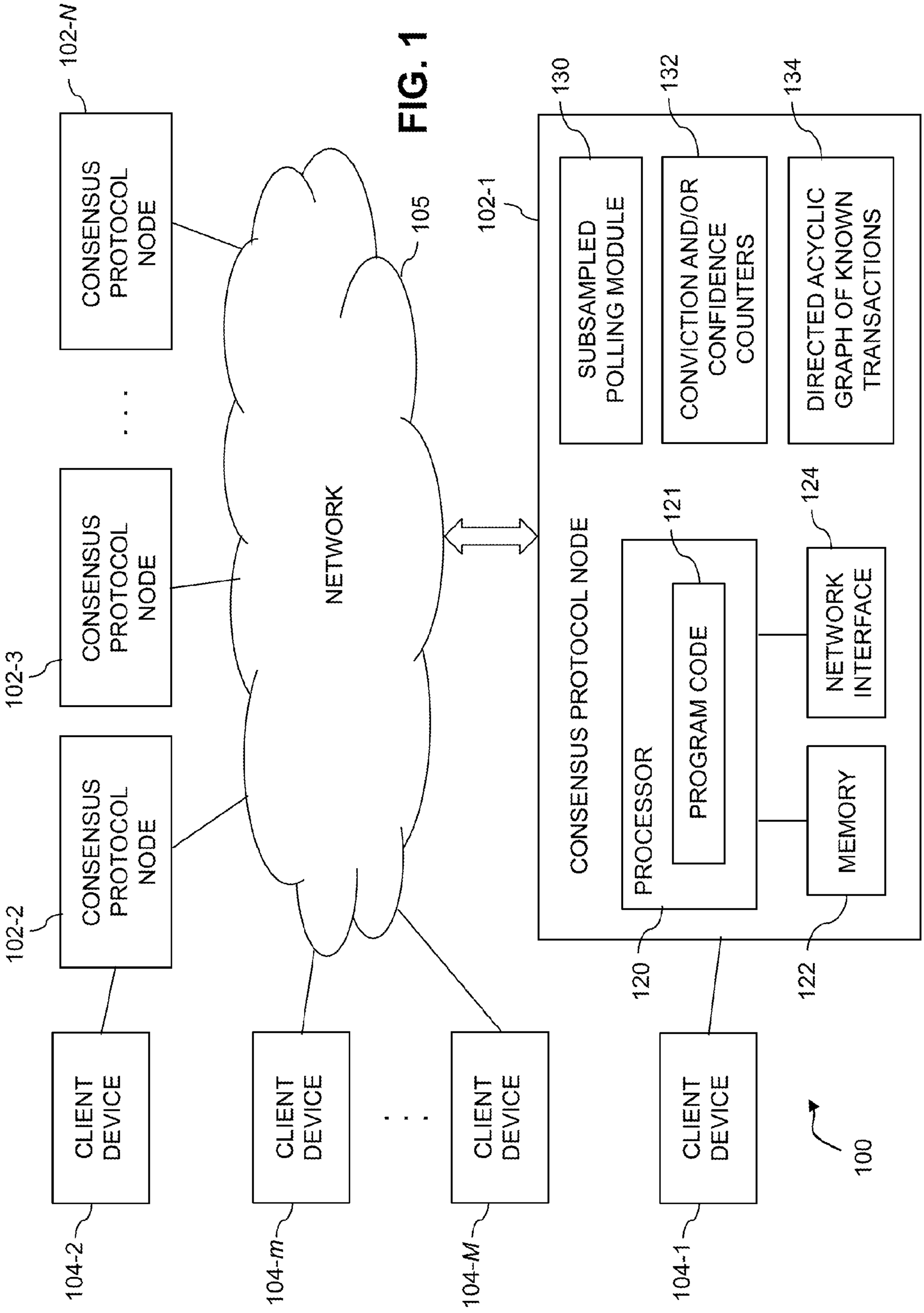
102-N
CONSENSUS PROTOCOL NODE
. . .
102-3
CONSENSUS PROTOCOL NODE
102-2
CONSENSUS PROTOCOL NODE
NETWORK
105
102-1
CONSENSUS PROTOCOL NODE
130
SUBSAMPLED POLLING MODULE
132
CONVICTION AND/OR CONFIDENCE COUNTERS
134
DIRECTED ACYCLIC GRAPH OF KNOWN TRANSACTIONS
121
PROCESSOR
PROGRAM CODE
124
NETWORK INTERFACE
120
122
MEMORY
CLIENT DEVICE
104-2
CLIENT DEVICE
104-m
CLIENT DEVICE
104-M
CLIENT DEVICE
104-1
100

