THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Toward Online Verification of Client Behavior in Distributed Applications

Robby Cochran
Mike Reiter

University of North Carolina at Chapel Hill

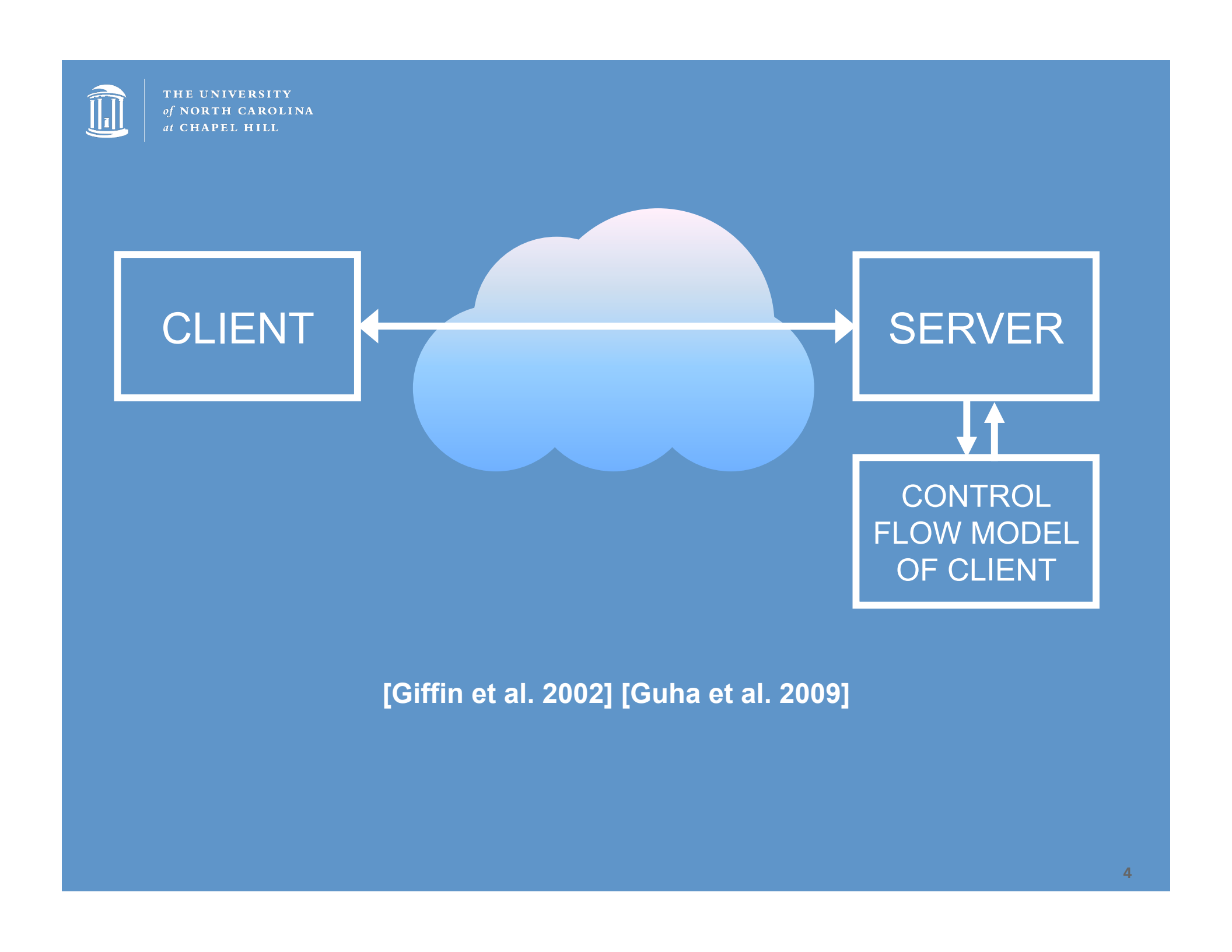
NDSS 2013


CLIENT
Is this message from an unmodified client?
SERVER
+
VERIFIER

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

