

Input Methods

Gnome.Asia Submit

Speakers (In vocabulary order)

- James Su: lead of [SCIM](#) community and project [IMBus](#).
- Peng Huang: author of [scim-python](#) and [iBus](#).
- Peng Wu: author of [Novel Pinyin](#)
- Yong Sun: [IIIMF](#) developer, and maintainer of [SunPinyin](#)

Input Method Frameworks

Existing Major Input Method Frameworks:

1. [SCIM](#) (Smart Common Input Methods), active community, widely adopted by many Linux/Gnu distributions, and currently available on Solaris/OpenSolaris.
 - [IIIMF](#) (Internet/Intranet Input Method Framework), the default IM framework on Solaris/OpenSolaris, designed for serving multiple users.
 - [OpenVanilla](#), a well-known IM framework on Mac OS.

... ..

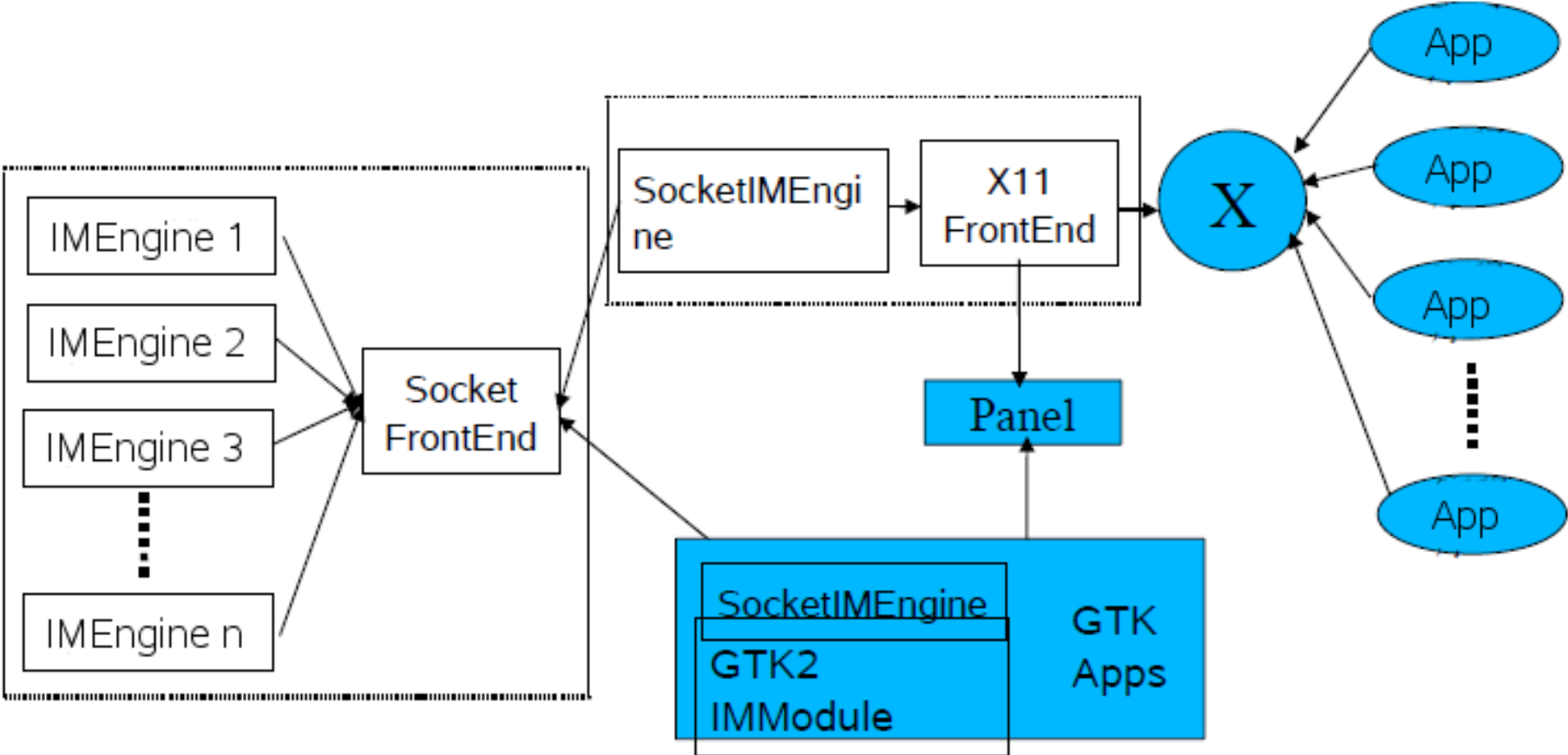
SCIM Overview

- Object-Oriented Design in Modern C++
 - Heavily employ on STL library
- Highly Modularized Components
 - Most components are modularized.
 - Customizable by combining different components.
- Easy Development for Input Engine
 - Only need to implement two subclass.
- Unicode-based
 - Independent with system locale.
- A lots of helper functions.
 - Encoding convert, network communication, file r/w, string manipulation, etc.

