of the amount of required USF funding. While D&P have proposed some model changes, they have failed to address others that are needed to arrive at an accurate assessment of the required funding.

Q. DOES THIS CONCLUDE YOUR TESTIMONY?

A. Yes it does.

JWS-One
Modifications to Tech.Pas
Replaces Lines ~255-269

In these code revisions, added code is indicated with an underline. Removed code is indicated with strikethrough fonts. The addition of a new procedure makes maintaining line numbers difficult. Wherever new code is added, existing code is maintained to simplify searching for the relevant section.

An electronic version of these changes is available if requested.

```
var
    . . .
    af              : double;
    b26             : double;
    cost            : array[copper26..fiber] of double;
    cmin            : double;
    techmin         : techtype;
    t               : techtype;

begin

. . .

          { For sizing cross connect box, we need cable size on both
feeder and distribution side. }

          {for n := 1 to NumXCBoxSizes do
          if l26  >= IntfeCost[n]^.NumLines
          then d26 := IntfeCost[n]^.NumLines; { D-side cable size }


          {f26 :=
SA_array^[i]^.ResLines/fill_factor_fn(SA_array^[i]^.density,0) +
                (SA_array^[i]^.BusLines   (one
PairsPerT1System/ChannelsPerT1System)*SA_array^[i]^.SwitchedDS1 +
                SA_array^[i]^.SpelAccessLines   (one
PairsPerT1System/ChannelsPerT1System)*SA_array^[i]^.SpelAccessDS1)/
                fill_factor_fn(SA_array^[i]^.density,0);

          for n := 1 to NumXCBoxSizes do
          if f26  >= IntfeCost[n]^.NumLines
          then f26 := IntfeCost[n]^.NumLines; { F-side cable size }


          tmp3 := zero;
          for n := 1 to NumXCBoxSizes do

          tmp2 :=

t1 terminal cost fn(SA array^[i]^.lines/FillFactor,n96,n24);
          tmp2 := tmp2*ac t1 term;
```

```
          { For sizing cross-connect box, we need cable size on both
feeder and distribution side. }

          if l26 > IntfcCost[NumXCBoxSizes]^.NumLines then
RunError(201);
          d26 := IntfcCost[1]^.NumLines;
          for n := 1 to NumXCBoxSizes-1 do
          if (l26 > IntfcCost[n]^.NumLines) and (l26 <=
IntfcCost[n+1]^.NumLines)
          then d26 := IntfcCost[n+1]^.NumLines; { D-side cable size }

          b26 :=
SA array^[i]^.ResLines/fill factor fn(SA array^[i]^.density,0) +
                (SA array^[i]^.BusLines - (one -
PairsPerT1System/ChannelsPerT1System)*SA array^[i]^.SwitchedDS1 +
                 SA array^[i]^.SpclAccessLines - (one -
PairsPerT1System/ChannelsPerT1System)*SA array^[i]^.SpclAccessDS1)/
                 fill factor fn(SA array^[i]^.density,0);

          if b26 > IntfcCost[NumXCBoxSizes]^.NumLines then
RunError(201);
          f26 := IntfcCost[1]^.NumLines;
          for n := 1 to NumXCBoxSizes-1 do
          if (b26 > IntfcCost[n]^.NumLines) and (b26 <=
IntfcCost[n+1]^.NumLines)
          then f26 := IntfcCost[n+1]^.NumLines; { F-side cable size }


if (d26+f26)  >= IntfcCost[n]^.NumLines
            { match on interface size large enough to handle both
cables, if not big enough, assign largest xcbox }

        if (d26+f26) > IntfcCost[NumXCBoxSizes]^.NumLines then
          begin
                tmp3 := IntfcCost[NumXCBoxSizes]^.NumLines;
          end
          else
          begin
                tmp3 := IntfcCost[1]^.NumLines;
                for n := 1 to NumXCBoxSizes-1 do
                    if ((d26+f26) > IntfcCost[n]^.NumLines) and
((d26+f26) <= IntfcCost[n+1]^.NumLines)

                    then tmp3 := IntfcCost[n+1]^.cost;

                    {modified then clause in accord with Pitkin
testimony}
          end;
```

JWS-Two
Modifications to Terminal.Pas
Replaces Lines ~260-269

In these code revisions, added code is indicated with an underline. Removed code is indicated with strikethrough fonts. The addition of a new procedure makes maintaining line numbers difficult. Wherever new code is added, existing code is maintained to simplify searching for the relevant section.

An electronic version of these changes is available if requested.

```
function drop_terminal_cost_fn(
                                lines                   : double;
                                density                 : double;
                                pct_ugd                 : double;
                                pct_bur                 : double;
                                pct_aer                 : double
                               ) : double;
var       i : integer;
          temp     : double;
begin
     temp     := zero;

     if lines < 1.0e-6 then drop_terminal_cost_fn := zero else

     if lines > 25 then lines:=lines*2;
     {Code added to reflect that sizing a Building Terminal requires
using
     both the F&D side demand.  As written in tech.pas}


     temp := zero;
     {for i := 1 to NumDropTerminalSizes do
     if lines >= DropTermCost[i]^.size then
     begin
          temp := pct_ugd*DropTermCost[i+1]^.CostUgd +
                  pct_bur*DropTermCost[i+1]^.CostBur +
                  pct_aer*DropTermCost[i+1]^.CostAer;


end;


     if (lines > DropTermCost[NumDropTerminalSizes]^.size) then
     begin
          temp := pct_ugd*DropTermCost[NumDropTerminalSizes]^.CostUgd +
                  pct_bur*DropTermCost[NumDropTerminalSizes]^.CostBur +
                  pct_aer*DropTermCost[NumDropTerminalSizes]^.CostAer;
     end
     else
     begin
          temp := pct_ugd*DropTermCost[1]^.CostUgd +
                  pct_bur*DropTermCost[1]^.CostBur +
                  pct_aer*DropTermCost[1]^.CostAer;
```

```
        for i := 1 to NumDropTerminalSizes-1 do
        if (lines > DropTermCost[i]^.size) and (lines <=
DropTermCost[i+1]^.size) then
        begin
            temp := pct_ugd*DropTermCost[i+1]^.CostUgd +
                    pct_bur*DropTermCost[i+1]^.CostBur +
                    pct_aer*DropTermCost[i+1]^.CostAer;

            {modified above assignment clause in accord with D&P}
end;
    end;
    {end;}


    drop_terminal_cost_fn := temp;
    END; { if lines >= 1 }
end;
```

JWS-3
Clusintf.Pas

In these code revisions, added code is indicated with an underline. Removed code is indicated with strikethrough fonts. The addition of a new procedure makes maintaining line numbers difficult. Wherever new code is added, existing code is maintained to simplify searching for the relevant section.