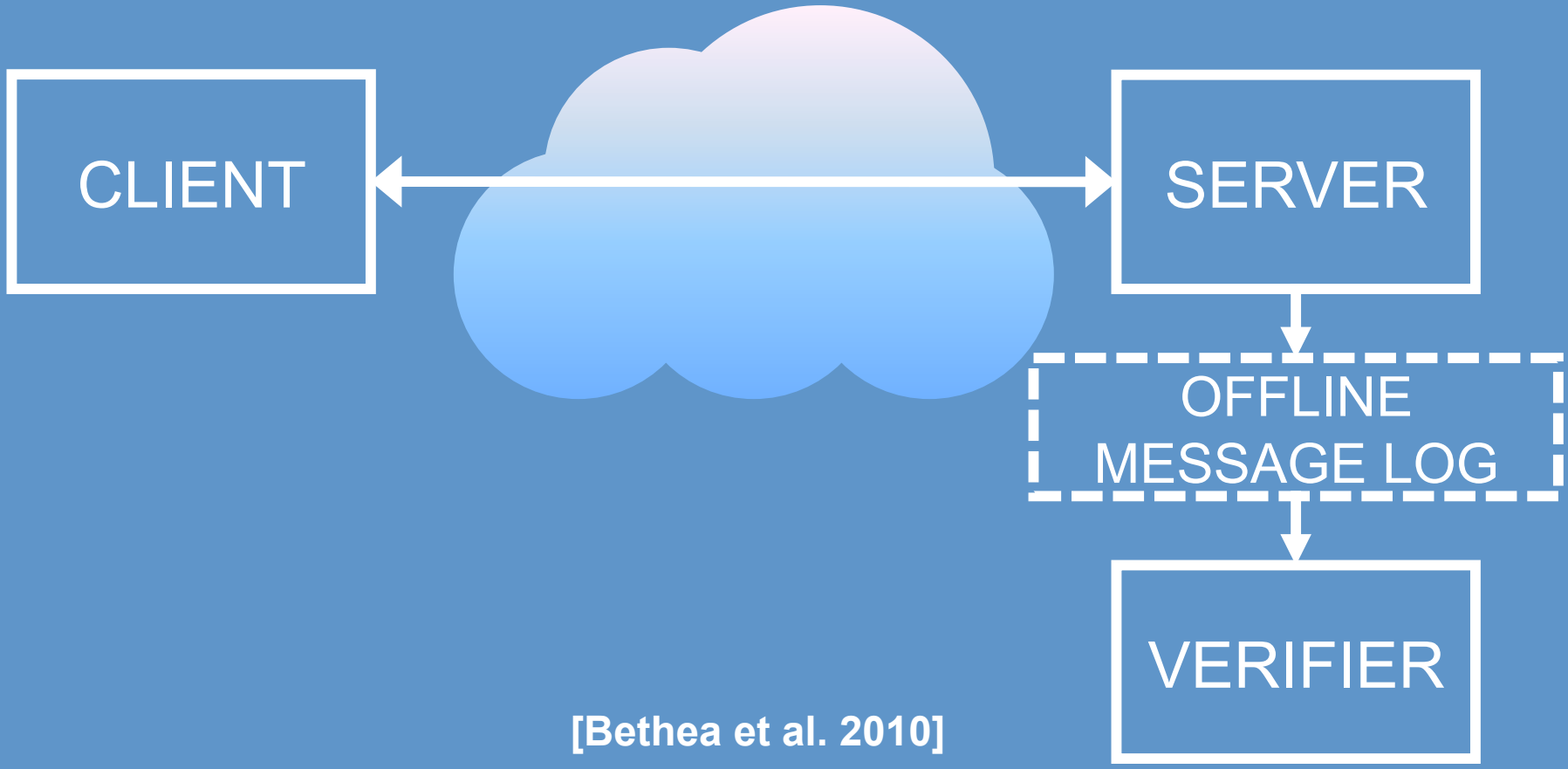
Goals

- Detect if client messages are consistent with the sanctioned client software …
  - Adversary might modify binary/memory
  - or rewrite message on the wire
- … Much more quickly than previous work
- Ideally we would do so online…

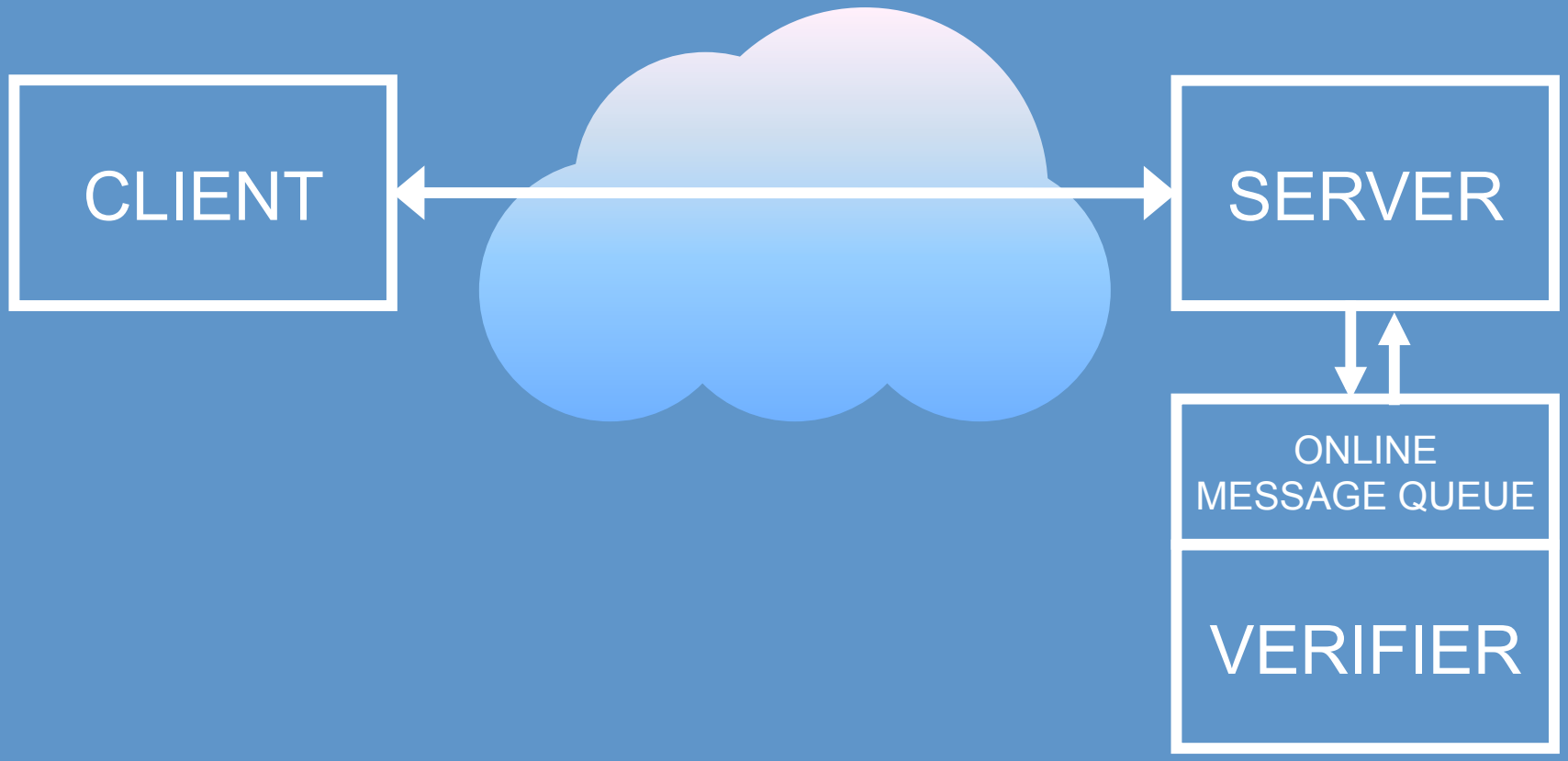


[Giffin et al. 2002] [Guha et al. 2009]