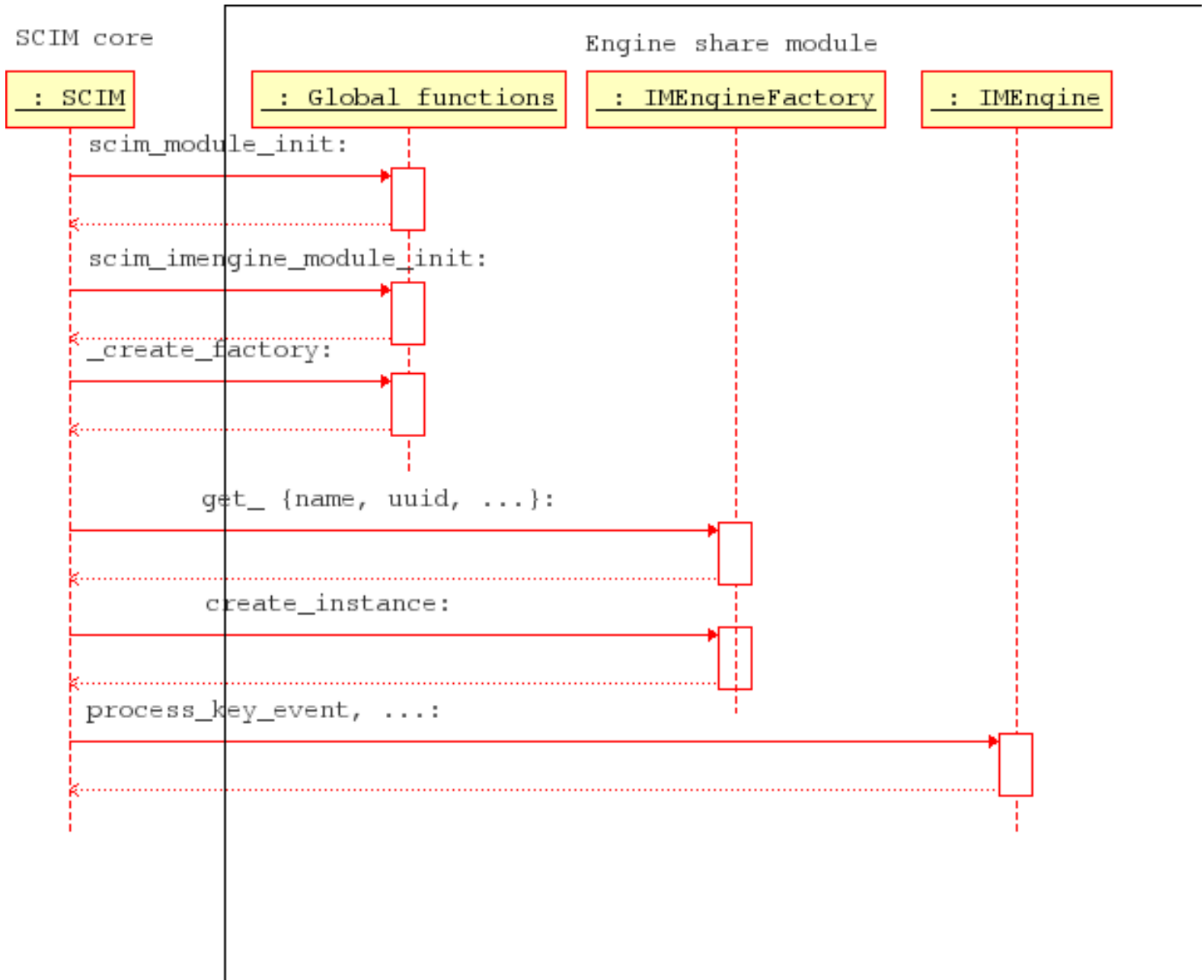
Components in SCIM

- IMEngine input method services(engine)
 - IMEngineFactory
 - IMEngineInstance
- FrontEnd
- Config
- Panel
- BackEnd
- Socket
 - SocketServer
 - SocketClient
 - SocketTransaction
- Helper classes and funtions. (Signal-Slot, LookupTable, IConv, Module etc.)

