

1/12

FIG. 1

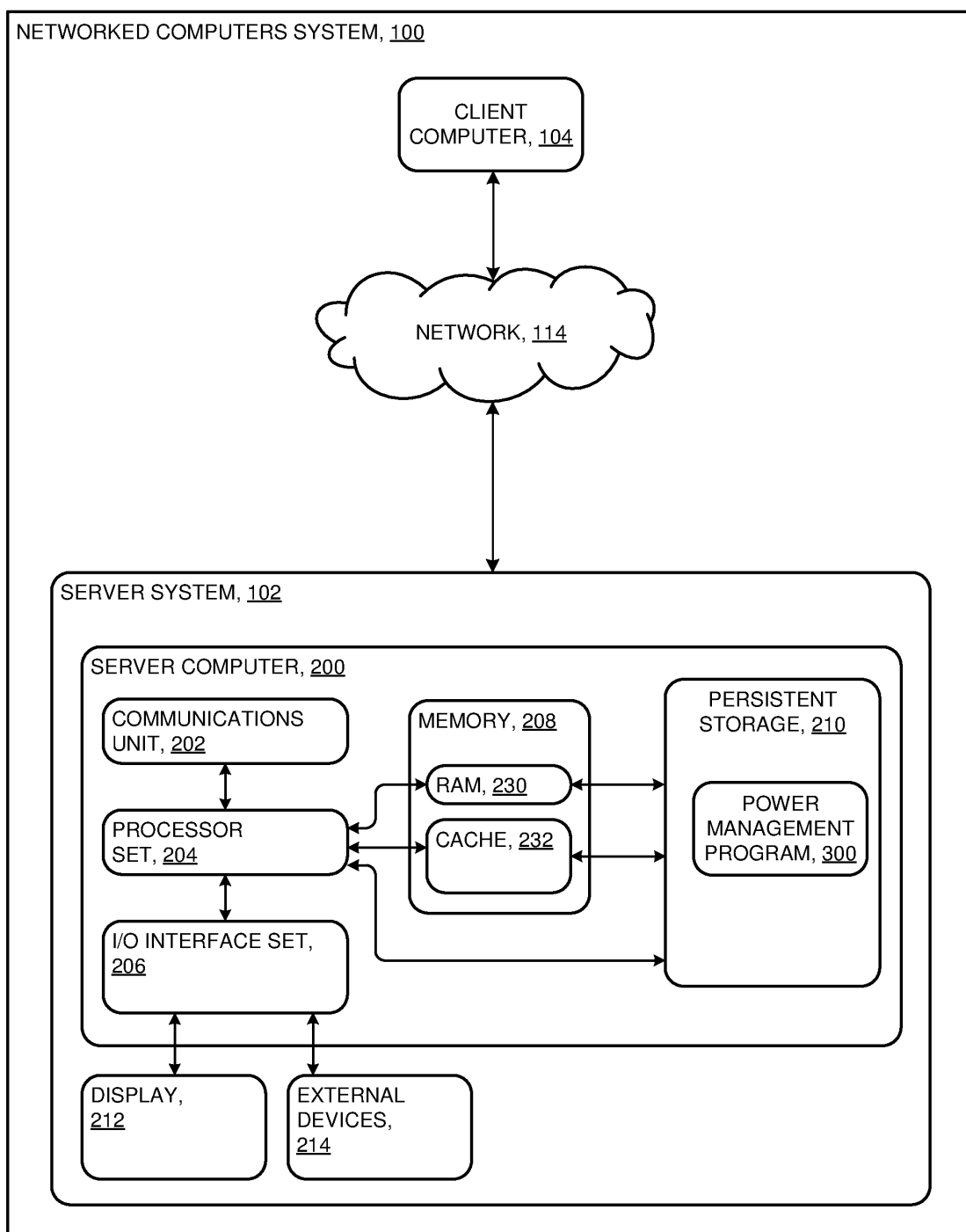


Fig. 2

