Symbolic ns-3 for Efficient Exhaustive Testing: Design, Implementation, and Simulations

Jianfei Shao, Minh Vu, Mingrui Zhang, Asmita Jayswal, **Lisong Xu**
School of Computing, University of Nebraska-Lincoln

https://symbolicns3.github.io

Outline

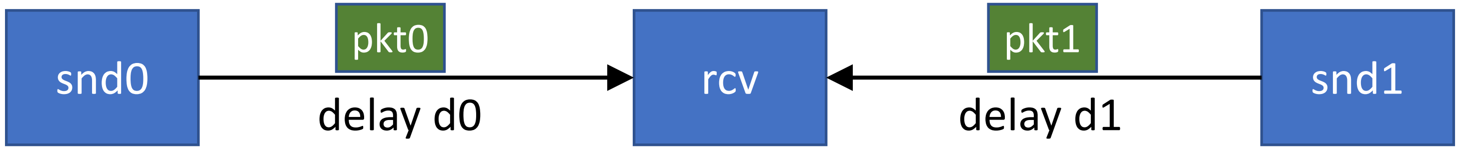
- ➡ Why Symbolic ns-3 (sym-ns-3)?
- How it works?
- How to make it more efficient?
- Conclusions

Exhaustive Testing

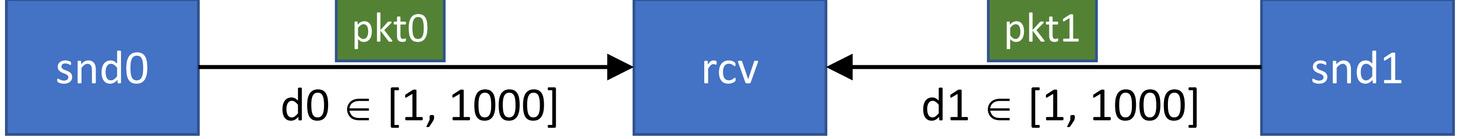
- What is it?
    - Exhaustively test something (protocol/network) for all possible cases
- When do we need it?
    - Evaluate all possible performance of a protocol/network
    - Find the worst-case performance of a protocol/network
    - Detect the bugs of a protocol/network

Exhaustive Testing Example 1

- Two senders each sends a packet to the receiver simultaneously



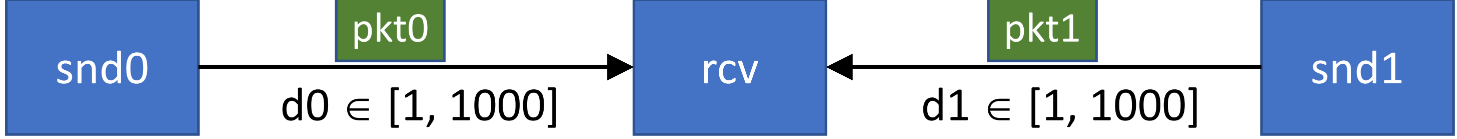
- Problem: What are all possible arrival time differences?
- Measurement: diff = Arrival time of pkt0 – arrival time of pkt1
- All possible link delays
  - d0 $\in$ [1, 1000] ms
  - d1 $\in$ [1, 1000] ms



Using ns-3

- How to find all possible diff values?
  - ns-3 script simulates the network for a *specific* (d0, d1) and reports diff
  - shell script runs the ns-3 script for *all possible* (d0, d1)
- Simulation result
  - All reported diff values = [-999, 999] ms
- Simulation time
  - The simulation time for one (d0, d1) ≈ 0.5 seconds
  - Total number of (d0, d1) = 1000 x 1000 = 1,000,000
  - Total simulation time for all possible (d0, d1) ≈ **6 days**
- **Exhaustive testing is time-consuming with ns-3!**

Using Our sym-ns-3



- How to find all possible diff values?
  - sym-ns-3 script simulates the network for a *symbolic* (d0, d1) and reports diff
- Simulation result
  - All reported diff values = [-999, 999] ms
  - **Same** simulation results as ns-3
- Simulation time
  - The simulation time for a symbolic (d0, d1) ≈ 1 minute
  - **Significantly faster** than ns-3
- **sym-ns-3 is more efficient for exhaustive testing than ns-3**

Outline

- Why sym-ns-3?
- How it works?
- How to make it more efficient?
- Conclusions

sym-ns-3

- Goal
  - Efficient exhaustive testing
- How?
  - Based on symbolic execution
  - Simulates a group of equivalent cases together instead of each case separately