[Vikram et al. 2009]



[Bethea et al. 2010]


CLIENT
SERVER
ONLINE
MESSAGE QUEUE
VERIFIER

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Introduction

- **Existing techniques to verify client behavior**
  - Imprecise
  - Increase bandwidth usage
  - Computationally expensive
- **Our method**
  - Precise: no false negatives and no false positives
  - No additional bandwidth *required*
  - Validates most legitimate behavior faster than previous techniques

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

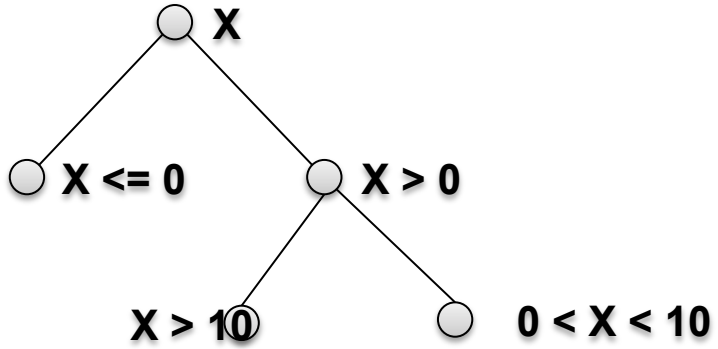
Overview

- *Introduction*
- ***Symbolic Execution***
- ***Key Insight #1: Common Case Optimization***
- ***Key Insight #2: Guided Search with History***
- ***Case Studies***
- ***Conclusion***

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Symbolic Execution
[Boyer 1975]

- A way of deriving the effects of a given program on a given system
  - Constraints on input are constructed based on each execution path
- Built on top of KLEE [Cadar et al. 2008]



THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Symbolic Environment

How can we use symbolic execution to verify a message?

```
1  loc = 0
2  while true do
3    key = symbolicReadKey()
4    if key == ESC then
5      sendQuitMsg()
6    else if key == UP then
7      loc = loc + 1
8    else if key == DN then
9      loc = loc – 1
10   end if
11     sendMsg(loc)
12 end while
```

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Symbolic Environment

```
1  loc = 0
2  while true do
3    key = symbolicReadKey()
4    if key == ESC then
5      sendQuitMsg()
6    else if key == UP then
7      loc = loc + 1
8    else if key == DN then
9      loc = loc – 1
10   end if
11     sendMsg(loc)
12 end while
```

Symbolic State	Execution Prefix
loc == 0 ∧ key == ESC	1,2,3,4,5

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Symbolic Environment

```
1  loc = 0
2  while true do
3    key = symbolicReadKey()
4    if key == ESC then
5      sendQuitMsg()
6    else if key == UP then
7      loc = loc + 1
8    else if key == DN then
9      loc = loc - 1
10   end if
11     sendMsg(loc)
12 end while
```

Symbolic State	Execution Prefix
loc == 0 ∧ key == ESC	1,2,3,4,5
loc == 1 ∧ key != ESC ∧ key == UP	1,2,3,4,6, 7,11

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Symbolic Environment

```
1  loc = 0
2  while true do
3    key = symbolicReadKey()
4    if key == ESC then
5      sendQuitMsg()
6    else if key == UP then
7      loc = loc + 1
8    else if key == DN then
9      loc = loc - 1
10   end if
11     sendMsg(loc)
12 end while
```

Symbolic State	Execution Prefix
loc == 0 ∧ key == ESC	1,2,3,4,5
loc == 1 ∧ key != ESC ∧ key == UP	1,2,3,4,6, 7,11
loc == -1 ∧ key != ESC ∧key != UP ∧ key == DN	1,2,3,4,6, 8,9,11

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Checking a Client Message

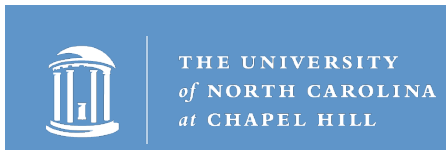
Client Message: msg_0

```
<location>1</location>
```

STP **Satisfiability Modulo Theory Solver** [Ganesh et al. 2007]