FIG. 1

```
1: procedure onQuery(v, col′)
2:     if col = ⊥ then col := col′
3:     respond(v, col)
4: procedure slushLoop(u, col0 ∈ {R, B, ⊥})
5:     col := col0 // initialize with a color
6:     for r ∈ {1 . . . m} do
7:         // if ⊥, skip until onQuery sets the color
8:         if col = ⊥ then continue
9:         // randomly sample from the known nodes
10:        K := sample(N \ u, k)
11:        P := [query(v, col)  for v ∈ K]
12:        for col′ ∈ {R, B} do
13:            if P.count(col′) ≥ α · k then
14:                col := col′
15:    accept(col)
```

FIG. 2

```
1: procedure SNOWFLAKELOOP(u, col_0 ∈ {R, B, ⊥})
2:     col := col_0, cnt := 0
3:     while undecided do
4:         if col = ⊥ then continue
5:         K := SAMPLE(V \ u, k)
6:         P := [QUERY(v, col)  for v ∈ K]
7:         for col' ∈ {R, B} do
8:             if P.COUNT(col') ≥ α · k then
9:                 if col' ≠ col then
10:                    col := col', cnt := 0
11:            else
12:                if ++cnt > β then ACCEPT(col)
```

FIG. 3

```
1:  procedure SNOWBALLLOOP(u, col_0 ∈ {R, B, ⊥})
2:      col := col_0, lastcol := col_0, cnt := 0
3:      d[R] := 0, d[B] := 0
4:      while undecided do
5:          if col = ⊥ then continue
6:          K := SAMPLE(V \ u, k)
7:          P := [QUERY(v, col)  for v ∈ K]
8:          for col' ∈ {R, B} do
9:              if P.COUNT(col') ≥ α · k then
10:                 d[col']++
11:     if d[col'] > d[col] then
12:         col := col'
13:     if col' ≠ lastcol then
14:         lastcol := col', cnt := 0
15:     else
16:         if ++cnt > β then ACCEPT(col)
```

FIG. 4

```
1: procedure AVALANCHELOOP
2:     while true do
3:         find T that satisfies T ∈ 𝒯 ∧ T ∉ 𝒬
4:         𝒦 := SAMPLE(𝒩 \ u, k)
5:         P := Σ_{v∈𝒦} QUERY(v, T)
6:         if P ≥ α · k then
7:             c_T := 1
8:             // update the preference for ancestors
9:             for T' ∈ 𝒯 : T' ←* T do
10:                if d(T') > d(P_{T'}.pref) then
11:                    P_{T'}.pref := T'
12:                if T' ≠ P_{T'}.last then
13:                    P_{T'}.last := T', P_{T'}.cnt := 0
14:                else
15:                    ++P_{T'}.cnt
16:        // otherwise, c_T remains 0 forever
17:        𝒬 := 𝒬 ∪ {T}    // mark T as queried
```

FIG. 5

```
1: procedure INIT
2:     T := ∅  // the set of known transactions
3:     Q := ∅  // the set of queried transactions
4: procedure ONGENERATETX(data)
5:     edges := {T' ← T : T' ∈ PARENTSELECTION(T)}
6:     T := Tx(data, edges)
7:     ONRECEIVETX(T)
8: procedure ONRECEIVETX(T)
9:     if T ∉ T then
10:        if P_T = ∅ then
11:            P_T := {T}, P_T.pref := T
12:            P_T.last := T, P_T.cnt := 0
13:        else P_T := P_T ∪ {T}
14:    T := T ∪ {T}, c_T := 0.
```

FIG. 6

1: **function** isPreferred(T)
2: **return** $T = \mathcal{P}_T$.pref
3: **function** isStronglyPreferred(T)
4: **return** $\forall T' \in \mathcal{T}, T' \overset{*}{\leftarrow} T$: isPreferred(T')
5: **function** isAccepted(T)
6: **return**
$((\forall T' \in \mathcal{T}, T' \leftarrow T$: isAccepted(T'))
$\wedge |\mathcal{P}_T| = 1 \wedge d(T) > \beta_1)$ // safe early commitment
$\vee (\mathcal{P}_T.\text{cnt} > \beta_2)$ // consecutive counter