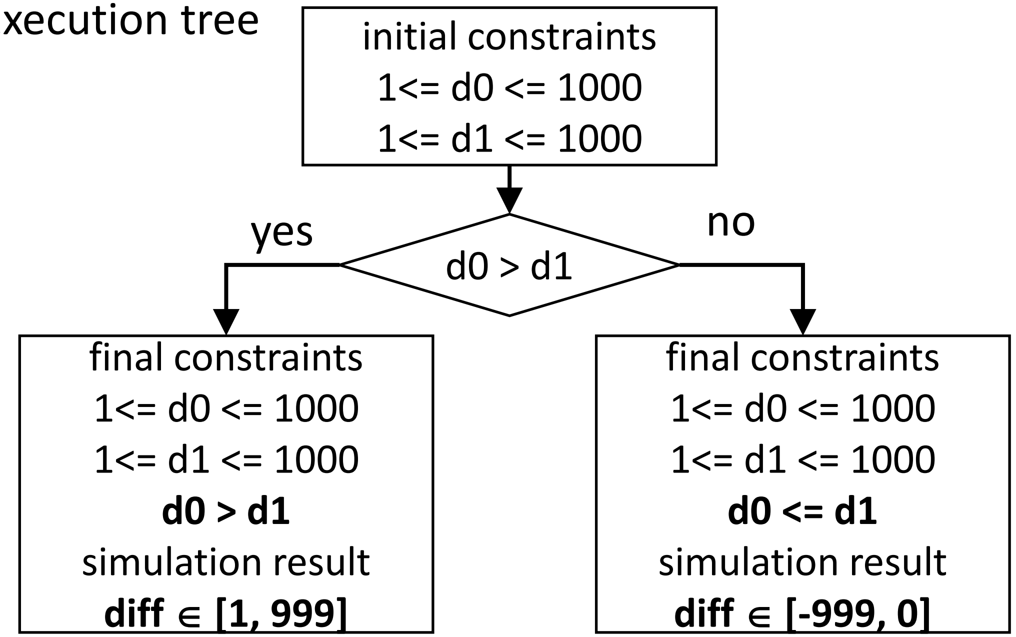
Background on Symbolic Execution

- A variable may have a symbolic value (a set of values specified by constraints) instead of only a specific value.
- When a program is executed symbolically, both branches instead of one branch of an if statement are explored.

Pseudocode example

```
sym 1<= d0 <= 1000
sym 1<= d1 <= 1000
if (d0 > d1){
     …// simulate accordingly
     diff = d0 - d1;
} else {
 …// simulate accordingly
     diff = d0 - d1;
}
```

Execution tree



initial constraints
1<= d0 <= 1000
1<= d1 <= 1000

d0 > d1

yes

no

final constraints
1<= d0 <= 1000
1<= d1 <= 1000
d0 > d1
simulation result
diff ∈ [1, 999]

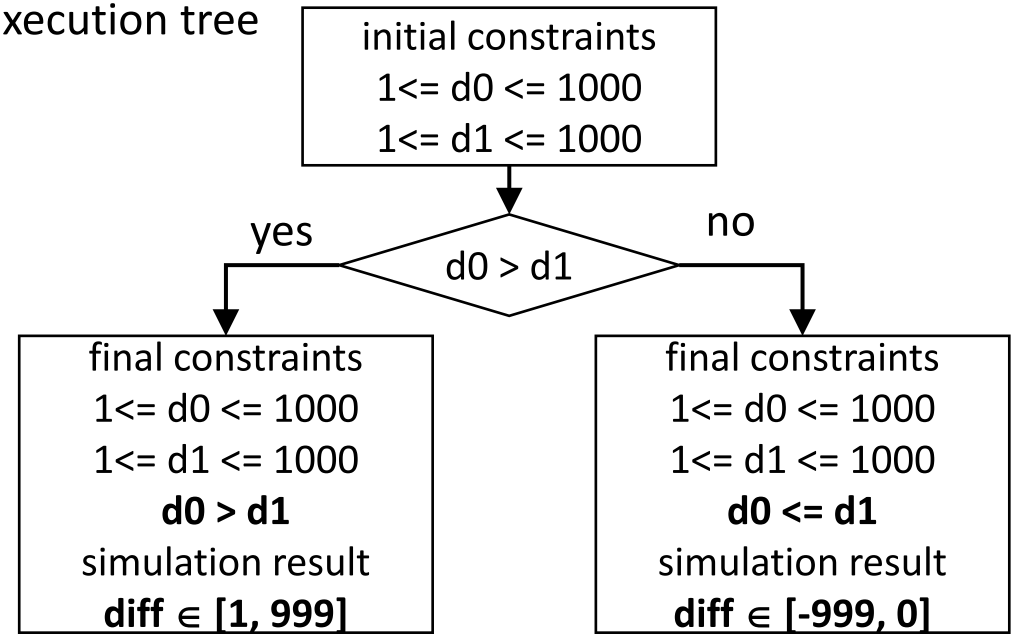
final constraints
1<= d0 <= 1000
1<= d1 <= 1000
d0 <= d1
simulation result
diff ∈ [-999, 0]

Symbolic execution runs equivalent cases together as branches, and thus is more efficient for exhaustive testing.