SCIM Architecture



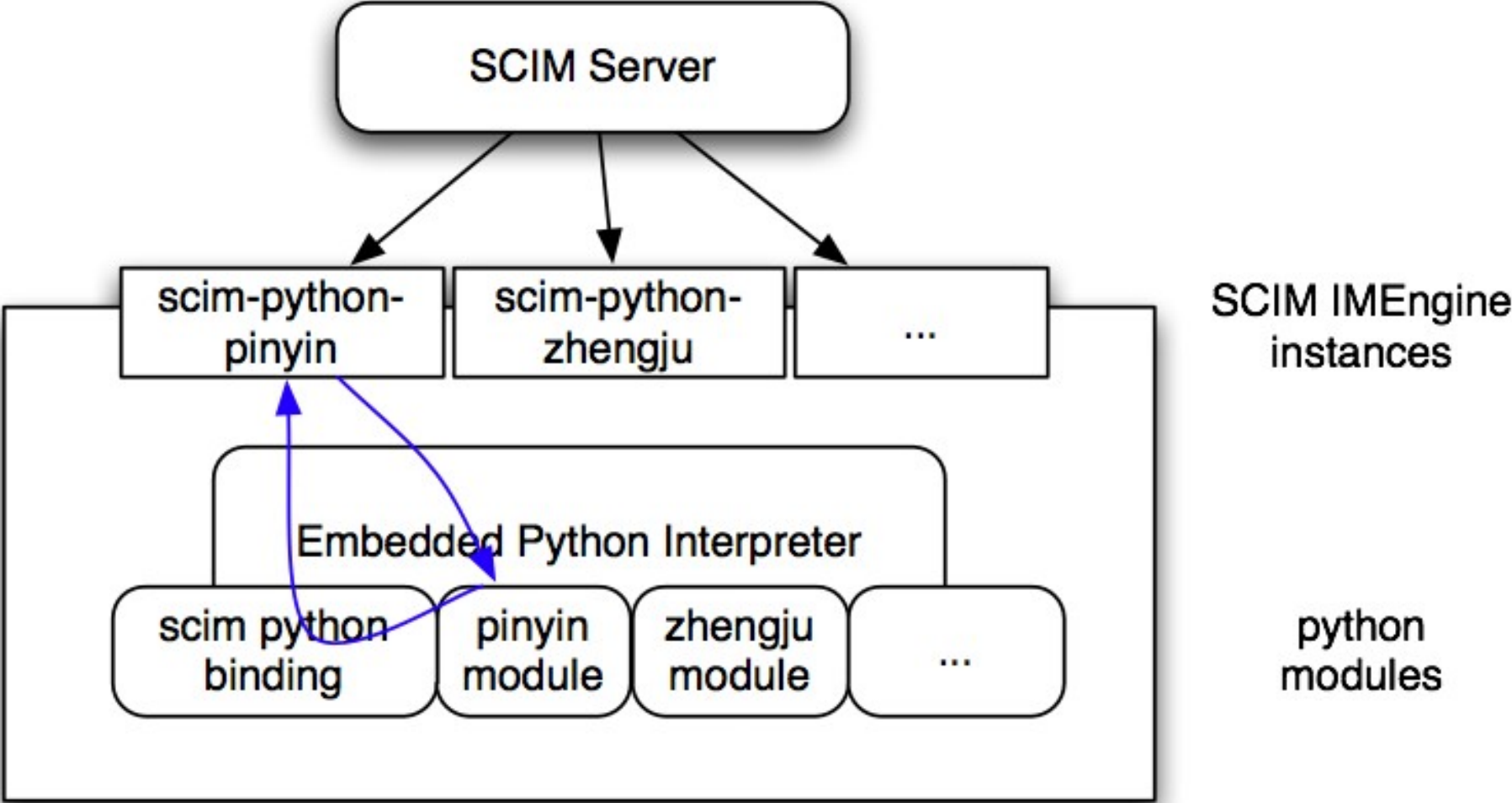
SCIM im module initialization



scim-python

- SCIM-python is a python binding for SCIM platform.
- Developers may use python language to develop input engines.
- SCIM-Python's partitions:
 - Python binding – by Huang Peng (C++, 9000 lines)
 - Python PinYin engine – by Huang Peng (Python, 5000 lines)
 - English Writer engine – by Huang Peng (Python, 300 lines)
 - ZhengJu engine – by [Yu Fan](#) (Python, 2000 lines)
 - XingMa engine – by [Acevery](#) (Python, 2500 lines)
- Included in Official Fedora 8, Fedora 9 and later
- Has been packaged for Debian, Ubuntu, Gentoo, Archlinux, OpenSUSE, and etc

scim-python Architecture



Python PinYin and ZhengJu

- Python PinYin:
 - Author: Huang Peng
 - It has about 1000000 phrases.
 - It likes most popular Chinese PinYin input method – Sogou & Google PinYin on Windows Platform.
 - It supports input Chinese Sentences
- ZhengJu Pinyin:
 - Author: Yu Fan
 - It supports input Chinese Sentences.
 - It is like Microsoft PinYin on Windows.

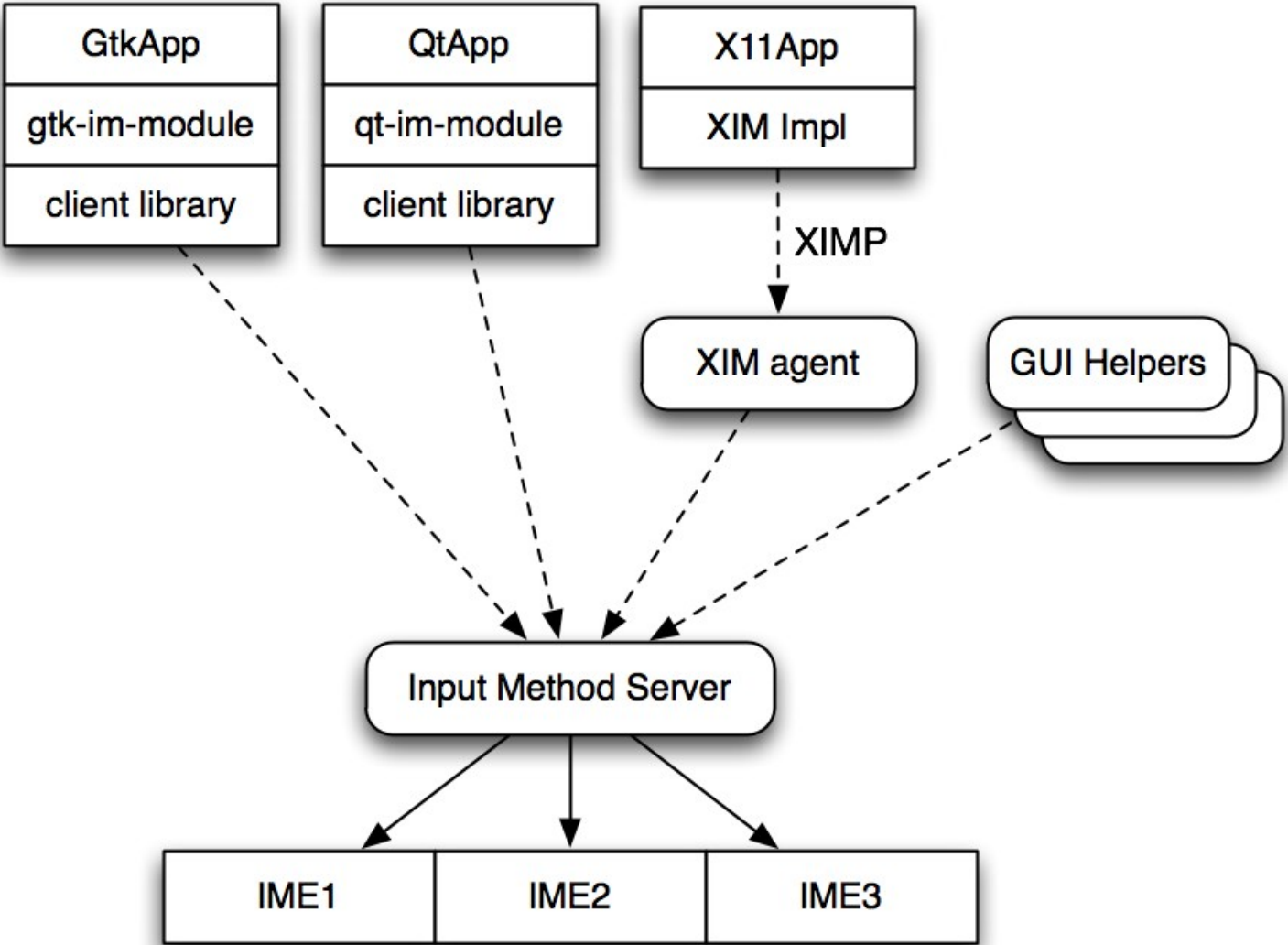
IIIMF Overview

- Designed by Hideki Hiura who designed XIM
- Designed for serving multiple users in a network environment. (Though it could be launched as single user mode)
- It's a protocol based architecture
- Multi-threaded Server (IIIMSF), written in C++
- more complexities (MT-safe ...)
- LEIF (Language Engine InterFace) in C
- lacks of an unified GUI component interface
- lacks of standardized configuration/storage facilities
- provides a keyboard layout simulation layer
- works more closely with Solaris features, such as kbd(1)

