Behind all theories: Effective network configuration for real cases

INET'01, June 6, 2001
Stockholm

Martin Bokämper

Maximilian Riegel

Overview

- The current state – theory and real world
- Configuration data requirements
- A particular solution: Zope
- Demonstration - Evolve a ‘small ISP’
  - Step#1: From config-files to Zope
  - Step#2: Static IP addresses for users
  - Step#3: Subnets for users
- Conclusion

State of the art

Network Management systems

- Detailed model of the network and network elements
- Network managers vs. element managers
- Both contain information models of great details
- Often used only for visualizing and monitoring networks.
- Configuration of network elements like routers and associated services like DNS or RADIUS is normally not done by NMSs.

Real networking

Real networks

- tend to grow fast with
  - frequent reorganization of the network and
  - exchange of most of the network equipment.

The operators of such networks are

- heavily loaded by the continuous reconfiguration of the growing network,
- not convinced to make long term planning for a well structured management system.

Configuration of real networks

Often done

- with script based systems,
- using well known tools like RCS, PERL, LDAP, ssh and many others.

Configuration

- evolves with the network,
- tends to grow in a unstructured manner.

Our perception

A large mismatch seems to exist between

- the highly abstract world of modeling of network management systems

and

- the actual management of configuration data in real-world networks.

Configuration data: Permissions and access control

- Access control on ‘object’ level (MIB-tree, config-file) vs. access control on ‘role’ level (can add new users, but not freely mess up user table).
- Complexity of permission management vs. flexibility.
- Ability to manage complete ‘subtrees’ as one object.

Configuration data: Consistency and ‘redundancy’

- Multiple (almost) identical servers need to be configured consistently
- The same basic information goes into the configuration of completely different systems.
- ‘Exceptions’ need to be possible … they can usually not be modeled up front.

Configuration data:
Revision control and history

- Who changed something, when and why
- Make multiple changes, test them and ‘release’ them as one transaction
- Be able to ‘go back’ to older configurations

Nothing new here - everything is known from the world of source code control.

Configuration data: Other issues

Comments and additional data

- Embedding notes or comments in configuration data is **sooo** useful.

And finally...

“Keep simple things simple”

Configuration data:
Modeling and representation

Detailed models: SNMP MIBs, DEN schemata

Alternative: Text-based config files, e.g. *sendmail*, *bind*, Cisco-routers

Issue	**Text**	**ASN.1 Model**
Revision Control	Lots of generic tools available (RCS, CVS, SCCS …)	Needs to be supported by model (examples?) or by special tools (examples?)
Embedding external data	Some generic tools available (preprocessors), lots of specialized tools/scripts.	
How to change	Lots of generic tools – editors, sed, awk, perl	Some generic tools
Add to or change model	Usually no problem, only few specialized tools need to be adapted	Extension of model usually difficult – more tools need to be adapted
Storing Comments	No problem	Usually not possible.

Configuration data: Modeling and representation

- Complexity and ‘variation’ of configuration data tends to increase with network layer.
- Conclusions:
  - There is not ‘the right’ solution,
    both approaches have their uses.
  - A solution ‘in the middle’ might be desirable
  - Creation of specialized tools needs to be simple

- Approach: **‘Model on demand’**

All this is not new - solutions exist.

- Similar problems exist in the management of web content.
- There are freely available tools solving these problems for web content,
  e.g. ***Zope*** (http://www.zope.org/)

- We will show, how Zope can be applied to solve the problems of the management of common tasks and services in an operational network.

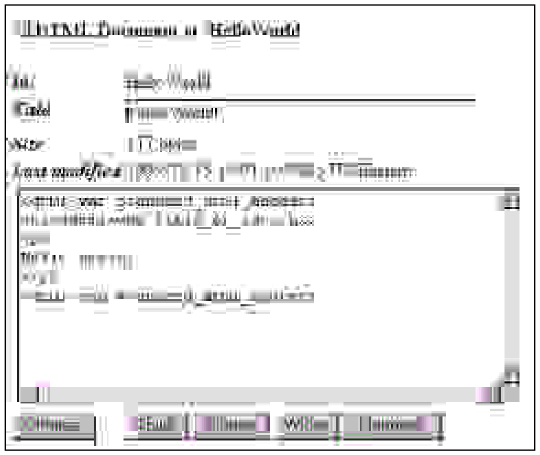
What is Zope?