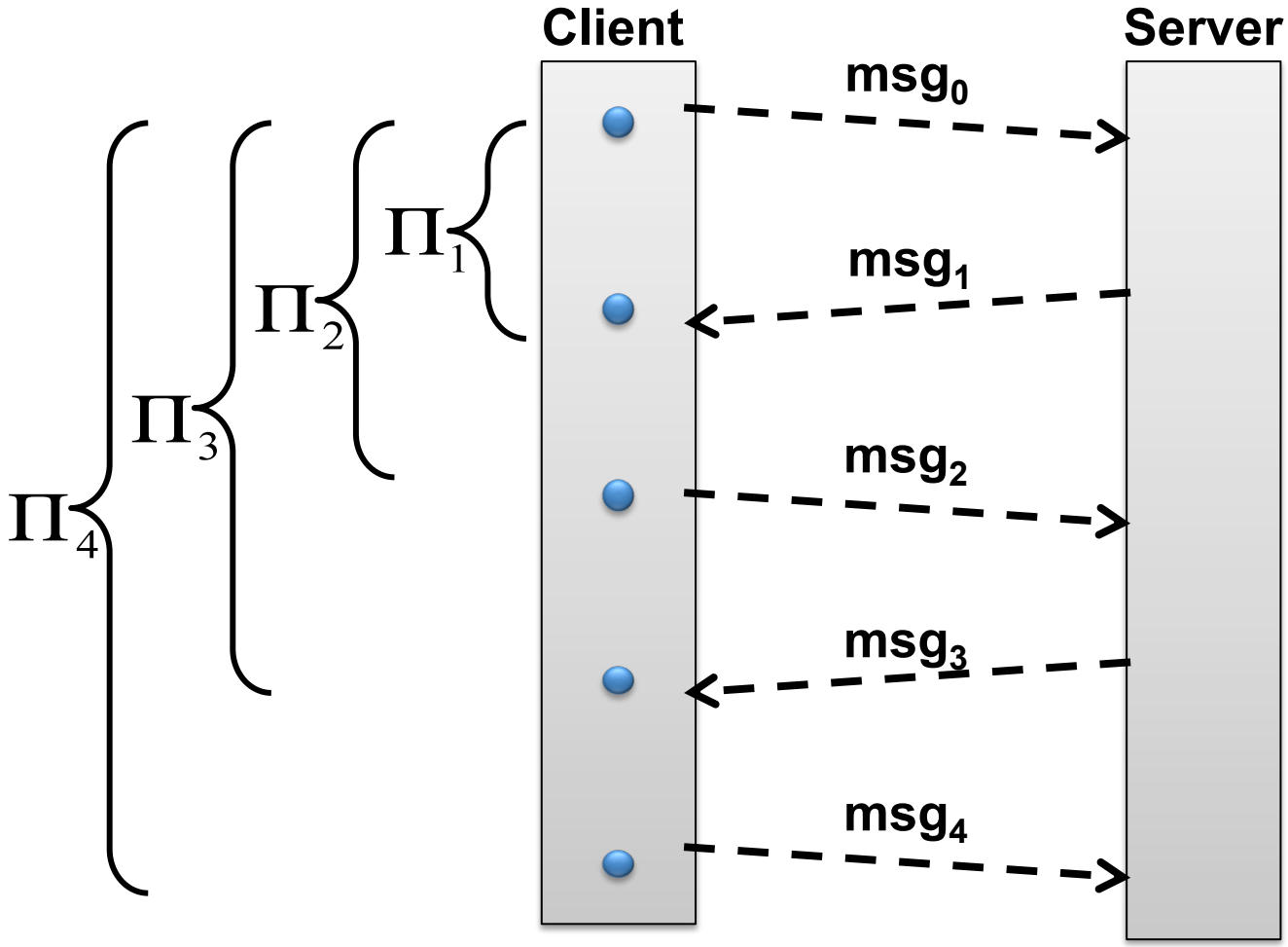
Check consistency of message with formulas generated via symbolic execution.

	Symbolic State	Execution Prefix
	loc == 0 ∧ key == ESC	1,2,3,4,5
	loc == 1 ∧ key != ESC ∧ key == UP	1,2,3,4,6, 7,11
	loc == -1 ∧ key != ESC ∧key != UP ∧ key == DN	1,2,3,4,6, 8,9,11



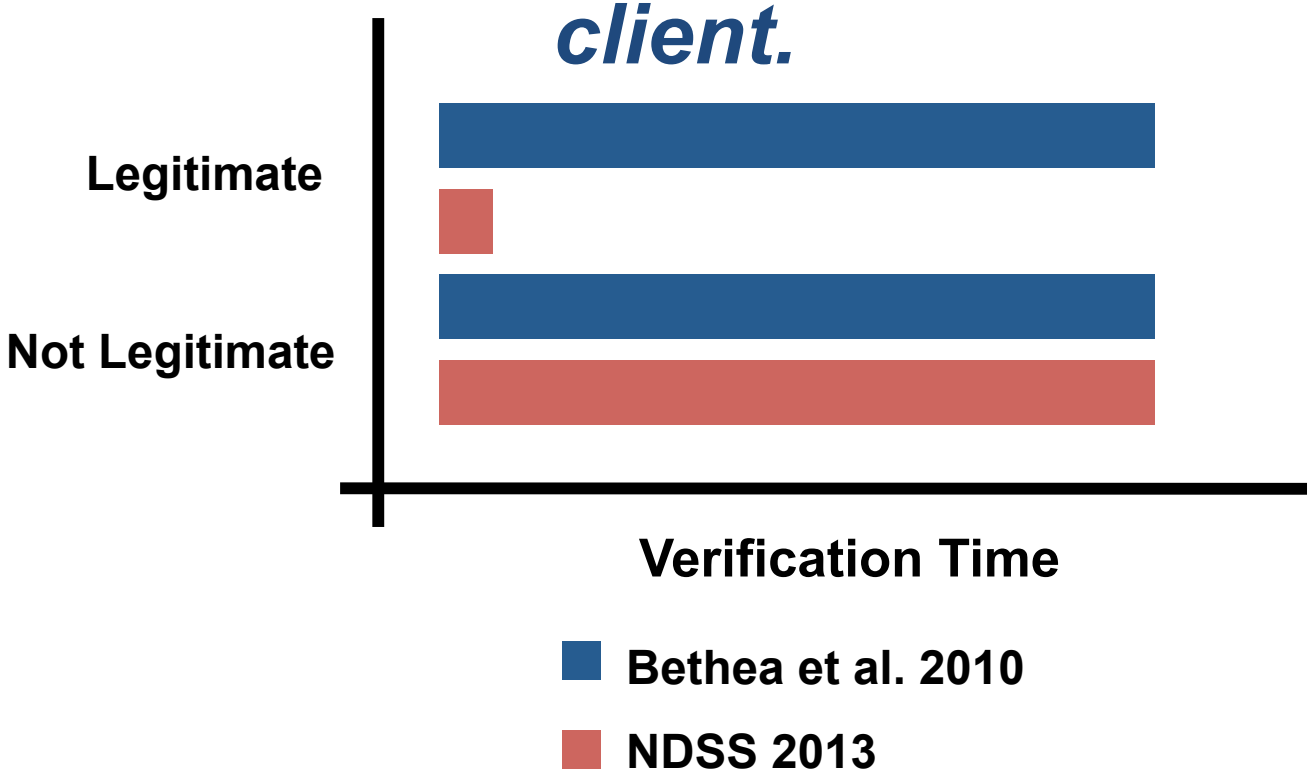
Verifying Client Messages

Iteratively find *execution prefix* $\prod$ consistent with msg_0, msg_1, …, msg_n



