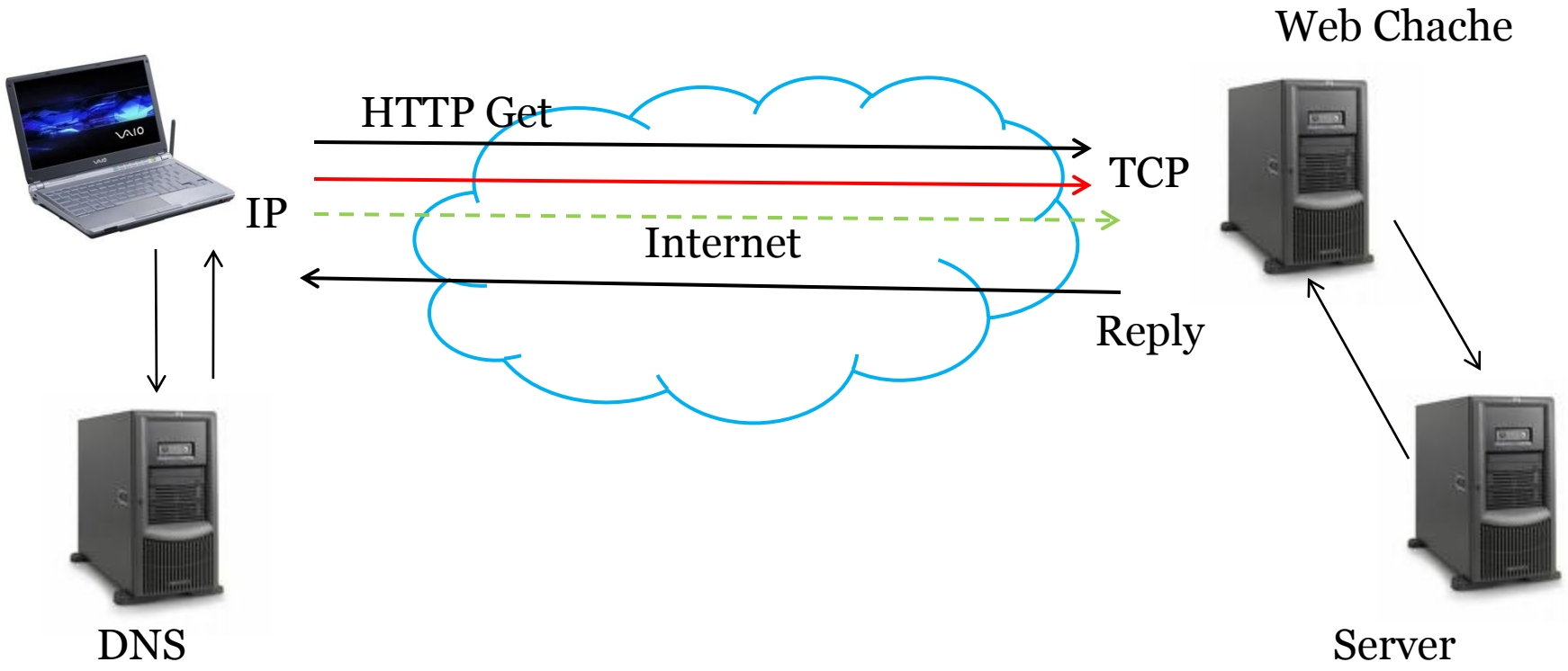
# X-Trace: A Pervasive Network Tracing Framework

R. Fonseca, G. Porter, R.H. Katz, S. Shenker and I. Stoica
NSDI 2007

Presented by:
Virajith Jalaparti
April 8th, 2010.

# A simple HTTP connection

- What are the network protocols/layers involved



# Distributed Debugging

- *Localize a fault*
  - *Logging*
  - Predicate checking
- Try to reproduce the fault
  - Sufficient Logging will help
  - Deterministic execution
- Fix it
  - Humans to be involved
- Efficiency/Usefulness
  - How convenient/complete is method?