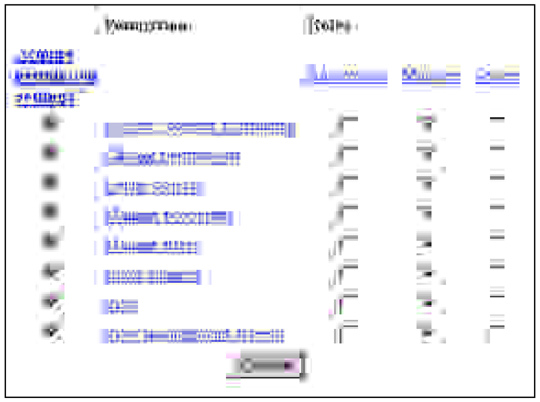
From the Zope web site:
http://www.zope.org/

The Leading Open Source Application Server

Zope leads the Open Source application server market because it is the most flexible solution in existence for creating and maintaining large web presences.
You can create a maintainable infrastructure that will grow with your needs using Zope's standards-based tool set. Zope gives you the power to create a site that uniquely addresses your business needs. Better yet, it's free!

What Zope is:



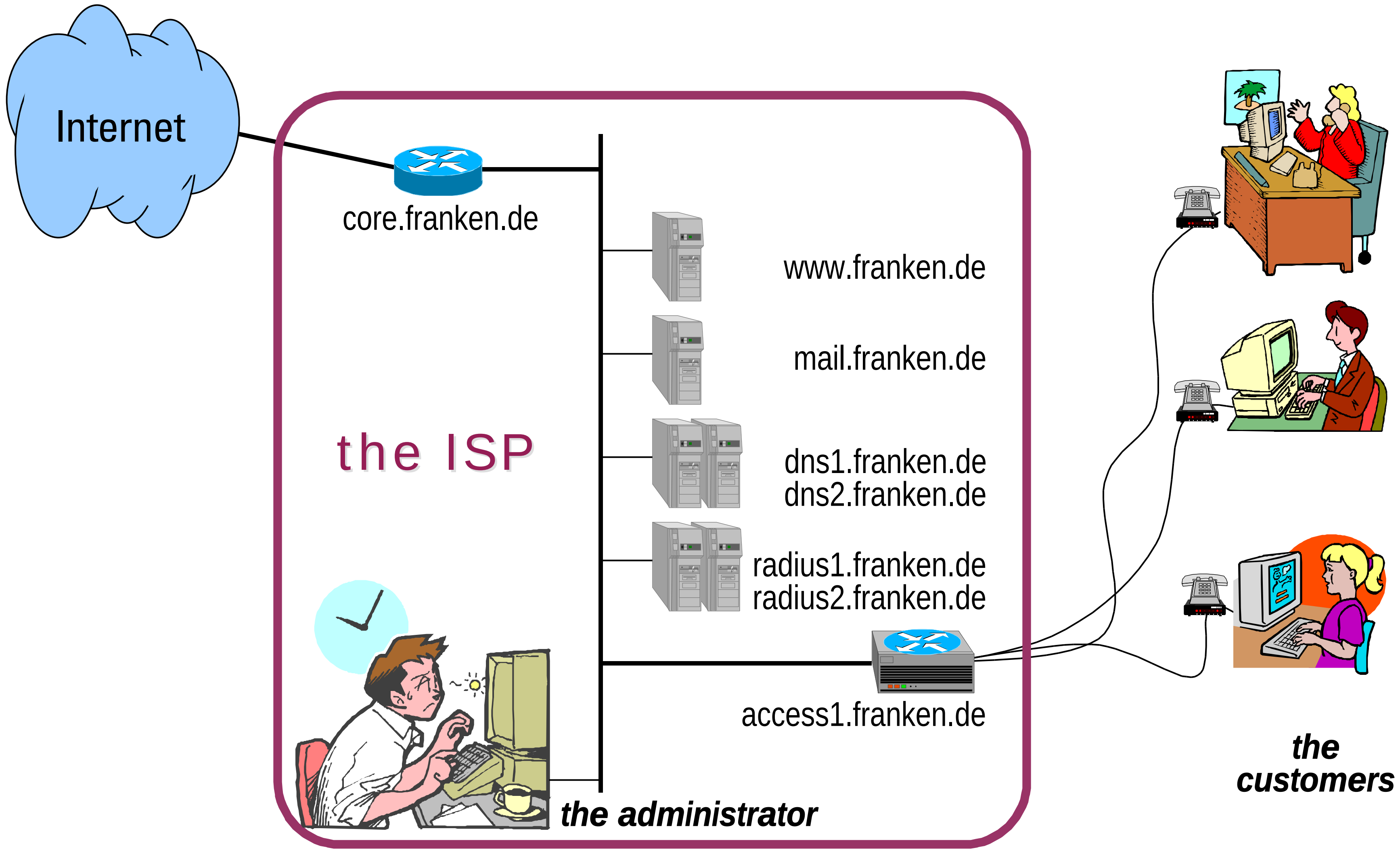


- OO and script development tool.
- Integrated transactional object database
- Powerful delegation and security model
- Web-based user interface
- Integrates Servers for multiple protocols: FTP, HTTP
- Extensible Interfaces to other systems, e.g. LDAP, SQL
- Open Source
- The ´default application´ is dynamic web content and application services

Demonstration

- Szenario:
  - ISP with
    - modem dial-in,
    - single connection to the internet,
    - mail-server, www-server,
    - redundant DNS servers,
    - redundant RADIUS servers
  - Growing user base, growing user demands
- Step #1: From config-files to Zope
- Step #2: Static IP addresses for users
- Step #3: Subnets for users