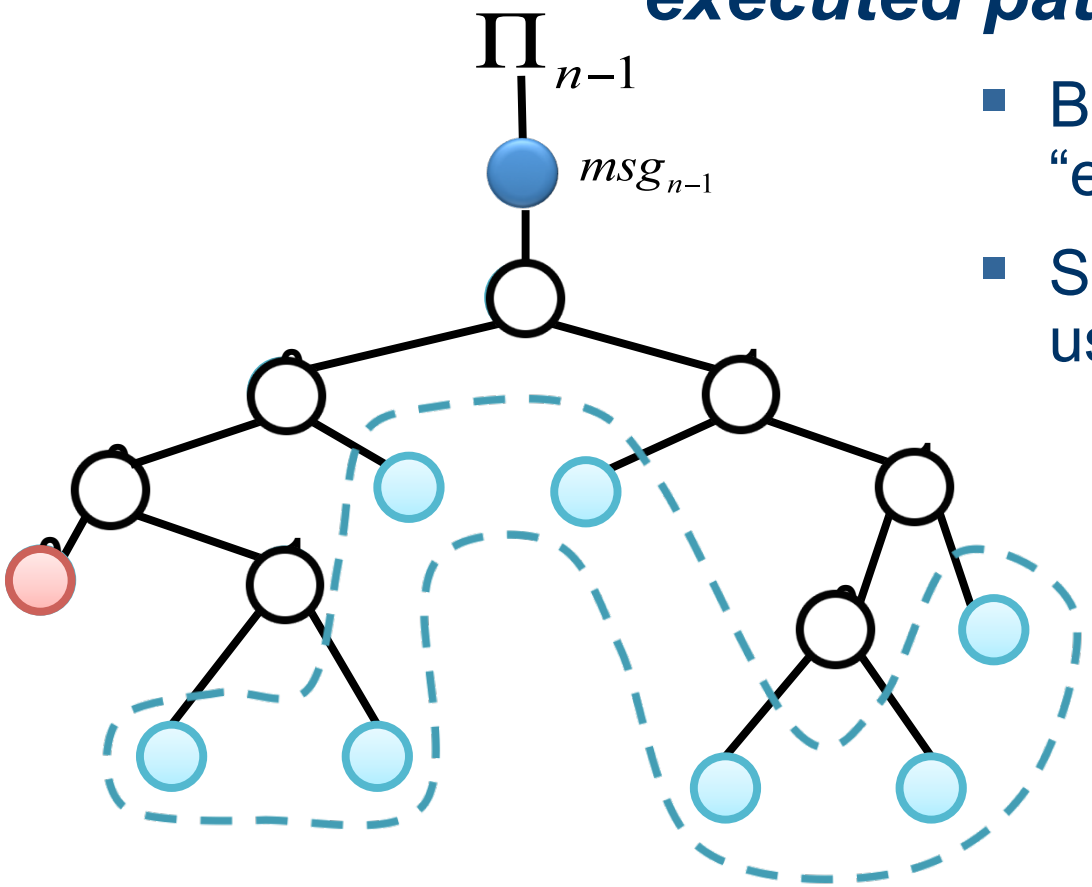
Verifying Client Messages

Key Insight #1: Optimize for the common case of a legitimate client.



Verification: Node Selection

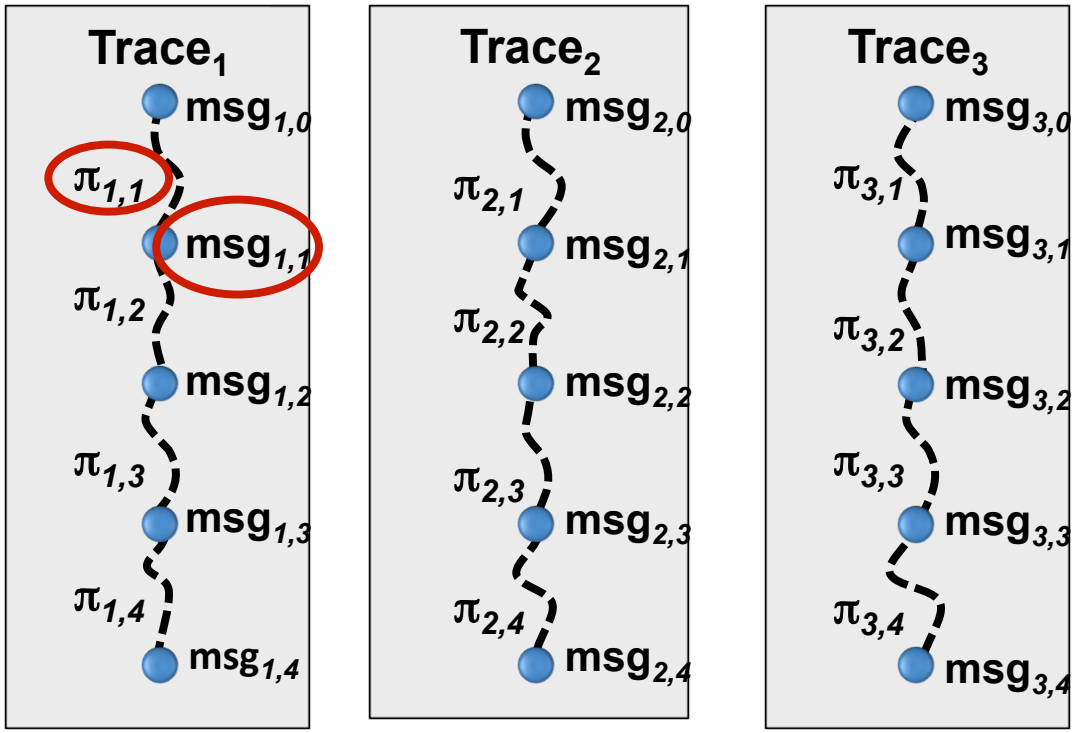
Key Insight #2:
Correlate message contents with previously executed paths



- Build training corpus of "execution fragments"
- Select next node to explore using training data

