

WAP5

Black-box Performance Debugging
for Wide-Area Distributed Systems

Patrick Reynolds
reynolds@cs.duke.edu

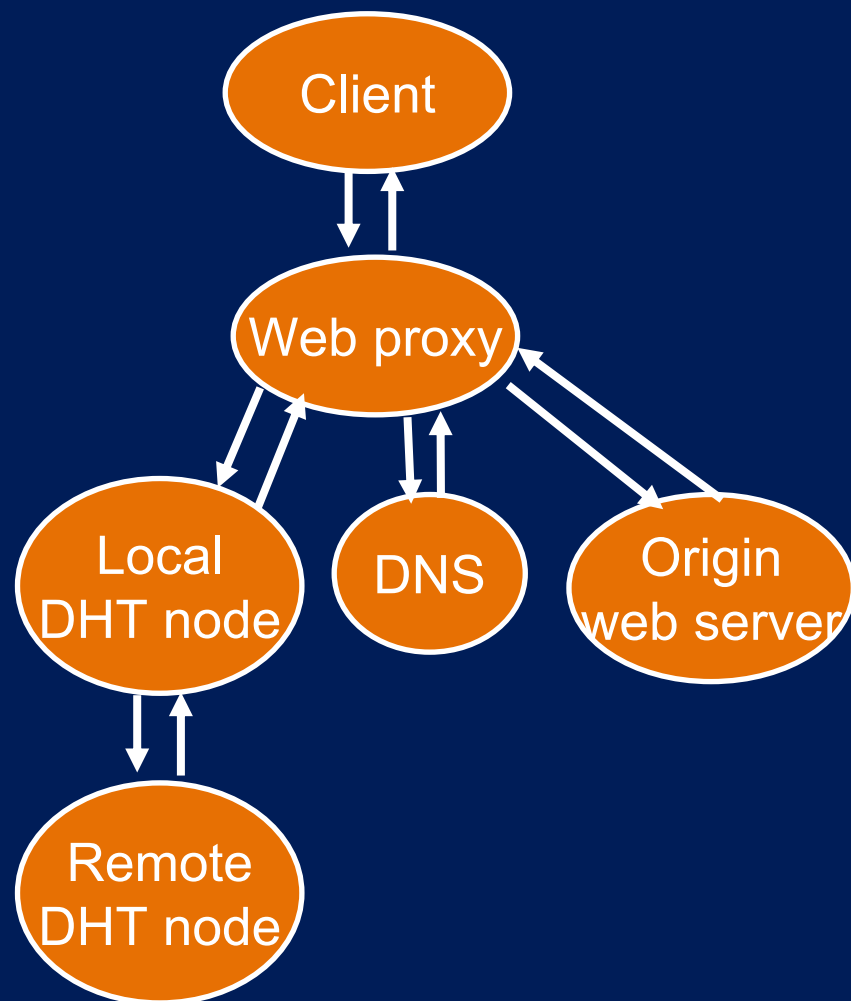
With:

Janet Wiener Marcos Aguilera
Jeffrey Mogul Amin Vahdat

<http://www.hpl.hp.com/research/project5/>

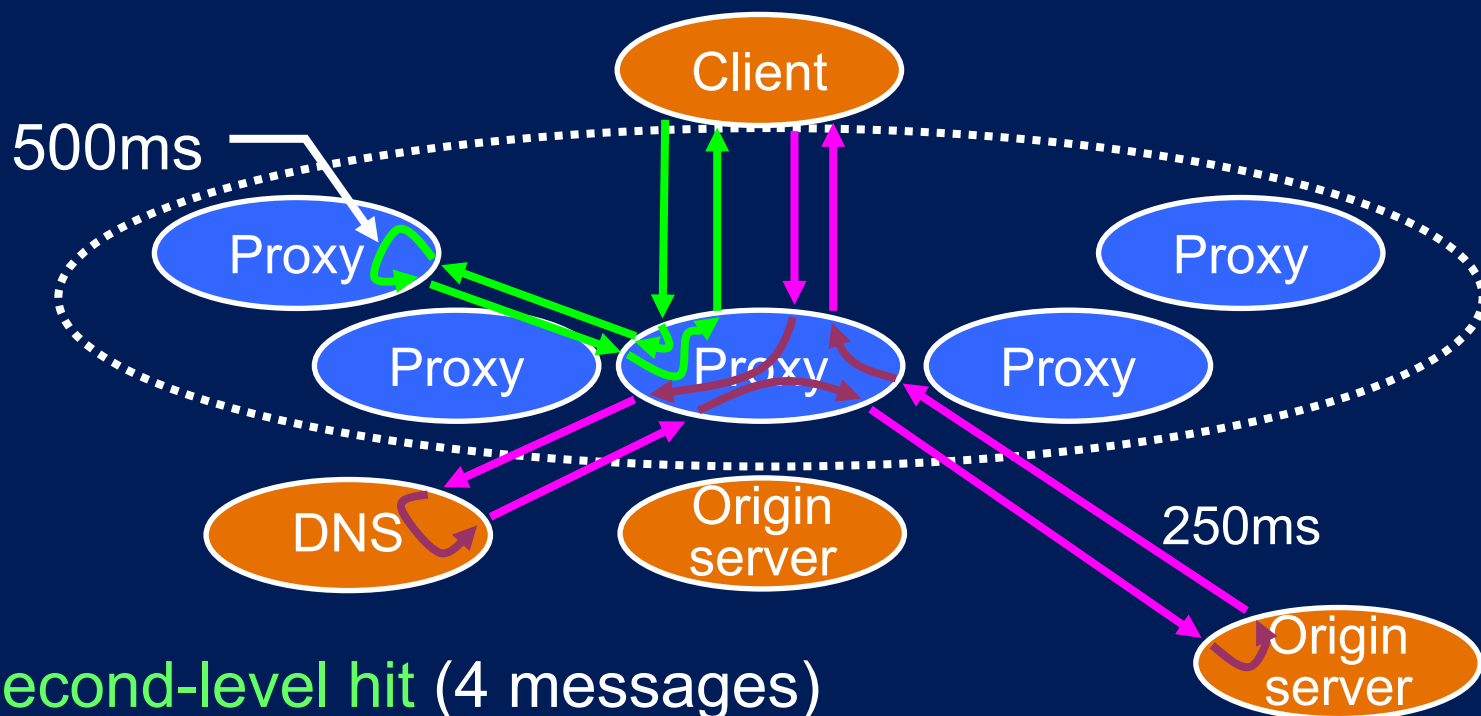
Motivation

- Discover structure and performance problems in large, wide-area systems
- Infer paths through nodes
 - One path per client request
 - Discover timing at each step
- Focus attention on nodes that are problematic
 - First step in performance debugging



