

# TESTING IN PRODUCTION: ENHANCING DEVELOPMENT AND TEST AGILITY IN A SANDBOX ENVIRONMENT

**Xiudong Fei**

[xiufei@microsoft.com](mailto:xiufei@microsoft.com)

**Sira Rao**

[sirarao@microsoft.com](mailto:sirarao@microsoft.com)

# Before we begin ...



## Thanks to our Reviewers

Moss Drake

Chris Blain

# Agenda

- What's Testing in Production (TiP)
- Sandbox environments and scope
- Why TiP
- What are challenges of TiP
- How SilverlightDetour addresses the challenges
- Results
- Takeaways

# What is Testing in Production (TiP)

- Using production environments, topologies to validate system or application
- Capture actual usage data to identify, verify reported issues
- Use of tools to provide a window into a production system

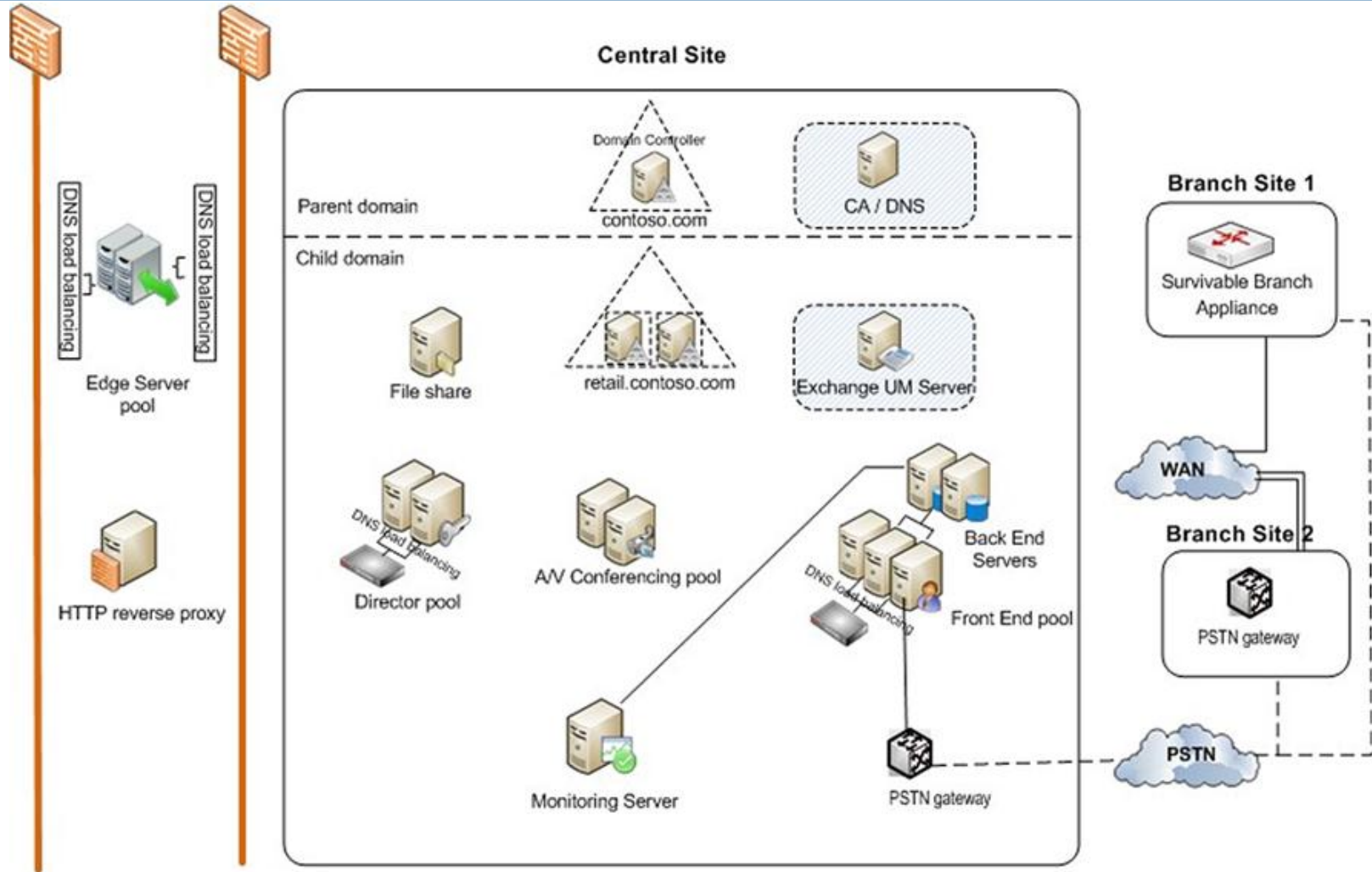
# Sandbox Environment and Scope

- Sandbox environment
  - Limits resources for guest program to run in, e.g. disk space, memory
  - Usually closed, restricts application from access to system resources and what application can modify
  - Scope is Plugin based applications e.g. Silverlight
  
