
PRESTO: A tool for Configuration Management at Massive Scale*

William Enck, Patrick McDaniel,
Shubho Sen, Panagiotis Sebos,
Sylke Spoerel, Albert Greenberg,
Sanjay Rao, and William Aiello



at&t

*To appear in Usenix Annual 2007

What Does It Mean to Program Networks?

- Insert new mechanism (e.g., new control plane mechanism) in the box, to customize/add value
 - Do we need this? Yes.
 - Otherwise, innovation is too slow, and implementations are ossified into costly, inefficient corners of the design space, optimized for obsolete constraints.
- Adapt the mechanism in the box to customize/add value
 - Do we need this? Yes.
 - Otherwise, we cannot sustain architectural integrity, reliability and performance at scale, and continuously upgrade service and service realization
 - Presto fits here.
 - A powerful tool to close the gap between architectural intent and network reality

Provisioning: Translating Intent to Reality

1. Data integration

- (Ever changing) service orders, architectural updates, policies, databases, spreadsheets, applications, ... all contribute to realizing *THE INTENT of the network architect*
 - Smashing square pegs into round holes
 - De-constructing square pegs, and re-mapping into round holes

2. Logistics

- Getting the right hardware to the right places at the right times
- Getting the right software (config) to the right places at the right times
 - Focus on the config payload: *THE REALITY*
 - Not config messaging (IETF netconf WG)



3. Test and turn up

- Process/workflow, validation/test

What Makes It Dangerous?

- Specifications are incomplete
 - “In that Empire, the Art of Cartography attained such Perfection that the map of a single Province occupied the entirety of a City, and the map of the Empire, the entirety of a Province.” *Borges*
 - Version 2.0: ... the College of Cartographers raised a Map of the Empire which was as extended as the Empire itself and coincided exactly with it
- Operations team members interpret the specs differently
 - Errors in configs
 - Discords between configs and specs
 - Time bombs in the configs: vulnerabilities, failures, ...
- Given the partitioning of design, test and implementation work, open loops may arise between architects, operations, and customers
 - Spec's, databases, checks ... and network get out of sync

Challenges and Difficulties

- Challenges
 - Express config guidelines unambiguously **in the language of network engineers (architects and operations)**
 - **Treat templates as first class objects and mechanize them**
- Difficulties
 - General truth: general data integration can be *very* difficult
 - ... projects delayed, truncated, failed, de-funded
- Presto approach
 - Deliver on the challenges, manage the difficulties
 - Quickness, with a bit of magic (by definition)

A Template Approach

- The PRESTO Approach
 - *Define configuration rules as close to the native configuration language as possible*
 - Create a hybrid language to “wrap” around configuration statements
- This is more than variable substitution
 - Sophisticated data modeling for each router (SQL)
 - Data driven template evaluation (loops, conditionals)
 - Custom language extensions for domain specific transformations
 - Represent some rules in tabular form for easy maintenance

Contextual and Functional Substitution

```
router bgp <BGP_1.CE_ASN>
no synchronization
bgp log-neighbor-changes
network <WAN_IF_1.NETIP:computeIpMask_Netip(<WAN_IF_1.IF_IP>,
255.255.255.252)> mask 255.255.255.252
network <WAN_IF_2.NETIP:computeIpMask_Netip(<WAN_IF_2.IF_IP>,
255.255.255.252)> mask 255.255.255.252
network <ROUTER.LOOPBACKIP> mask 255.255.255.255
```

Context Substitution

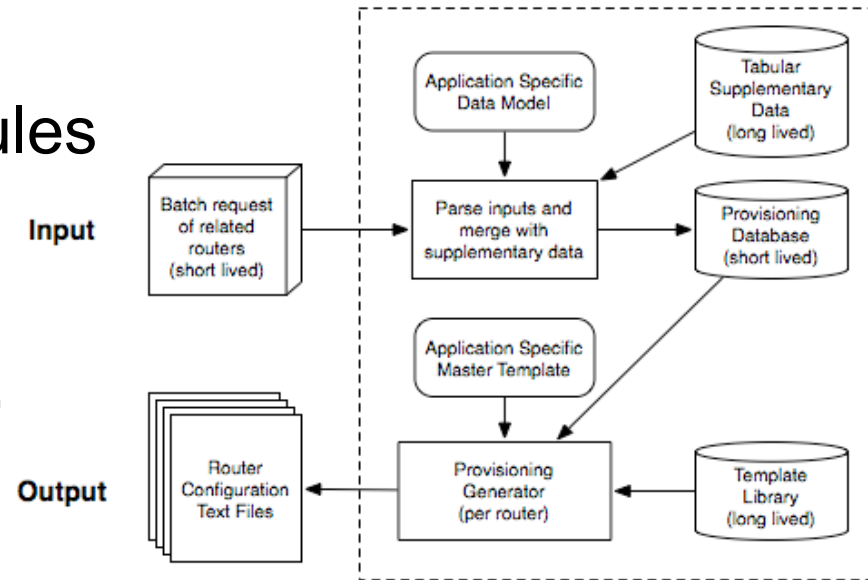
Functional Substitution

```
[ACLIST: SELECT (*) FROM (BASE_ACL) WHERE (COMMENT=DEFAULT)]
access-list <ACLIST.ACL_NUMBER> <ACLIST.ACL_ACTION>
    <ACLIST.ACL_ACTION_DETAIL>
[/ACLIST]
```