IIIMF Client Components

- IIIMXCF - IIIM X Window System Client Framework
 - iim-xbe: the XIM backend
 - xiiimp.so: an implementation of XIM interface
- IIIMJCF - IIIM Java2 Client Framework
- IIIMECF - IIIM Emacs Client Framework
- IIIMGCF - IIIM GTK+ client Framework (aka iimf-gtk immodule)
- IIIMWCF - IIIM Windows Client Framework
- IIIMQCF - IIIM Qt Client Framework
- libiiimcf - IIIM generic C client Framework library
- libiiimp - IIIM protocol

Server-Centric Architecture

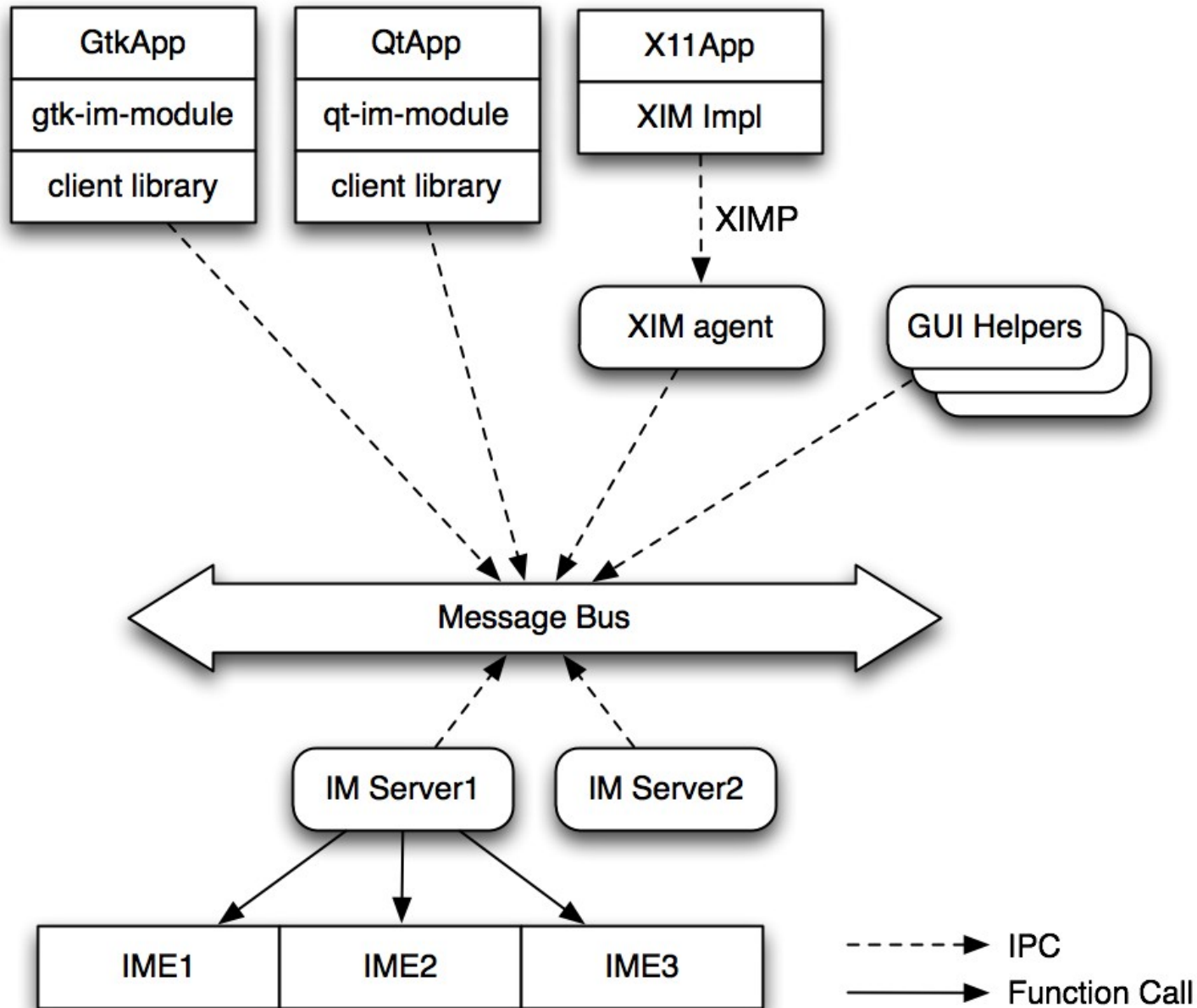


---> IPC
—> Function Call

Disadvantages of Server-Centric Architecture

- There's usually only 1 server per desktop session, which loads all engines as shared objects. So one buggy engine would crash the entire server.
- Engines are developed with either C or C++, it's difficult to add other language bindings.
- In current architecture, a binding basically means plugging an interpreter into server's process space.
- Engines are not able to be loaded or unloaded on the fly.