Coral example

- **Causal path:** a sequence of related messages and processing, annotated with timing/delays



- **Second-level hit** (4 messages)
- **Second-level miss** (6 messages)
- Also: DHT lookups

Goals

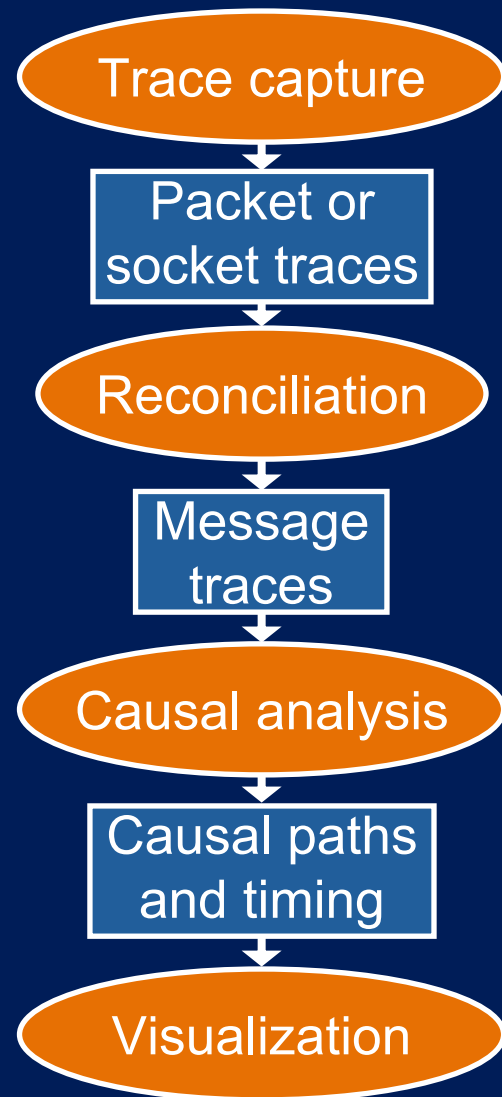
- **Find bugs in wide-area applications**
 - **Performance bugs**: too much or too little time at any point
 - **Structure bugs**: incorrect ordering or placement of processing or communication
- **Expose causal paths**
 - Structure discovery
 - Measure latency for processing and communication
 - Unexpected structure or timing
 - **Indicates possible bugs**
- **Black-box approach**
 - Do not require source code access
 - Allow heterogeneity

Three target audiences

- **Primary programmer**
 - Debugging or optimizing his/her own system
- **Secondary programmer**
 - Inheriting a project or joining a programming team
 - **Discovery**: learning how the system behaves
- **Operator**
 - Monitoring a running system for unexpected behavior
 - Performing regression tests after a change

Contributions

- **New causality analysis algorithm**
- **Full tool chain**
 - Trace capture library
 - Causal path analysis
 - Visualization
- **Results with two PlanetLab CDNs**
 - Coral and CoDeeN

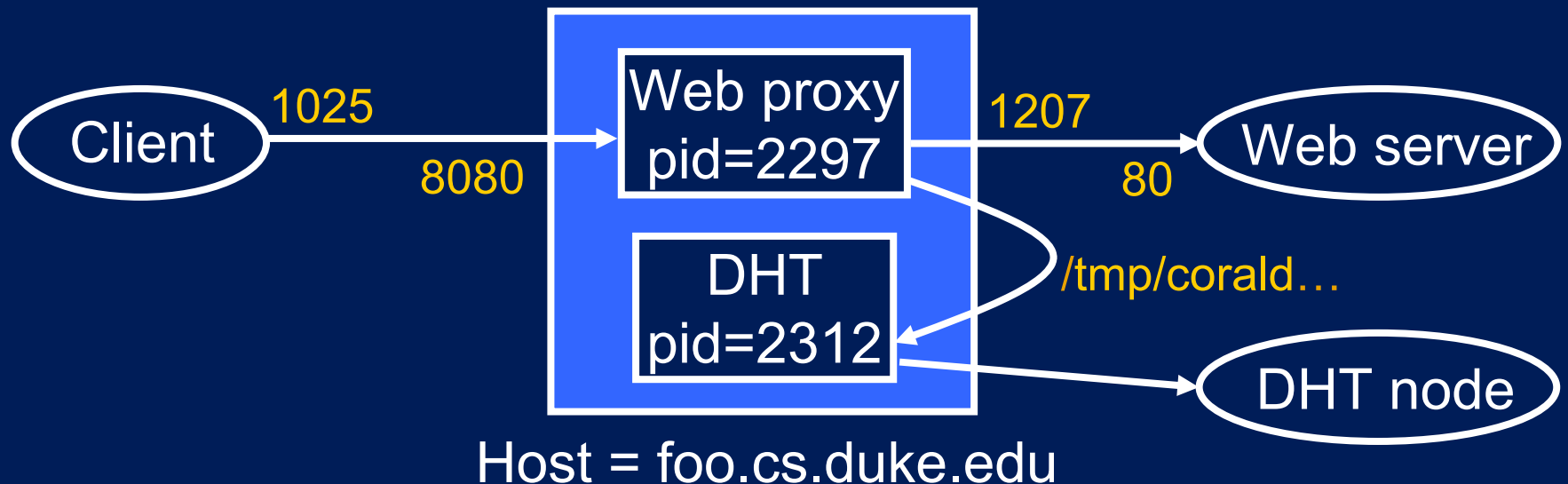


Outline

- Introduction
- **Naming**
- Trace capture
- Reconciliation
- Causality analysis
 - Message linking algorithm
- Results with CoDeeN & Coral