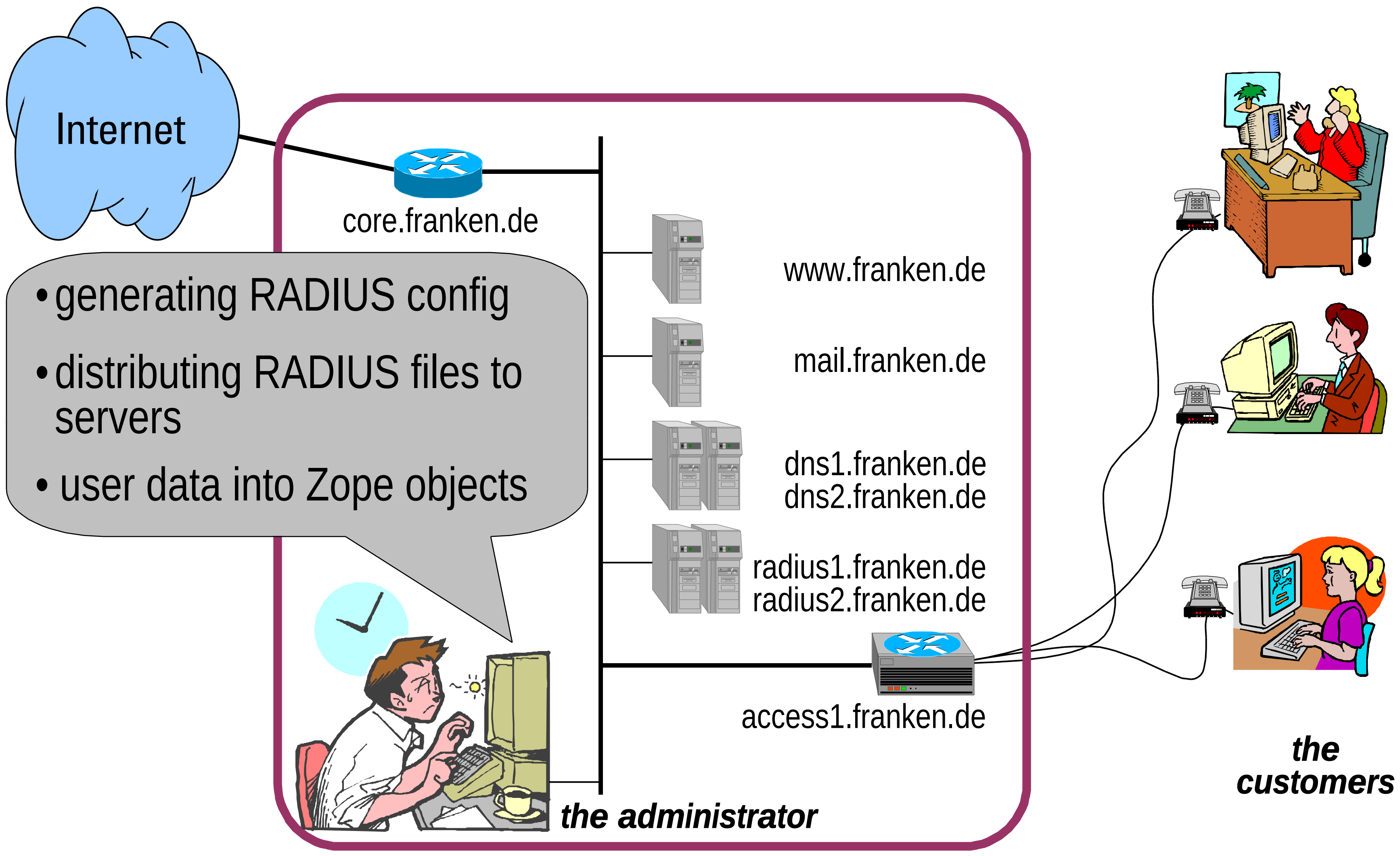
The demonstration szenario



Step #1: From config-files to Zope

- RADIUS configuration is generated from Zope-controlled templates
- Distribution of the files to the servers (http)
- Users with dynamic addresses will be extracted from plain user file and will be modeled in ZOPE.