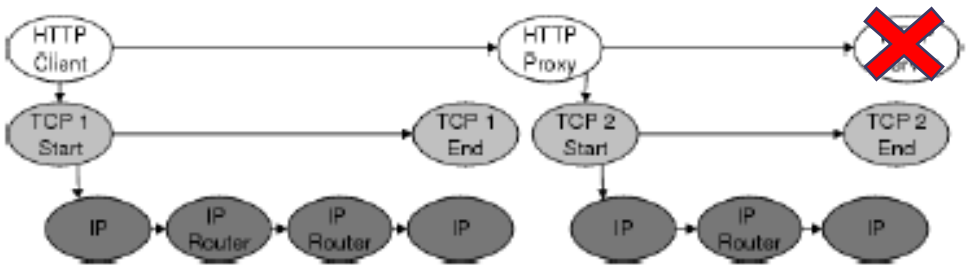
# Myriad of Protocols

- Applications: DNS, web, databases
- Layering: Transport, Network
- Administrative domains: ISPs
- NATs, Proxies, VPNs etc.
- Existing solutions for diagnosis are very specific
  - Traceroute
  - Http monitoring

# X-Trace: a one-for-all solution

- Integrated tracing framework
- Associates metadata with each "*task request*"
- Constructs a "Task Tree"
  - Captures causal relations between different network protocols.

- Simp
  exten




for

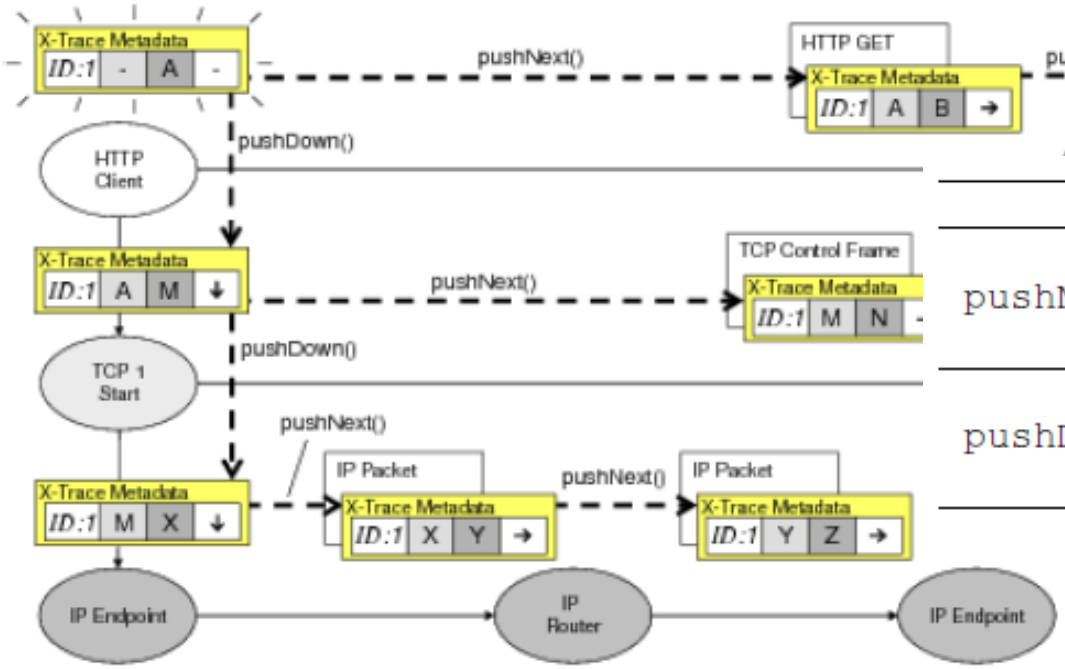
# Design Principles

- Trace done in-band with actual request
  - *Delays execution of each step*
- Trace data collected out-of-band

- Entity that receives the traces decoupled from that which requests them
  - *Requires agreements/authentication etc.*