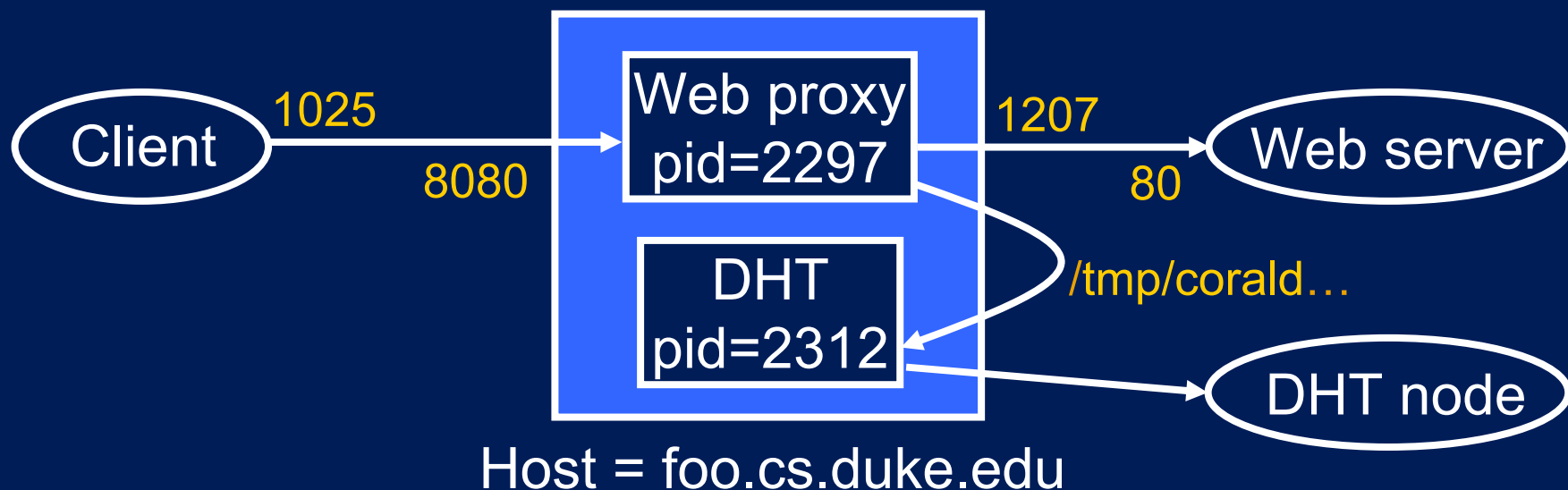
Naming

- Message is single read/write system call
 - May be many TCP or UDP packets
- Node can be process or host
- Endpoint can be socket path or <IP address, port>



Naming

- **Node names are causal names**
 - Message into a process/host can cause messages out
- **Endpoint names guide aggregation**
 - Calls to `foo:8080` are different from calls to `foo:53`
 - Client hosts and ports can be ignored

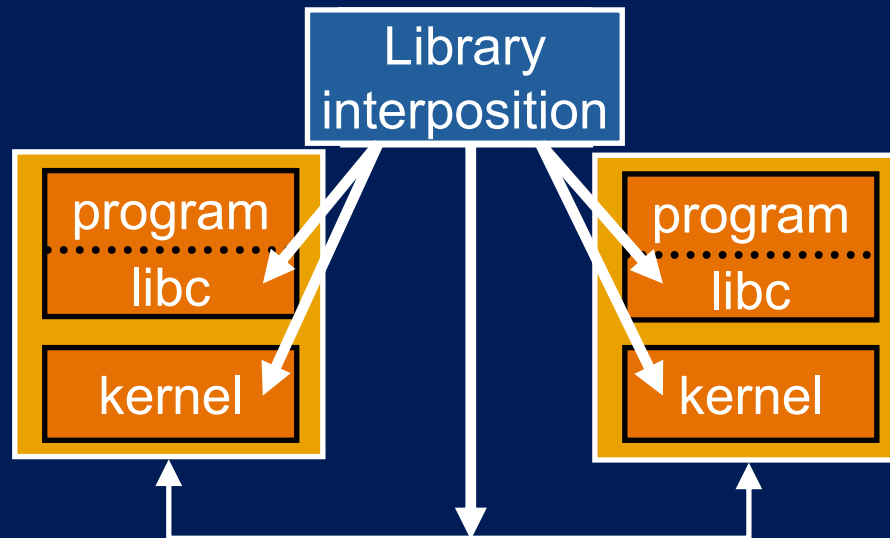


Outline

- Introduction
- Naming
- Trace capture
- Reconciliation
- Causality analysis
 - Message linking algorithm
- Results with CoDeeN & Coral

Trace capture

- Capture events using host/net sniffing or library interposition
 - All three choices: no modifications to applications
 - On PlanetLab: sniffing on host only, limited flexibility
- We capture events using library interposition
 - Captures all calls that create, modify, or use a socket



Outline

- Introduction
- Naming
- Trace capture
- **Reconciliation**
- Causality analysis
 - Message linking algorithm
- Results with CoDeeN & Coral

Reconciliation: Convert socket calls to logical messages

- Assign endpoint names to each call

| | |
|--------------------|--|
| client pid=5040 | bind(fd=6, addr={15.1.2.3:33250}) connect(fd=6, addr={16.5.6.7:80}) send(fd=6, len=10, time=0.592) recv(fd=6, len=12, time=2.033) |
|--------------------|--|

| | |
|--------------------|--|
| server pid=8712 | bind(fd=4, addr={16.5.6.7:80}) accept(lfd=4, addr={15.1.2.3:33250}) = 5 recv(fd=5, len=10, time=0.852) send(fd=5, len=12, time=1.705) |
|--------------------|--|

| | | | |
|-------------|-------------|-------|-------|
| client/5040 | server/8712 | 0.592 | 0.852 |
| server/8712 | client/5040 | 1.705 | 2.033 |

Reconciliation:

Convert socket calls to logical messages

- **Combine *send* and *recv* events for each message**
 - Detect dropped or reordered UDP packets
 - Detect differing message (buffer) boundaries

| | |
|--------------------|--|
| client pid=5040 | <code>bind(fd=6, addr={15. 1. 2. 3: 33250})</code> <code>connect(fd=6, addr={16. 5. 6. 7: 80})</code> <code>send(fd=6, len=10, time=0.592)</code> <code>recv(fd=6, len=12, time=2.033)</code> |
|--------------------|--|

| | |
|--------------------|--|
| server pid=8712 | <code>bind(fd=4, addr={16. 5. 6. 7: 80})</code> <code>accept(lfd=4, addr={15. 1. 2. 3: 33250}) = 5</code> <code>recv(fd=5, len=10, time=0.852)</code> <code>send(fd=5, len=12, time=1.705)</code> |
|--------------------|--|

| | | | |
|-------------|-------------|-------|-------|
| client/5040 | server/8712 | 0.592 | 0.852 |
| server/8712 | client/5040 | 1.705 | 2.033 |