Step #1: From config-files to Zope

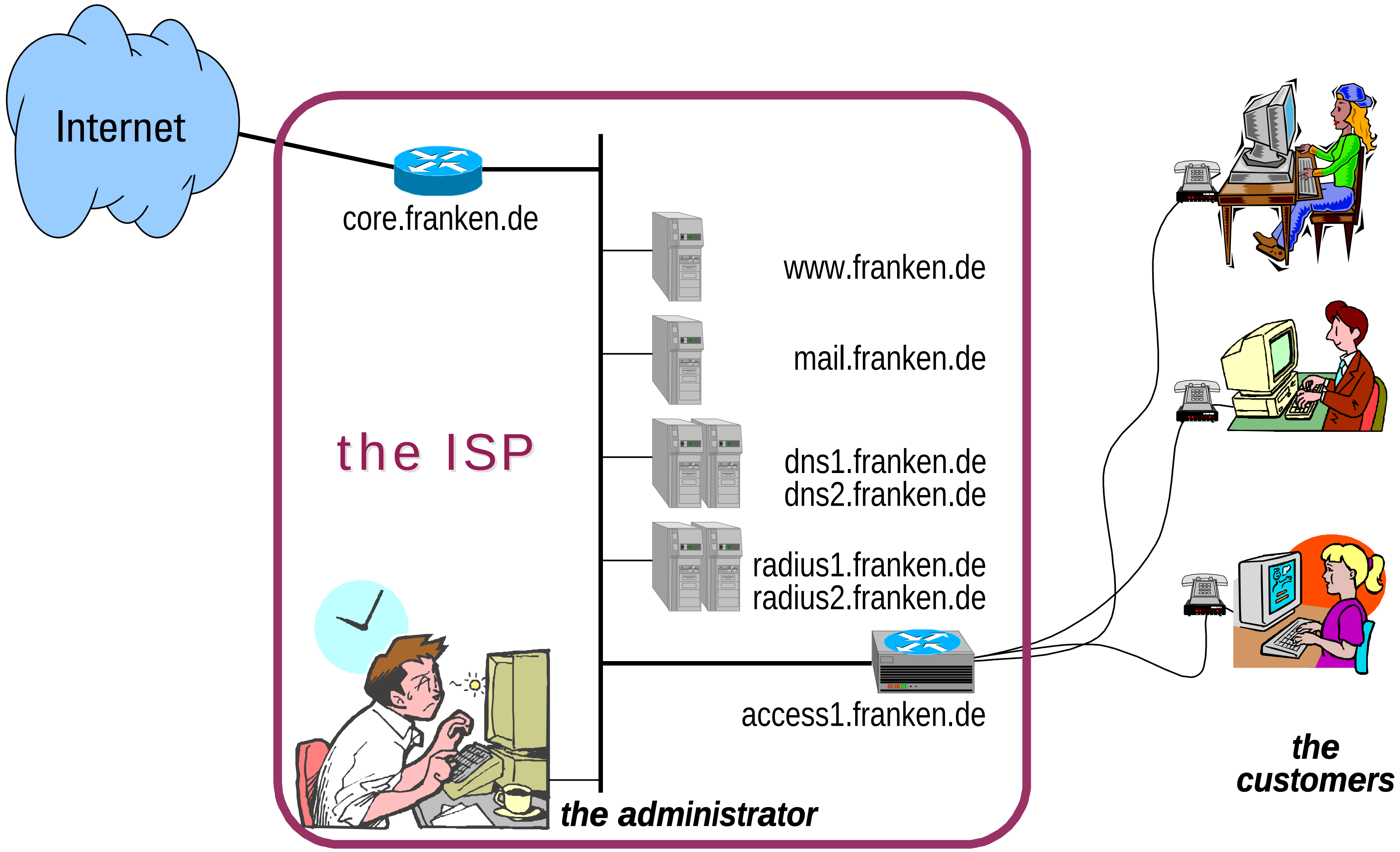


- generating RADIUS config
- distributing RADIUS files to servers
- user data into Zope objects

Acquisition in Zope

- All persistent Zope objects are contained by some other object in the database – usually a ‘Folder’
- When the class & superclasses do not provide a specific attribute for an object, it can be ‘acquired’ from it’s containing object – recursively.
- The list of ‘containing objects’ (context) can be temporarily manipulated
- Acquisition is used heavily in Zope – makes templates visible while having the option to override.