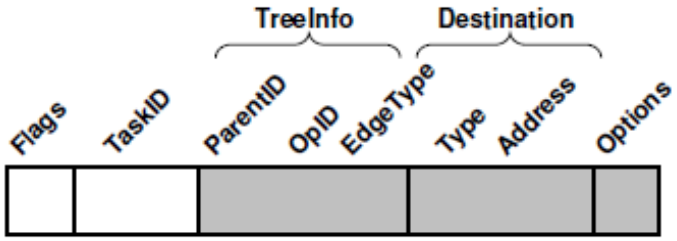
# X-Trace metadata

- Inserted by a client/each layer to construct the task tree




| *TreeInfo* operations | |
|---|---|
| `pushNext()` | next.parentID ⇐ current.opID<br>next.opID ⇐ unique()<br>next.type ⇐ NEXT |
| `pushDown()` | next.parentID ⇐ current.opID<br>next.opID ⇐ unique()<br>next.type ⇐ DOWN |

# X-Trace metadata




- Flags: for specifying which options are present
- Treeinfo: used for constructing the task tree
- Destination: to which the trace report has to be sent
- Options: (type, length,  payload)

# Making Implementations X-Tracable

Original Forwarding Code

```
forwardMessage(msg)
    dest = nextHop(msg)
    lowerLayer.send(msg,dest)
```

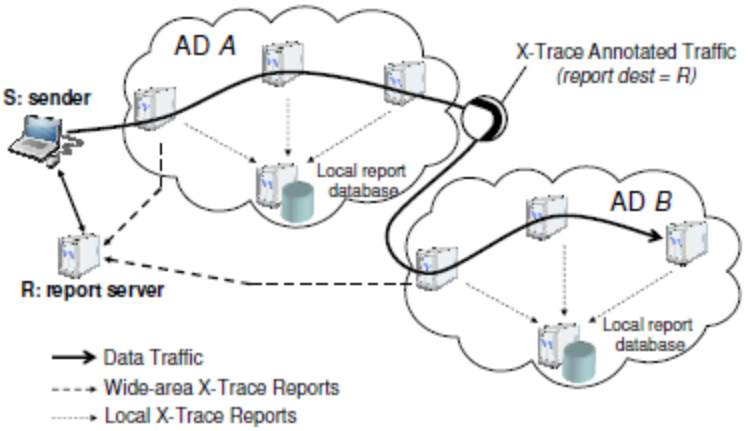
With added X-Trace Propagation

```
forwardMessage(msg)
    dest = nextHop(msg)
    xtr = msg.getXTraceMetadata()
    /* Propagate to the next hop */
    msg.setXTraceMetadata(xtr.pushNext())
    /* Propagate to the lower layer */
    lowerLayer.setXTraceMetadata(xtr.pushDown())
    lowerLayer.send(msg,dest)
```

- Requires keeping track of causal relations for propagating the metadata received by an application

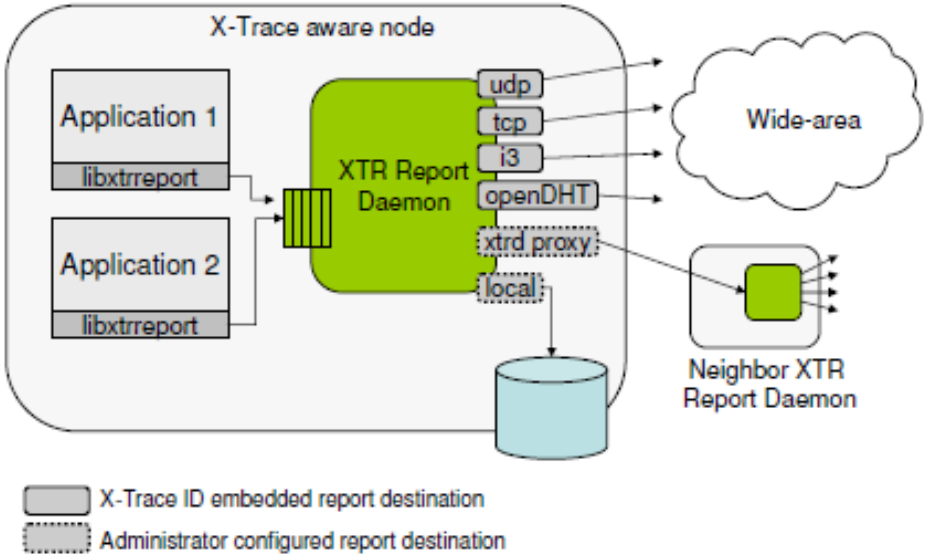
# Task Tree reconstruction

- Metadata used to specify recipient of the data collected at a node
  - Need not be the initiator of the task
  - Can be different across different domains
- Each laye… …the task was processed
  - *What sh… …be useful?*
- Tree reco… …ers in trace data