Reconciliation:

Convert socket calls to logical messages

- Assign node (process) names to each message

| | |
|---------------------------|--|
| client pid=5040 | <code>bind(fd=6, addr={15.1.2.3:33250})</code> <code>connect(fd=6, addr={16.5.6.7:80})</code> <code>send(fd=6, len=10, time=0.592)</code> <code>recv(fd=6, len=12, time=2.033)</code> |
|---------------------------|--|

| | |
|---------------------------|--|
| server pid=8712 | <code>bind(fd=4, addr={16.5.6.7:80})</code> <code>accept(lfd=4, addr={15.1.2.3:33250}) = 5</code> <code>recv(fd=5, len=10, time=0.852)</code> <code>send(fd=5, len=12, time=1.705)</code> |
|---------------------------|--|

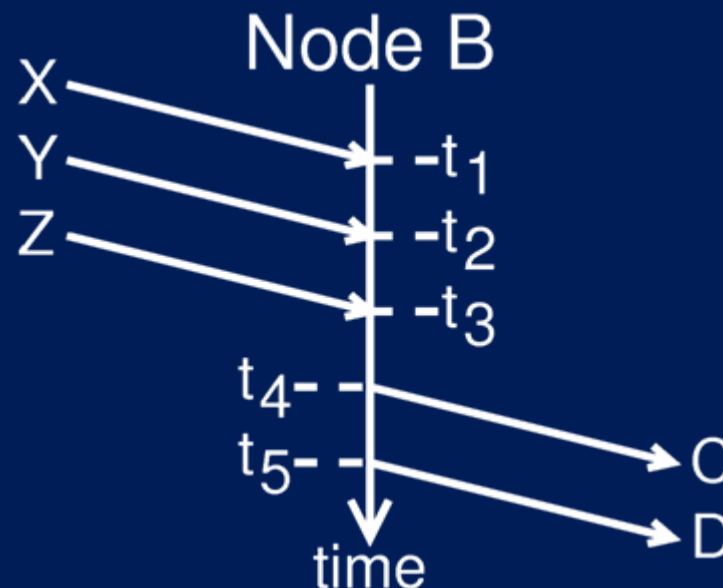
| | | | |
|--------------------|--------------------|-------|-------|
| client/5040 | server/8712 | 0.592 | 0.852 |
| server/8712 | client/5040 | 1.705 | 2.033 |

Outline

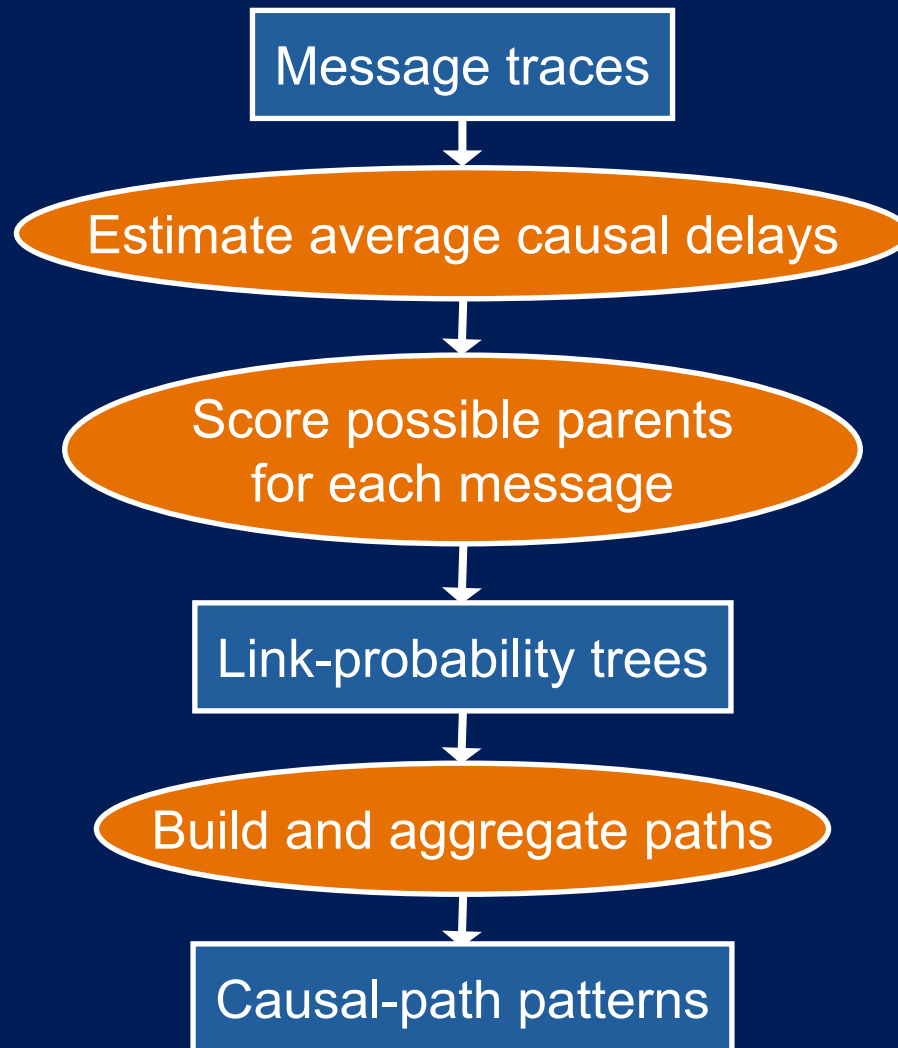
- Introduction
- Naming
- Trace capture
- Reconciliation
- **Causality analysis**
 - **Message linking algorithm**
- Results with CoDeeN & Coral

Causal path analysis

- Which call to B caused outgoing calls?
 - Could be spontaneous action
 - May be ambiguous
 - Make good guesses
 - Use statistics over whole trace
 - Try multiple possibilities
- Build paths by combining calls