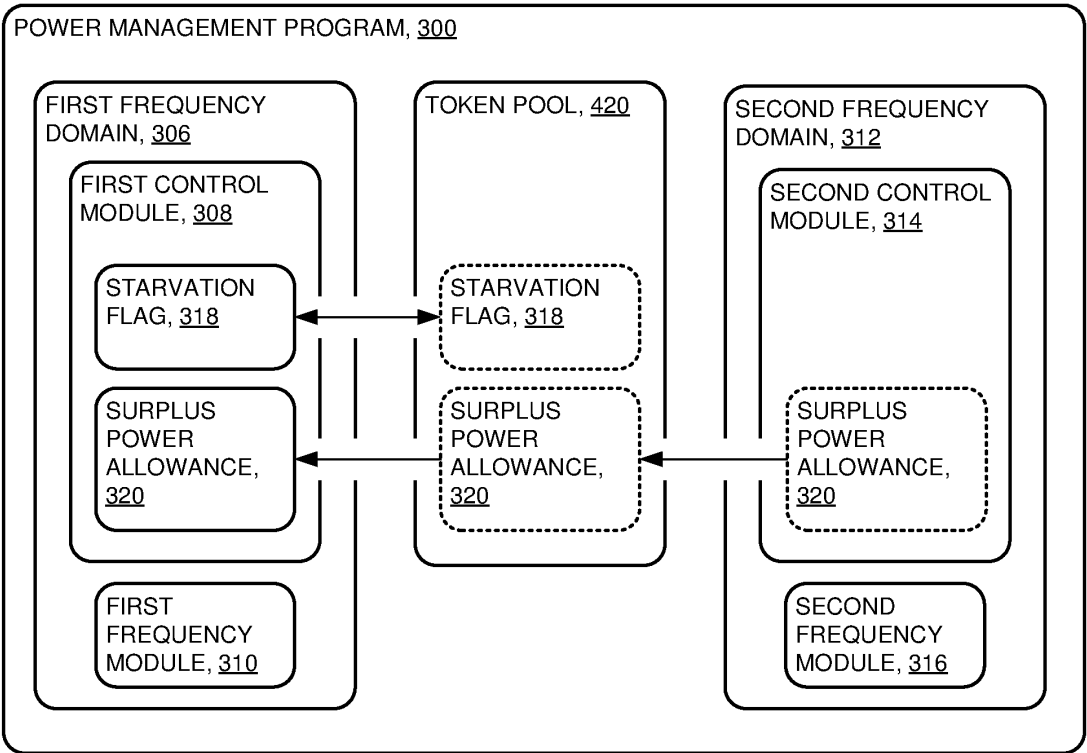
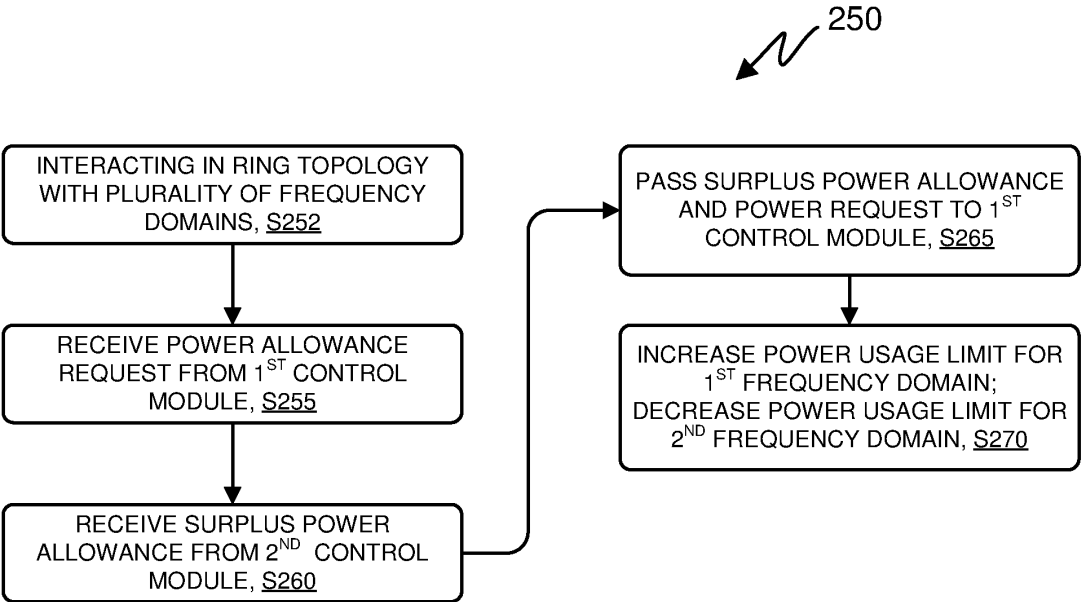


