Algorithm 3 pt2ptDistance2(Source indoor position p_s, destination indoor position p_t)

$v_s \leftarrow$ getHostPartition(p_s)
$v_t \leftarrow$ getHostPartition(p_t)
$doors_s \leftarrow P2D_{\sqsubset}(v_s)$
$doors_t \leftarrow P2D_{\sqsupset}(v_t)$
for each door $d_s \in doors_s$ **do**
  $np \leftarrow$ the partition in $D2P_{\sqsubset}(d_s) \setminus \{v_s\}$
  if $P2D_{\sqsubset}(np) = \{d_s\}$ **and** $np \neq v_t$ **then**
    remove d_s from $doors_s$
$dist_m \leftarrow \infty$
for each door $d_s \in doors_s$ **do**
  $doors \leftarrow \emptyset$
  for each door $d_t \in doors_t$ **do**
    if $dist_V(p_s, d_s) + dist_V(p_t, d_t) < dist_m$ **then**
      add d_t to $doors$
  initialize a min-heap H
  for each door $d_i \in \mathcal{S}_{door}$ **do**
    if $d_i \neq d_s$ **then**
      $dist[d_i] \leftarrow \infty$
    else
      $dist[d_i] \leftarrow 0$
    enheap($H, \langle d_i, dist[d_i] \rangle$)
  while H is not empty **do**
    $\langle d_i, dist[d_i] \rangle \leftarrow$ deheap(H)
    if $d_i \in doors$ **then**
      $doors \leftarrow doors \setminus \{d_i\}$
      if $dist_m > dist_V(p_s, d_s) + dist[d_i] + dist_V(p_t, d_i)$ **then**
        $dist_m \leftarrow dist_V(p_s, d_s) + dist[d_i] + dist_V(p_t, d_i)$
      if $doors = \emptyset$ **then**
        break
    mark door d_i as visited
    $parts \leftarrow D2P_{\sqsubset}(d_i)$
    for each partition $v \in parts$ **do**
      for each unvisited door $d_j \in P2D(v)$ **do**
        if $d_j \in P2D_{\sqsubset}(v)$ **then**
          if $dist[d_i] + G_{dist}.f_{d2d}(v, d_i, d_j) < dist[d_j]$ **then**
            $dist[d_j] \leftarrow dist[d_i] + G_{dist}.f_{d2d}(v, d_i, d_j)$
return $dist_m$