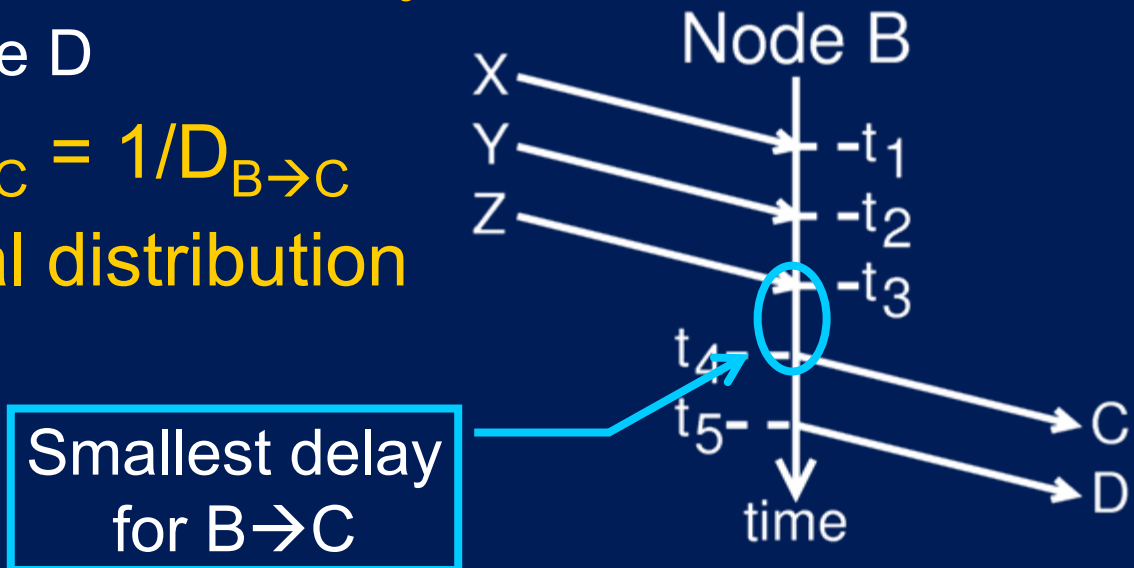


Message linking algorithm



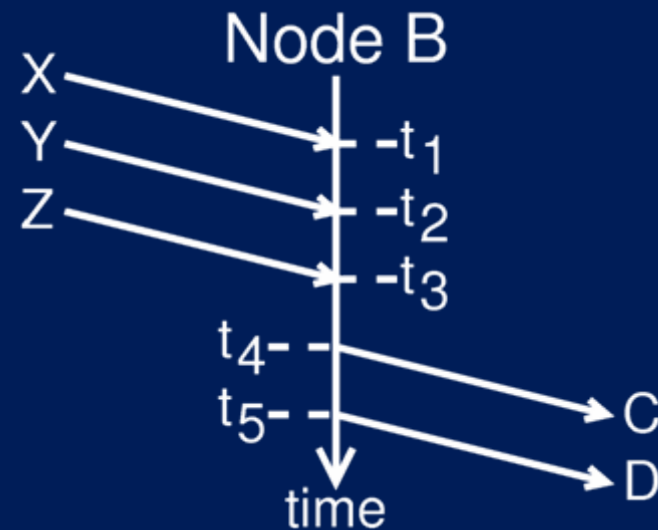
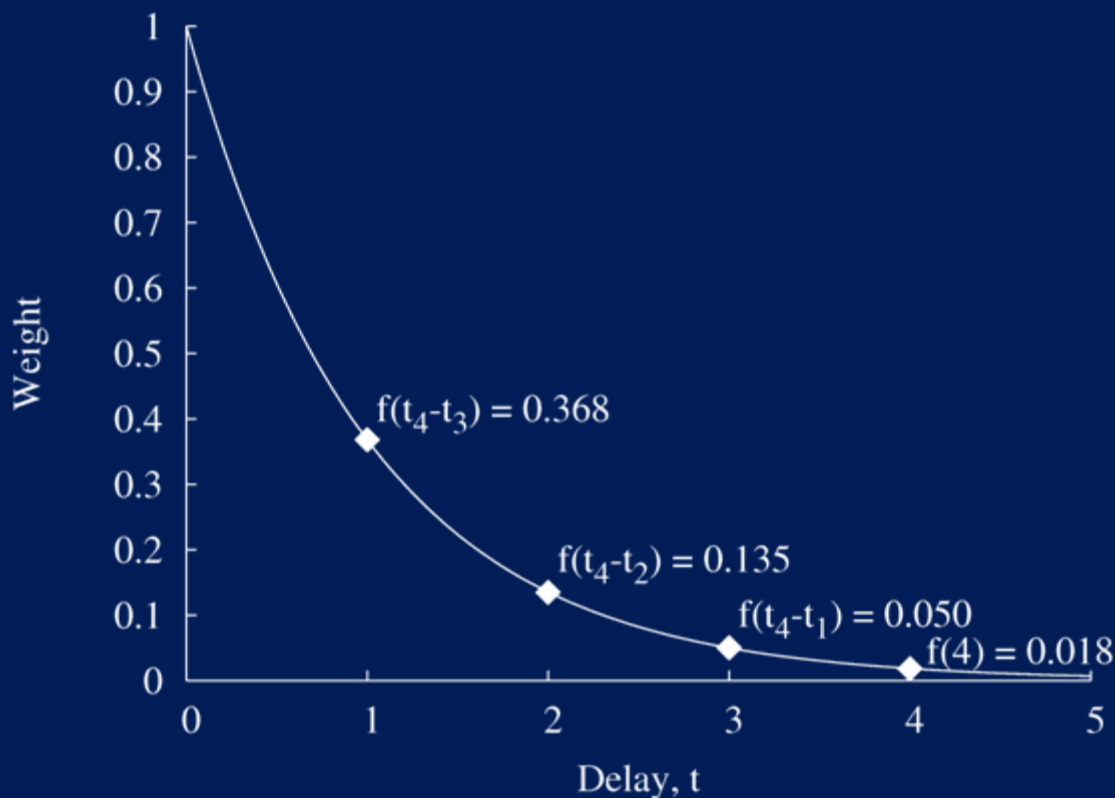
Estimate average causal delay

- Look at all messages into B, plus all $B \rightarrow C$ messages
 - Take **smallest delay** before each $B \rightarrow C$ message
 - Trace-specific upper limit
- $D_{B \rightarrow C}$ = average of these delays
 - Might underestimate D
- Scaling factor $\lambda_{B \rightarrow C} = 1/D_{B \rightarrow C}$
- Create exponential distribution
 - $f(t) = \lambda e^{-\lambda t}$



