

Na Kika

Towards an Open Edge-Side Computing Network

**Robert Grimm, Guy Lichtman,
Nikos Michalakis**

New York University

A New Class of Web-Based Applications

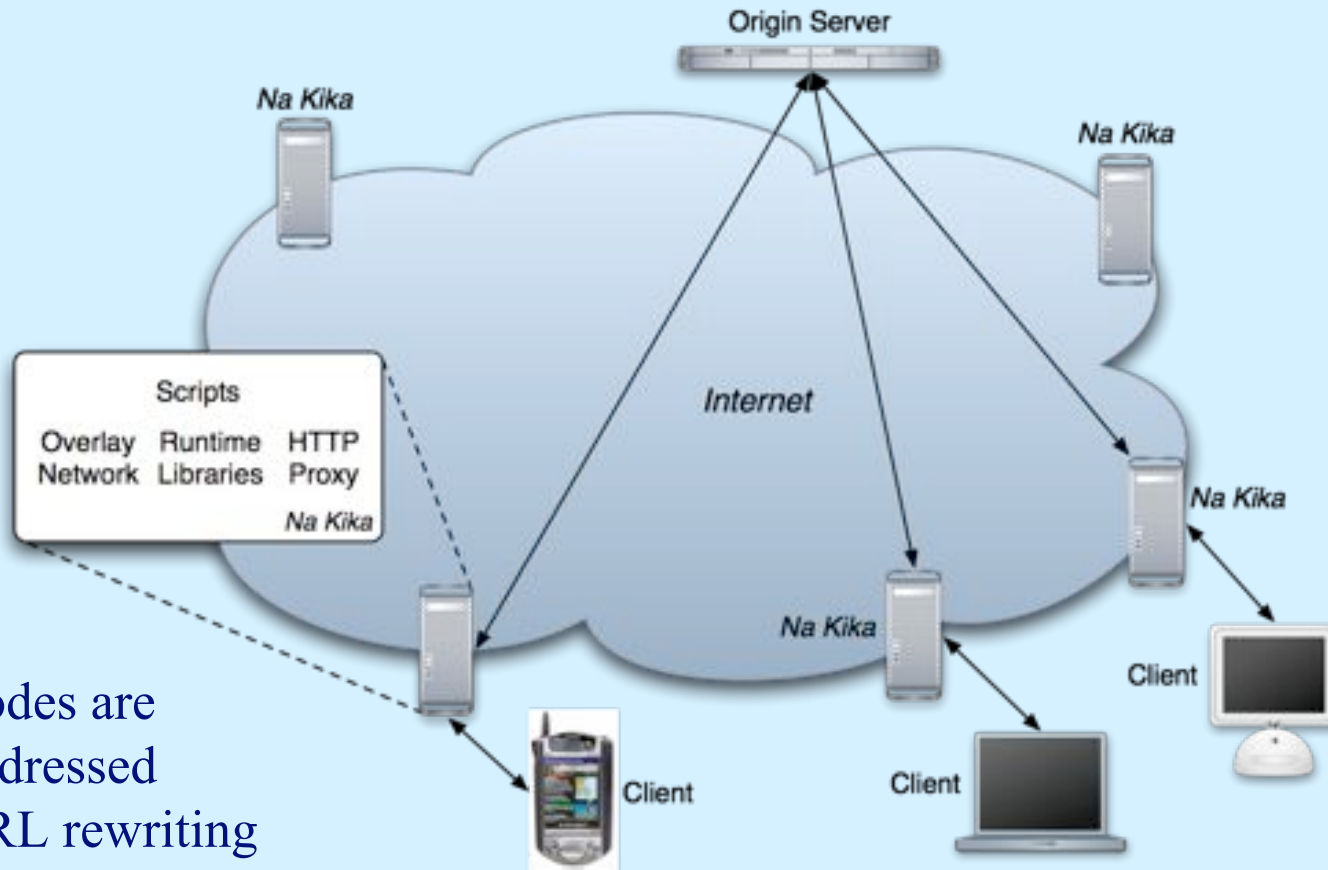
- E.g., NYU's Surgical Interactive Multimedia Modules
 - Cover individual medical conditions
 - Organize content along narrative lines
 - Restore context missing in clinical practice
 - Rich-media enhanced lectures, annotated imaging studies, pathology data, animated and real-life surgical footage
 - Adapt to students' learning needs
 - Already integral part of curriculum at NYU
 - Also deployed at other U.S. and Australian medical schools
- Multimedia heavy, personalized, collaborative
 - Created by community for use by community