- Branch 1
  - All the equivalent cases following the same red execution path
- Branch 2
  - All the equivalent cases following the same green execution path

Execution tree

```
sym 1<= d0 <= 1000
sym 1<= d1 <= 1000
if (d0 > d1){
      …// simulate accordingly
      diff = d0 – d1;
} else {
 …// simulate accordingly
      diff = d0 – d1;
}
```

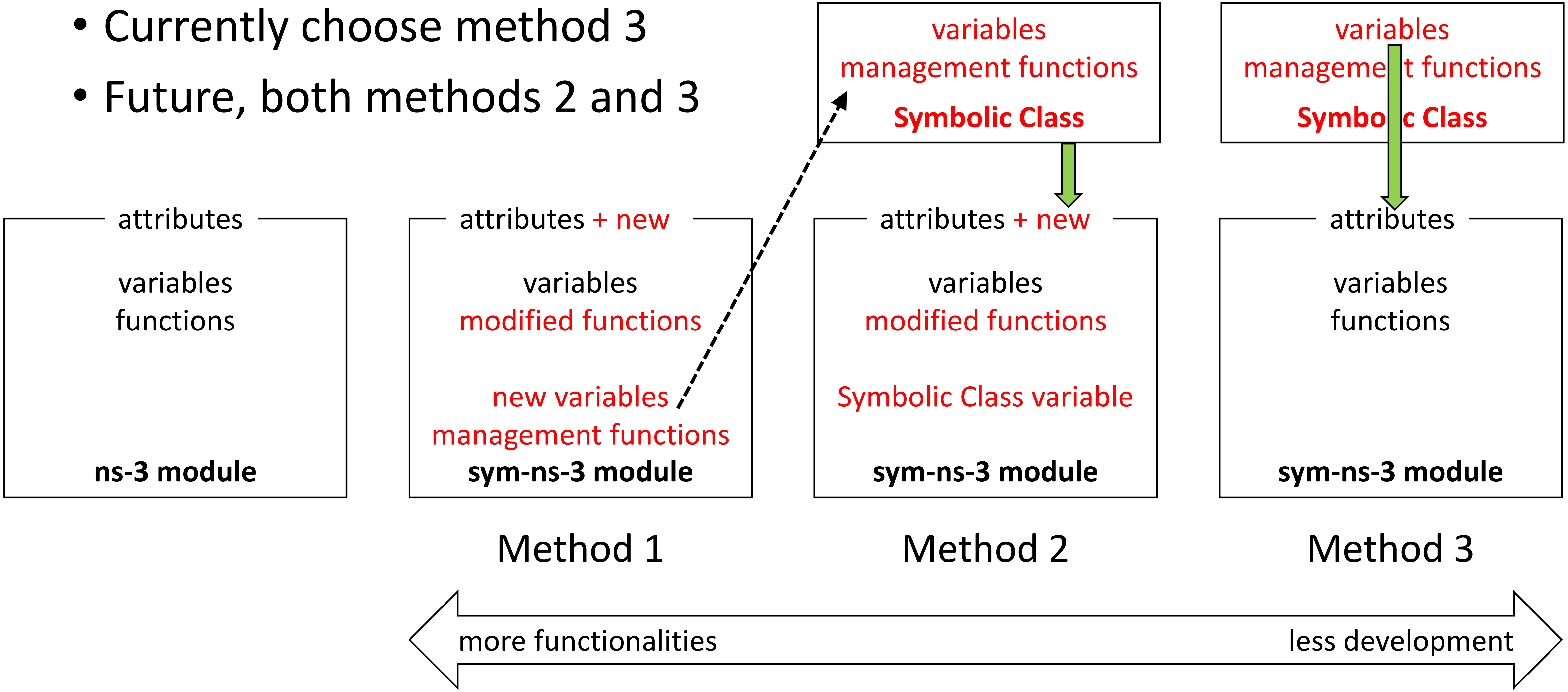


Branch 1 **Branch 2**

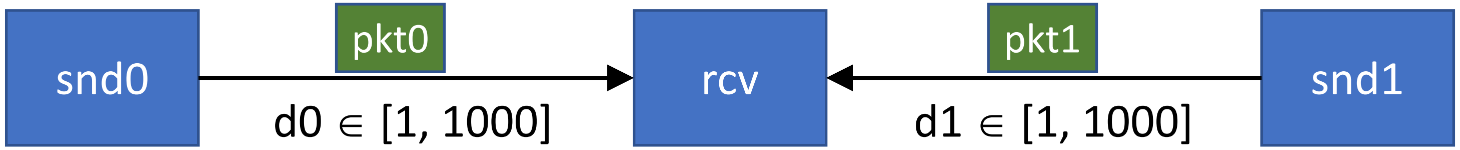
```
sym 1<= d0 <= 1000
sym 1<= d1 <= 1000
if (d0 > d1){
      …// simulate accordingly
      diff = d0 – d1;
} else {
 …// simulate accordingly
      diff = d0 – d1;
}
```

How sym-ns-3 modifies ns-3?