FCC code was used as base for this comparison. An electronic version of these changes is available if requested.

```
type  d4vector  = array[1..4] of double;
      i4vector  = array[1..4] of integer;
      string2   = string[2];
      string4   = string[4];
      string8   = string[8];
      string14  = string[14];
      string132 = string[132];
      s2500     = array[1..2500] of single; {ADD following Four lines}
      i2500     = array[1..2500] of integer;
      s2500 ptr = ^s2500;
      i2500 ptr = ^i2500;


var
    CLUfile1          : text;
    CLUfile2          : text;
    OUTfile           : text; {Add line}

    BINFile           : file of GridRecordType;
    do_batch          : boolean;
    areaname          : string8;
    CloseParam        : string2;
    dataline          : string132;
    ClusterNumber     : integer;
    CloseWindow       : boolean;
    SetRaster         : boolean;
    WeightDensity     : boolean;
    GlobalRasterSize  : double;
    CmdParam          : array[1..4] of string8;
    code              : integer;
    dens              : double;
    DataError         : boolean;
    TerrainDataError  : boolean;

    {Add three lines}
    GR RLines         : s2500 ptr;
    GR BLines         : s2500 ptr;
    GR Index          : i2500 ptr;


   {Following two lines commented out}
{    ResTrueUp         : double;}
```

```
{    BusTrueUp          : double;}

. . .

{***The following sort procedure was added to Clustintf.pas.  It is
analogous to one used elsewhere in the model.  Existing code is shown
at the beginning to indicate a possible insertion point.***}


99: dispose(rra);

END;
{Add new procedure}

PROCEDURE SortIndex(n: integer; rv: s2500 ptr; rn: i2500 ptr);
(*
        This routine was borrowed from Press, et al., Numerical
Recipes, 1st edition, 1986.
 *)
LABEL 99;
VAR
   l,j,ir,i: integer;
   rrv, rrv2, rrv3: single;
   rrn: integer;
BEGIN
   IF(n<2)Then Goto 99;    {BJB}
   l := (n DIV 2)+1;
   ir := n;
   WHILE true DO BEGIN
      IF (l > 1) THEN BEGIN
         l := l-1;
         rrn := rn^[l];
         rrv := Frac (rv^[rrn]);
      END ELSE BEGIN
         rrn := rn^[ir];
         rrv := Frac (rv^[rrn]);
         rn^[ir] := rn^[1];
         ir := ir-1;
         IF (ir = 1) THEN BEGIN
            rn^[1] := rrn;
            GOTO 99
         END
      END;
      i := l;
      j := l+1;
      IF rrv >= 0.5 THEN rrv := rrv - 0.5 ELSE rrv := rrv + 0.5;
      WHILE (j <= ir) DO BEGIN
         rrv2 := Frac (rv^[rn^[j]]);
         IF rrv2 >= 0.5 THEN rrv2 := rrv2 - 0.5 ELSE rrv2 := rrv2 +
0.5;
         IF (j < ir) THEN BEGIN
            rrv3 := Frac (rv^[rn^[j+1]]);
            IF rrv3 >= 0.5 THEN rrv3 := rrv3 - 0.5 ELSE rrv3 := rrv3 +
0.5;
            IF (rrv2 > rrv3) THEN BEGIN
               rrv2 := rrv3;
```

```
                j := j+1;
            END;
        END;
        IF (rrv > rrv2) THEN BEGIN
            rn^[i] := rn^[j];
            i := j;
            j := j+j;
        END ELSE j := ir+1;
    END;
    rn^[i] := rrn;
  END;
99:
END;

. . .

procedure rasterize( ClusterNumber : integer);

type d8000     = array[1..8000] of double;
     i8000     = array[1..8000] of integer;
     d8000_ptr = ^d8000;
     i8000_ptr = ^i8000;

const big : double=1.0e10;
      mil : double=0.0010000;
      fifty : double=50.0000;

var
     x                 : d8000_ptr;
     y                 : d8000_ptr;
     ResLines          : d8000_ptr;
     BusLines          : d8000_ptr;
     NumberOfPoints    : integer;
     i                 : integer;
     c                 : integer;
     code              : integer;
     vstr              : string14;
     xtry              : double;
     ytry              : double;
     numtry            : integer;
     m                 : double;
     llx               : double;
     lly               : double;
     urx               : double;
     ury               : double;
     data2             : string132;
     indata            : string132;
     j                 : integer;
     k                 : integer;
     RasterSize        : double;
     RasterTooLarge    : boolean;
     ResResidualLines  : double;
     BusResidualLines  : double;
     z                 : double;
     dResLines         : double;
     dBusinessLines    : double;
```

```
     bpop              : array[1..50,1..50] of boolean;
     rpop              : array[1..50,1..50] of boolean;
     n2                : integer;
     d1                : double;
     d2                : double;
     dr1               : double;
     nr1               : integer;
     R_Lines           : array[1..50,1..50] of single; {Added for D&P
Changes}
     B_Lines           : array[1..50,1..50] of single;

 {Added for D&P Changes}
     AllowDeletes      : boolean; {Add to correct true up}
     iResLines         : integer; {Add to correct true up}
     iBusLines         : integer; {Add to correct true up}
begin

(*
     Now calculate the number of residential primary lines, using logic
akin to that above.
*)
     GR^.PrimaryLines := round( dr1 ) + GR^.HousingUnits - nr1;

     writeln(OUTfile, '     Number of data points: ',NumberOfPoints);
     {Added above to support an auditing file}
     writeln('     Number of data points: ',NumberOfPoints);


GR^.gBusinessLines := round(dBusinessLines);
     GR^.gPrivateLines := 0;
     GR^.gSpecialLines := 0;
     writeln(OUTfile, '     ResLines, Business Lines:
',GR^.gResLines,',',GR^.gBusinessLines);
      {Added above to support an auditing file}
      writeln('     ResLines, Business Lines:
',GR^.gResLines,',',GR^.gBusinessLines);


     if ( GR^.HousingUnits>0 ) then GR^.lines_per_house :=
dResLines/GR^.HousingUnits else
                                   GR^.lines_per_house := one;
     if ( GR^.BusinessUnits>0 ) then GR^.lines_per_bus   :=
dBusinessLines/GR^.BusinessUnits else
                                    GR^.lines_per_bus := one;

     llx := llx-mil; {Added for D&P}
     lly := lly-mil; {Added for D&P}
     urx := urx+mil; {Added for D&P}
     ury := ury+mil; {Added for D&P}

     RasterSize := GlobalRasterSize;

{ if raster size is not given externally, or it is too large, calculate
it here. }
```

```
     RasterTooLarge := false;
     if ( max( (urx-llx), (ury-lly) )/RasterSize > fifty ) then
RasterTooLarge := true;

     if RasterTooLarge or (SetRaster = false) then
     begin
          RasterSize := max( (urx-llx), (ury-lly) )/fifty;
     end
     else
     begin
       RasterSize:=max((urx-llx),(ury-lly))/round(max((urx-llx),(ury-
lly))/Rastersize+half);


     end;
     {added above for D&P}

. . .

{***The following section of code is excerpted in entirety to place the
code changes in context with existing code.  Changes/additions are
underlined.***}

(*   These loops handle the actual rasterization. *)

     for i := 1 to GR^.nrow do
         for j := 1 to GR^.ncol do
         begin
              GR^.ResLines[i,j] := 0;
              GR^.BusLines[i,j]   := 0;
              bpop[i,j] := false;
              rpop[i,j] := false;
              R Lines[i,j] := zero;
              B Lines[i,j] := zero;

         end;

     for k := 1 to NumberOfPoints do
     begin
          i := round( abs(y^[k] - lly)/RasterSize + half );
          j := round( abs(x^[k] - llx)/RasterSize + half );


          GR^.ResLines[i,j] := GR^.ResLines[i,j] + round(ResLines^[k]);
          {remove the strikethrough line}
          R Lines[i,j] := R Lines[i,j] + ResLines^[k];
          {add underline line}
          if ResLines^[k] > zero then rpop[i,j] := true;
          GR^.BusLines[i,j]   := GR^.BusLines[i,j] +
round(BusLines^[k]);
          {remove the strikethrough line}
          B Lines[i,j] := B Lines[i,j] + BusLines^[k];
          {add underline line}
          if BusLines^[k] > zero then bpop[i,j] := true;
     end;
```

```
    writeln(OUTfile, '     Completed basic rasterization.');

    writeln('     Completed basic rasterization.');

(*  Now we need to handle residual lines caused by fractions of lines
in input data. *)

    ResResidualLines := dResLines;
    BusResidualLines := dBusinessLines;

    iResLines := 0;
    iBusLines := 0;
    for i := 1 to GR^.nrow do
        for j := 1 to GR^.ncol do
        begin
            GR^.ResLines[i,j] := round(R_Lines[i,j]);
            GR^.BusLines[i,j] := round(B_Lines[i,j]);

            ResResidualLines := ResResidualLines - GR^.ResLines[i,j];
