Na Kika: Secure Service Execution and Composition in an Open Edge-Side Computing Network

Robert Grimm, Guy Lichtman, Nikolaos Michalakis
Amos Elliston, Adam Kravetz, Jonathan Miller, Sajid Raza
New York University
Dynamic Content: Easy to Build, Hard to Scale

- Dynamic content is increasingly popular, easy to create and publish
- Example: mashups
  - chicagocrimes.org = Crime reports over Google Maps
  - zillow.com = Real Estate stats over Microsoft Virtual Earth
- Easy to realize on a home server
  - PHP, Python, ASP, JSP, ...
- Collaborate, plug together
- Does not scale
We Need a New Delivery Platform

- Need a platform that is scalable, extensible and secure
  - Near the client, supports mixing/mashing, controls hosted code
- Clusters amplify resources, not necessarily near the client
  - [TACC, Veritas, Linux-HA]
- Edge-side hosting targeted at big trusted sites
  - [Akamai, ACDN, ColTrES, Tuxedo, vMatrix, WebSphere]
- P2P collaborative architectures limited to static content
  - [Coral, CoDeeN, CobWeb]
- Some efforts provide containment but not composition
  - [Active Cache, SDT]
- Na Kika reconciles extensibility with security
Na Kika Architecture

- DNS redirects clients to nearby proxies
- Proxies organized in structured overlay for caching static content
- Sites publish scripts that are treated like static content
- Scripts are executed and composed at the edge for scaling dynamic content
Programming Model

- Write scripted code:
  - Easy, already familiar
  - Javascript
- Structure functionality inside event handlers
  - onRequest handler
  - onResponse handler
- Specify handlers as a Policy object

Client → Na Kika proxy → Server

```javascript
p = new Policy();
p.onRequest = function() { ... }
p.onResponse = function() { ... }
p.register();
```
Service Modularity

- Leverage descriptive nature of HTTP messages
  - URL
  - Client IP address
  - Method
  - Headers
- Select handlers based on HTTP message properties
- Execute the most specific match

```javascript
p1 = new Policy();
p1.url = ["*.zillow.com/*"];
p1.client = ["0.0.0.0/0"];
p1.method = ["GET"];
p1.onRequest = function(){ ... }
p1.onResponse = function(){ ... }
p1.register();
```

```javascript
p2 = new Policy();
p2.url = ["*.zillow.com/*"];
p2.client = ["128.122.0.0/16"];
p2.method = ["GET"];
p2.onRequest = function(){ ... }
p2.onResponse = function(){ ... }
p2.register();
```
Event handler pair mimics proxy structure

A series of event handlers is called a **pipeline**

Compose handlers via the `nextStages` property.
Admission & Emission Control

• Reuse same mechanisms
• Handler selection
• Composition

• Make security policies extensible

• Insert two extra pipeline stages
  • ClientWall near client
  • ServerWall near server

```javascript
var p = new Policy();
p.method = ["GET", "POST"];
p.onRequest = function()
  { Request.terminate(ACCESS_DENIED); }
p.register();
```
Containing Hosted Code

- Scripts are sandboxed
  - Select native libraries, no direct access to system

- No hard quota
  - Hard to set appropriate limits on a shared resource

- Control consumption *only* under congestion
  - Congestion control enforces collaboration
  - If no congestion, do nothing (no hard limits)
  - Otherwise, throttle requests
  - Terminate largest consumers as a last resort.
Evaluation

- How does Na Kika compare to a single server?
  - Wise-MD web-based learning application

- How easy is it to extend functionality/security?
  - Examples of Na Kika extensions

- Is throttling/termination effective?
  - Both under overload and malicious scripts [see paper]
Wise-MD

- Wise-MD is a web-based education tool developed at NYU medical
  - (formerly known as SIMMs)
- Global participation
  - U.S. + Australia
- Multimedia intensive
  - 1 GB total content
- Dynamic
  - HTML generated from XML and XSL stylesheet
Wise-MD on Na Kika

- 1 developer, 100 + 130 lines of code, 2 days to port
- Comparison between single server and Na Kika
  - Clients and proxies run on 12 PlanetLab nodes each
  - For multimedia content accessed by 240 clients

<table>
<thead>
<tr>
<th></th>
<th>Clients seeing more than 140Kbps bandwidth</th>
<th>Failures seen by clients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Server</td>
<td>0%</td>
<td>60%</td>
</tr>
<tr>
<td>Na Kika Cold Cache</td>
<td>11.5%</td>
<td>5.6%</td>
</tr>
<tr>
<td>Na Kika Warm Cache</td>
<td>80.3%</td>
<td>1.9%</td>
</tr>
</tbody>
</table>
Extensibility in Action

- Na Kika Pages (NKP)
  - Programming model similar to PHP, JSP, ASP
- Image transcoding
  - Transforms images to JPG, scales them down
- Annotated Wise-MD
  - Layer electronic post-it notes over Wise-MD
- Content blocking
  - First additional stage creates policy based on blacklist
  - Second new stage executes policy, rejecting illegal URLs
Annotated Wise-MD In Action

Module Selection:

**Adrenal Adenoma:**
In this module, you will learn how to work up an adrenal adenoma and gain comprehension of the complex physiology of the adrenal gland.

**Carotid Stenosis:**
In this module, you will learn about cerebral vascular occlusive disease and its relation to symptoms.