- Have explored three different methods to modify ns-3
- Currently choose method 3
- Future, both methods 2 and 3



Example 1 scripts of ns3 vs sym-ns-3



```
... // Other setup code




uint32_t d0 = 1;
p2p[0].SetChannelAttribute("Delay",TimeValue(Time(d0)));




uint32_t d1 = 1;
p2p[1].SetChannelAttribute("Delay",TimeValue(Time(d1)));

... // Simulation execution
```

ns-3 script

```
... // Other setup code

Ptr<Symbolic> sym0 = CreateObject<Symbolic>();
sym0->SetMinMax(1, 1000);
uint32_t d0 = sym0->GetSymbolicUintValue();
p2p[0].SetChannelAttribute("Delay",TimeValue(Time(d0)));

Ptr<Symbolic> sym1 = CreateObject<Symbolic>();
sym1->SetMinMax(1, 1000);
uint32_t d1 = sym1->GetSymbolicUintValue();
p2p[1].SetChannelAttribute("Delay",TimeValue(Time(d1)));

... // Simulation execution
```

Symbolic Class

a symbolic management function

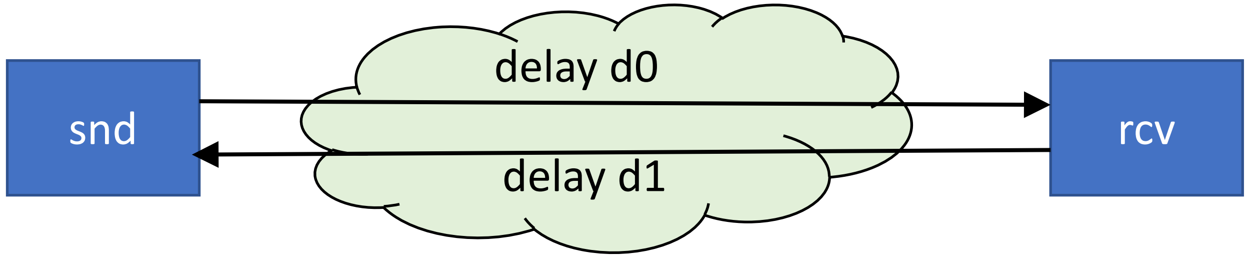
get symbolic value

use existing attribute

sym-ns-3 script (method 3)

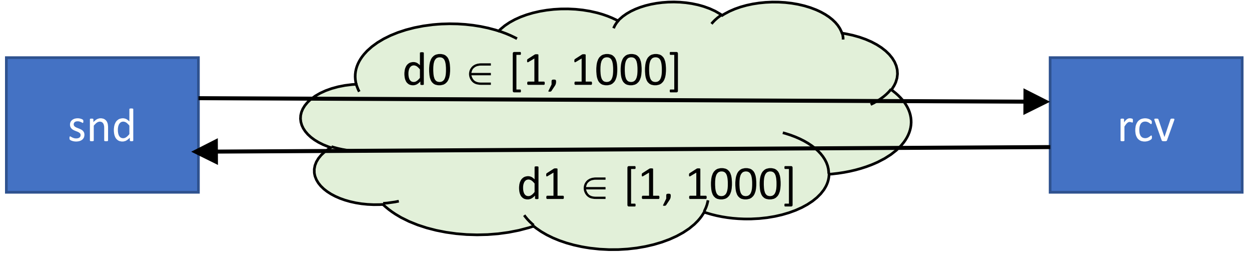
Exhaustive Testing Example 2 – TCP Performance

- Problem: Find all possible performance of TCP



- All possible network delays
  - Forward delay: d0 ∊ [1, 1000] ms
  - Backward delay: d1 ∊ [1, 1000] ms
- Measurement: Number of received data packets in 2000 ms
- Limit the max number of data packets to 2 in order to manually check the simulation results

Results



snd

rcv

d0 ∈ [1, 1000]

d1 ∈ [1, 1000]