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Building Training Corpus

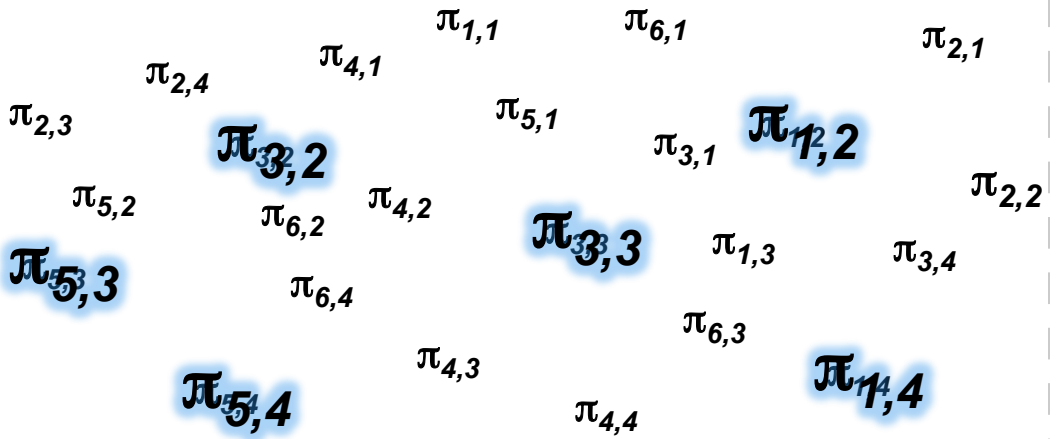


Execute client software to generate a set of message traces.

Split each trace into a set of execution fragments $\pi_{i,j}$

Associate each $\pi_{i,j}$ with the set of messages it is consistent with.

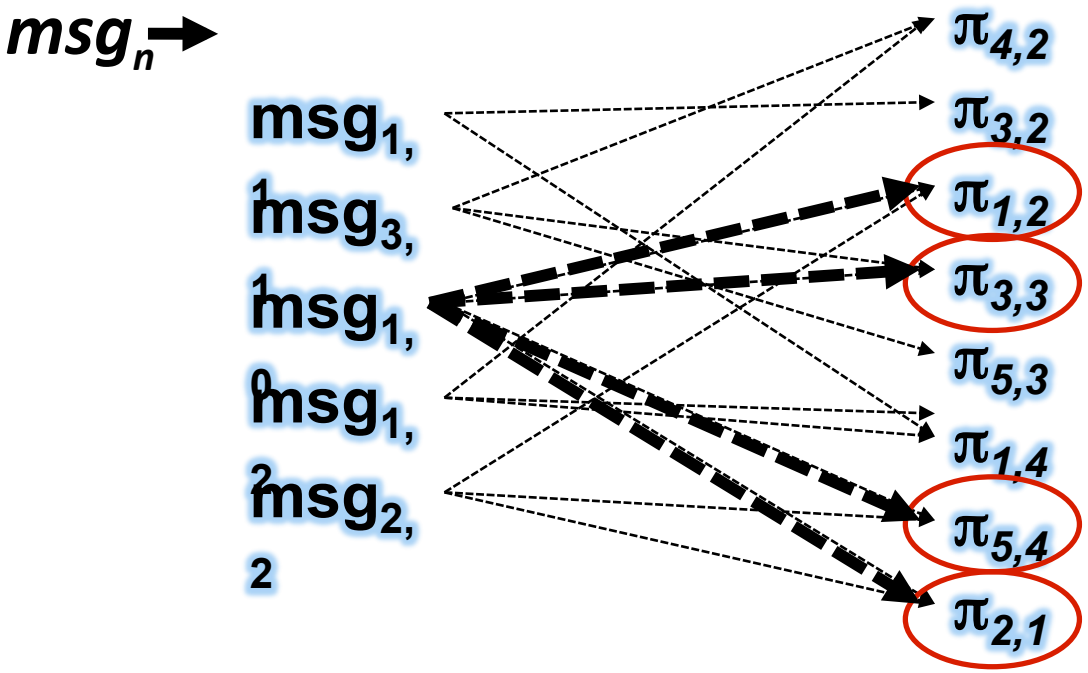
Clustering Training Data



Cluster execution fragments π by edit distance using k-medoids clustering.

Cluster messages by edit distance using k-medoids clustering.

Using the Clusters



Map message medoids to execution fragment medoids.

Find message medoid closest to msg_n via edit distance.

Use associated execution fragment medoids to guide search.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

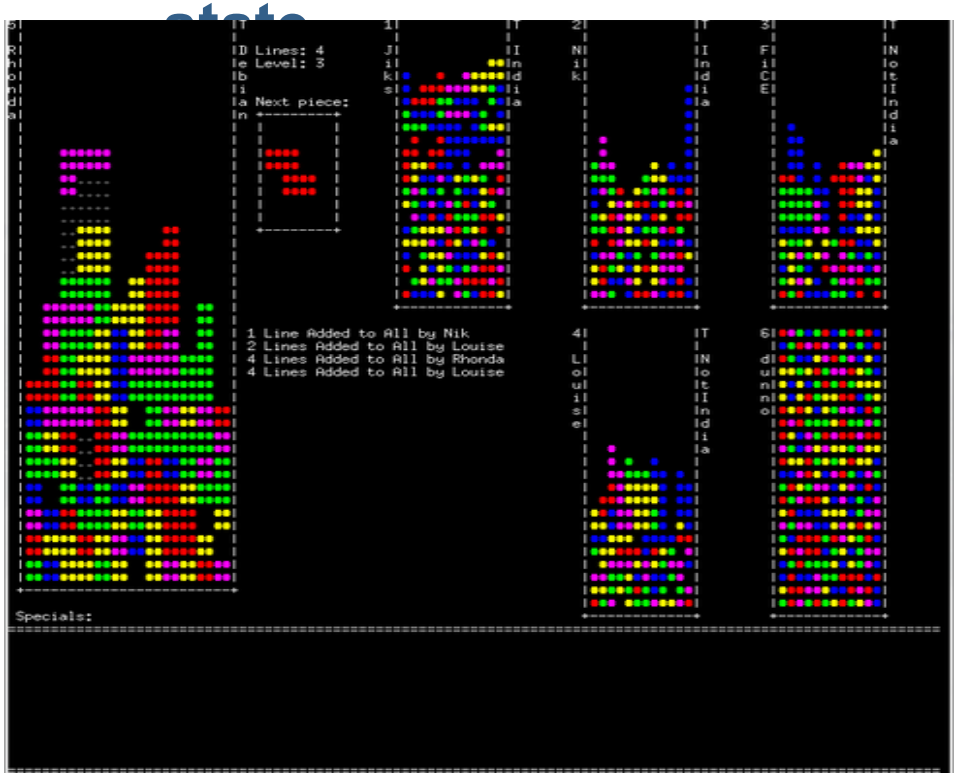
Case Studies

- **Networked games are popular!**
  - Some > 10 million subscribers
  - Gaming industry revenues of $67 billion (2012)
- **Cheaters reduce the quality of the game for honest players**
  - Our technique detects any cheat that requires a modification to the client binary or memory