Fig. 3

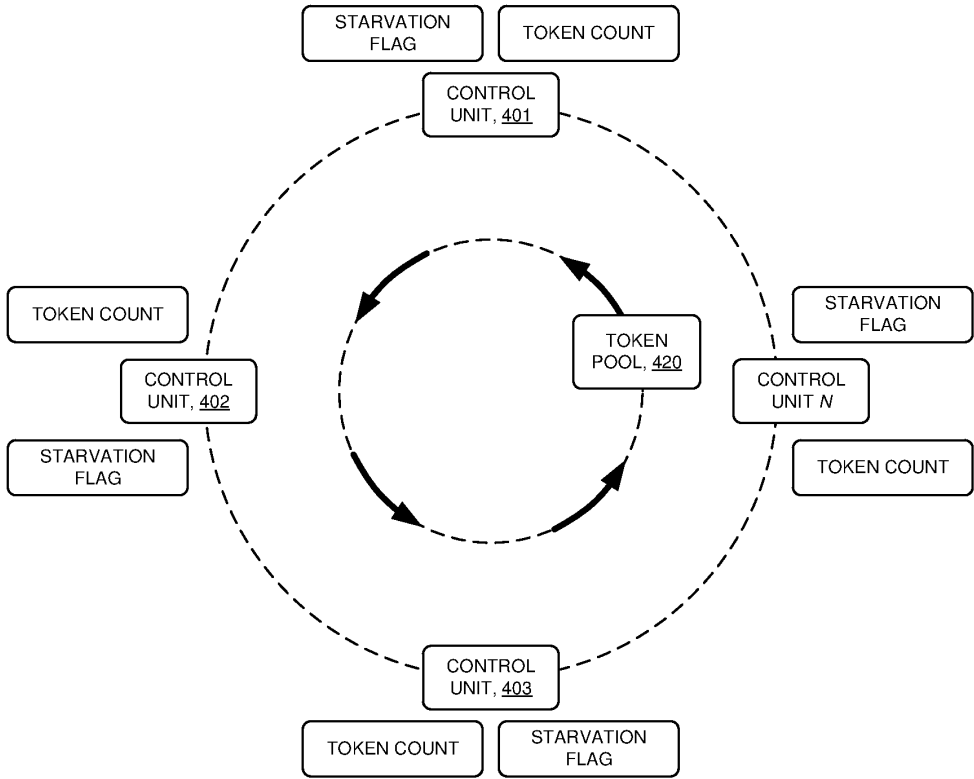


Fig. 4A

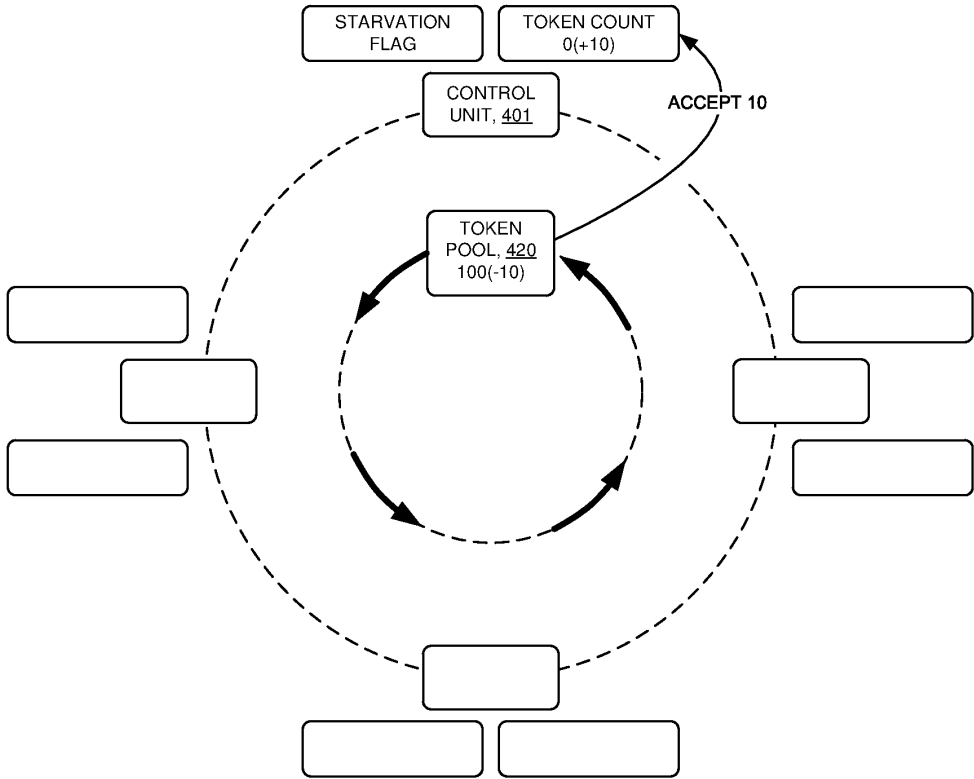


Fig. 4B

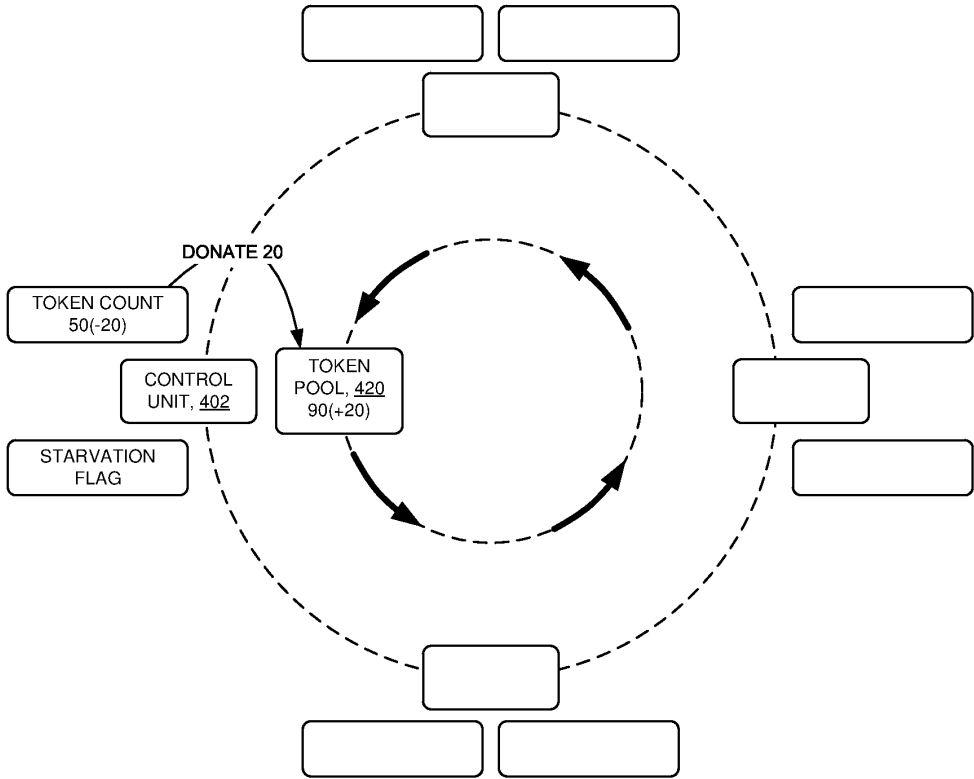


Fig. 4C

6/12

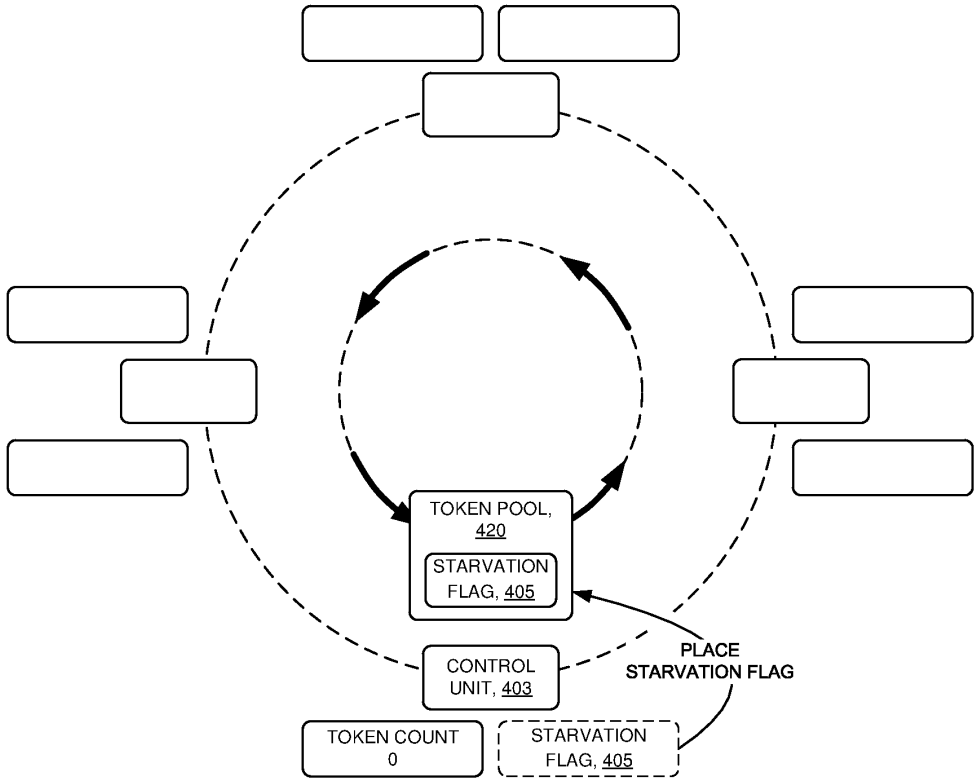


Fig. 4D

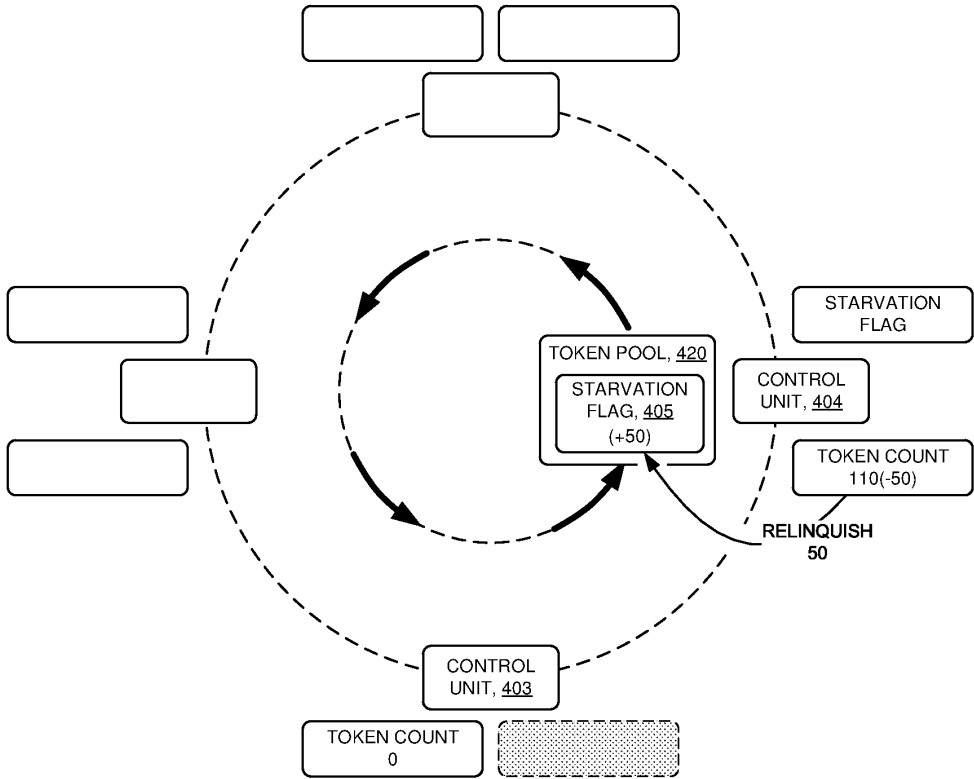


Fig. 4E

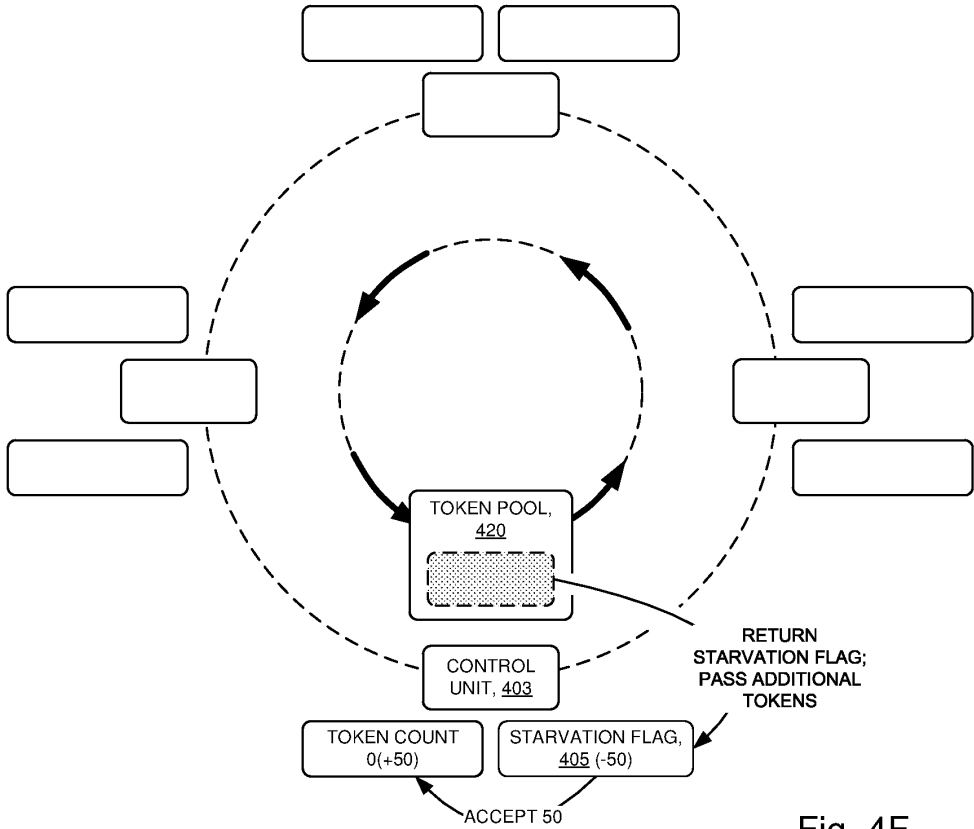


Fig. 4F

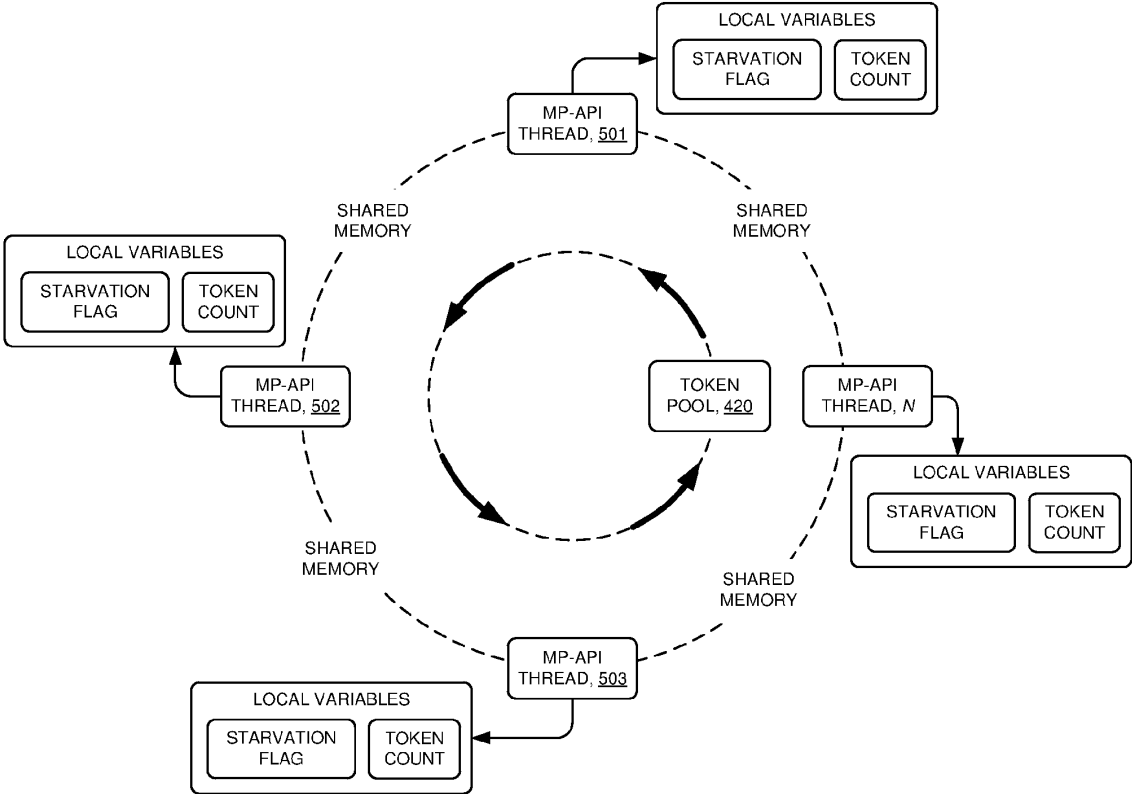


Fig. 5

10/12

Algorithm 1 TokenGov Initialization

```

1: procedure INIT
2:    $tokenPool \leftarrow 200$ 
3:    $turn \leftarrow 0$ 
4:   For N FDs,  $P_i.tokens \leftarrow 0$ ,  $i \in [0, N)$ 
5:   Spawn N Policy Daemon  $P_N$ 
6:   for i in 0..15 do
7:     DAEMON(i)

```

Fig. 6A

Algorithm 2 mips