- Silverlight – Plugin, app framework; enables Interactive media experiences and Rich business apps
  
- Microsoft Lync Web app (LWA) – Real time communications, collaboration app

# Why should we TiP

- Limitations of test environments
  - Load Balancers, PSTN GW
  - Concurrent users, user activities
- Find problems at a larger scale than test environments
- Uses complexity of production environments to our benefit
- Could reach faster development and test cycles

# Complexity of Production Environments



# Challenges of TiP

- Production environment affects agility of development, testing
  - Install / Upgrade back-end environments
  - Access permissions
- Difficult to perform debugging, troubleshooting tasks
- Logging Limitations
  - Hard or impossible to get logs e.g. application crash or hang
  - Control amount of logging, at different layers
  - Limited size and Logs often overwritten
- How to Test and Dogfood early and Iterate on Feedback



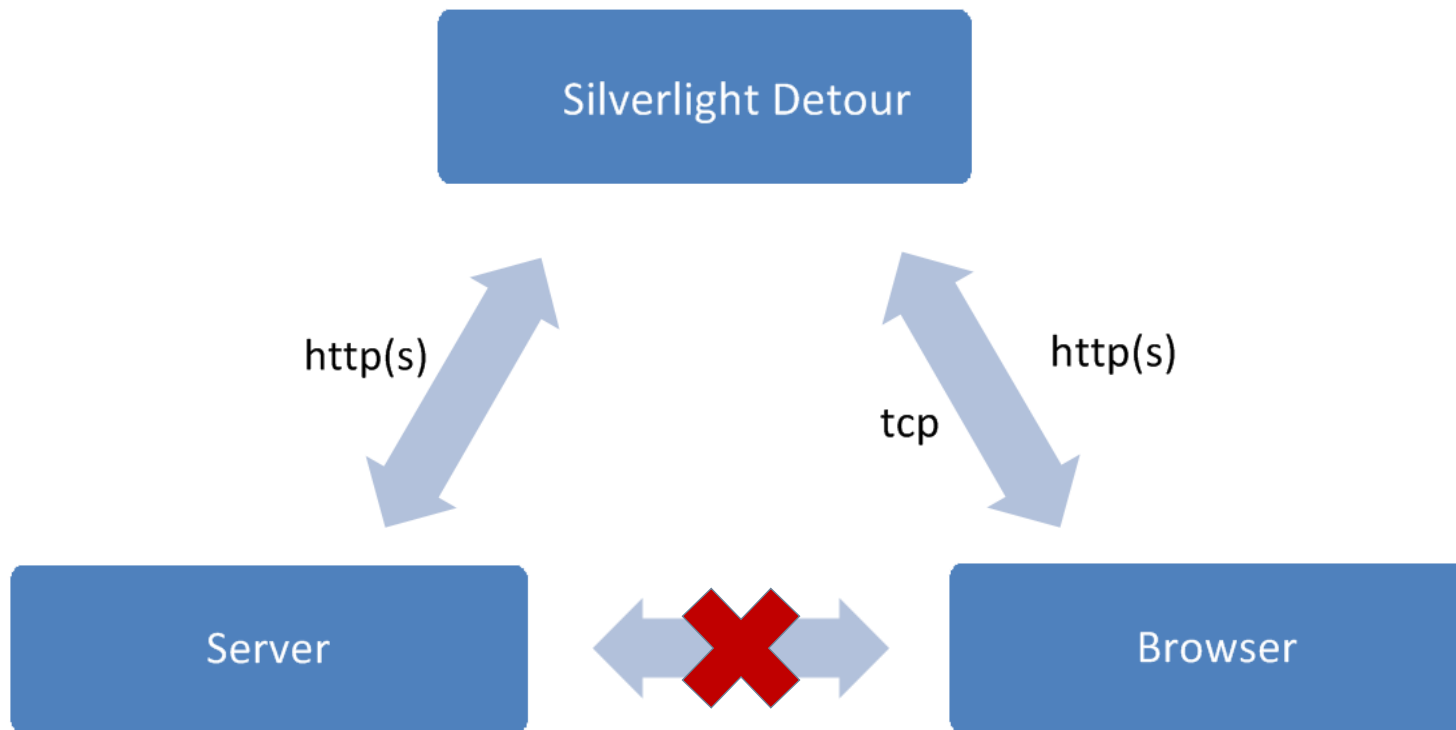
# SilverlightDetour Addresses the Challenges

