ORIGINAL

EX PARTE OR LATE FILED

AT&T

February 28, 2000

RECEIVED
FEB 28 2000
FEDERAL COMMUNICATIONS COMMISSION
OFFICE OF THE SECRETARY

Ms. Magalie Roman Salas
Secretary
Federal Communications Commission
445 Twelfth Street, S.W.
Washington, D.C. 20554

RE: Ex Parte Presentation
CC Docket No. 96-45 – Universal Service/Proxy Cost Models
CC Docket No. 97-160 – Forward-Looking Cost Mechanism

Dear Ms. Salas:

Attached please find a written document and a CD-ROM that contains several of the modifications that AT&T proposes be made to the Synthesis Model's code. Also included is some more in-depth analysis of the Prim algorithm's next node selection criteria.

Two copies of this Notice are being submitted to the Secretary of the FCC in accordance with Section 1.1206(a)(2) of the Commission's rules. A copy of this Notice and the CD-ROM is also being provided to ITS.

Sincerely,

Richard N. Clarke/ha

Richard N. Clarke

Attachments

cc: Katie King
Bob Loube
Bill Sharkey (w/ CD-ROM)
Bryan Clopton
Gene Fullano
Jeff Prisbrey (w/ CD-ROM)
Sheryl Todd

No. of Copies rec'd 0+2
List ABCDE

Suggested Modifications to the Synthesis Model

Implementing Code Modifications

02/17/00

Drop Terminal Locations and Orientation

PrimDist.pas

Original Code

```
procedure get_lines( GR          : GridRecordType_ptr;
                     density     : double;
                     row         : integer;
                     col         : integer;

                     NS_lots     : integer;
                     EW_lots      : integer;
*
*
*
line_vector^[n] := factor*lines_per_lot;

x^[n] := GR^.LowerLeftX + (col-1)*GR^.MicroGridEW+ i*(one/EW_lots);
y^[n] := GR^.LowerLeftY + (row-1)*GR^.MicroGridNS + j*(one/NS_lots);

tmp := structure_cost_fn(line_vector^[n],0,density,GR^.hardness,GR^.DepthToBedrock,
```

Modified Code

```
procedure get_lines( GR          : GridRecordType_ptr;
                     density     : double;
                     row         : integer;
                     col         : integer;
                     divider_row : integer;
                     divider_col : integer;
                     NS_lots     : integer;
                     EW_lots      : integer;
*
*
*
line_vector^[n] := factor*lines_per_lot;

if (row<=divider_row) and (col<=divider_col) then
begin
   x^[n] := GR^.LowerLeftX + (col-1)*GR^.MicroGridEW + i*(GR^.MicroGridEW/EW_lots);
   y^[n] := GR^.LowerLeftY + (row-1)*GR^.MicroGridNS + j*(GR^.MicroGridNS/NS_lots);
end;
if(row<=divider_row) and (col>divider_col) then
begin
  x^[n] := GR^.LowerLeftX + (col-1)*GR^.MicroGridEW + (EW_lots-i)*(GR^.MicroGridEW/EW_lots);
  y^[n] := GR^.LowerLeftY + (row-1)*GR^.MicroGridNS+ j*(GR^.MicroGridNS/NS_lots);
end;
if(row>divider_row) and (col<=divider_col) then
begin
  x^[n] := GR^.LowerLeftX + (col-1)*GR^.MicroGridEW + i*(GR^.MicroGridEW/EW_lots);
  y^[n] := GR^.LowerLeftY + (row-1)*GR^.MicroGridNS+ (NS_lots-j)*(GR^.MicroGridNS/NS_lots);
end;
if (row>divider_row) and (col>divider_col) then
begin
  x^[n] := GR^.LowerLeftX + (col-1)*GR^.MicroGridEW + (EW_lots-i)*(GR^.MicroGridEW /EW_lots);
  y^[n] := GR^.LowerLeftY + (row-1)*GR^.MicroGridNS+ (NS_lots-j)*(GR^.MicroGridNS/NS_lots);
end;

  tmp := structure_cost_fn(line_vector^[n],0,density,GR^.hardness,GR^.DepthToBedrock,
```