Bus-Centric Architecture



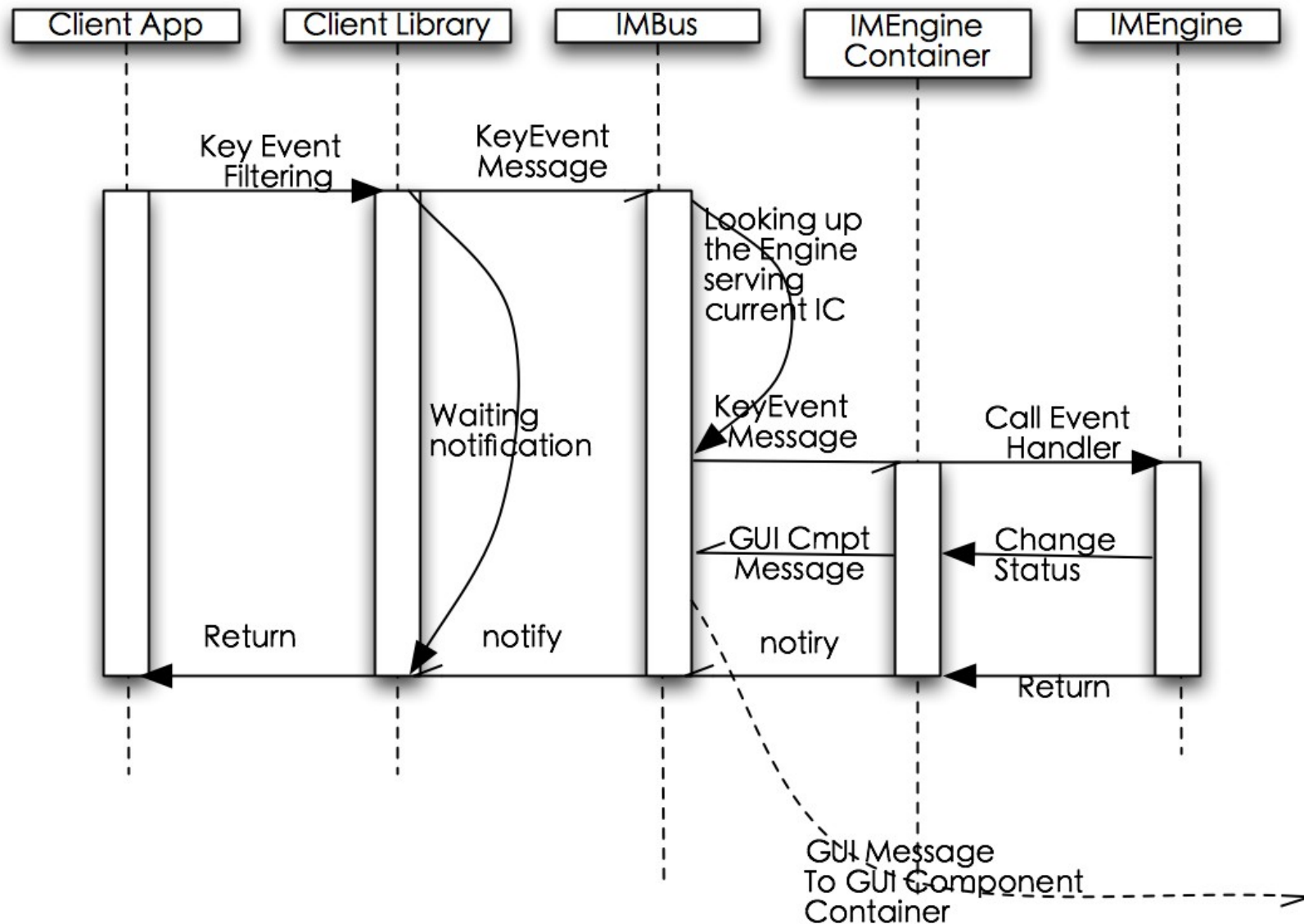
The NG Input Method Frameworks

- [IMBus](#)
 - Initiated by [NEAOSS](#) (North-Eastern Asia OpenSource Software)
 - To unify the IM SPI on various platforms
 - [IM Engine SPI specification \(2nd draft\)](#)
- [iBus](#)
 - Proposed by Peng Huang and Yong Sun
 - Inspired by IMBus and [scim-python](#)
 - Implemented with python
 - Use dbus for IPC, and gobject as object system
- Facing new Opportunities/Challenges
 - Multiple Pointers (MPX)

IMBus Overview

- Inspired from the design of D-Bus
- A Bus-Centric Architecture
 - imbus-daemon, libimbus:
 - IMEngine and its Containers:
 - GUI Component and its Containers:
 - Configuration/Storage Component:
- A Brand-new Event Model
 - Event Producer, Consumer, and Observer
 - Service declaration and registering
- Written in pure C to avoid compatibility issues
- Fully object oriented and Extensible
- Platform and GUI independent
- Compatible with XKB

An Example of Event Flow



IBus

Intelligent Pinyin Input Methods

1. [SunPinyin](#)
2. [NovelPinyin](#)
3. [ZhengJu](#) in scim-python
4. [Davepy](#)

... ..

SunPinyin Overview

- Built on a back-off n-gram language model (3-gram)
- Supports multiple input schemes (Instant & Classic)
- Supports 2-gram history cache, or user's customized model
- Ported to various OS/platforms:
 - IIMF, maintained by Yong Sun
 - SCIM, by Kov Chai
 - BeOS, by Anthony Lee (author of BeCJK)
 - MacOS, by jjgod (Jiang Jiang) and Yong Sun
- Hosted in input-method project on OS.o, dual-licensed with CDDL+LGPLv2.1
- Applying the acceptance of Debian, contributed packaging on Ubuntu, OpenSuse and Mandriva ...

Statistical Language Model

To calculate the probability of sentence $S=(W_1,W_2,W_3\dots W_n)$

$$\begin{aligned} P(S) &= P(W_1) \cdot P(W_2 | W_1) \cdot P(W_3 | W_1, W_2) \cdot P(W_4 | W_1, W_2, W_3) \dots P(W_n | W_1, W_2, \dots, W_{n-1}) \\ &= P(W_1) \cdot \prod_{i=2}^n P(W_i | W_1, W_2, \dots, W_{i-1}) \end{aligned}$$

$$P(S) = P(W_1) \prod_{i=2}^n P(W_i | W_1, W_2, \dots, W_{i-1})$$

In reality, due to the data sparseness, it's impossible to calculate the probability in this way. A practical method is, to assume the $P(W_i | W_1, W_2, \dots, W_{i-1})$ only depends on the previous N words, i.e., $W_{i-N+1}, W_{i-N+2}, \dots, W_{i-1}$.

