C-12.19

We can model this problem using a graph. We associate a vertex of the graph with each switching center and an edge of the graph with each line between two switching centers. We assign the weight of each edge to be its bandwidth. Vertices that represent switching centers that are not connected by a line do not have an edge between them.

We use the same basic idea as in Dijkstra's algorithm. We keep a variable $d[v]$ associated with each vertex v that is the bandwidth on any path from a to this vertex. We initialize the d values of all vertices to 0, except for the value of the source (the vertex corresponding to a) that is initialized to infinity. We also keep a π value associated with each vertex (that contains the predecessor vertex).

The basic subroutine will be very similar to the subroutine *Relax* in Dijkstra. Assume that we have an edge (u, v). If $\min\{d[u], w(u, v)\} > d[v]$ then we should update $d[v]$ to $\min\{d[u], w(u, v)\}$ (because the path from a to u and then to v has bandwidth $\min\{d[u], w(u, v)\}$, which is more than the one we have currently).

Algorithm Max-Bandwidth(G,a,b)

1. FOR (all vertices v in V)
   $d[v] \leftarrow 0$

2. $d[a] \leftarrow \infty$
3. $Known \leftarrow \emptyset$
4. $Unknown \leftarrow V$
5. WHILE ($Unknown \neq \emptyset$)
   $u \leftarrow ExtractMax(Unknown)$
   $Known \leftarrow Known \cup \{u\}$
   FOR (all vertices v in $Adj[u]$)
   IF ($min(d[u], w(u, v)) > d[v]$)
   $d[v] \leftarrow min(d[u], w(u, v))$

6. return (d[b])

Here, the function $ExtractMax(Unknown)$ finds the node in Unknown with the maximum value of d.

Complexity (this has not been asked for):
If we maintain $Unknown$ as a heap, then there are $n = |V|$ ExtractMax operations on it and up to $m = |E|$ changes to the values of elements in the heap. We can implement a data structure which takes $O(\log n)$ time for each of these operations. Hence, the total running time is $O((n+m)\log n)$.