**Cholecystitis:**
In this module you will be introduced to the pathophysiology of Acute Cholecystitis and other diseases related to Cholelithiasis.
Extensibility in Action

- Na Kika Pages
  - Programming model extension similar to PHP, JSP, ASP

- Image transcoding
  - Transforms images to JPG, scales them down

- Annotated Wise-MD
  - Layer electronic post-it notes over Wise-MD

- Content blocking
  - First additional stage creates policy based on blacklist
  - Second new stage executes policy, rejecting illegal URLs
Easy to build, Easy to Scale

- Less than 100 lines of code for each application
  - Annotations relied on 180 lines of external code
- Less than 8 hours to write and debug
- Deployment at the edge scales
Limitations & On-going Work

- Source code must be made public
- Sites gain capacity, but lose control over performance
- Unsuitable for applications with large databases
  - Hard state replication in place, SPECweb99 [see paper]
  - Better replication strategies
- Proxies assumed trusted
  - Protection against misbehaving/malicious proxies
- Resource management as congestion control
Conclusions

- Na Kika scales dynamic content
  - Focus on collaborative efforts

Contributions

- Same mechanism for defining functionality and policies
- Congestion-based resource management
www.nakika.org
unused
Na Kika: Secure Service Execution and Composition in an Open Edge-Side Computing Network

Robert Grimm, Guy Lichtman, Nikolaos Michalakis, Amos Elliston, Adam Kravetz, Jonathan Miller, Sajid Raza
the octopus-god

*Na Kika*: The octopus-god of the Gilbert Islands. His many arms served him well when he shoved up the earth from the bottom of the sea to form the islands, the beaches and the rocks...

**Encyclopedia Mythica™**
We Need a New Delivery Platform

- Trusted deployments limited to amplifying existing sites.
  - [Akamai, aCDN, ColTrES, Tuxedo, vMatrix, WebSphere]

- Open architectures limited to static content
  - [Coral, CoDeeN, CobWeb]

- Some open architectures do provide code containment but no composition.
  - [Active Cache, SDT]

- Need a platform that has it all:
  - scalable → move resources near clients.
  - easily and openly extensible, composable
  - secures the hosted code
The keys to Na Kika’s extensible architecture:

- Applications are built through pipelines.
- Pipeline stages are developed using Policy objects.
- Stages are selected by matching HTTP predicates (vertical extensibility).
- Policy URLs used for linking stages by different providers (horizontal extensibility).
edge-side scripting

- Javascript: easy, familiar to programmers.
- Access to HTTP requests and responses via `Request` and `Response` objects.
- Objects, called `vocabularies`, provide access to:
  - soft state
  - hard state
  - content processing
  - selected system functionality
- One object however is special...
  - The `Policy` object serves as an application building block.
security challenges

- Secure the pipeline:
  - client - Na Kika
  - stage - stage
  - Na Kika - server

- Secure script execution on Na Kika host:
  - resource access control
  - resource management

  without sacrificing extensibility
securing the host

- **Pipeline isolation:**
  - pipeline = process
  - access system through vocabularies only.

- **Script isolation:**
  - stage = script execution context.
  - scripting engine isolates contexts.

- But host resources can still be abused.
securing the host

- Scripting helps again:
  - Easier to control a small execution engine.
  - Only selected system functionality is exposed via platform libraries.
  - Each script runs on its own isolated environment.
motivation

- An application such as the SIMMs:
  - Delivers multimedia-heavy, personalized content.
  - Scales to large, often global audience.
  - Developed and used by a large, geographically diverse community.
  - Has multiple content creators and service developers.
  - High degree of collaboration among organizations.

- Can it be supported by current architectures?
single server bottleneck

- most common, inexpensive solution
- but it’s a bottleneck
  - CPU, bandwidth are limited.
  - administration requires a form of organization.
  - same for collaboration/development.
layering still a bottleneck

- straightforward service layering helps with:
  - administration
  - collaboration/development
- still it’s a performance bottleneck (CPU, bandwidth)
Security Issues

- Servers and Clients are not trusted
  - Control admission and emission in the pipeline
- Hosted scripts are not trusted
  - Limit access to system resources
  - Control resource consumption
Implementation

- Implementation based on Apache
- Event handlers = Apache input and output filters
- Pipeline isolation = Different processes
- Scripting on Apache via user-level threads
  - Hide piecemeal content handling
  - Isolate execution
- Resource Management = external process
  - Tracks per process usage
  - Signals throttling and termination
We Need a New Delivery Platform

Need a platform that is

- Scalable ➔ amplify resources + collocate with clients
- Extensible ➔ allow composition, mixing, mashing
- Secure ➔ protected from untrusted code

Clusters amplify resources but not near the client

- [TACC]

Edge-side hosting targeted for trusted deployments

- [Akamai, ACDN, ColTrES, Tuxedo, vMatrix, WebSphere]

P2P architectures limited to static content

- [Coral, CoDeeN, CobWeb]

Some provide containment but not composition

- [Active Cache, SDT]
Wise-MD on Na Kika (latency)

The graph shows the performance of Wise-MD on Na Kika with varying latency times. The x-axis represents the total time in seconds, and the y-axis represents the percentage of data processed. The graph includes lines for different client counts and cache states (warm and cold), with 120, 180, and 240 clients. The single server is indicated on the far right of the graph.