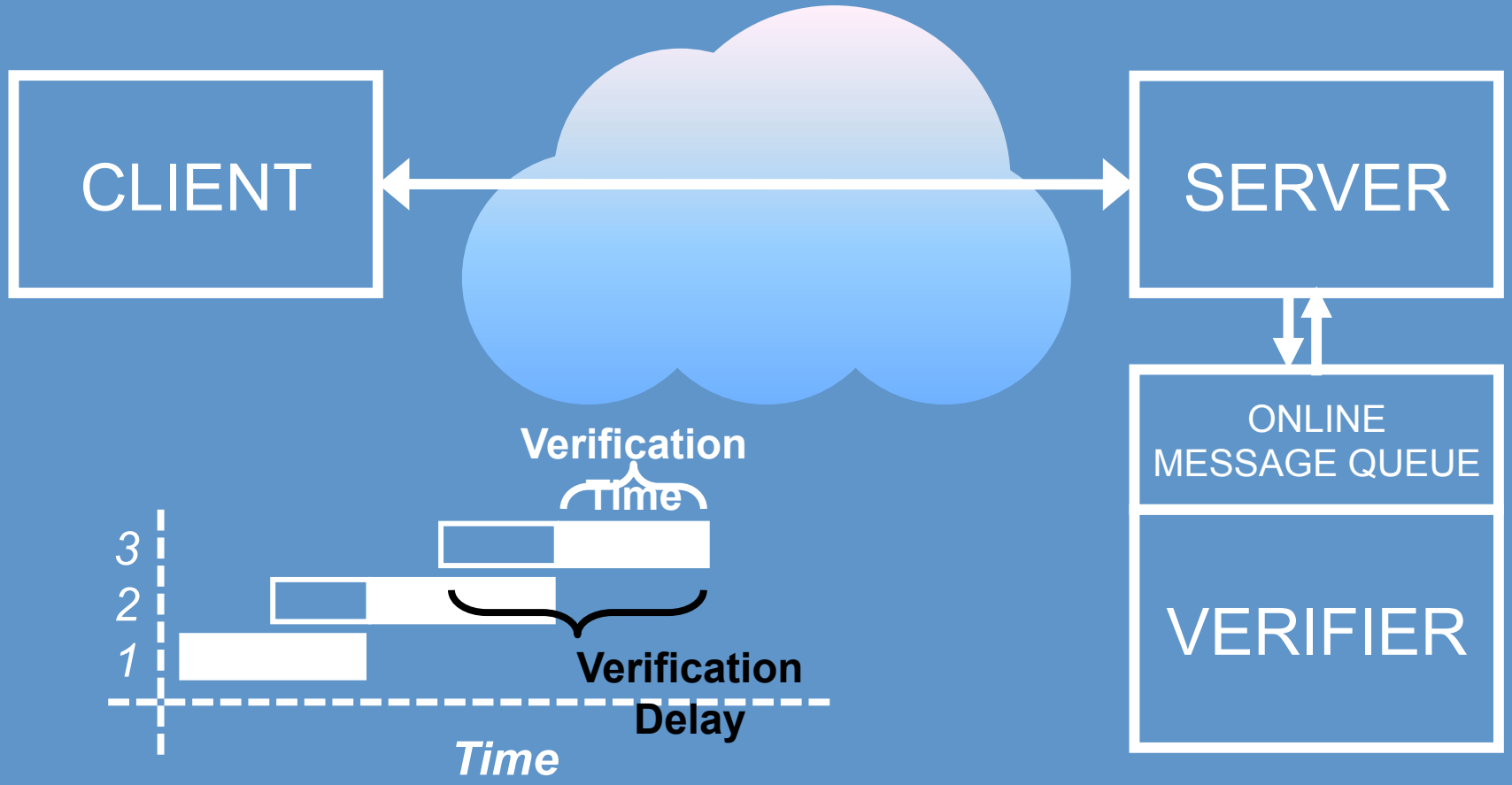
Case Studies

- **XPilot**
  - Open-source multiplayer 2D shooter
  - 150000 lines of C
  - Little client-side state

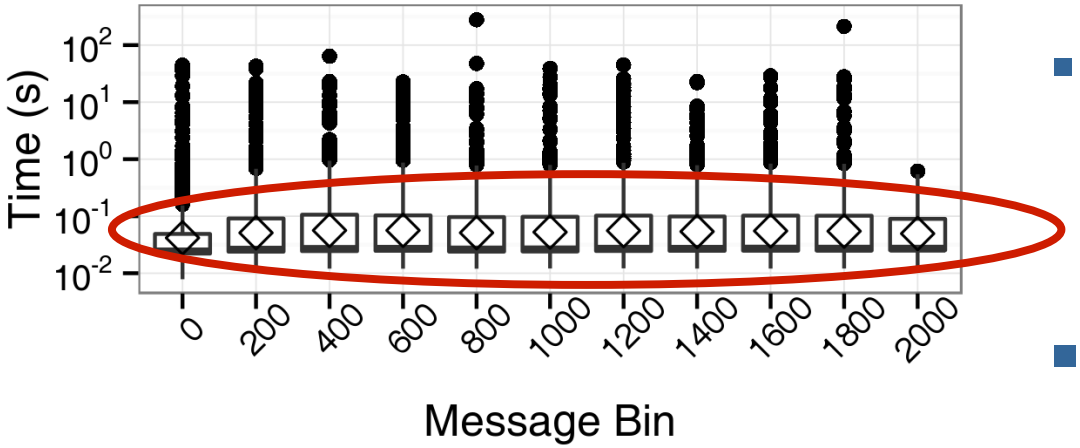
- **TetriNET**
  - Text-based multiplayer Tetris game
  - 5000 lines of C
  - More ambiguity at server about client