#2: Static IP addresses for users

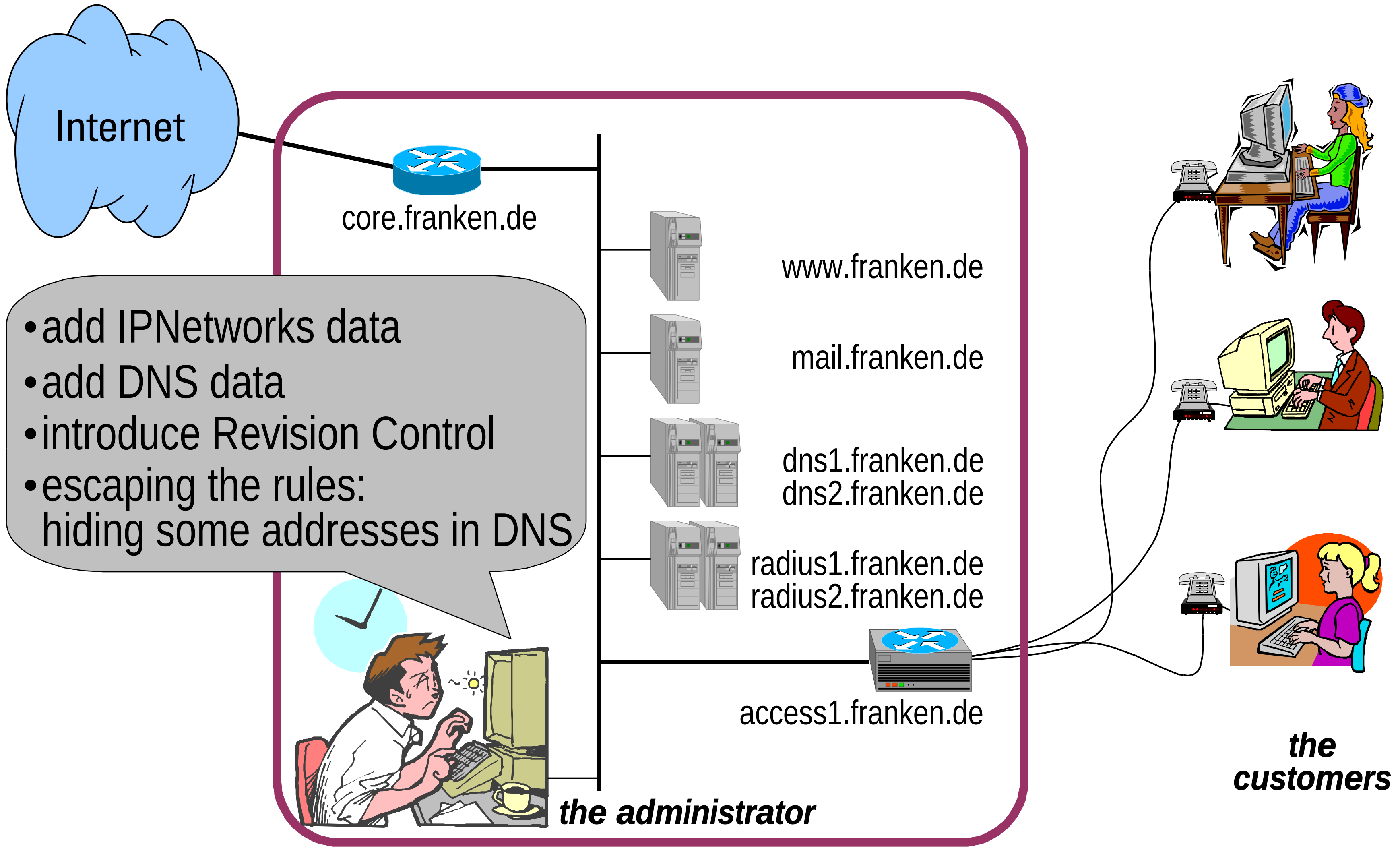


mbokaemper@unispherenetworks.com, maximilian.riegel@icn.siemens.de

#2: Static IP addresses for users

- Enable static IP address assignment for users
  - add IPNetworks data and DNS data
  - introduce Revision Control
  - escaping the rules:
    hiding some IP addresses in DNS