            BusResidualLines := BusResidualLines - GR^.BusLines[i,j];

            iResLines := iResLines + GR^.ResLines[i,j];
            iBusLines := iBusLines + GR^.BusLines[i,j];
            GR_RLines^[(i - 1) * GR^.ncol + j] := R_Lines[i,j];
            GR_BLines^[(i - 1) * GR^.ncol + j] := B_Lines[i,j];
            GR_Index^[(i - 1) * GR^.ncol + j] := (i - 1) * GR^.ncol +
j;
        end;

    ResResidualLines := round(ResResidualLines);
    BusResidualLines := round(BusResidualLines);

    while abs( ResResidualLines ) > 1.0e-4 do
    begin
        z := ran3(idum);
        z := z*(GR^.nrow-1);
        i := round(z)+1;
        z := ran3(idum);
        z := z*(GR^.ncol-1);
        j := round(z)+1;

{remove the strikethrough lines above}


{sort grids so those that contributed to the most to any overage are at
the beginning of the list and those that contributed the most to any
underage are at the end of the list
}
    SortIndex (GR^.nrow * GR^.ncol, GR_RLines, GR_Index);
{
    for i := 1 to GR^.nrow * GR^.ncol do writeln (OUTfile,
GR_RLines^[GR_Index^[i]]);
}
    if ResResidualLines >= zero then
    begin
```

```
        k := 1;
        while abs(ResResidualLines) > 1.0e-4 do
        begin
            i := ((GR Index^[k] - 1) Div GR^.ncol) + 1;
            j := ((GR Index^[k] - 1) Mod GR^.ncol) + 1;

            if rpop[i,j] then { only add lines to populated
microgrids }
            begin
                writeln(OUTfile, '     added 1 res to (', i, ',', j,
')', GR RLines^[GR Index^[k]]);

                GR^.ResLines[i,j] := GR^.ResLines[i,j]+1;
                GR^.ResLines[i,j] + 1;
                ResResidualLines := ResResidualLines - one;
                iResLines := iResLines + 1;
            end;
            k := k + 1;
            if k > GR^.nrow * GR^.ncol then k := 1;

        end;
    end
    else
    begin
        AllowDeletes := false;
        k := GR^.nrow * GR^.ncol;
        while abs(ResResidualLines) > 1.0e-4 do
        begin
            i := ((GR Index^[k] - 1) Div GR^.ncol) + 1;
            j := ((GR Index^[k] - 1) Mod GR^.ncol) + 1;
            {

              subtract lines if there is currently at least TWO lines
or if this is second time through the list and there originally
EXACTLY one line
            }

            if ((AllowDeletes = false) and (GR^.ResLines[i,j] > 1))
then
            begin
                writeln(OUTfile, '     removed 1 res from (', i,
',', j, ')', GR RLines^[GR Index^[k]]);

                GR^.ResLines[i,j] :=
               GR^.ResLines[i,j] 1;
               GR^.ResLines[i,j] - 1;
                ResResidualLines := ResResidualLines + one;
            end;
        end;
    end;
    iResLines := iResLines - 1;
            end;
            if ((AllowDeletes = true) and (GR^.ResLines[i,j] > 0) and
(round (GR RLines^[GR Index^[k]]) = 1)) then
            begin
```

```
                    writeln(OUTfile, '     deleted res from (', i, ',',
j, ')', GR_RLines^[GR_Index^[k]]);
                    GR^.ResLines[i,j] := GR^.ResLines[i,j] - 1;
                    ResResidualLines := ResResidualLines + one;
                    iResLines := iResLines - 1;
               end;

               k := k - 1;
               if k < 1 then
               begin
                    k := GR^.nrow * GR^.ncol;
                    AllowDeletes := not AllowDeletes;
               end;
          end;
     end;

     SortIndex (GR^.nrow * GR^.ncol, GR_BLines, GR_Index);
     if BusResidualLines >= zero then
     begin
          k := 1;
          while abs(BusResidualLines) > 1.0e-4 do
          begin
          z := ran3(idum);
          z := z*(GR^.nrow-1);
          i := round(z)+1;
          z := ran3(idum);
          z := z*(GR^.ncol-1);
          j := round(z)+1;
          if BusResidualLines > zero then
          begin
          i := ((GR_Index^[k] - 1) Div GR^.ncol) + 1;
               j := ((GR_Index^[k] - 1) Mod GR^.ncol) + 1;
               if bpop[i,j] then { only add lines to populated
microgrids }
               begin
                    writeln(OUTfile, '     added 1 bus to (', i, ',', j,
')', GR_BLines^[GR_Index^[k]]);
                    GR^.BusLines[i,j] := GR^.BusLines[i,j] + 1;
                    BusResidualLines := BusResidualLines - one;
                    iBusLines := iBusLines + 1;
               end;
               k := k + 1;
               if k > GR^.nrow * GR^.ncol then k := 1;
          end;
     end
     else
     begin
          AllowDeletes := false;
          k := GR^.nrow * GR^.ncol;
          while abs(BusResidualLines) > 1.0e-4 do
          begin
               i := ((GR_Index^[k] - 1) Div GR^.ncol) + 1;
               j := ((GR_Index^[k] - 1) Mod GR^.ncol) + 1;
               {
                 subtract lines if there is currently at least TWO lines
or
```

```
                if this is second time through the list and there
originally
                EXACTLY one line
            }
            if ((AllowDeletes = false) and (GR^.BusLines[i,j] > 1))
then
            begin
                writeln(OUTfile, '     removed 1 bus from (', i,
',', j, ')', GR BLines^[GR Index^[k]]);
                GR^.BusLines[i,j] := GR^.BusLines[i,j] - 1;
                BusResidualLines := BusResidualLines + one;
                iBusLines := iBusLines - 1;
            end;
            if ((AllowDeletes = true) and (GR^.BusLines[i,j] > 0) and
(round (GR BLines^[GR Index^[k]]) = 1)) then
            begin
                writeln(OUTfile, '     deleted bus from (', i, ',',
j, ')', GR BLines^[GR Index^[k]]);
                GR^.BusLines[i,j] := GR^.BusLines[i,j] - 1;
                BusResidualLines := BusResidualLines + one;
                iBusLines := iBusLines - 1;
            end;

            k := k - 1;
            if k < 1 then
            begin
                k := GR^.nrow * GR^.ncol;
                AllowDeletes := not AllowDeletes;
            end;
        end;
    end;


{
    while abs( ResResidualLines ) > 1.0e-4 do
    begin
        z := ran3(idum);
        z := z*(GR^.nrow-1);
        i := round(z)+1;
        z := ran3(idum);
        z := z*(GR^.ncol-1);
        j := round(z)+1;
        if ResResidualLines > zero then
        begin
            if rpop[i,j] then { only add lines to populated
microgrids
 }
{           begin
                GR^.ResLines[i,j] := GR^.ResLines[i,j]+1;
                ResResidualLines := ResResidualLines - one;
            end;
        end
        else
        begin
```

```
                if GR^.ResLines[i,j] > 0 then { only subtract lines if
there is currently at least one there
}

{               begin
                    GR^.ResLines[i,j] := GR^.ResLines[i,j]-1;
                    ResResidualLines := ResResidualLines + one;
                end;
            end;
        end;

        while abs(BusResidualLines) > 1.0e-4 do
        begin
            z := ran3(idum);
            z := z*(GR^.nrow-1);
            i := round(z)+1;
            z := ran3(idum);
            z := z*(GR^.ncol-1);
            j := round(z)+1;
            if BusResidualLines > zero then
            begin
                if bpop[i,j] then  { only add lines to populated
microgrids
}
{               begin

                    GR^.BusLines[i,j] := GR^.BusLines[i,j]+1;
                    BusResidualLines := BusResidualLines - one;
                end;
            end
            else
            begin
                if GR^.BusLines[i,j] > 0 then { only subtract lines if
there is currently at least one there
}
{               begin
                    GR^.BusLines[i,j] := GR^.BusLines[i,j]-1;
                    BusResidualLines := BusResidualLines + one;
                end;
            end;
        end;
}

    if (iResLines <> GR^.gResLines) or (iBusLines <>
GR^.gBusinessLines) then
        writeln(OUTfile, ' *** TRUE-UP FAILED! ***')
    else
        writeln(OUTfile, '     Completed true-up for fractional
lines.');

     writeln('     Completed true-up for fractional lines.');

(*  Now calculate density, using area defined by convex hull of
populated raster cells. *)

     GR^.density := dist_density( GR );
```

```
    writeln(OUTfile, '      Completed density calculations.');

    writeln('      Completed density calculations.');

    END; { if not DataError }

    dispose(x); dispose(y); dispose(ResLines); dispose(BusLines);
end;
```