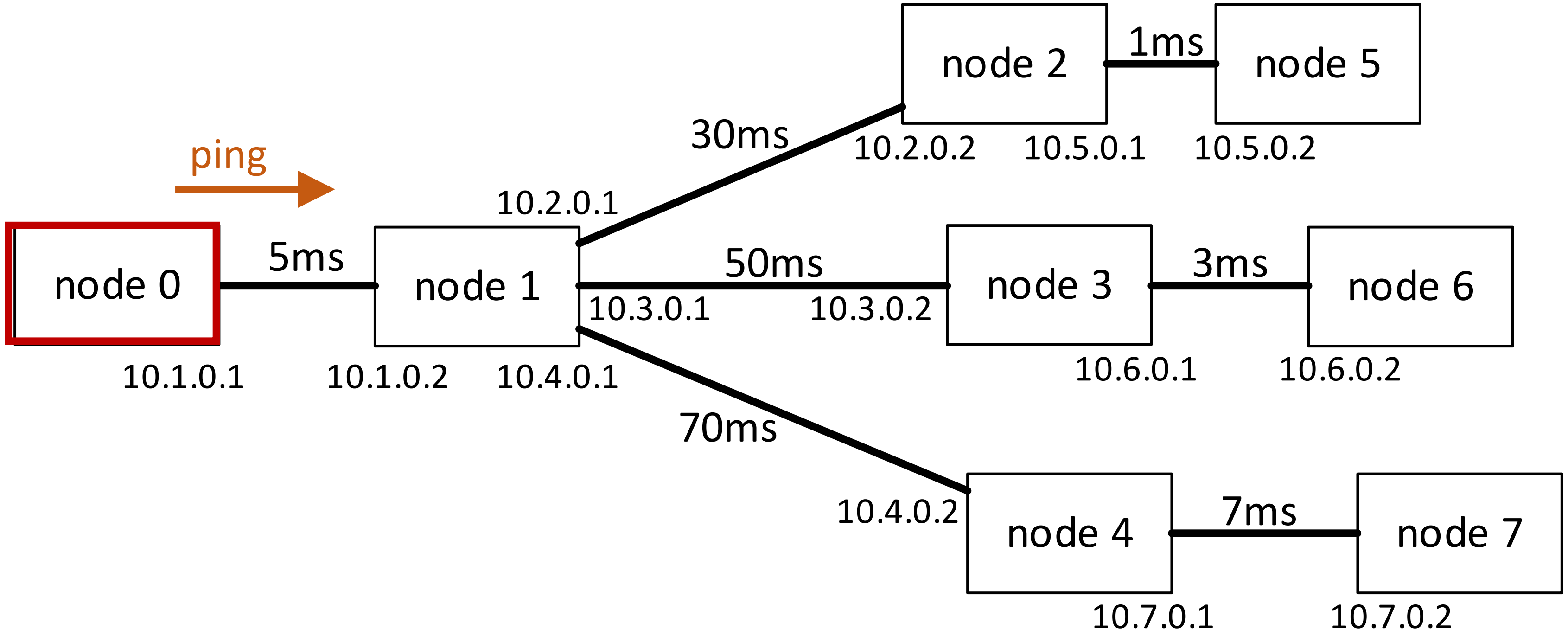
- ns-3
  - Take **about 6 days** to run **1000x1000 (d0, d1) cases**, each about 0.5 seconds
- sym-ns-3
  - Take **about 3 hours** and explore **about 140 branches** for symbolic (d0, d1)
- Simulation result summary

2d0 + d1 (3-way handshake)	3d0 + 2d1 (3-way handshake + 1 RTT)	Num of received data packets	Comments
[1999, 3000]	[2999, 5000]	0	
[1000, 1998]	[1999, 3497]	1	
[3, 1331]	[5, 1998]	2	

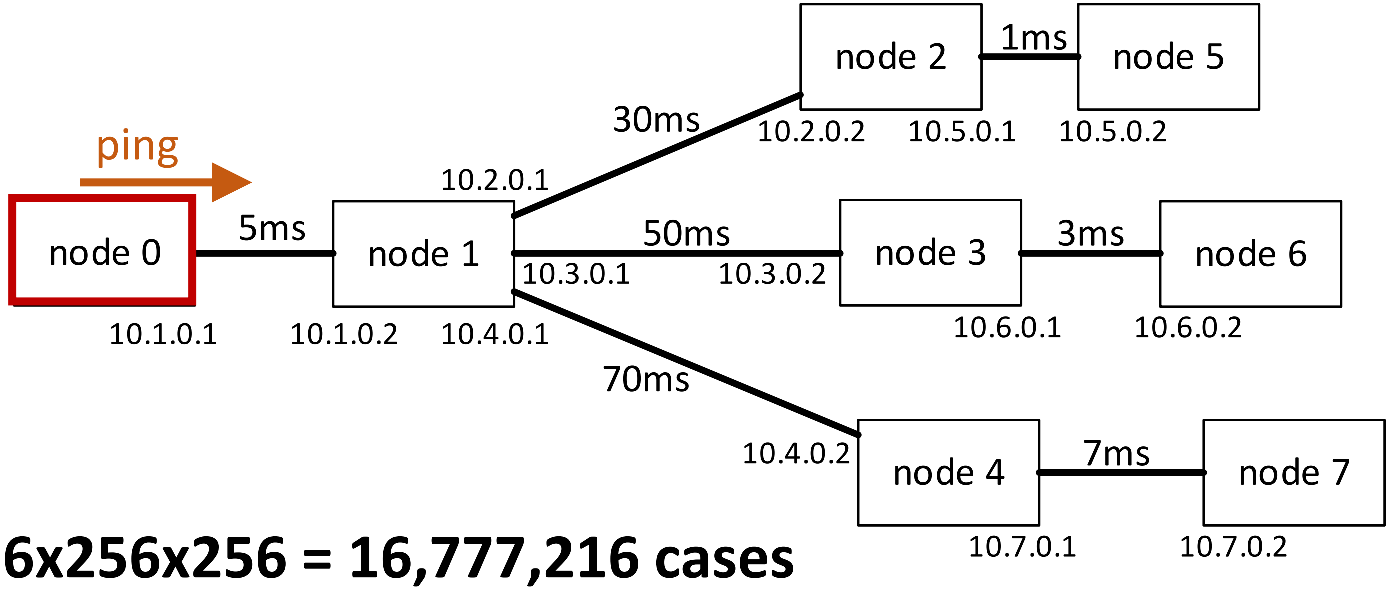
Takes only about 3 hours for 1 millions of TCP tests