- Bypass production Install, Access issues
- Provides Out of process Debugging with no disruption to Production system
- Provides Logging mechanisms
- Provides seamless and transparent replacement of client application for testing and Dogfood

# SilverlightDetour Solution Architecture

## DETOUR concept

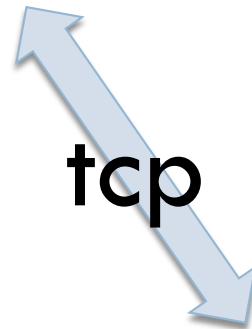
- Intercept product package
- Replace the whole product package
- Inject test dlls, javascript



# Addressing the Logging Challenges



Limited logging size in Sandbox

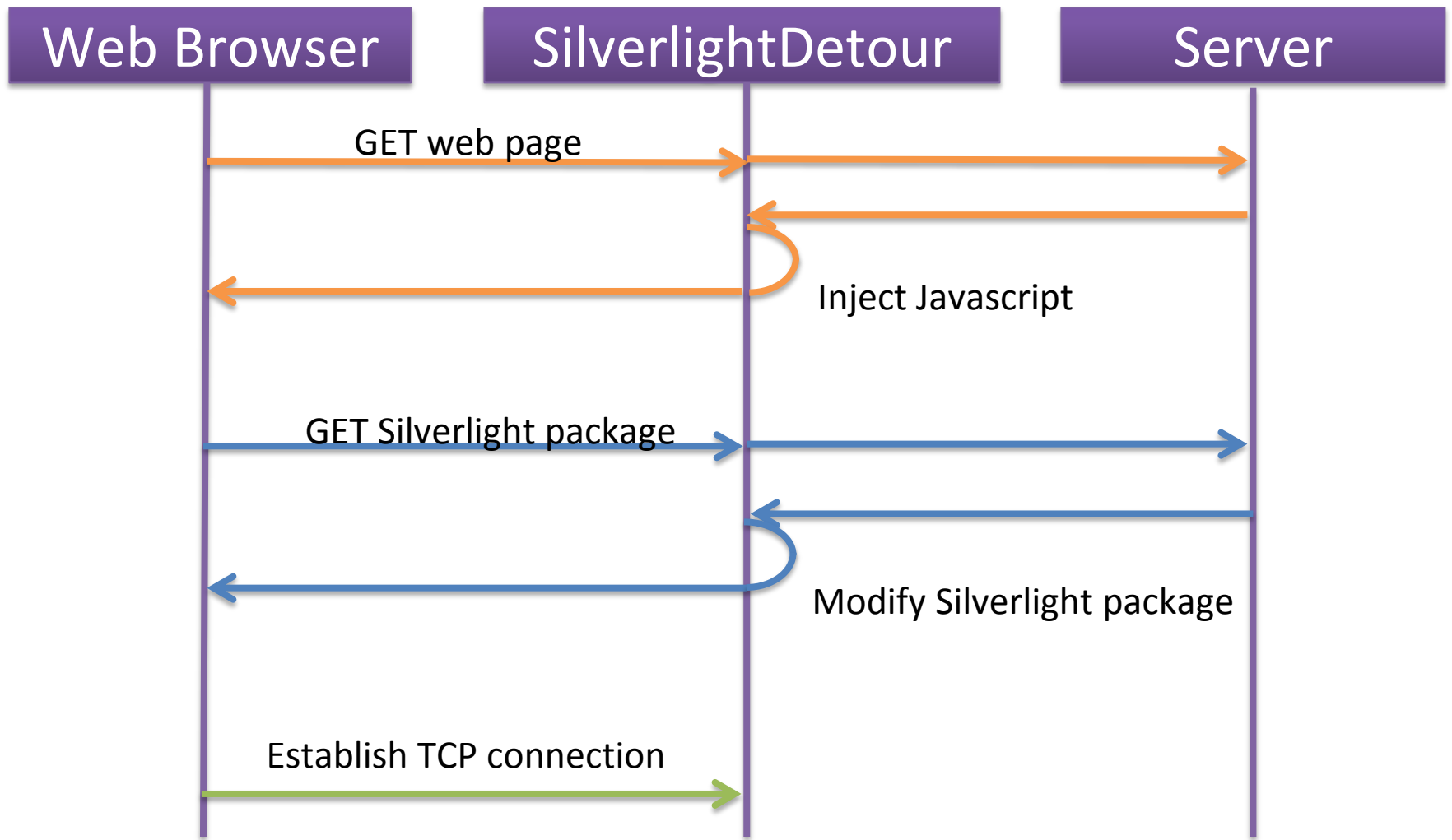


Unlimited space

**SilverlightDetour**



# Sequence Diagram



# Debugging Made Easy

```
PS>$disp = [MS.SLDetour]::GetDisplayNameTextBox()
```

```
PS>$disp.Text = "SilverlightDetour"
```

```
PS>$disp.TextAlignment
```

Left

```
PS>disp.TextAlignment = 2
```

```
PS>[MS.SLDetour]::ClickJoinButton()
```

The screenshot displays two windows side-by-side. On the left is a PowerShell console window titled 'SilverlightDetour is listening on port 8998@10.121.129.229 [XIUODN]'. It lists various DLLs and assemblies loaded, such as \AppShare\XML\, \AppShare\JS\, and \PowerPoint\DhtmltoXaml\. It shows a log of an incoming Silverlight connection and the execution of PowerShell commands: `$disp = [MS.SLDetour]::GetDisplayNameTextBox()`, `$disp.Text = "SilverlightDetour"`, `$disp.TextAlignment` (returns Left), `disp.TextAlignment = 2`, and `[MS.SLDetour]::ClickJoinButton()`. On the right is a Microsoft Lync Web App window in Internet Explorer. The page title is 'Microsoft Lync Web App'. The language is set to English. Under the 'Join as