Input Modifications

To Recognize Drop Engineering Problem

Summary of Drop and NID adjustment based on PNR customer data for GA

Lines per Record (including SA) From PNR "IN" files	# of PNR Records	Estimated adjustment to arrive at HCPM modeled lots	Estimated HCPM modeled Lots	Cable Size (assuming a 50% Distribution Sizing Factor)	Drop Cost per foot (from Cable Size Table below)	TotalDrop (assuming 1 foot of drop per lot)		Nid Cost Differential from BST Input of $24.52	Total NID Differential from BST Input
<=1	1,049,650	0.67	703,266	Regular Drop	0.318	223,638		-1.77	(1,244,780)
>1, <=2	1,887,263	0.95	1,792,900	Regular Drop	0.318	570,142		6.06	9,235,227
>2, <=3	81,070	1	81,070	Regular Drop	0.318	25,780		17.43	1,413,050
>3, <=4	63,664	1	63,664	Regular Drop	0.318	20,245		25.26	1,608,153
>4, <=6	63,282	1	63,282	12	3.934	248,958		0	-
>6, <=9	34,431	1	34,431	18	3.934	135,455		-	-
>9, <=12.5	14,968	1	14,968	25	3.934	58,886		-	-
>12.5, <=25	10,984	1	10,984	50	4.651	51,087		-	-
>25, <=50	2,872	1	2,872	100	5.887	16,907		-	-
>50, <=100	881	1	881	200	8.079	7,118		-	-
>100, <=200	279	1	279	300	10.544	2,942		-	-
>200, <=300	23	1	23	400	13.160	303		-	-
>300, <=450	6	1	6	600	18.024	108		-	-
>450, <=600	1	1	1	900	25.115	25		-	-
>600	9	1	9	1,200	31.790	286		-	-
Total	*3,209,383*		*2,768,635*			*1,361,880*			*11,011,650*
					Adjusted Drop Per Foot Input (Total Drop divided by Estimated Total Lots)	**0.49**		**Adjusted NID Input Differential (Total NID divided by Estimated Total Lots)**	**3.98**

Cable Sizes	Average Installed Cost For Building Connection Cable (assumes plant mix of Density=100)
4200	$ 111.078
3600	$ 95.396
3000	$ 80.368
2400	$ 65.091
2100	$ 57.554
1800	$ 47.406
1200	$ 31.790
900	$ 25.115
600	$ 18.024
400	$ 13.160
300	$ 10.544
200	$ 8.079
100	$ 5.887
50	$ 4.651
25	$ 3.934
18	$ 3.934
12	$ 3.934
6	$ 3.934