Exhaustive Testing Example 3 – IP Reachability

- Problem: Find all 10.x.x.x addresses reachable from node 0 using ping



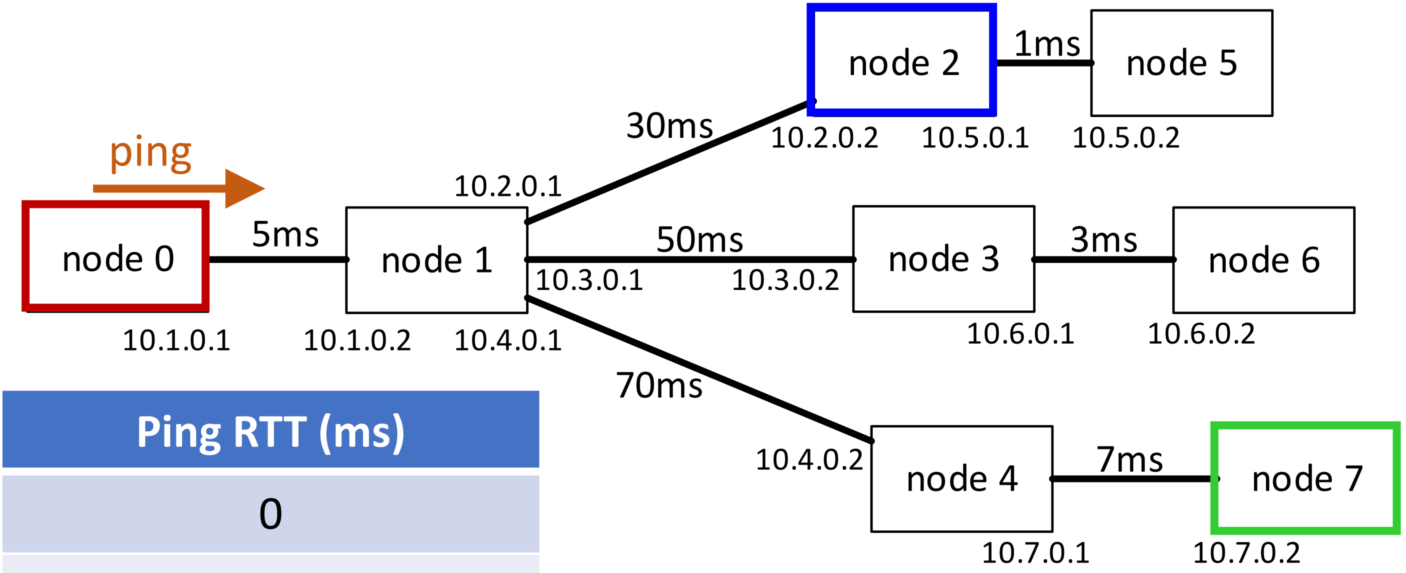
Simulation Times



- ns-3
  - Take **about 100 days** to run **256x256x256 = 16,777,216 cases** (10.x.x.x), each about 0.5 seconds
- sym-ns-3
  - Take **about 15 minutes** and explore **about 30 branches** for symbolic IP destination 10.x.x.x

Necessary to check each IP to detect all possible bugs

Reported Ping RTTs



Destination IP	Ping RTT (ms)
10.1.0.1	0
10.1.0.2, 10.1.255.255, 10.2.0.1, 10.2.255.255, 10.3.0.1, 10.3.255.255, 10.4.0.1, 10.4.255.255	10
10.2.0.2, 10.5.0.1, 10.5.255.255	70
10.5.0.2	72
10.3.0.2, 10.6.0.1, 10.6.255.255	110
10.6.0.2	116
10.4.0.2, 10.7.0.1, 10.7.255.255	150
10.7.0.2	164
Others	No reply for ping

Takes only about 15 minutes for 16 millions of ping tests

Outline

- Why sym-ns-3?
- How it works?
- How to make it more efficient?
- Conclusions

Making sym-ns-3 More Efficient

- Notice we can make sym-ns-3 even more efficient
  - Goal: Reduce the number of branches
  - How? Redesign and rewrite simulator so that different cases share the same execution path as much as possible
- So far, we have redesigned and rewritten
  - ns-3 event schedulers (ACM Transactions on Modeling and Computer Simulation 2022)
  - ns-3 routers (this WNS3 paper)