Particularly, we have unigram ($N=0$, context-free grammar), bigram ($N=1$), trigram ($N=2$), and fourgram ($N=3$). The most commonly used is trigram.

General Terms

- types: the size of vocabulary (or dictionary)
- tokens: the size of corpus
- vector/solution space:
 - for a N-gram model, $V = \text{types}^N$
- data sparseness:
 - tokens $\ll V$, tokens is much smaller than V
- Maximum likelihood: (C, occurrence count)
 - $P_{ML}(W_i|W_{i-2}, W_{i-1}) = C(W_{i-2}, W_{i-1}, W_i) / C(W_{i-2}, W_{i-1})$

Smoothing

- Due to the data sparseness, many possible word sequences may not be collected in training corpus, if $P_{ML}(W_i|h)$ is 0, then the probability of entire sentence becomes 0.
 - E.g., if we only see "Tom read a book" in corpus, while don't see anything like "Yong read xxx", then $P_{ML}(\text{Yong read a book}) = 0$.
- Smoothing Methodologies:
 - Simple Smoothing
 - Add-one, Add-delta ...
 - Discounting Smoothing
 - Witten-Bell, Good-Turing, Liner-Discounting ...
 - Compound Smoothing
 - Back-off Smoothing
 - Interpolated Smoothing

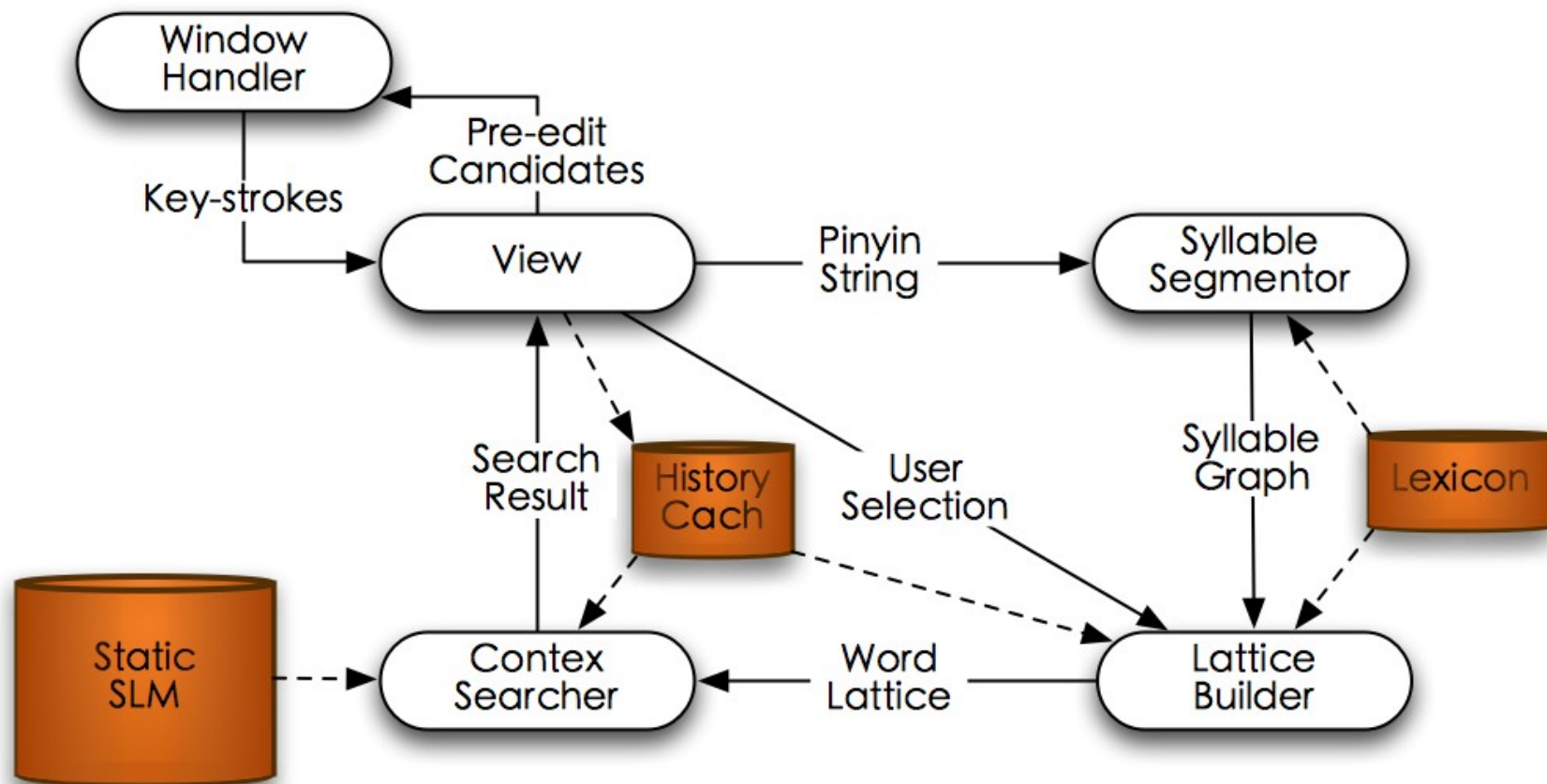
Back-off smoothing

A general back-off model could be expressed as following:

$$P_s(W_i | h) = P_d(W_i | h) \quad \text{-- } C(h, W_i) > \theta$$
$$\text{bow}(h) \cdot P_s(W_i | h') \quad \text{-- } C(h, W_i) \leq \theta$$

- h' is the history h truncated by the first word. For trigram (A, B, W_i) , h is (A, B) , so h' is B .
- $P_d(W_i | h) < P_{ML}(W_i | h)$
- $\text{bow}(h)$ is the back-off weight, for a given h , $\text{bow}(h)$ is a **const number**, and could be determined by:
 - $\sum_i (P_s(W_i | h)) = P_s(W_1 | h) + P_s(W_2 | h) + \dots + P_s(W_n | h) = 1$
- this is a recursive expression, if the occurrence number of (h', W_i) is 0, then back-off continues, it may back-off to W_i , even to a average distribution (if W_i is not seen).
- if h is not seen in training stage, $P(W_i | h) = P(W_i | h')$