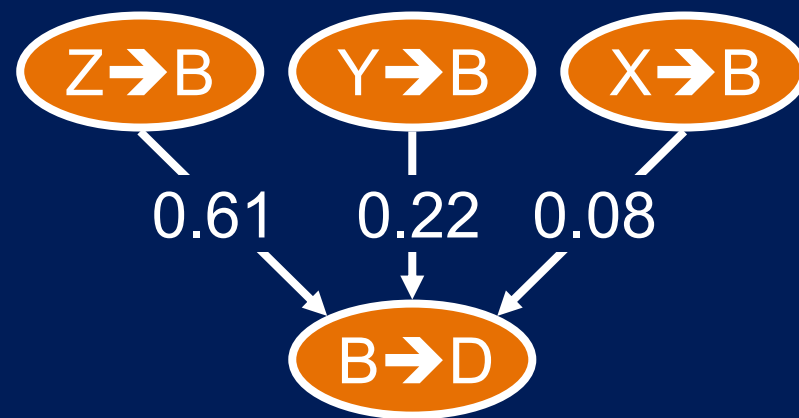
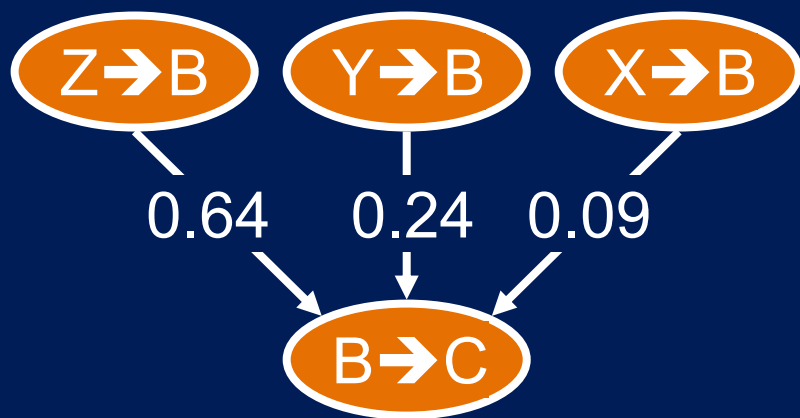
Find and weight possible parent messages

- Use $f(t)$ to find weight of link from each parent



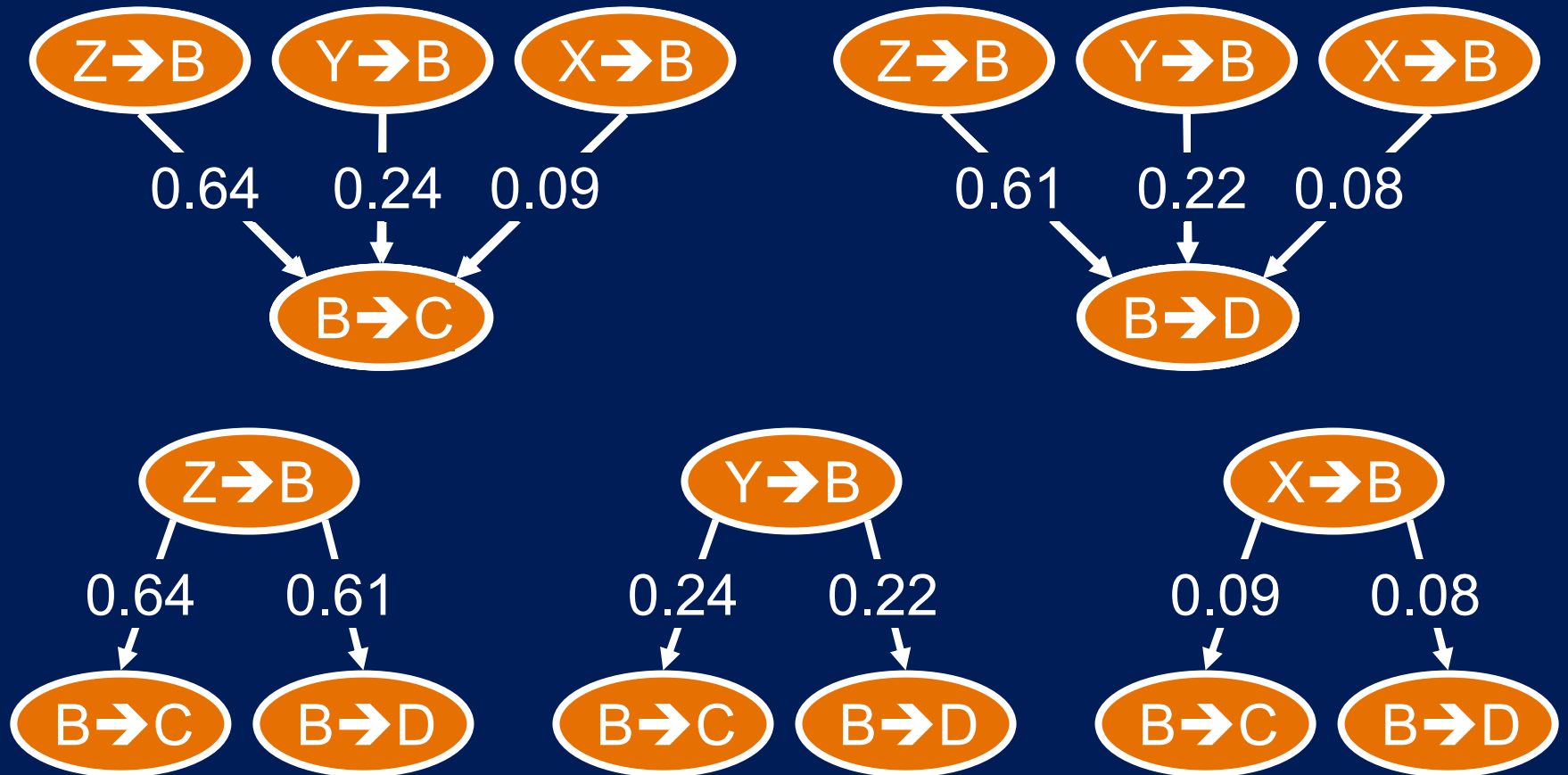
Find and weight possible parent messages

- Normalize so sum of weights to each child = 1
- Possible-parent trees
 - Spontaneous action has small probability, not shown
 - Links to $B \rightarrow D$ are slightly less likely



