Supplemental information

Honeycomb: An open-source distributed system for smart buildings

Tian Xing, Hu Yan, Kailai Sun, Yifan Wang, Xuetao Wang, and Qianchuan Zhao
SUPPLEMENTAL ITEMS

S1. Asynchronous Leaderless Spanning Tree

This supplement shows the pseudocode and function modules of the ALST algorithm. Table S1 lists the main functions of the proposed algorithms.

| Function | Definition |
|----------|------------|
| reset() | reset the variables, including setting flag = false, parent = -1, child = null, and all elements of edges = false. |
| stop() | stops the execution of the application |
| send(a, b) | sends the message a to the port set b. |
| read() | waiting for receipt of messages and will return the port of the sender. This function is blocking. |
| ALST() | ALST algorithm module, is the parts of Algorithm 1 from line 6 to line 41. |
| HbD() | in time, this function removes the connection to the neighbor and returns the port of the neighbor. |

Algorithm 1. Asynchronous Leaderless Spanning Tree (ALST):

```
Input: x
Output: leader, parent, child
1  reset()
2  x_min ← x
3  flag ← true
4  send(x_min, "search"), nbres
5  port, (rx, msg) ← read()
6  while true do
7    if rx <= x_min then
8      if rx < x_min then
9        x_min ← rx
10       reset()
11      end if
12     edges[port] ← true
13     switch msg do
14       case "search"
15         if (flag == false) then
16           flag ← true
17           parent ← port
18         end if
19       case "join"
20         append port to child
21       case "end"
22         if (port == parent) then
23           send("end", x_min, child)
24         end if
25       end if
26     end switch
27     if all elements of edges are true then
28       if x_min = x then
29         leader ← true
30         send(x_min, "end"), child
31         stop()
32       else
33         leader ← false
34         send(x_min, "join"), parent
35       end if
36     end if
37   end while
38   stop()
```
S2. ALST-based generation and maintenance of communications

This supplement shows the pseudocode of ALST-based generation and maintenance of communications.

Algorithm 2. ALST-based generation and maintenance of communications:

Input: x
Output: leader, parent, child
1 reset()
2 $x_{min} \leftarrow x$
3 flag $\leftarrow$ true
4 step $\leftarrow 1$
5 send($x_{min}$, “search”), nbrs
6 port, (rx, msg) $\leftarrow$ read()
7 while true do
8 if step $== 1$ then
9 ALST()
10 step $\leftarrow 2$
11 end if
12 if step $== 2$ then
13 if port $== HBD()$ then
14 remove port from edges
15 if port $== parent$ then
16 reset()
17 $x_{min} \leftarrow x$
18 flag $\leftarrow$ true
19 step $\leftarrow 1$
20 send($x_{min}$, “search”), nbrs
21 port, (rx, msg) $\leftarrow$ read()
22 end if
23 if port in child then
24 remove port from child
25 end if
26 else
27 if port, (rx, msg) $\leftarrow$ read() then
28 if port $== parent$ then
29 reset()
30 $x_{min} \leftarrow x$
31 flag $\leftarrow$ true
32 step $\leftarrow 1$
33 if $rx > x_{min}$ then
34 send($x_{min}$, “search”), nbrs
35 end if
36 else
37 if port in child then
38 remove port from child
39 end if
40 edges[port] $\leftarrow$ false
41 if $rx < x_{min}$ then
42 step $\leftarrow 1$
43 end if
44 if $rx > x_{min}$ then
45 send($x_{min}$, “search”), port
46 end if
47 if $rx == x_{min}$ then
48 edges[port] $\leftarrow$ true
49 switch msg do
50 case “search”
51 send($x_{min}$, “search”), port
52 case “join”
SUPPLEMENTAL EXPERIMENTAL PROCEDURES

S3. Evaluating the scalability of Honeycomb

This supplemental experiment shows the performance of Honeycomb at different scales. To evaluate the scalability of Honeycomb, we calculated the communication cost of constructing a communication spanning tree. We conducted the experiments on a computer with AMD R7-4800H CPU and 16GB RAM, simulating smart nodes with CPU processes. Linear-shaped, square-shaped, and cube-shaped topologies are set up for different node scales (parameters settings are listed in Table S2). Considering the parallelism of smart nodes, we regard the running time of the root node as the time to construct the spanning tree, and count the sum of the number of messages for all nodes. The experiments were repeated five times for each case, and then the results were averaged, as shown in Figure S1.

| Number of smart nodes | Line-shaped topology | Square-shaped topology | Cube-shaped topology |
|-----------------------|----------------------|------------------------|---------------------|
| 4                     | 4                    | 2×2                    | 1×2×2               |
| 16                    | 16                   | 4×4                    | 2×2×4               |
| 64                    | 64                   | 8×8                    | 4×4×4               |
| 128                   | 128                  | 8×16                   | 4×4×8               |
| 256                   | 256                  | 16×16                  | 4×8×8               |
| 384                   | 384                  | 16×24                  | 6×8×8               |
| 512                   | 512                  | 16×32                  | 8×8×8               |
| 640                   | 640                  | 20×32                  | 8×8×10              |
| 768                   | 768                  | 24×32                  | 8×8×12              |

Figure S1. Running parameters of spanning tree construction at different scales. (A) Running time. For the linear fitted curves, the R-Square values (coefficient of determination) of the topologies are 0.9943, 0.9742, and 0.9728. (B) The number of messages. For the quadratic fit curves, the R-Squares for three topologies are 0.9997, 0.9978, and 0.9993.
The results of the experiments show that: (1) The running time is linear with the scales of smart nodes, which is consistent with the time complexity $O(diam)$ in the theoretical analysis. (2) The number of messages is quadratic with the scales of smart nodes, also in agreement with the theoretical analysis (the number of edges $|E|$ of these topologies is proportional to the number of nodes $n$. Thus, the total number of messages $O(n|E|)$ can be approximated as $O(n^2)$. In fact, the topologies of single-layer and multi-layer BAS can be regarded as square-shaped and cubic-shaped (the linear-shaped topology is an extreme case). It can be found that these BASs have low communication at a large scale and good scalability.