Drop Terminal Locations and Orientation-Cont

PrimDist.pas

Original Code

```
*
*
*
573   procedure calculate_prim_distribution_cost(
*
*
*
616   var i            : integer;
617       j            : integer;
618       n            : integer;
619       num_terms    : integer;
620       k            : integer;

621       midx         : double;
622       midy         : double;
*
*
*
695 n := 1;
696

697 for i := 1 to GR^nrow do
698   for j := 1 to GR^.ncol do
699   if (flag^[i,j]=k) and (lines^[i,j]>zero) then
700    begin
701      lots := round(GR^.ResLines[i,j]/GR^.lines_per_house) +
702      round(GR^BusLines[i,j]/GR^.lines_per_bus);
703
704      lot_divide( lots,NS_lots,EW_lots );
705
706      get_lines(GR, density, i, j, round(NS_lots), round(EW_lots), lines^[i,j], n, line_vector,
707                x, y, grid_dropterm_cost, grid_nid_cost, grid_drop_cost, grid_drop_feet);

708
709      prim_drop_cost := prim_drop_cost + grid_drop_cost;
```

Modified Code

```
*
*
*
573   procedure calculate_prim_distribution_cost(
*
*
*
616   var i            : integer;
617       j            : integer;
618       n            : integer;
619       num_terms    : integer;
620       k            : integer;
          divider_row  : integer;
          divider_col  : integer;
621       midx         : double;
622       midy         : double;
*
*
*
695 n := 1;
696
    divider_row := round( abs(SAIY[n]-GR^.LowerLeftY)/GR^.MicroGridNS);
    divider_col := round( abs(SAIX[n]-GR^.LowerLeftX)/GR^MicroGridEW );
697 for i := 1 to GR^nrow do
698   for j := 1 to GR^.ncol do
699   if (flag^[i,j]=k) and (lines^[i,j]>zero) then
700    begin
701      lots := round(GR^.ResLines[i,j]/GR^.lines_per_house) +
702      round(GR^BusLines[i,j]/GR^.lines_per_bus);
703
704      lot_divide( lots,NS_lots,EW_lots );
705
         get_lines(GR, density, i, j, divider_row, divider_col, round(NS_lots), round(EW_lots),
                   lines^[i,j], n, line_vector, x, y, grid_dropterm_cost, grid_nid_cost,
                   grid_drop_cost, grid_drop_feet);
708
709      prim_drop_cost := prim_drop_cost + grid_drop_cost;
```

Creating Lots Within Microgrids

Lot_Div.pas

Original Code

```
018 procedure lot_divide(
*
*
*
054         NS_try_d := number_of_lots/EW_try ;
055         NS_try := round(NS_try_d);

056         waste := NS_try*EW_try - number_of_lots;
057         if (waste < 0) then waste := number_of_lots;
058         if (waste <= minwaste) then
059         begin
```

Modified Code

```
018 procedure lot_divide(
*
*
*
054         NS_try_d := number_of_lots/EW_try ;
055         NS_try := round(NS_try_d);
            while (EW_try*NS_try < number_of_lots) do NS_try := NS_try + 1;
056         waste := NS_try*EW_try - number_of_lots;
057         if (waste < 0) then waste := number_of_lots;
            if (waste <= minwaste) and (2*EW_try >= NS_try) and (2*NS_try >= EW_try) then
059         begin
```

Residual Line Counts

ClusIntf.pas

```
0558    d1          : double;
0559    d2          : double;
0560    dr1         : double;
0561    nr1         : integer;

0562
0563  begin
*
*
*
0791        GR^.ResLines[i,j] := 0;
0792        GR^.BusLines[i,j]  := 0;
0793         bpop[i,j] := false;
0794         rpop[i,j] := false;

0795     end;
*
*
*
0800        j := round( abs(x^[k] -llx)/RasterSize + half );
0801
0802        GR^.ResLines[i,j] := GR^.ResLines[i,j] + round(ResLines^[k]);
0803        if ResLines^[k] > zero then rpop[i,j] := true;
0804        GR^.BusLines[i,j]  := GR^.BusLines[i,j] + round(BusLines^[k]);
0805        if BusLines^[k] > zero then bpop[i,j] := true;
0806     end;
*
*
*
0817     begin

0818        ResResidualLines :=ResResidualLines - GR^.ResLines[i,j];
0819        BusResidualLines :=BusResidualLines - GR^.BusLines[i,j];
0820     end;
```

```
0558    d1          : double;
0559    d2          : double;
0560    dr1         : double;
0561    nr1         : integer;
        R_Lines     : array[1..50,1..50] of single;
        B_Lines     : array[1..50,1..50] of single;
0562
0563  begin
*
*
*
0791        GR^.ResLines[i,j] := 0;
0792        GR^.BusLines[i,j]  := 0;
0793         bpop[i,j] := false;
0794         rpop[i,j] := false;
             R_Lines[i,j] := zero;
             B_Lines[i,j] := zero;
0795     end;
*
*
*
0800        j := round( abs(x^[k] -llx)/RasterSize + half );
0801
             R_Lines[i,j] := R_Lines[i,j] + ResLines^[k];
             if ResLines^[k] > zero then rpop[i,j] := true;
             B_Lines[i,j] := B_Lines[i,j] + BusLines^[k];
             if BusLines^[k] > zero then bpop[i,j] := true;
0806     end;
*
*
*
0817     begin
             GR^.ResLines[i,j] := round(R_Lines[i,j]);
             GR^.BusLines[i,j] := round(B_Lines[i,j]);
0818         ResResidualLines :=ResResidualLines - GR^.ResLines[i,j];
0819         BusResidualLines :=BusResidualLines - GR^.BusLines[i,j];
0820     end;
```