Network Item	BST Filed Cost
Drop (per foot)	0.32
NID	24.52

NID Components	Cost per Unit
2 pair Housing	4.84
6 Pair Housing	8.38
Labor	7.26
Exempt	2.82
Int/Prot	7.83

Total NID Cost on Pairs per lot basis	
1Pr	22.75
2pr	30.58
3pr	41.95
4pr	49.78

Exhibit Five-JWS

Table One-Summary at Wire Center level

HCPM Cluster Analysis

SWCLLI	Individual Rectilinear	Individual Road Path	% of Rect
Total Distribution AIVLGAMA	5,708,426	7,258,972	127.2%
Total Distribution ATLNGAHR	8,646,591	14,075,189	162.8%
Total Distribution CLMBGAMT	28,725,522	42,422,091	147.7%
Total Distribution CVTNGAMT	48,432,209	77,085,974	159.2%
Total Distribution FRSYGAMA	17,375,901	25,527,771	146.9%
Total Distribution GRFNGAMA	46,437,250	63,149,812	136.0%
Total Distribution MLLNGAMA	12,827,658	17,194,636	134.0%
Total Distribution SENOGAMA	8,502,041	11,414,791	134.3%
Total Distribution WRNSGAMA	9,596,809	12,775,133	133.1%
Total Distribution WRRBGAMA	23,359,932	33,458,885	143.2%
Grand Total	209,612,339	304,363,254	145%

Table Two-Detail at Cluster Level

HCPM Cluster Analysis

SWCLLI	CANbr	Individual Rectilinear	Individual Road Path	% of Rect
AIVLGAMAXXX	1	693,244	754,264	108.8%
AIVLGAMAXXX	2	725,853	1,196,482	164.8%
AIVLGAMAXXX	3	459,070	540,319	117.7%
AIVLGAMAXXX	4	1,025,553	1,244,076	121.3%
AIVLGAMAXXX	5	136,356	258,267	189.4%
AIVLGAMAXXX	6	507,290	668,035	131.7%
AIVLGAMAXXX	7	571,946	679,486	118.8%
AIVLGAMAXXX	8	1,002,089	979,281	97.7%
AIVLGAMAXXX	9	587,025	938,762	159.9%
Total Distribution AIVLGAMA		5,708,426	7,258,972	127.2%
ATLNGAHRXXX	1	470,482	533,972	113.5%
ATLNGAHRXXX	7	596,559	700,827	117.5%
ATLNGAHRXXX	2	465,450	665,914	143.1%
ATLNGAHRXXX	3	189,323	286,192	151.2%
ATLNGAHRXXX	4	462,864	741,162	160.1%
ATLNGAHRXXX	5	640,024	1,853,048	289.5%
ATLNGAHRXXX	6	802,234	1,085,244	135.3%
ATLNGAHRXXX	8	730,314	870,120	119.1%
ATLNGAHRXXX	9	815,887	1,955,627	239.7%
ATLNGAHRXXX	10	849,132	1,596,371	188.0%
ATLNGAHRXXX	11	831,382	1,476,630	177.6%
ATLNGAHRXXX	12	394,418	575,857	146.0%
ATLNGAHRXXX	13	595,426	670,781	112.7%
ATLNGAHRXXX	14	803,096	1,063,444	132.4%
Total Distribution ATLNGAHR		8,646,591	14,075,189	162.8%

HCPM Cluster Analysis

SWCLLI	CANbr	Individual Rectilinear	Individual Road Path	% of Rect
CLMBGAMTXXX	1	93,575	109,408	116.9%
CLMBGAMTXXX	2	113,219	235,361	207.9%
CLMBGAMTXXX	3	1,720,693	4,039,113	234.7%
CLMBGAMTXXX	4	948,162	1,534,910	161.9%
CLMBGAMTXXX	5	1,088,550	1,975,547	181.5%
CLMBGAMTXXX	6	862,647	1,225,288	142.0%
CLMBGAMTXXX	7	427,899	853,649	199.5%
CLMBGAMTXXX	8	389,110	530,806	136.4%
CLMBGAMTXXX	9	421,769	461,926	109.5%
CLMBGAMTXXX	10	791,569	1,044,307	131.9%
CLMBGAMTXXX	11	540,264	1,004,558	185.9%
CLMBGAMTXXX	12	972,564	1,699,638	174.8%
CLMBGAMTXXX	13	731,514	808,363	110.5%
CLMBGAMTXXX	14	985,210	1,345,602	136.6%
CLMBGAMTXXX	15	607,838	926,455	152.4%
CLMBGAMTXXX	16	890,405	1,918,959	215.5%
CLMBGAMTXXX	17	1,350,048	2,818,937	208.8%
CLMBGAMTXXX	18	289,783	323,620	111.7%
CLMBGAMTXXX	19	858,781	985,862	114.8%
CLMBGAMTXXX	20	500,321	753,403	150.6%
CLMBGAMTXXX	21	892,877	1,185,211	132.7%
CLMBGAMTXXX	22	517,614	654,713	126.5%
CLMBGAMTXXX	23	934,948	1,359,210	145.4%
CLMBGAMTXXX	24	203,683	249,982	122.7%
CLMBGAMTXXX	25	229,691	275,827	120.1%
CLMBGAMTXXX	26	556,030	595,250	107.1%
CLMBGAMTXXX	27	34,405	51,193	148.8%
CLMBGAMTXXX	28	378,181	428,223	113.2%
CLMBGAMTXXX	29	183,522	205,646	112.1%
CLMBGAMTXXX	30	326,616	342,661	104.9%
CLMBGAMTXXX	31	256,663	289,000	112.6%
CLMBGAMTXXX	32	324,469	408,029	125.8%
CLMBGAMTXXX	33	138,721	147,171	106.1%
CLMBGAMTXXX	34	155,004	188,771	121.8%
CLMBGAMTXXX	35	101,454	108,115	106.6%
CLMBGAMTXXX	36	101,167	112,603	111.3%
CLMBGAMTXXX	37	93,004	105,132	113.0%
CLMBGAMTXXX	38	40,709	41,561	102.1%
CLMBGAMTXXX	39	37,588	43,219	115.0%
CLMBGAMTXXX	40	48,793	50,978	104.5%
CLMBGAMTXXX	41	452,461	531,992	117.6%
CLMBGAMTXXX	42	240,999	398,140	165.2%
CLMBGAMTXXX	43	819,245	953,318	116.4%
CLMBGAMTXXX	44	238,787	414,547	173.6%
CLMBGAMTXXX	45	672,886	753,847	112.0%
CLMBGAMTXXX	46	485,426	523,684	107.9%
CLMBGAMTXXX	47	318,099	350,505	110.2%
CLMBGAMTXXX	48	299,013	574,624	192.2%
CLMBGAMTXXX	49	518,428	662,251	127.7%
CLMBGAMTXXX	50	362,742	566,860	156.3%
CLMBGAMTXXX	51	570,881	653,378	114.5%
CLMBGAMTXXX	52	341,275	395,520	115.9%
CLMBGAMTXXX	53	243,771	269,945	110.7%