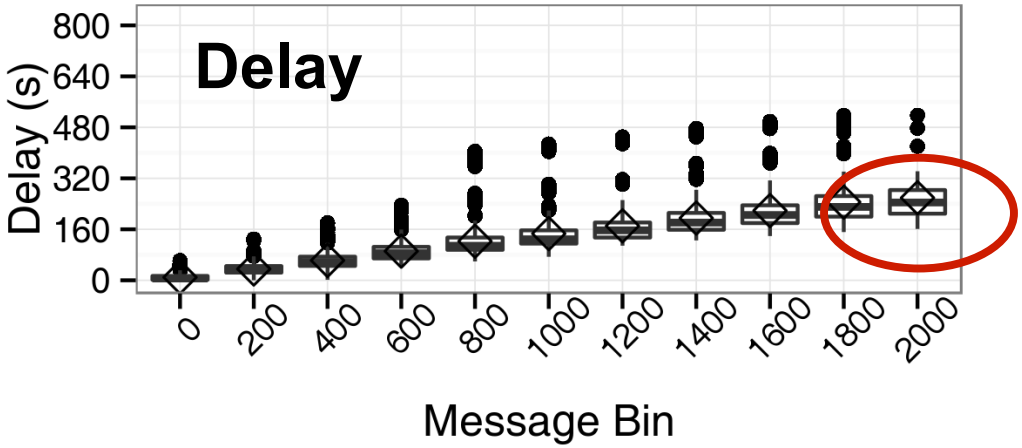


Measuring Performance



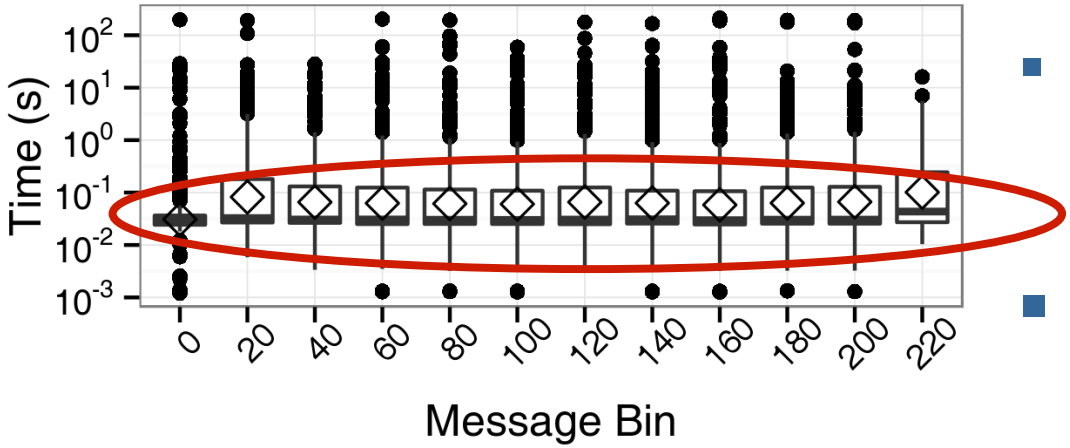
XPilot Results

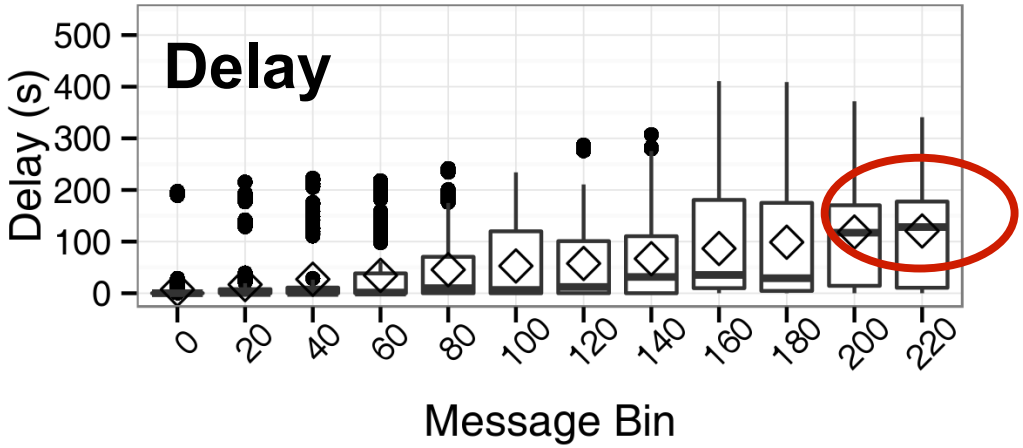




- 40-fold cross validation
  - 4000 points per bin
  - Average: 32 messages/ second
- **Average verification time < 100ms**
- **Average delay at end of queue is less than 5 min**
- **100x faster than previous work**

TetriNET Results





- 20-fold cross validation
  - Average game 6.5 minutes
  - 20x20 = 400 points per bin
- **Average verification time < 100ms**
- **Average Delay at end of queue is less than 2 min**
- **See paper for a variation that often keeps up with gameplay with a small increase in bandwidth**

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Conclusion

- **Key Insights**
  - Optimize for the common case legitimate client
  - Use a search heuristic that correlates message contents with previously executed paths
  - Optimize symbolic execution components for our specific needs (see paper)
- **Contribution: Precise client checking algorithm**
  - Dramatically improves performance over previous work with similar design goals
  - In some cases, verification comes very close to keeping pace with the application (see paper)

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Questions?