Build causality trees

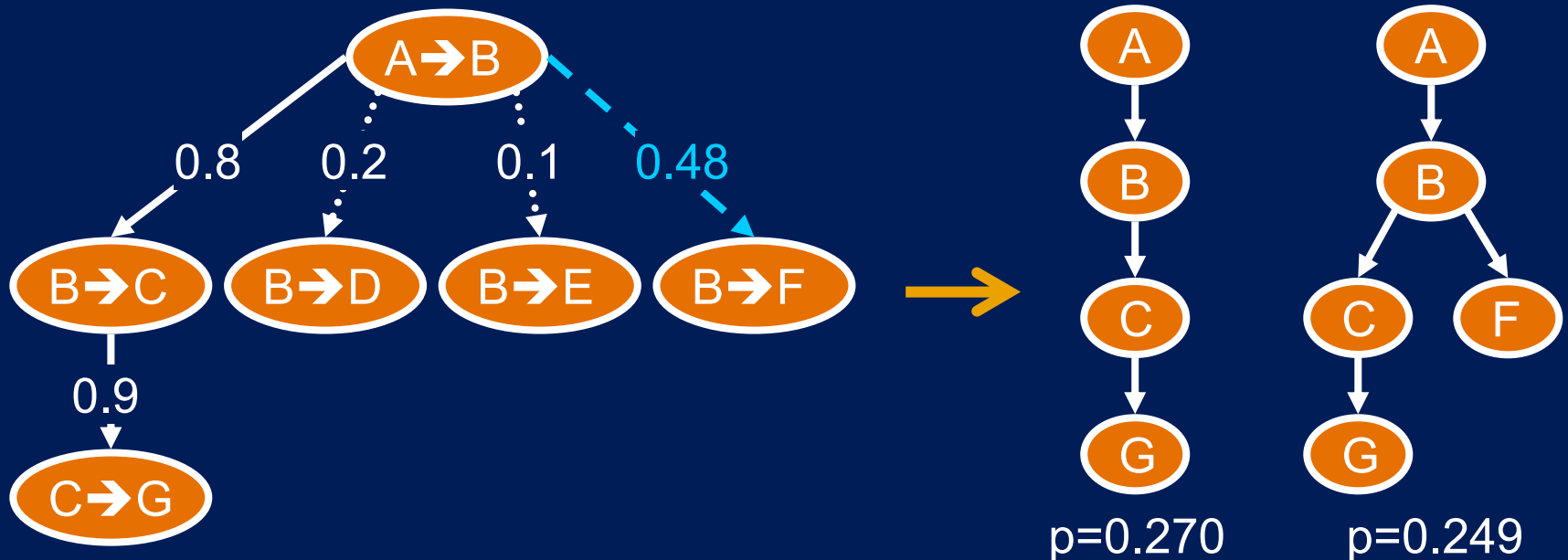
- Invert to get possible-child trees



Build causality trees

- **Build trees from individual links**
 - Use probability to decide whether or not to keep child
 - Some links are “try-both” and generate 2 trees
- **Tree probability is product of link probabilities**

$$p = 0.8 * 0.9 * (1-0.2) * (1-0.1) * (1-0.48) \approx 0.270$$



Build causality trees

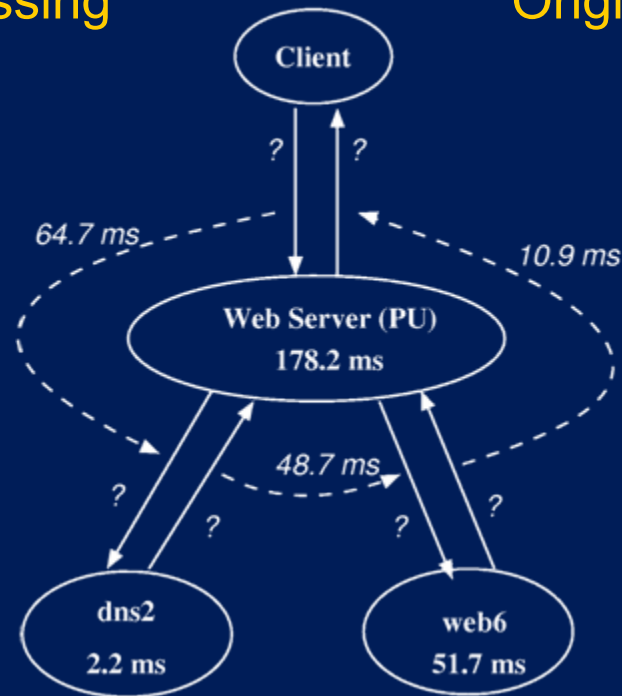
- **Aggregate trees with identical structure**
 - Combine client names and ports for better aggregation
- **Total probabilities for each pattern → ranking**
 - Expected number of instances
 - Highlights paths that appear **many times** with **high confidence**

Outline

- Introduction
- Naming
- Trace capture
- Reconciliation
- Causality analysis
 - Message linking algorithm
- **Results with CoDeeN & Coral**