HCPM Cluster Analysis

SWCLLI	CANbr	Individual Rectilinear	Individual Road Path	% of Rect
CLMBGAMTXXX	54	107,618	121,674	113.1%
CLMBGAMTXXX	55	373,711	383,444	102.6%
CLMBGAMTXXX	56	240,738	259,065	107.6%
CLMBGAMTXXX	57	563,315	645,037	114.5%
CLMBGAMTXXX	58	211,509	334,227	158.0%
CLMBGAMTXXX	59	359,723	467,641	130.0%
CLMBGAMTXXX	60	425,718	589,029	138.4%
CLMBGAMTXXX	61	235,687	257,744	109.4%
CLMBGAMTXXX	62	504,430	877,412	173.9%
Total Distribution CLMBGAMT		28,725,522	42,422,091	147.7%
CVTNGAMTXXX	1	267,862	272,549	101.7%
CVTNGAMTXXX	2	1,865,549	3,015,865	161.7%
CVTNGAMTXXX	3	189,469	170,509	90.0%
CVTNGAMTXXX	4	385,229	486,548	126.3%
CVTNGAMTXXX	5	647,895	2,227,463	343.8%
CVTNGAMTXXX	6	274,457	462,909	168.7%
CVTNGAMTXXX	7	439,267	456,631	104.0%
CVTNGAMTXXX	8	99,084	157,992	159.5%
CVTNGAMTXXX	9	1,410,650	2,085,939	147.9%
CVTNGAMTXXX	10	827,811	1,241,726	150.0%
CVTNGAMTXXX	11	500,944	919,579	183.6%
CVTNGAMTXXX	12	2,481,161	5,396,082	217.5%
CVTNGAMTXXX	13	1,002,070	1,175,299	117.3%
CVTNGAMTXXX	14	317,049	416,437	131.3%
CVTNGAMTXXX	15	1,302,766	2,102,724	161.4%
CVTNGAMTXXX	16	654,885	861,039	131.5%
CVTNGAMTXXX	17	750,107	889,346	118.6%
CVTNGAMTXXX	18	510,061	925,485	181.4%
CVTNGAMTXXX	19	447,402	638,322	142.7%
CVTNGAMTXXX	20	1,502,387	2,038,345	135.7%
CVTNGAMTXXX	21	1,167,838	1,292,732	110.7%
CVTNGAMTXXX	22	1,770,826	2,433,176	137.4%
CVTNGAMTXXX	23	1,207,880	1,783,116	147.6%
CVTNGAMTXXX	24	564,698	693,882	122.9%
CVTNGAMTXXX	25	406,828	861,061	211.7%
CVTNGAMTXXX	26	1,494,499	1,855,677	124.2%
CVTNGAMTXXX	27	719,795	1,018,709	141.5%
CVTNGAMTXXX	28	2,043,386	4,304,042	210.6%
CVTNGAMTXXX	29	930,505	2,176,778	233.9%
CVTNGAMTXXX	30	515,335	596,443	115.7%
CVTNGAMTXXX	31	1,242,218	1,456,185	117.2%
CVTNGAMTXXX	32	697,064	1,332,989	191.2%
CVTNGAMTXXX	33	1,286,890	2,304,582	179.1%
CVTNGAMTXXX	34	674,588	891,618	132.2%
CVTNGAMTXXX	35	1,762,606	1,937,130	109.9%
CVTNGAMTXXX	36	1,051,066	2,182,297	207.6%
CVTNGAMTXXX	37	1,326,480	1,911,842	144.1%
CVTNGAMTXXX	38	1,378,898	3,554,899	257.8%
CVTNGAMTXXX	39	1,556,322	1,913,563	123.0%
CVTNGAMTXXX	40	2,195,493	4,775,377	217.5%
CVTNGAMTXXX	41	1,682,619	2,333,110	138.7%

HCPM Cluster Analysis

SWCLLI	CANbr	Individual Rectilinear	Individual Road Path	% of Rect
CVTNGAMTXXX	42	2,020,686	2,463,644	121.9%
CVTNGAMTXXX	43	2,000,227	2,983,117	149.1%
CVTNGAMTXXX	44	948,664	1,611,782	169.9%
CVTNGAMTXXX	45	578,669	858,059	148.3%
CVTNGAMTXXX	46	1,118,457	1,327,720	118.7%
CVTNGAMTXXX	47	213,567	291,655	136.6%
Total Distribution CVTNGAMT		48,432,209	77,085,974	159.2%
FRSYGAMAXXX	1	259,937	268,628	103.3%
FRSYGAMAXXX	2	630,100	1,276,558	202.6%
FRSYGAMAXXX	3	1,234,621	1,615,314	130.8%
FRSYGAMAXXX	4	637,143	601,782	94.5%
FRSYGAMAXXX	5	493,435	907,647	183.9%
FRSYGAMAXXX	6	375,734	420,816	112.0%
FRSYGAMAXXX	7	602,883	871,037	144.5%
FRSYGAMAXXX	8	335,614	483,496	144.1%
FRSYGAMAXXX	9	169,547	212,447	125.3%
FRSYGAMAXXX	10	304,112	440,445	144.8%
FRSYGAMAXXX	11	280,276	284,116	101.4%
FRSYGAMAXXX	12	627,496	770,431	122.8%
FRSYGAMAXXX	13	178,386	236,625	132.6%
FRSYGAMAXXX	14	330,338	474,288	143.6%
FRSYGAMAXXX	15	1,252,137	1,651,307	131.9%
FRSYGAMAXXX	16	246,749	307,956	124.8%
FRSYGAMAXXX	17	903,765	1,228,443	135.9%
FRSYGAMAXXX	18	320,812	407,513	127.0%
FRSYGAMAXXX	19	334,658	356,657	106.6%
FRSYGAMAXXX	20	337,123	419,623	124.5%
FRSYGAMAXXX	21	483,109	635,656	131.6%
FRSYGAMAXXX	22	710,180	912,597	128.5%
FRSYGAMAXXX	23	453,250	756,428	166.9%
FRSYGAMAXXX	24	691,613	802,336	116.0%
FRSYGAMAXXX	25	866,399	1,698,324	196.0%
FRSYGAMAXXX	26	525,964	840,783	159.9%
FRSYGAMAXXX	27	636,515	1,235,175	194.1%
FRSYGAMAXXX	28	562,543	1,376,633	244.7%
FRSYGAMAXXX	29	659,470	829,711	125.8%
FRSYGAMAXXX	30	768,929	1,189,777	154.7%
FRSYGAMAXXX	31	1,163,063	2,015,222	173.3%
Total Distribution FRSYGAMA		17,375,901	25,527,771	146.9%
GRFNGAMAXXX	1	165,840	184,538	111.3%
GRFNGAMAXXX	2	1,249,780	1,711,614	137.0%
GRFNGAMAXXX	3	567,605	762,643	134.4%
GRFNGAMAXXX	4	693,117	977,511	141.0%
GRFNGAMAXXX	5	1,105,803	1,792,346	162.1%
GRFNGAMAXXX	6	380,580	1,477,435	388.2%
GRFNGAMAXXX	7	957,860	1,118,154	116.7%
GRFNGAMAXXX	8	1,298,567	1,768,379	136.2%
GRFNGAMAXXX	9	1,122,309	1,369,923	122.1%
GRFNGAMAXXX	10	1,494,312	2,293,078	153.5%