Iteration via Context Substitution

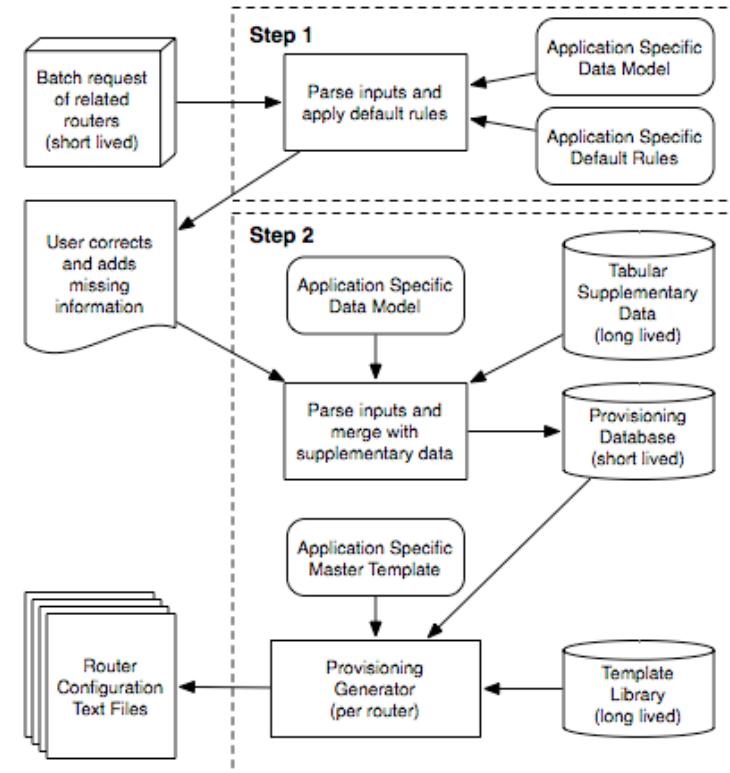
Stateless Pluggable Design

- Many data repositories already exist
 - No need to create one more
- PRESTO stores configuration rules
 - Two types of rules: *tabular* and *logic*
- Tabular rules merge with input
- Domain experts write *configlets*, effectively *active templates*, describing different parts of the configuration
- A master template assembles the configlets depending on the request



Data Integration Adaptors

- Inputs frequently require “human eyes”
 - Multiple inconsistent data sources **Input**
 - Missing information
 - Defaults do not always apply
- Approaching full automation
 - Best effort data reconnaissance
 - Apply default rules to resolve inconsistencies and fill in missing information **Output**
 - Require only one manual confirmation



Scalable and Extensible

- New interface types
 - Augment interface card table
- New feature
 - Add a new active template for associated configlet to the library
- New wrinkle on an existing feature
 - Add a new active template, reusing portions common to already supported features
- New router model:
 - Reuse router model-independent parts of the config (substantial overlap).
 - Focus new effort only on router model-dependant pieces and for features not yet supported

It Works

- Existing standards documents mapped nicely to configlets
 - Examples made the basis of configuration statements
- Two-step process was necessary and unavoidable
 - Sometimes information was simply unavailable at configuration time
- Diverse initial router configurations quickly created once information was available
 - Configuration time dependent on retrieval of information
- Near full automation achieved
- In use at AT&T

Questions?

William Enck

Systems and Internet Infrastructure Security (SIIS) Laboratory

The Pennsylvania State University

<http://www.cse.psu.edu/~enck>

enck@cse.psu.edu