#2: Static IP addresses for users



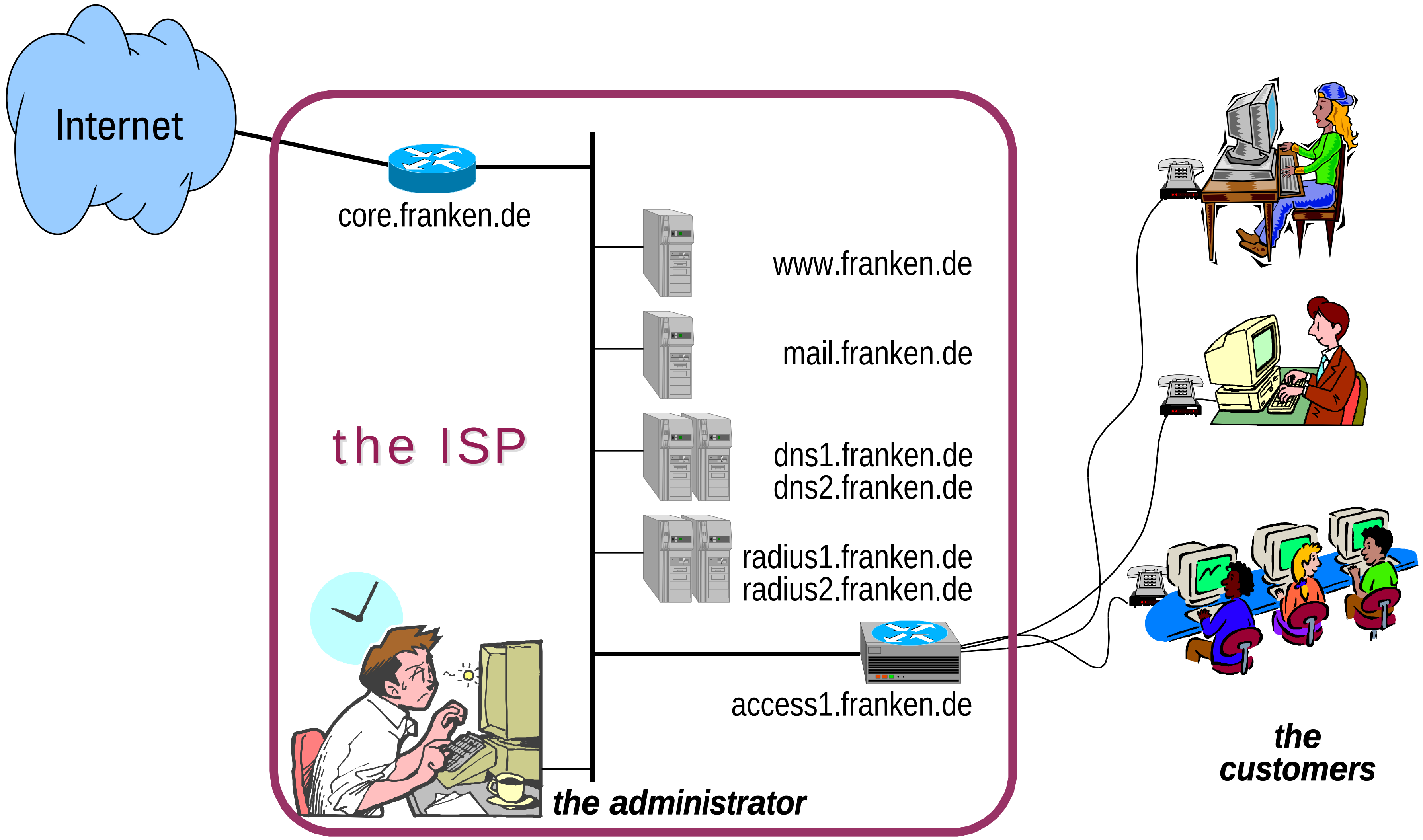
- add IPNetworks data
- add DNS data
- introduce Revision Control
- escaping the rules:
  hiding some addresses in DNS

Revision Control in Zope

- All old versions of an object are kept in the database until cleanup is explicitly requested.
- User-defined classes inherit this behavior.
- “Version” objects provide branching and merging

- Changed objects are locked – no concurrent work is possible.