HCPM Cluster Analysis

SWCLLI	CANbr	Individual Rectilinear	Individual Road Path	% of Rect
GRFNGAMAXXX	11	1,702,265	2,457,955	144.4%
GRFNGAMAXXX	12	739,023	2,528,046	342.1%
GRFNGAMAXXX	13	1,233,366	1,790,276	145.2%
GRFNGAMAXXX	14	1,266,525	1,387,679	109.6%
GRFNGAMAXXX	15	1,175,294	1,375,971	117.1%
GRFNGAMAXXX	16	419,026	1,111,241	265.2%
GRFNGAMAXXX	17	1,278,517	1,682,508	131.6%
GRFNGAMAXXX	18	1,962,268	2,370,027	120.8%
GRFNGAMAXXX	19	977,934	1,060,866	108.5%
GRFNGAMAXXX	20	2,056,007	2,413,396	117.4%
GRFNGAMAXXX	21	1,332,835	1,591,897	119.4%
GRFNGAMAXXX	22	2,053,007	2,355,552	114.7%
GRFNGAMAXXX	23	1,310,724	2,132,464	162.7%
GRFNGAMAXXX	24	1,049,604	2,438,331	232.3%
GRFNGAMAXXX	25	2,635,131	2,884,086	109.4%
GRFNGAMAXXX	26	1,390,424	1,671,570	120.2%
GRFNGAMAXXX	27	1,305,642	1,706,033	130.7%
GRFNGAMAXXX	28	926,057	1,325,039	143.1%
GRFNGAMAXXX	29	519,623	663,531	127.7%
GRFNGAMAXXX	30	1,099,881	1,356,512	123.3%
GRFNGAMAXXX	31	997,884	1,307,978	131.1%
GRFNGAMAXXX	32	585,373	698,450	119.3%
GRFNGAMAXXX	33	541,135	571,939	105.7%
GRFNGAMAXXX	34	1,206,399	1,396,979	115.8%
GRFNGAMAXXX	35	420,633	489,339	116.3%
GRFNGAMAXXX	36	385,230	526,117	136.6%
GRFNGAMAXXX	37	186,696	198,370	106.3%
GRFNGAMAXXX	38	1,199,363	1,442,225	120.2%
GRFNGAMAXXX	39	1,620,743	2,187,215	135.0%
GRFNGAMAXXX	40	675,648	799,056	118.3%
GRFNGAMAXXX	41	877,864	1,031,085	117.5%
GRFNGAMAXXX	42	973,266	1,142,314	117.4%
GRFNGAMAXXX	43	1,298,090	1,600,144	123.3%
Total Distribution GRFNGAMA		46,437,250	63,149,812	136.0%
MLLNGAMAXXX	1	823,116	896,000	108.9%
MLLNGAMAXXX	2	388,887	677,251	174.2%
MLLNGAMAXXX	3	414,532	478,768	115.5%
MLLNGAMAXXX	4	667,102	1,056,006	158.3%
MLLNGAMAXXX	5	393,571	669,402	170.1%
MLLNGAMAXXX	6	266,062	412,269	155.0%
MLLNGAMAXXX	7	281,847	394,002	139.8%
MLLNGAMAXXX	8	291,824	398,768	136.6%
MLLNGAMAXXX	9	346,493	379,179	109.4%
MLLNGAMAXXX	10	323,648	500,534	154.7%
MLLNGAMAXXX	11	201,517	244,533	121.3%
MLLNGAMAXXX	12	361,593	423,815	117.2%
MLLNGAMAXXX	13	463,822	655,991	141.4%
MLLNGAMAXXX	14	314,101	467,642	148.9%
MLLNGAMAXXX	15	523,983	786,850	150.2%
MLLNGAMAXXX	16	254,691	396,847	155.8%
MLLNGAMAXXX	17	595,754	773,884	129.9%