```

1: procedure CALCMIPS(P)
2:    $ips \leftarrow \max(IPS_0, IPS_1, \dots, IPS_{15})$ 
3:    $std \leftarrow \sigma(mips[0], mips[1], \dots, mips[15])$ 
4:    $mips[top] \leftarrow ips$ 
5:    $std2 \leftarrow \sigma(mips[0], mips[1], \dots, mips[15])$ 
6:   /* if change in std is <5% then set new MIPS */
7:   if  $(|std - std2| \div std) < 0.05$  then
8:      $MIPS \leftarrow \max(mips[0], mips[1], \dots, mips[15])$ 
9:   /* Else ignore current IPS */
10:   $P.MIPS \leftarrow MIPS$ 

```

Fig. 6B

11/12

Algorithm 3 Increase token request count

```

1: procedure INCTOKENS(P)
2:   /* tokens to be accepted */
3:   if  $P_i.lastTokenRequest * 2 > P_i.request$  then
4:      $P_i.request \leftarrow P_i.request$ 
5:   else
6:      $P_i.request \leftarrow P_i.lastTokenRequest * 2$ 
7:   if  $P_i.request > 100$  then
8:      $P_i.request = 100$ 
9:    $lastTokenRequest \leftarrow P_i.request$ 

```

Fig. 6C

12/12**Algorithm 4** Daemon Thread per FD

```

1: procedure DAEMON(ID)
2:    $i \leftarrow ID$ 
3:    $winSize \leftarrow 10$ 
4:    $mips[winSize] \leftarrow 0$ 
5:    $MIPS \leftarrow 0$ 
6:    $prevMIPS \leftarrow 0$ 
7:   /* Track MIPS in case of load increase */
8:    $trackMIPS \leftarrow False$ 
9:   /* Requesting token is incremental starting from 1 */
10:   $lastTokenRequest \leftarrow 0$ 