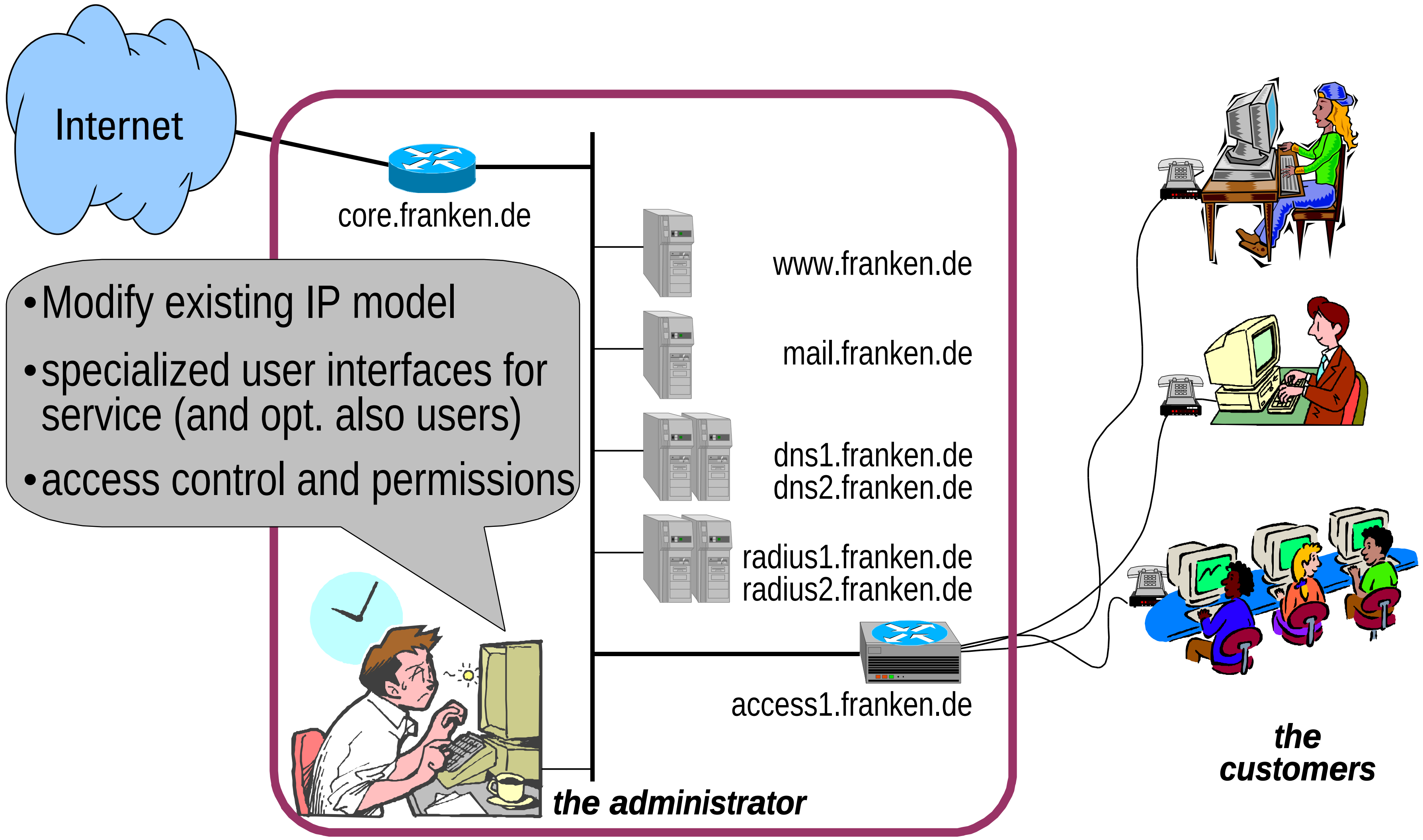
Step #3: Multiple IP addresses for users



Step #3: Multiple IP addresses for users

- Allow assignment of additional addresses to some users
  - extend the object model for address management
  - specialized user interfaces for service people (and optionally end users)
  - access control and permissions.

Step #3: Multiple IP addresses for users



- Modify existing IP model
- specialized user interfaces for service (and opt. also users)
- access control and permissions

Access Control in Zope

- Acquisition works on access information!
- Many specific permissions can be defined
- Permissions can be bundled to ‘roles’ and assigned to users.
- (Executable) objects can have their own rights that may exceed the permissions of the calling user. (Similar to ‘setuid flag’ in unix.)
- User definitions work by acquisition – a UserFolder is only valid in the subtree it is part of. This simplifies delegation of responsibility A LOT.

Conclusion

- Some simple cases shown – most concepts covered
  - Step#1: From config-files to Zope
  - Step#2: Static IP addresses for users
  - Step#3: Multiple IP addresses for users
- Possible next steps, e.g.:
  - Knowledge of IP Networks has other applications, e.g.:
    - Generation of filters in routers
    - Mail server rules to restrict relay to local addresses
  - User self definable policy rules and services
- Not covered in Zope: Dependencies (‚make‘)

The end

- Thank you for your attention.
- Questions?