HCPM Cluster Analysis

SWCLLI	CANbr	Individual Rectilinear	Individual Road Path	% of Rect
MLLNGAMAXXX	18	407,589	469,247	115.1%
MLLNGAMAXXX	19	221,049	428,372	193.8%
MLLNGAMAXXX	20	102,681	156,997	152.9%
MLLNGAMAXXX	21	483,851	634,221	131.1%
MLLNGAMAXXX	22	370,868	544,182	146.7%
MLLNGAMAXXX	23	484,087	720,339	148.8%
MLLNGAMAXXX	24	431,017	624,169	144.8%
MLLNGAMAXXX	25	385,021	392,582	102.0%
MLLNGAMAXXX	26	233,340	320,966	137.6%
MLLNGAMAXXX	27	623,452	919,951	147.6%
MLLNGAMAXXX	28	396,814	514,531	129.7%
MLLNGAMAXXX	29	383,622	443,901	115.7%
MLLNGAMAXXX	30	1,391,724	1,413,437	101.6%
Total Distribution MLLNGAMA		12,827,658	17,194,636	134.0%
SENOGAMAXXX	1	1,722,304	1,742,686	101.2%
SENOGAMAXXX	2	950,457	1,056,025	111.1%
SENOGAMAXXX	3	694,126	802,553	115.6%
SENOGAMAXXX	4	782,153	911,816	116.6%
SENOGAMAXXX	5	686,490	1,159,008	168.8%
SENOGAMAXXX	6	1,123,850	1,554,382	138.3%
SENOGAMAXXX	7	949,092	2,001,139	210.8%
SENOGAMAXXX	8	748,263	1,062,623	142.0%
SENOGAMAXXX	9	845,306	1,124,559	133.0%
Total Distribution SENOGAMA		8,502,041	11,414,791	134.3%
WRNSGAMAXXX	1	1,796,834	1,772,330	98.6%
WRNSGAMAXXX	2	431,752	441,343	102.2%
WRNSGAMAXXX	3	388,538	585,825	150.8%
WRNSGAMAXXX	4	353,544	596,331	168.7%
WRNSGAMAXXX	5	644,830	920,038	142.7%
WRNSGAMAXXX	6	500,861	658,867	131.5%
WRNSGAMAXXX	7	381,755	435,494	114.1%
WRNSGAMAXXX	8	384,797	786,982	204.5%
WRNSGAMAXXX	9	877,296	958,503	109.3%
WRNSGAMAXXX	10	311,880	383,676	123.0%
WRNSGAMAXXX	11	361,291	573,566	158.8%
WRNSGAMAXXX	12	320,141	436,138	136.2%
WRNSGAMAXXX	13	253,814	595,941	234.8%
WRNSGAMAXXX	14	523,259	780,722	149.2%
WRNSGAMAXXX	15	470,027	557,217	118.5%
WRNSGAMAXXX	16	363,920	727,607	199.9%
WRNSGAMAXXX	17	185,527	381,185	205.5%
WRNSGAMAXXX	18	541,321	639,012	118.0%
WRNSGAMAXXX	19	228,716	270,318	118.2%
WRNSGAMAXXX	20	276,706	274,038	99.0%
Total Distribution WRNSGAMA		9,596,809	12,775,133	133.1%
WRRBGAMAXXX	1	490,738	713,808	145.5%

HCPM Cluster Analysis

SWCLLI	CANbr	Individual Rectilinear	Individual Road Path	% of Rect
WRRBGAMAXXX	2	1,440	2,327	161.6%
WRRBGAMAXXX	3	724,000	1,101,110	152.1%
WRRBGAMAXXX	4	831,351	1,204,585	144.9%
WRRBGAMAXXX	5	609,450	1,036,650	170.1%
WRRBGAMAXXX	6	820,573	1,083,536	132.0%
WRRBGAMAXXX	7	497,558	713,061	143.3%
WRRBGAMAXXX	8	251,945	278,970	110.7%
WRRBGAMAXXX	9	467,382	893,967	191.3%
WRRBGAMAXXX	10	1,311,214	2,250,997	171.7%
WRRBGAMAXXX	11	1,212,107	1,884,417	155.5%
WRRBGAMAXXX	12	197,762	305,215	154.3%
WRRBGAMAXXX	13	524,546	755,653	144.1%
WRRBGAMAXXX	14	736,035	1,888,687	256.6%
WRRBGAMAXXX	15	391,786	478,863	122.2%
WRRBGAMAXXX	16	328,661	379,957	115.6%
WRRBGAMAXXX	17	977,622	1,213,637	124.1%
WRRBGAMAXXX	18	479,508	610,723	127.4%
WRRBGAMAXXX	19	560,506	862,041	153.8%
WRRBGAMAXXX	20	1,173,641	1,344,012	114.5%
WRRBGAMAXXX	21	465,393	672,594	144.5%
WRRBGAMAXXX	22	125,273	137,655	109.9%
WRRBGAMAXXX	23	581	587	101.0%
WRRBGAMAXXX	24	656,668	895,523	136.4%
WRRBGAMAXXX	25	3,759	6,102	162.3%
WRRBGAMAXXX	26	655	1,298	198.2%
WRRBGAMAXXX	27	7,672	10,545	137.4%
WRRBGAMAXXX	28	27,080	35,928	132.7%
WRRBGAMAXXX	29	444,837	473,537	106.5%
WRRBGAMAXXX	30	21,364	33,410	156.4%
WRRBGAMAXXX	31	790,948	970,107	122.7%
WRRBGAMAXXX	32	605,260	829,916	137.1%
WRRBGAMAXXX	33	328,623	399,903	121.7%
WRRBGAMAXXX	34	388,687	601,596	154.8%
WRRBGAMAXXX	35	168,352	196,266	116.6%
WRRBGAMAXXX	36	485,444	895,652	184.5%
WRRBGAMAXXX	37	479,954	611,832	127.5%
WRRBGAMAXXX	38	558,403	718,484	128.7%
WRRBGAMAXXX	39	337,539	780,440	231.2%
WRRBGAMAXXX	40	1,056,167	1,175,879	111.3%
WRRBGAMAXXX	41	187,647	302,849	161.4%
WRRBGAMAXXX	42	744,935	889,185	119.4%
WRRBGAMAXXX	43	1,010,200	1,343,975	133.0%
WRRBGAMAXXX	44	569,904	761,001	133.5%
WRRBGAMAXXX	45	389,978	462,957	118.7%
WRRBGAMAXXX	46	28,004	40,727	145.4%
WRRBGAMAXXX	47	887,654	1,207,463	136.0%
WRRBGAMAXXX	48	533	590	110.7%
WRRBGAMAXXX	49	593	668	112.6%
Total Distribution Total WRRBGAMA		23,359,932	33,458,885	143.2%
All Wire Centers Total		**209,612,339**	**304,363,254**	**145.2%**

CERTIFICATE OF SERVICE

I, Kelseau Powe, Jr., do hereby certify that I have caused the foregoing **REPLY COMMENTS OF BELLSOUTH CORPORATION, QWEST COMMUNICATIONS INC., AND SPRINT CORPORATION** to be 1) filed with the Secretary of the FCC via hand delivery (original and four copies), 2) served, via hand delivery and email, on the FCC's contractor International Transcription Services, Inc. and 3) served, via First Class United States mail, postage prepaid, on the other parties listed on the attached service list.

Kelseau Powe, Jr.

August 27, 2001

International Transcription
Services, Inc.
1231 20th Street, N.W.
Washington, DC 20036

ddisanti@itsdocs.com

Sheryl Todd
Accounting Policy Division
Federal Communications Commission
Room 5-A422
445 12th Street,, S.W.
Washington, DC 20554

3 copies

David L. Lawson.....AT&T
Christopher T. Shenk
Sidley Austin Brown & Wood
1501 K Street, N.W.
Washington, DC 20005

Mark C. Rosenblum
Judy Sello
AT&T Corp.
Room 1135L2
295 North Maple Avenue
Basking Ridge, NJ 07920

Joseph DiBella
Michael E.Glover
Edward Shakin
Verizon Telephone Companies
8th Floor
1320 North Court House Road
Arlington, VA 22201

Chris Frentrup
WorldCom, Inc.
1133 19th Street, N.W.
Washington, DC 20036

cc96-45i.doc
Updated 8/27/2001