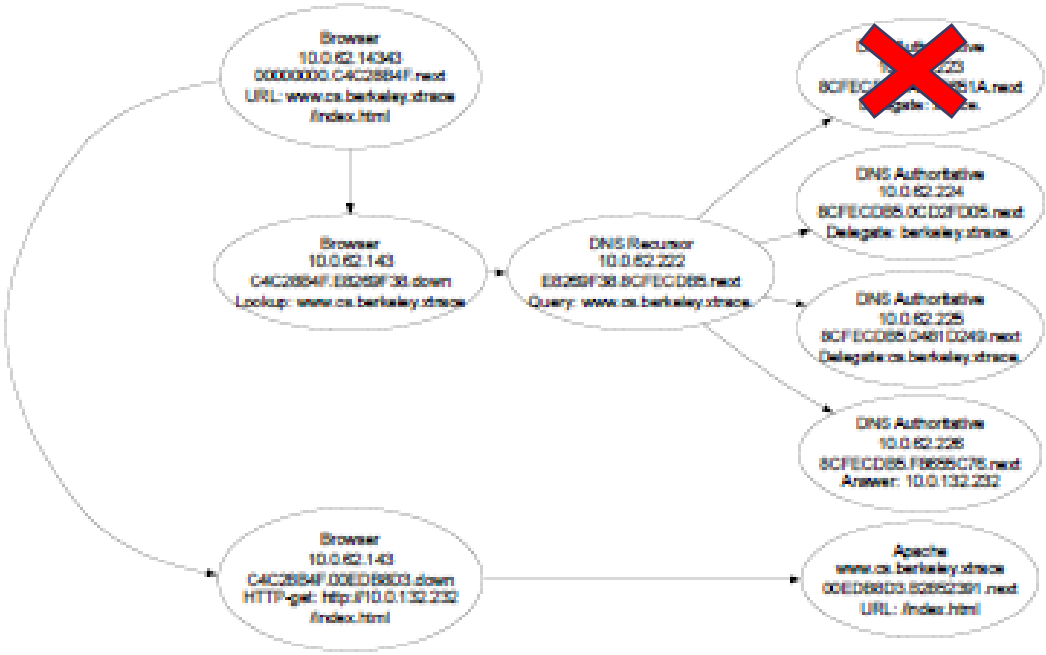
# Generating Reports

- Libxtrreport: thin library

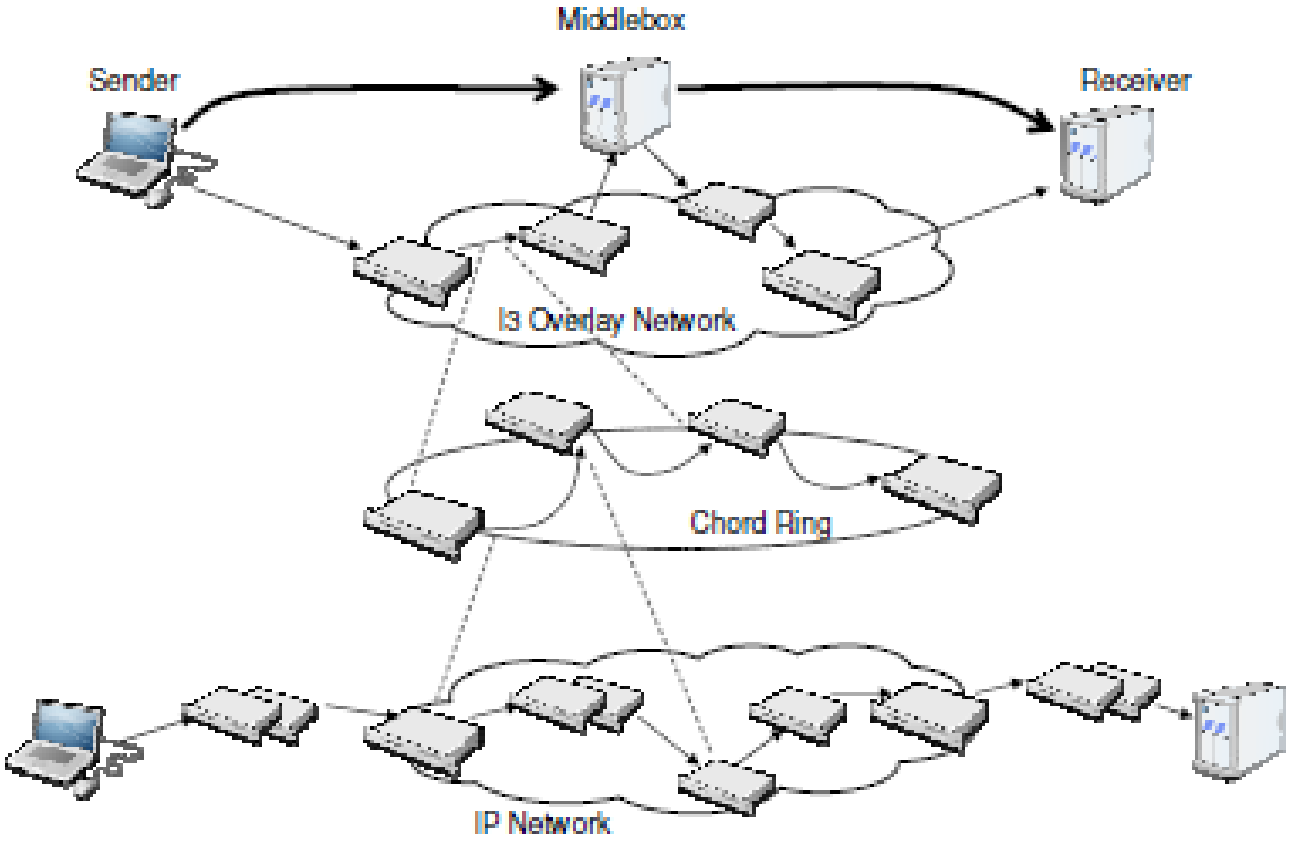


# X-Tracing Web Requests

- EDNSO options used



# X-Tracing an overlay network



# Experimental Setup

- 3 nodes used
  - I3 nodes
  - Chord nodes
- Simple number application
  - Source
  - Middlebox
  - destination

