops; $prop++) {
    printf "%-14s:", @cname[ $prop];
    $avg = 0;
    $cnt = 0;
    for ($cat = 0; $cat < $ncategories; $cat++) {
        if ($maxscores[ $prop][ $cat]  != -1) {
            $avg += $maxscores[ $prop][ $cat];
            printf "%4.1f ", $maxscores[ $prop][ $cat];
           $cnt++;
       } else {
           printf "ABST ";
       }
    }
    printf "%4.1f\n", $avg/$cnt;
}

# print the standard deviation
print "\nStandard Deviation\n";
header 1;
for ($prop = 0; $prop < $nprops; $prop++) {
    printf "%-14s:", @cname[ $prop];
    $avg = 0;
    $cnt = 0;
    for ($cat = 0; $cat < $ncategories; $cat++) {

        # If only one score in a category, standard deviation is invalid (use 0)
        if ($numscores[ $prop][ $cat]  <= 1) {
           $d = 0;
           print " N/A ";
       } else {
            $d = sqrt(($devscores[ $prop][ $cat]  - $avgscores[ $prop][ $cat]  * $sumscores[ $prop][ $cat] ) /
($numscores[ $prop][ $cat]  - 1));
           $cnt++;
           $avg += $d;
           printf "%4.1f ", $d;
       }
    }

    printf "%4.1f\n", $avg/$cnt;
}

# print the averaged results
print "\nAverage Scores\n";
header(1);
for ($prop = 0; $prop < $nprops; $prop++) {
    $sum = 0;
    $cnt = 0;
    @passing[ $prop]  = 0;
    printf "%-14s:", @cname[ $prop];
    for ($cat = 0; $cat < $ncategories; $cat++) {
        $score = $avgscores[ $prop][ $cat];
        if ($score >= $pass) {
            @passing[ $prop]  = 1;
       }
        if ($score == -1) {
           print " ABST";
```

```
      } else {
          printf "%4.1f ", $score;
          $sum += $score;
          $cnt++;
      }
   }
   @avg[ $prop] = $sum/$cnt;
   printf "%4.1f\n", @avg[ $prop];    # print the average of all categories
}

# print the average of the "overall" category
print "\nAverage of the \"overall\" category\n";
for ($prop = 0; $prop < $nprops; $prop++) {
    printf "%-14s: %4.1f\n", @cname[ $prop], @sumoverall[ $prop]/@numoverall[ $prop];
}

# print proposals, sorted by rank, high score first
# create a list of proposal indices
#for ($prop = 0; $prop < $nprops; $prop++) { push @plist, $prop; }
#sub decreasing {
#   @avg[ $b] <=> @avg[ $a];
#}
#
#@sorted = sort decreasing @plist;
#foreach $prop (@sorted) {
#   printf "%-14s:%4.1f\n", @cname[ $prop], @avg[ $prop];
#}

# print the proposals that pass
print "\nProposals meeting the criteria:\n";
if (scalar(@passing) == 0) {
   print "None\n";
} else {
   for ($prop = 0; $prop < $nprops; $prop++) {
      if (@passing[ $prop]) {
          print "@cname[ $prop]\n";
      }
      else {
          $failing .= "@cname[ $prop]\n";
      }
   }
}

#print the proposals that failed
print "\nProposals not meeting the criteria:\n";
if ($failing eq "") {
   print "None\n";
} else {
   print $failing;
}
```