7: **procedure** onQuery(j, T)
8: onReceiveTx(T)
9: respond(j, isStronglyPreferred(T))

FIG. 7

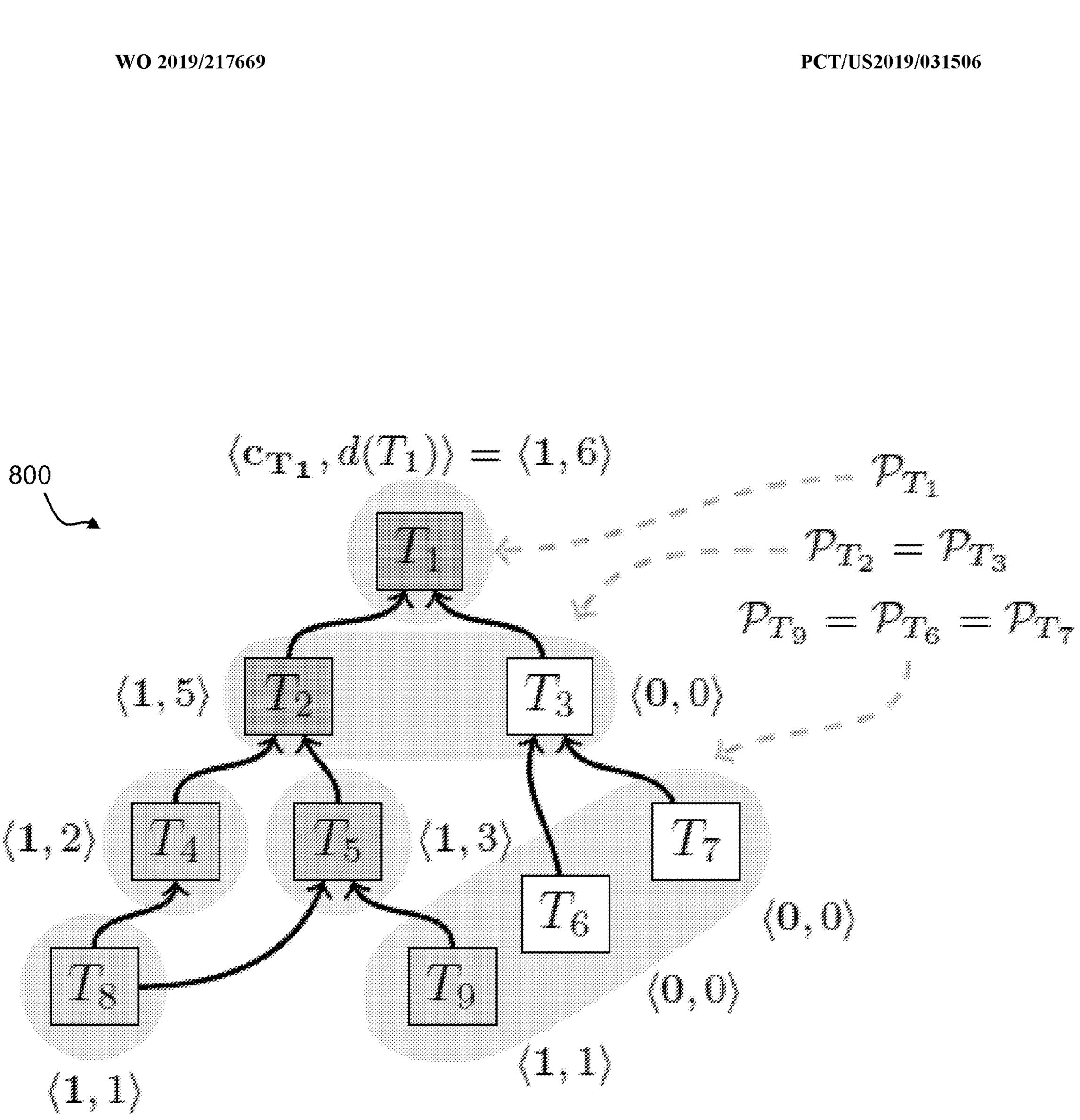
800
$\langle c_{T_1}, d(T_1)\rangle = \langle 1, 6\rangle$
T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8
T_9
$\langle 1, 5\rangle$
$\langle 0, 0\rangle$
$\langle 1, 2\rangle$
$\langle 1, 3\rangle$
$\langle 0, 0\rangle$
$\langle 0, 0\rangle$
$\langle 1, 1\rangle$
$\langle 1, 1\rangle$
p_{T_1}
$p_{T_2} = p_{T_3}$
$p_{T_9} = p_{T_6} = p_{T_7}$