We Need a New Delivery Platform

- Scalable → on the internet's edge
 - Computing power and bandwidth to serve local clients
- Easily extensible and composable
- Secure
 - Contain code, enforce extensible security policies
- But existing systems fall short
 - Trusted deployments are limited to amplifying existing sites
 - Akamai, ACDN, ColTrES, Tuxedo, vMatrix, Websphere
 - Active Cache, SDT provide only basic containment
 - No security policy enforcement, no composition model
 - Coral and CoDeeN limited to static content

Enter *Na Kika*

- In a nutshell: Execute scripted event handlers on edge nodes organized into a structured overlay



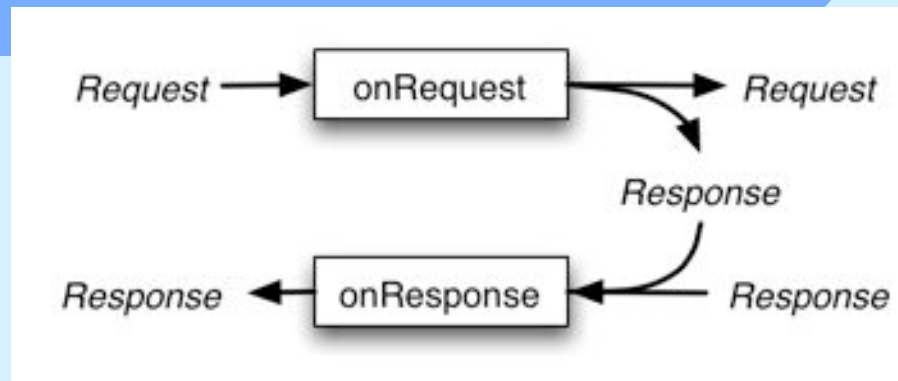
Na Kika nodes are proxies, addressed through URL rewriting

Enter *Na Kika* (cont.)

- Scripting
 - Safe; easier to secure small, interpreted runtime
 - Familiar to web developers; with low cognitive complexity
 - Uniform mechanism for functionality and security policies
- Structured overlay
 - Incrementally scalable and deployable
 - Helps absorb load spikes
 - One cached copy of either content or scripts is enough

Secure Extensibility and Extensible Security

- Event handlers process requests and responses
 - Interpose on message flow clients ↔ cache
 - Selected through predicates on HTTP messages
 - Provides modularity, declarative specification



```
p          = new Policy();
p.url      = [ "med.nyu.edu",
              "medschool.pitt.edu" ];
p.contentType = "text/html";
p.client    = [ "nyu.edu", "pitt.edu" ];
p.onResponse = function() { ... }
p.register();
```

Secure Extensibility and Extensible Security (cont.)