Redesign IP Routers

- Redesign the code that compares symbolic IP addresses
- Details in our WNS3 paper
- Illustrating example - find the interface for a destination IP (dst)
  - original code: 5 branches (num of entries)
  - rewritten code: 3 branches (num of interfaces), keeping same simulation results

original code

entry1

entry2

entry3

entry4

other

```
if (dst matches entry1)       //branch 1
    return interface1
else if (dst matches entry2) //branch 2
    return interface1
else if (dst matches entry3) //branch 3
    return interface2
else if (dst matches entry4) //branch 4
    return interface2
else                          //branch 5
    return interface0
```

rewritten code

entry1

entry2

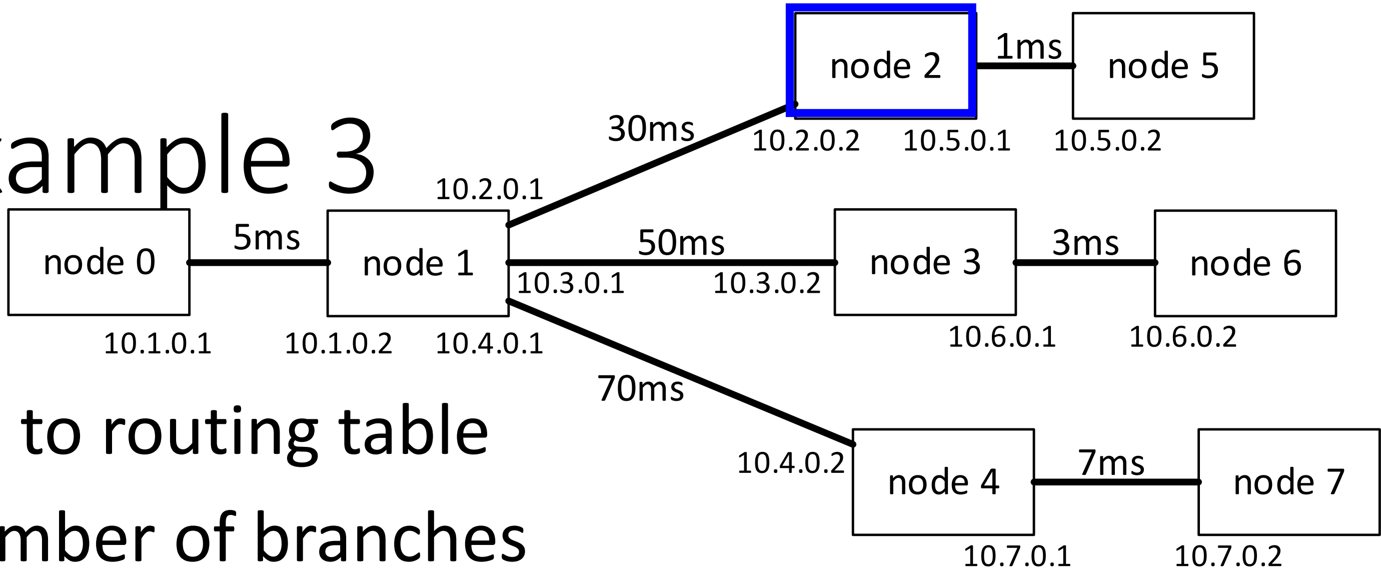
entry3

entry4

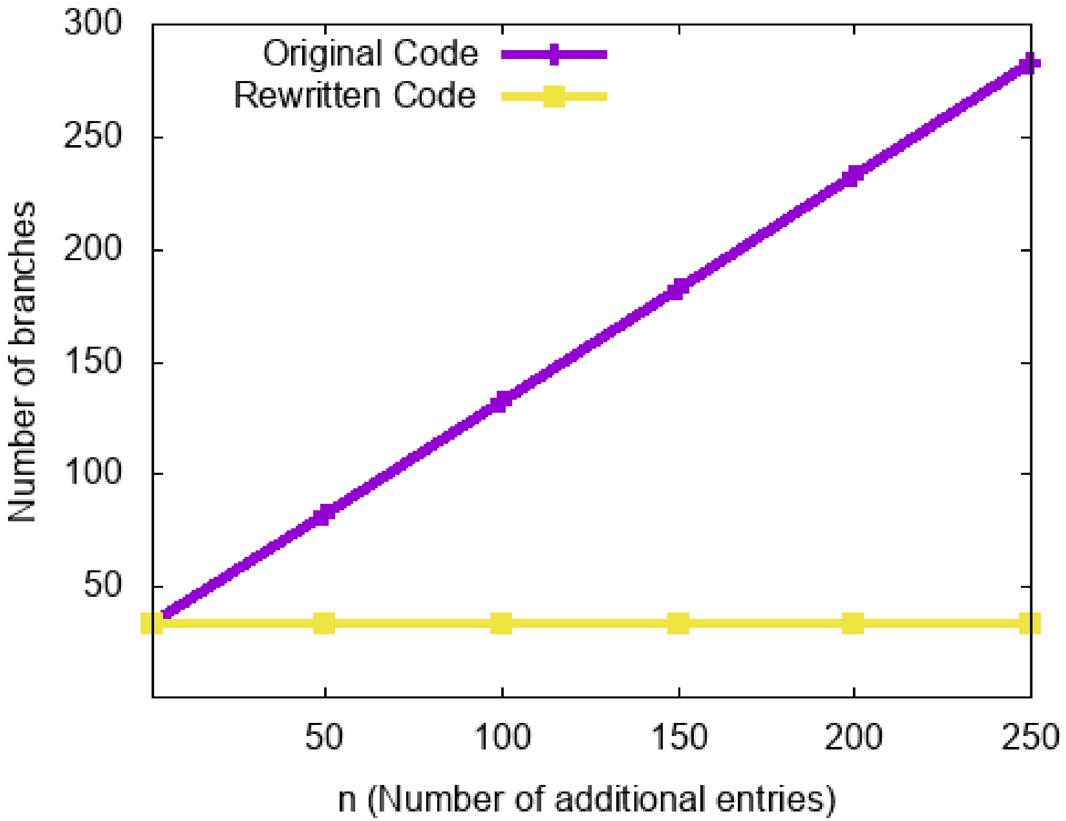
other

```
if (dst matches entry1 or entry2)       //branch 1
    return interface1
else if (dst matches entry3 or entry4) //branch 2
    return interface2
else                                    //branch 3
    return interface0
```