Figure from CS525 Spring 2009 presentation by Chi-Hung Lu

# Complete Task Tree

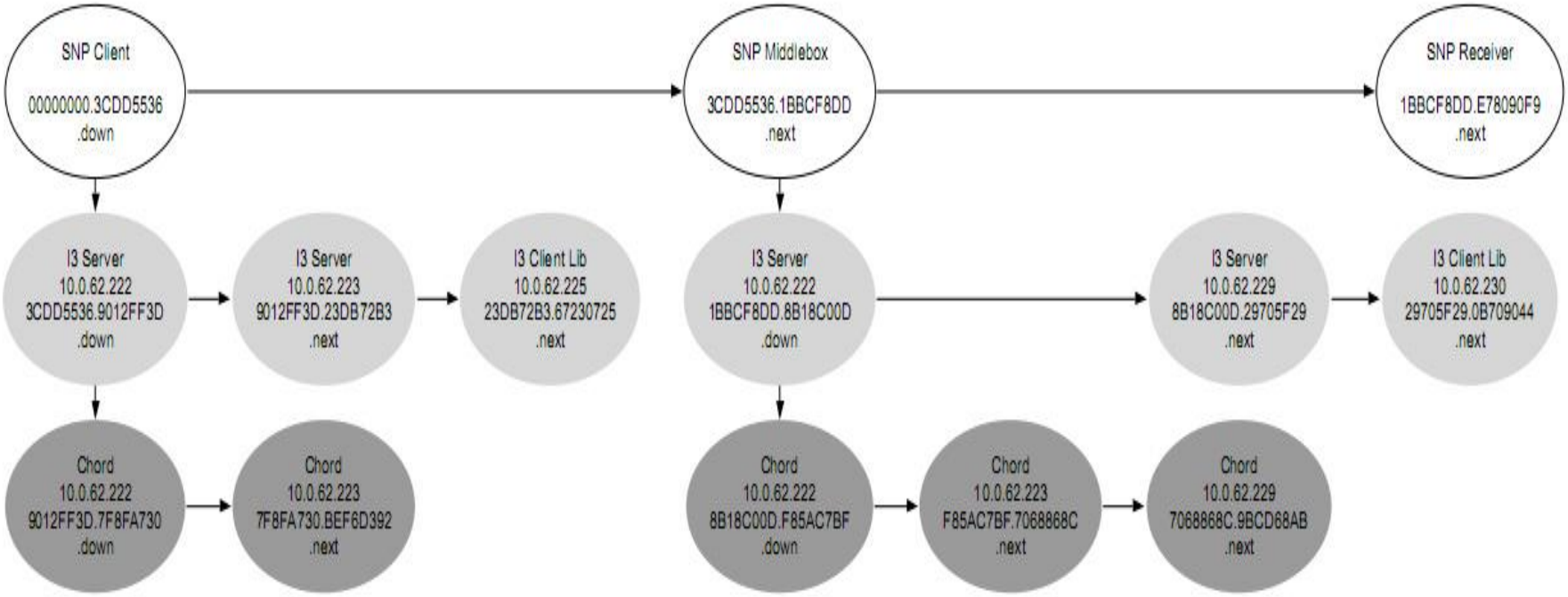


Figure from CS525 Spring 2009 presentation by Chi-Hung Lu

# Receiver fails

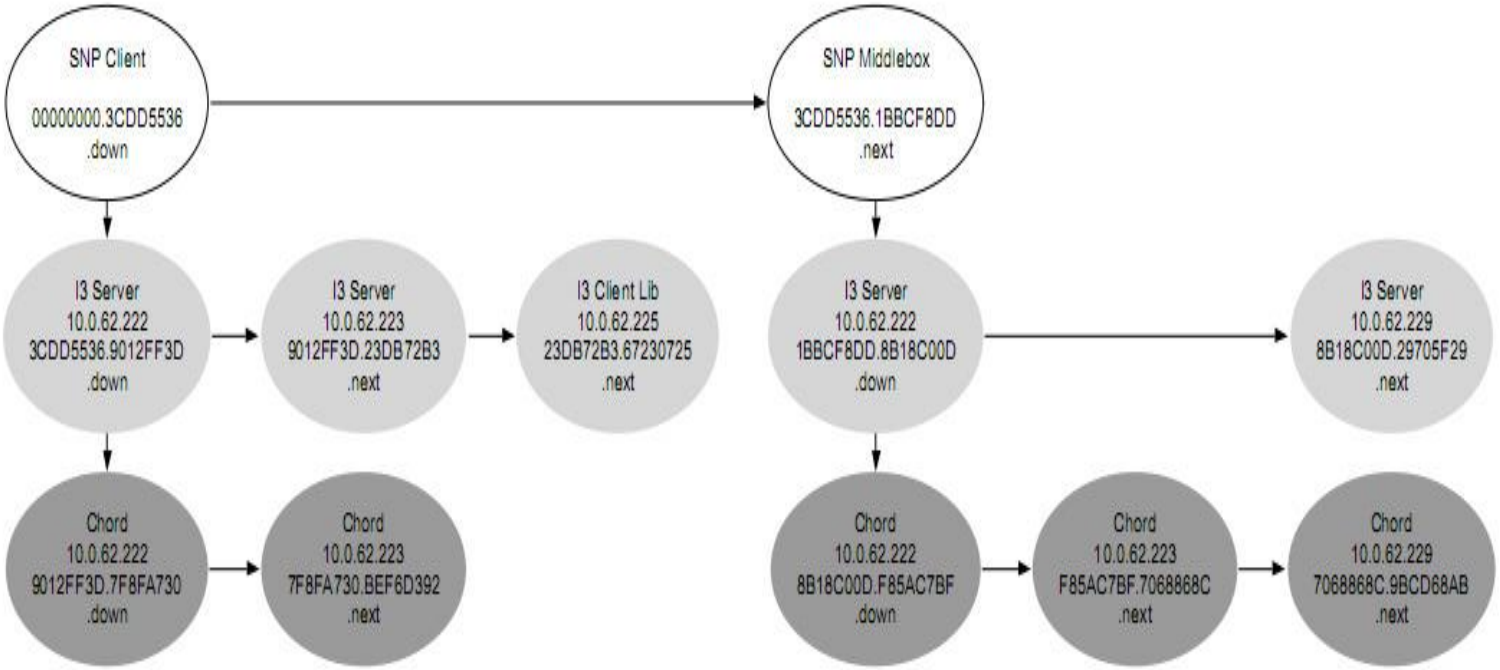


Figure from CS525 Spring 2009 presentation by Chi-Hung Lu

# Middlebox Process crashes

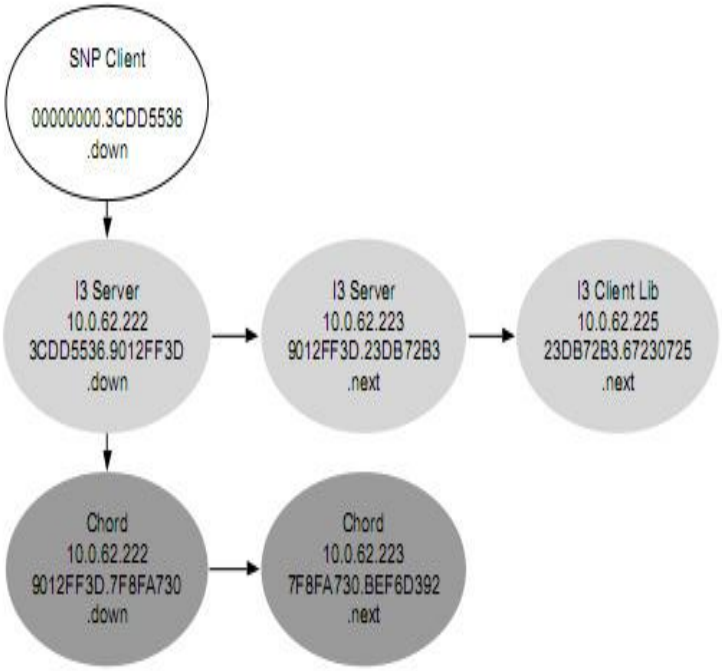
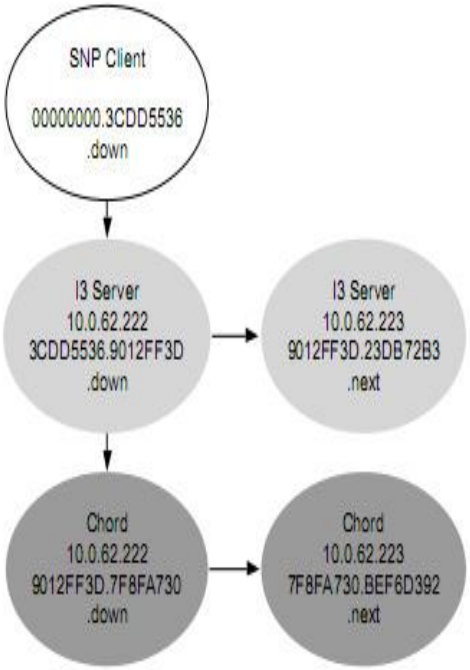


Figure from CS525 Spring 2009 presentation by Chi-Hung Lu

# Middlebox host fails



# Discussion

- What does X-Trace provide?
  - Generally useful only to locate point of crash
  - Reports need to very detailed if they have to be used for debugging purposes!
- Can we use X-Trace for Routing?
  - Need not result in a tree!
  - Where does it end?
  - No simple concept of task
- Modifying Applications/Protocols!
  - Guidelines for designing new applications

# Discussion

- Partial Deployment
  - Better than none!
- Privacy concerns
  - Can be used to easily keep track of a user's tasks
- Requires unique task id
  - <Ip address, rand number> can be used but many hosts don't have a public Ip.
- Humans have to identify/report errors!