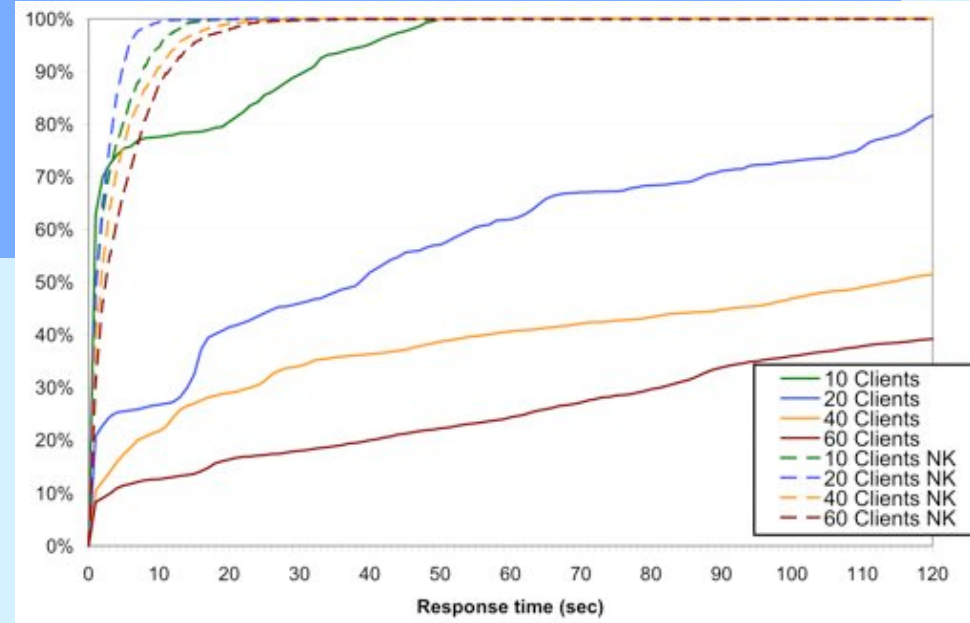
- Pipeline stages compose collections of event handlers
 - First stage provides administrative control
 - Mediates all requests and responses, enforces security policies
 - Fetched from <http://nakika.net/admin.js>, can be locally overridden
 - Second stage implements site-specific processing
 - Fetched relative to domain: <http://domain/policy.js>
 - Additional stages can be scheduled dynamically
 - Execute right after scheduling stage, but before other stages
 - Provide additional functionality, notably content transformation
- Same declarative event handler dispatch for functionality and policies

Resource Management

- Fixed quotas are hard to set
 - Functionality ranges from annotations to H.264 re-encoding
- Predicate-based quotas are more flexible
 - But also amplify the configuration problem
- Congestion-based resource management
 - Pipelines can use as many resources as they need, as long as they do not interfere with other pipelines
 - Manager monitors individual and total consumption
 - On congestion, manager notifies pipelines of congestion
 - If congestion persists, manager terminates pipelines
 - Current heuristics based on adaptation, degree of consumption

The SIMMs in Action

- App developer ported SIMMs to *Na Kika*
 - XSL processing, multi-media moved to edge
 - 2 days: 4 hours for port, rest for debugging
 - 130 lines changed in JSP/servlets, 100 new lines for policy
- Accelerated log replay from 12 PlanetLab nodes
 - 240 simultaneous clients, 90th percentile latency: 4.6s on *Na Kika*, 150s with single server
 - More clients increase latency on *Na Kika* (up to 2.4x) and for a single server (up to 1.5x) while also increasing the failure rate for a single server (up to 44%)



Extensibility at Work

- *Na Kika* Pages
 - Alternative programming model similar to PHP, JSP, ASP
 - Process content with `.nkp` extension or `text/nkp` MIME type
 - Replace code in `<?nkp ... ?>` with result of execution
- Image transcoding
 - Provides building block for other sites
 - Transforms images to JPG, while also scaling them
- Annotated SIMMs
 - Layer electronic post-it notes over SIMMs
 - Schedule original pipeline after new stage
 - Executed as one pipeline on edge nodes

Extensibility at Work (cont.)

- Content blocking
 - First additional stage creates policy based on blacklist
 - Second new stage executes policy, rejecting illegal URLs
- For each: < 100 lines of code, <= 8 hours to write/test
 - HTML-based annotations themselves 180 lines of code

Challenge: Trusted Nodes

- Now: nodes trusted to correctly cache, execute code
 - Limits participating organizations, vulnerable to attacks
 - Content integrity *and* security depend on correctness
- Next: Probabilistic verification model
 - A few trusted nodes manage registration database
 - Fraction of untrusted nodes selected as “secret agents”
 - Do not participate in regular edge-side processing
 - Agents monitor other nodes by repeating all processing
 - Report violators back to trusted nodes
 - Some open issues, to be explored through simulation
 - How to shape monitoring requests, how to punish violators

Challenge: Hard State Replication

- Now: application-specific replication
 - Builds on Gao et al., WWW '03
 - Node-local dbs, persistent message queues, scripted message handlers
 - SPECweb99 with *Na Kika* Pages already runs well
- Next: Two open challenges
 - When to replicate? How to partition?
 - What are the right application trade-offs?
 - What are the right (global) resource management trade-offs?
 - How to best implement?
 - Current implementation uses Java-based JORAM, which is too slow and heavy-weight

<http://www.cs.nyu.edu/rgrimm/nakika/>