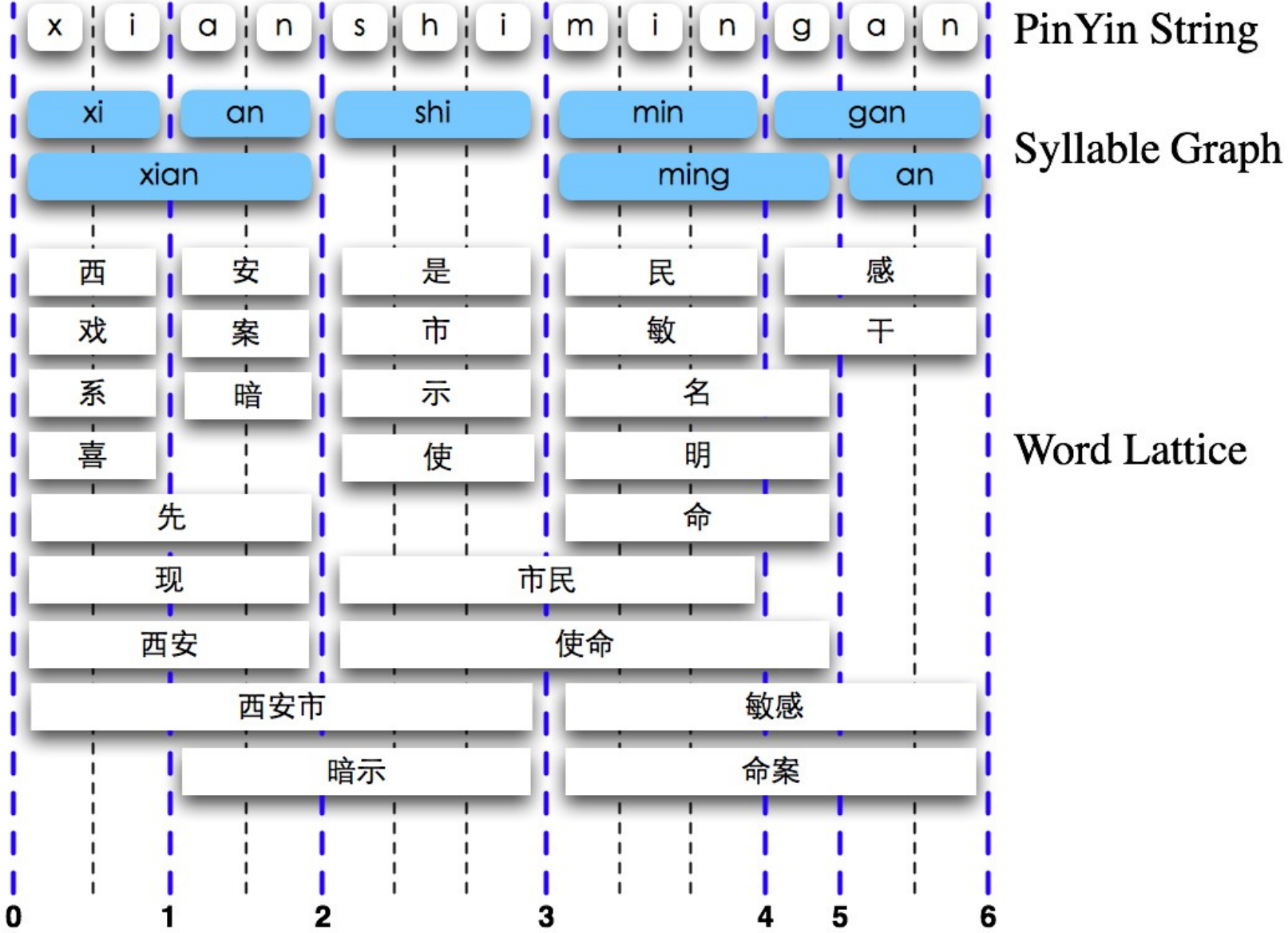
Conceptual Model of SunPinyin



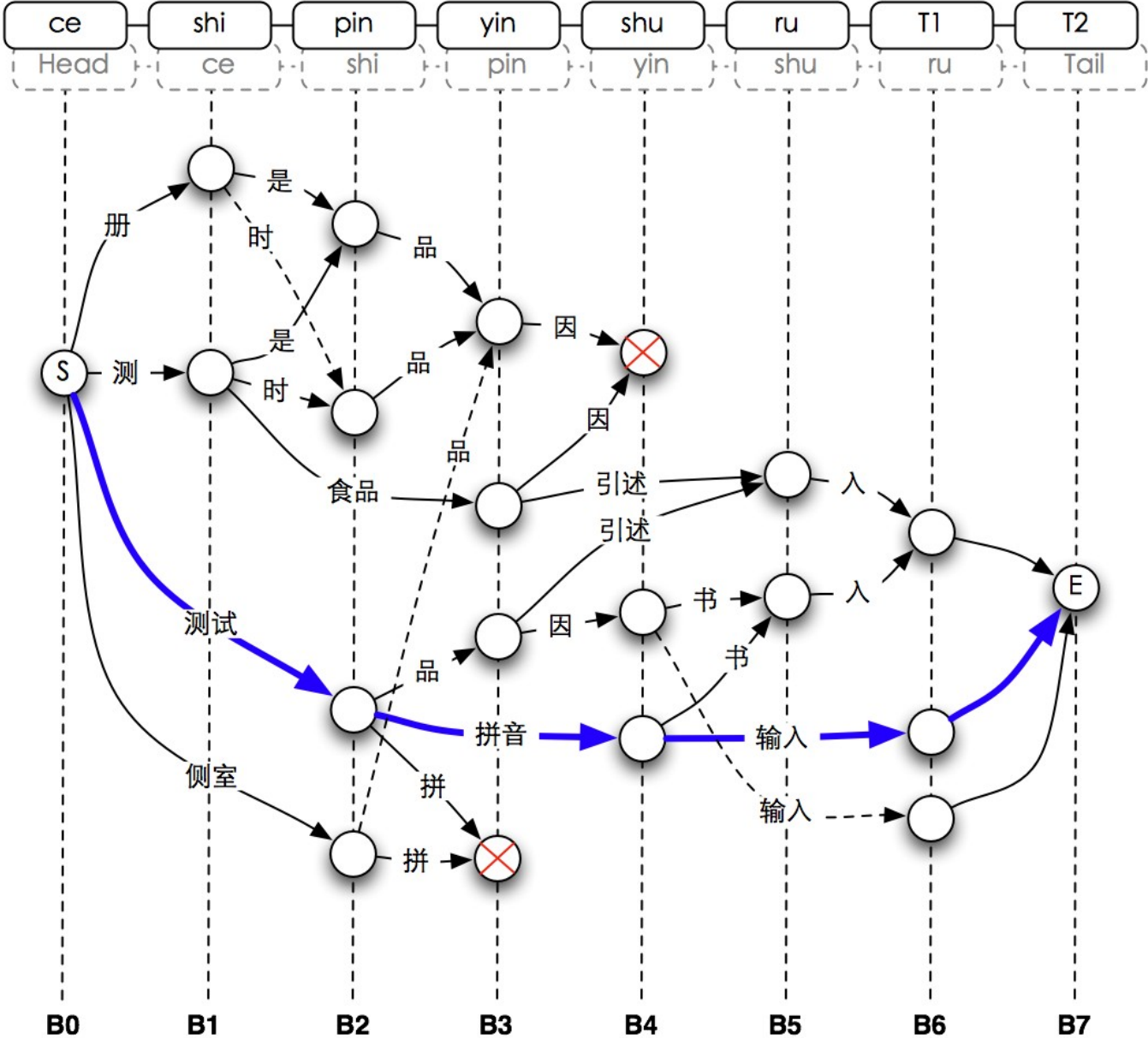
$$P(z|x, y) = \lambda P_{smooth}(z|x, y) + (1 - \lambda) P_{cache}(z|y)$$

$$P_{cache}(z|y) = \lambda_1 \frac{C(y, z)}{C(y)} + (1 - \lambda_1) \frac{C(z)}{history\ size}$$

pinyin-string -> syllable-graph -> word lattice



Searching the Lattice



References of SunPinyin

- <http://src.opensolaris.org/source/xref/.../sunpinyin/TODO>
- <http://www.opensolaris.org/os/project/input-method>
- SunPinyin Code Tour Documents
- <http://blogs.sun.com/yongsun/tags/sunpinyin>

Novel Pinyin Overview

- Based on an interpolation smoothing of Hidden Markov Model. (bi-gram)
- Complete support for ShuangPin schemes, fuzzy pinyin and incomplete pinyin, inherited from scim-pinyin.
- Improved user input self-learning.
- Appears on some distro, SUSE, Mandriva, aur etc.

Mathematic Model

$$H = \underset{H}{\operatorname{Argmax}} P(H|P)$$

- H stands for Hanzi sequences, P stands for pinyin sequences.
- $P(H|P)$ stands for the possibility of the corresponding Hanzi when Pinyin is given.

Mathematic Model (Cont.)