11:  while true do
12:    /* Wait for the tokenPool to reach to use */
13:     $while(turn \neq i);$ 
14:     $P_i.load \leftarrow \max(L_0, L_1, \dots, L_{15})$ 
15:     $P_i.required = P_i.load$ 

16:    if  $P_i.required > P_i.tokens$  then
17:       $request = P_i.required - P_i.tokens$ 
18:       $incTokens(P)$ 
19:       $calcMIPS(P_i)$ 
20:      /* If we already gave token previously but MIPS
21:      do not increase then exit */
22:      if  $trackMIPS == True$  AND  $MIPS < prevMIPS$  then
23:        goto exit
24:      else
25:        if  $trackMIPS == False$  then
26:           $trackMIPS \leftarrow True$ 
27:        /* Accept tokens from pool */
28:        if  $tokenPool > P_i.request$  then
29:           $P_i.tokens \leftarrow P_i.tokens + P_i.request$ 
30:           $tokenPool \leftarrow tokenPool - P_i.request$ 
31:        else
32:           $P_i.tokens \leftarrow P_i.tokens + tokenPool$ 
33:           $tokenPool \leftarrow 0$ 
34:        else
35:          /* Donate tokens */
36:           $extra \leftarrow P_i.tokens - P_i.required$ 
37:           $tokenPool \leftarrow tokenPool + extra$ 
38:           $P_i.tokens \leftarrow P_i.tokens - extra$ 
39:      exit:
40:         $turn \leftarrow (turn + 1) \bmod N$ 
41:        Set frequency  $\leftarrow P_i.tokens$ 
42:      /* While loop ends here */

```

Fig. 6D