Exhaustive Testing Example 3



- IP reachability example
- Add multiple additional entries to routing table
- Rewritten code reduces the number of branches



Destination	Mask	Interface
10.5.1.0	255.255.255.0	2
…	…	…
10.5.n.0	255.255.255.0	2

Additional entries to routing table of node 2

Outline

- Why sym-ns-3?
- How it works?
- How to make it more efficient?
- Conclusions

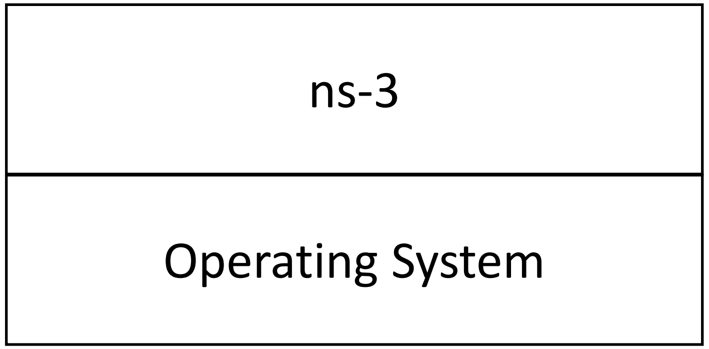
Conclusion

- sym-ns-3 for efficient exhaustive testing
- Future work
  - Continue improving the efficiency
  - More support for symbolic floating-point numbers
- Project webpage (code, documents): https://symbolicns3.github.io

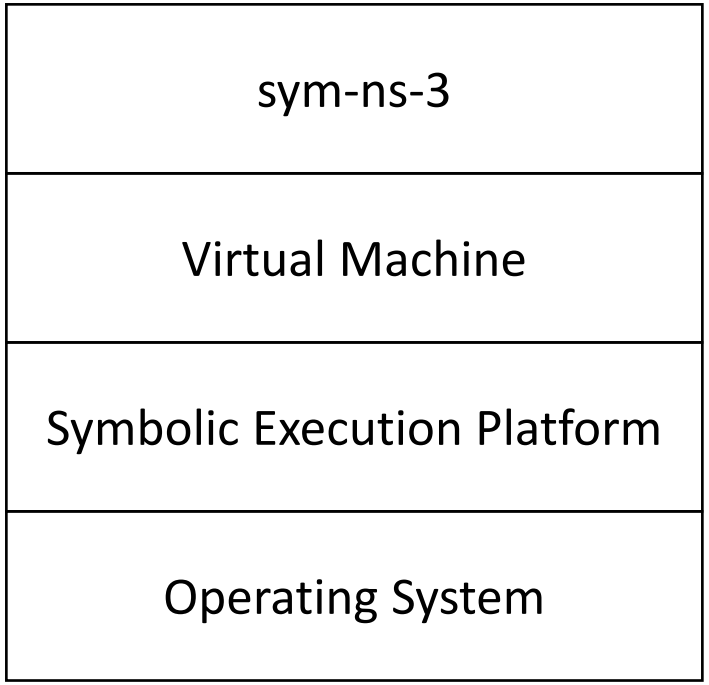
- Acknowledgement: Supported in part by NSF-CCF-1918204

Backup Slides

Running ns-3 vs sym-ns-3



Running ns-3



Each branch is conceptually a virtual machine running a copy of sym-ns-3.

S2E symbolically executes big software systems
https://github.com/S2E/s2e

Running sym-ns-3

Redesign Event Schedulers

- Redesign the code that compares symbolic event timestamps
- Details in ACM Transactions on Modeling and Computer Simulation 2022
- Illustrating example - determine whether event e1 or e2 executes first
  - original code: 3 branches
  - rewritten code: 2 branches, while keeping same simulation results

original code

before

same time

after

```
if (e1.t < e2.t) {            //branch 1
    … // execute event e1
    … // execute event e2
} else if (e1.t == e2.t) { //branch 2
    … // execute event e1
    … // execute event e2
} else {                      //branch 3
    … // execute event e2
    … // execute event e1
}
```

rewritten code

before

same time

after

```
if (e1.t <= e2.t) {           //branch 1
    … // execute event e1
    … // execute event e2
} else {                      //branch 2
    … // execute event e2
    … // execute event e1
}
```