Node Selection Criteria - (a)Distance

PrimDist.pas

Original Code

```
400   for i := 1 to n do
401   begin
402      a^[i] := true;
403      b^[i] := 0;
404      c^[i] := dlarge;
405      d1^[i] := zero;
406   end;
*
*
*
421          begin
422             dist := dmtx^[j]^[k];
423
424             cost := provisional_cost(k,num_terms,line_vector^[k],GR,dist,(dist+d2s^[j]),density,
425                     FillFactor);
426
427
428             if cost < c^[k] then
429             begin
430                c^[k]  := cost;
431                d1^[k] := dist;
432                b^[k]  := j;
433             end;
434
435            if min > c^[k] then
436            begin
437               min := c^[k];
438               l := k;
439               dist2 := d1^[k] + d2s^[ b^[k] ];
440            end;
441          end; { if a[k] }
442      end; { for k }
443      j := l;
444      a^[j] := false;
445      d2s^[l] := dist2;
446   end; { for i }
```

Modified Code

```
400   for i := 1 to n do
401      begin
402         a^[i] := true;
403         b^[i] := 0;
404         c^[i] := dlarge;
            d1^[i] := dlarge;
406      end;
*
*
*
421          begin
422             dist := dmtx^[j]^[k];
423
426
427
428             if dist  < d1^[k] then
429             begin
431                d1^[k] := dist;
432                b^[k]  := j;
433             end;
434
435            if min > d1^[k] then
436            begin
437               min := d1^[k];
438               l := k;
439               dist2 := d1^[k] + d2s^[ b^[k] ];
440            end;
441          end; { if a[k] }
442      end; { for k }
443      j := l;
444      a^[j] := false;
445      d2s^[l] := dist2;
446   end; { for i }
```

Node Selection Criteria - (b)Total Cost

PrimDist.pas

```
346     if lines>0 then
347         provisional_cost := (dist*ac_structure + dist2SAI*ac_cable)*penalty/lines
348     else
349         provisional_cost := zero;
```

```
346     if lines>0 then
            provisional_cost := (dist*ac_structure + dist2SAI*ac_cable)*penalty
348     else
349         provisional_cost := zero;
```

Overlapping Microgrids

ClusIntf.pas

Original Code

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


llx := llx-half;
lly := lly-half;
urx := urx+half;
ury := ury+half;

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;

GR^.MicroGridNS := RasterSize;
GR^.MicroGridEW := RasterSize;
```

Modified Code

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


llx := llx-mil;
lly := lly-mil;
urx := urx+mil;
ury := ury+mil;

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;

GR^.MicroGridNS := RasterSize;
GR^.MicroGridEW := RasterSize;
```

Attachment A

DOCUMENT OFF-LINE

This page has been substituted for one of the following:

o An oversize page or document (such as a map) which was too large to be scanned into the ECFS system.

o Microfilm, microform, certain photographs or videotape.

o Other materials which, for one reason or another, could not be scanned into the ECFS system.

The actual document, page(s) or materials may be reviewed by contacting an Information Technician at the FCC Reference Information Center, at 445 12th Street, SW, Washington, DC, Room CY-A257. Please note the applicable docket or rulemaking number, document type and any other relevant information about the document in order to ensure speedy retrieval by the Information Technician.

One CD-ROM