FIG. 8

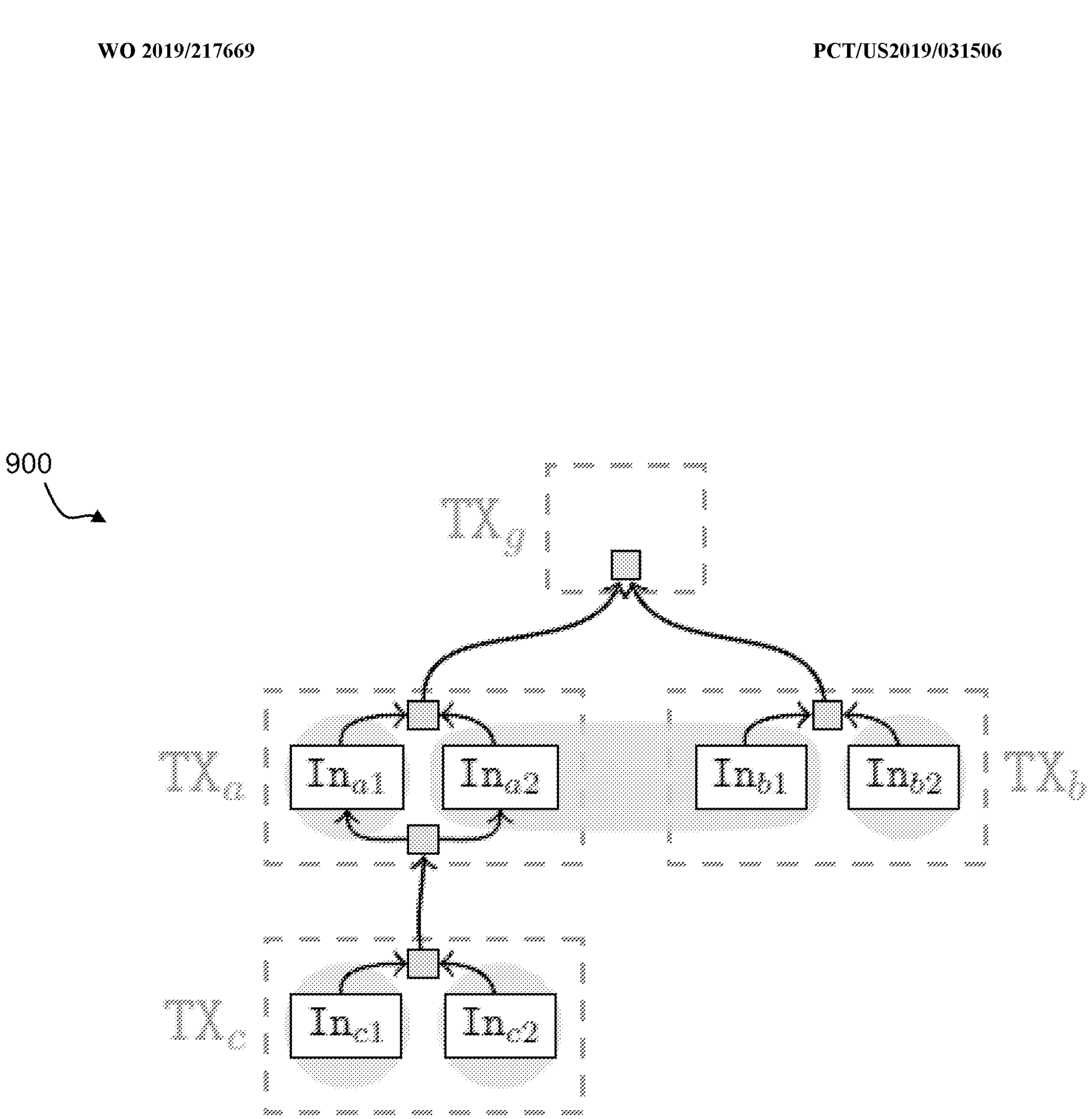

FIG. 9