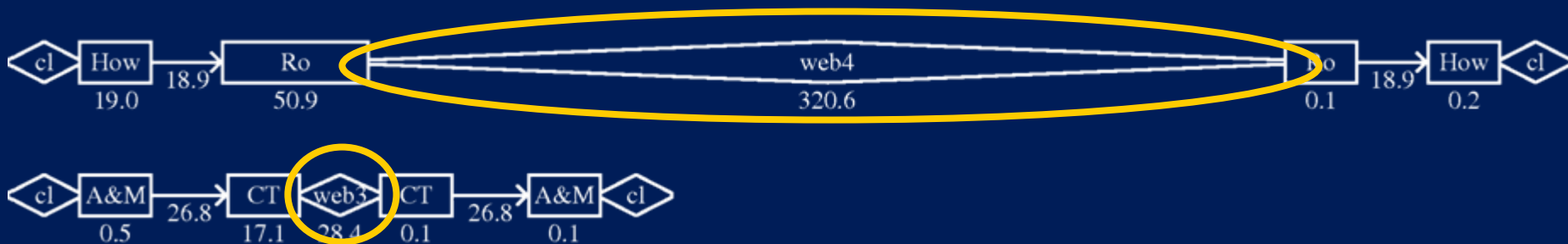
Results: Timeline vs. call tree

- Coral miss path with DNS lookup



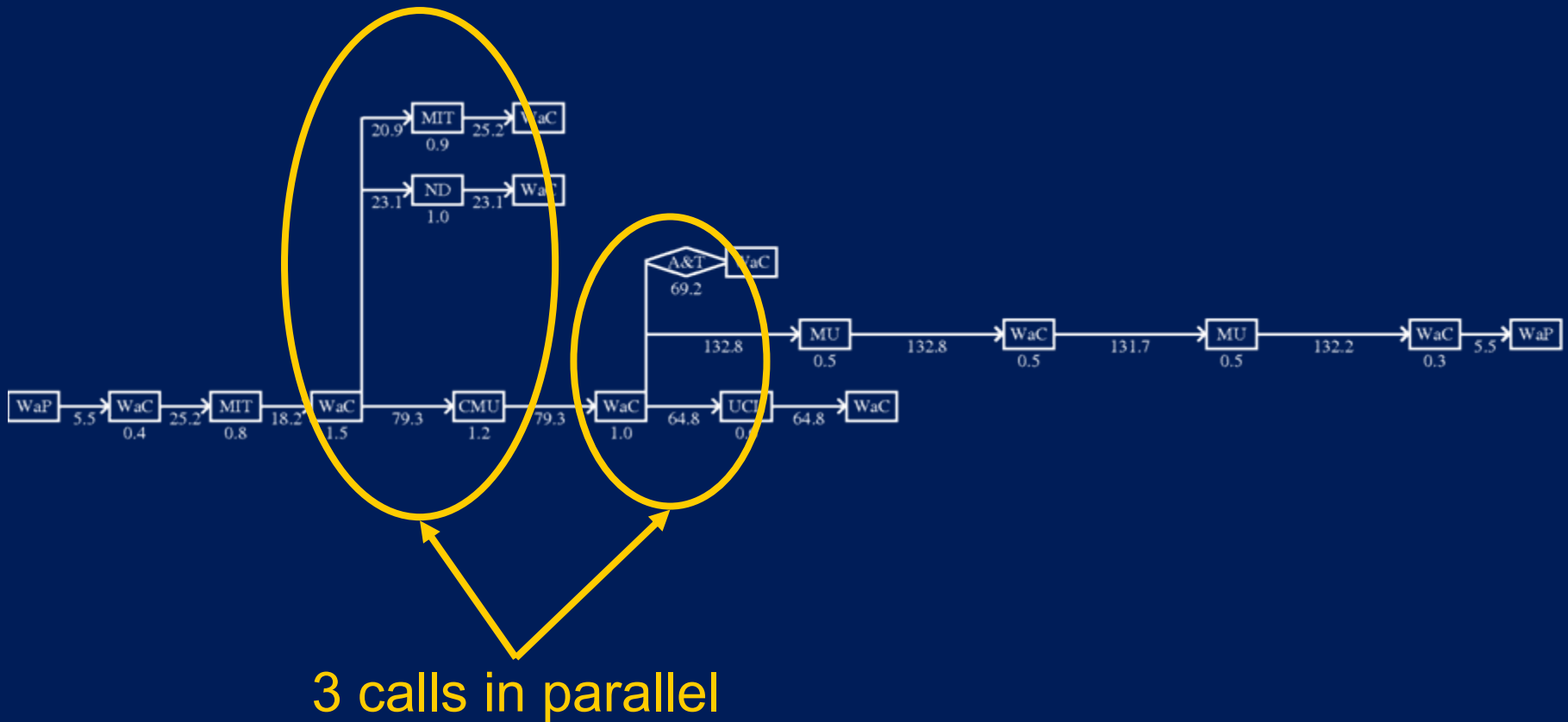
Results: Two CoDeeN miss paths

- Different mean delays at proxies
 - 0.20 to 4.86 ms in different proxies
- Different delays at origin web servers
- All clients aggregated together



Results: Coral DHT lookup

- Three-level DHT lookups



Conclusions

- **WAP5 exposes structure and timing of wide-area applications**
 - Particularly PlanetLab applications
- **Successful analysis of CoDeeN and Coral traces**
 - We found paths that match authors' descriptions of systems
 - We characterized delays at each step and found outliers

<http://www.hpl.hp.com/research/project5/>

Extra slides

Future work

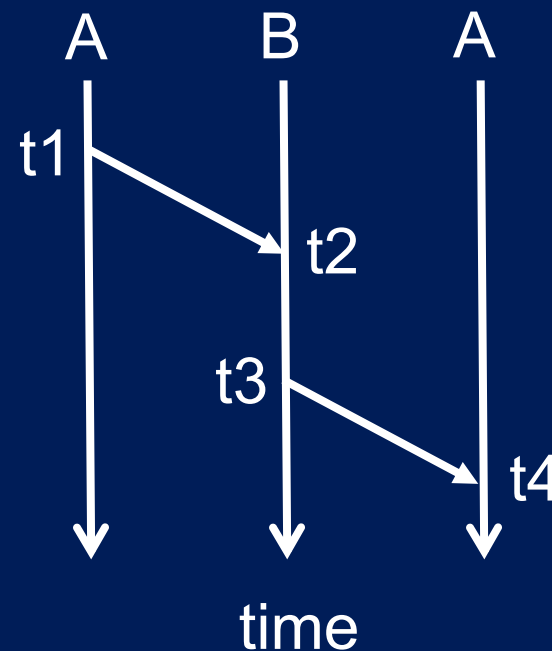
- **Phased behavior**
 - Time-varying patterns
 - Time-varying delays
- **Better aggregation**
 - Coalesce similar path patterns
 - Paths through DHTs
- **Better visualization**

Related work

- **Trace-based analysis tools**
 - Our LAN-based work in SOSP 2003
 - Magpie: detailed picture per machine by using OS-level instrumentation (Event Tracing for Windows)
 - Pinpoint: instrument middleware
 - Others instrument applications
- **Inference-based performance analysis**
 - SLIC uses statistical induction to correlate low-level metrics with SLO violations
- **Interposition tools**
 - Trickle, ModelNet

Node and network latency

- **Node latency = $t_3 - t_2$**
 - Not affected by clock offset
 - All timestamps are local to B
- **Network latency = $t_2 - t_1$; $t_4 - t_3$**
 - Correct for clock offset [Paxson98]
 - $RTT = (t_4 - t_3) + (t_2 - t_1)$
 - $Skew = (t_2 - t_1) - RTT/2$



LibSockCap interposition library

- **Low overhead**
- **Easy to deploy in CoDeeN and Coral**
 - Use existing framework to push out new software
 - Restart process to begin/end trace
- **Advantages**
 - Logical message semantics
 - Per process, not per machine
 - Capture UNIX, TCP, and UDP sockets
- **Disadvantages**
 - Timestamps combine OS and network latency
 - No control packets or fragments