a guest' section, the name 'SilverlightDetour' is entered in a text box, and the 'Join Meeting' button is highlighted with a purple border. Below this, it says 'Joining the meeting...' and a 'Cancel Joining' button is visible. The footer of the Lync Web App shows the Office logo and copyright information: '© 2010 Microsoft Corporation. All Rights Reserved.'

# Other Possibilities of SilverlightDetour

- Provides Remote Assistance
- Alternate UX Automation mechanism
  - Uses TCP connection
  - Manipulate Silverlight objects
- Fault Injection and Security tests
  - Change error codes
  - Change XML schema and Fuzz test XML parsers

# Results

**Agility Improved** in development, testing and iterated faster to achieve desired quality levels in production environments

**Savings** of about 2 person days per month on install / upgrade / maintenance of test topologies

**Increased Overall productivity** of Dev, test teams and better collaboration within Product teams

**Tool identified issues** impacting various aspects of product – UI, functionality, stability, security and back-end environment

**Dogfooding** improved and Users provided with latest version of Lync Web app

# Takeaways

- We envision Plugin-based applications could employ our framework for TiP
- Teams developing Web-based apps can employ principles mentioned
- These techniques will help improve development, test agility
- Use framework for UI automation, Remote assistance and Fault Injection

Need thought, frameworks, solutions that will help each of us embrace Testing in Production ...



# Thanks and Questions?