$$\begin{aligned}\hat{H} &= \operatorname{argmax}_H \frac{\Pr(P|H)\Pr(H)}{\Pr(P)} = \operatorname{argmax}_H \Pr(P|H)\Pr(H) \\ &= \operatorname{argmax}_H p(w_1) \prod_{i=2}^n p(w_i|w_{i-1}) \Pr(P|H) \\ &= \operatorname{argmax}_H p(w_1) \prod_{i=2}^n p(w_i|w_{i-1}) p(p_1 p_2 \cdots p_n | w_1 w_2 \cdots w_n) \\ &= \operatorname{argmax}_H p(w_1) \prod_{i=2}^n p(w_i|w_{i-1}) \prod_{i=1}^n p(p_i|w_i)\end{aligned}$$

Concrete Example

- Example: zhong'guo'ren
- $P(\text{中国人} | \text{zhong'guo'ren})$
- $= P(\text{中国人}) * P(\text{zhong'guo'ren} | \text{中国人})$
- $= P(\text{中国}) * P(\text{人} | \text{中国}) * P(\text{zhong'guo} | \text{中国}) * P(\text{ren} | \text{人})$
- $= 0.01 * 0.1 * 0.7 * 0.5$
- $= 3.5 * 10^{-4}$

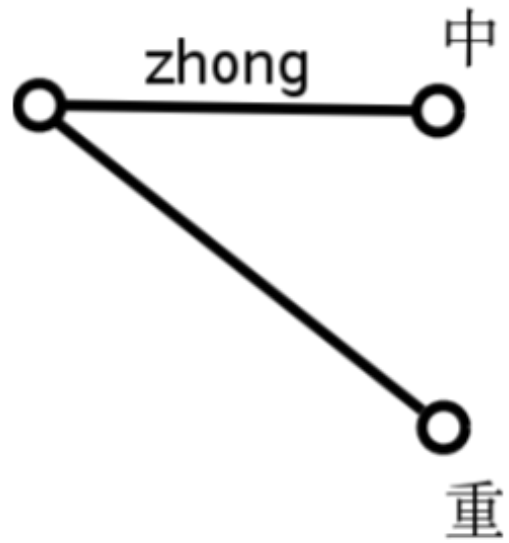
Concrete Example(Cont.)

- $P(\text{种果人} | \text{zhong'guo'ren})$
- $= P(\text{种果人}) * P(\text{zhong'guo'ren} | \text{种果人})$
- $= P(\text{种果}) * P(\text{人} | \text{种果}) * P(\text{zhong'guo} | \text{种果}) * P(\text{ren} | \text{人})$
- $= 0.01 * 0.01 * 0.8 * 0.5$
- $= 4.0 * 10^{-5}$
- $< 3.5 * 10^{-4} = P(\text{中国人} | \text{zhong'guo'ren})$
- So we will choose 中国人 as a result.

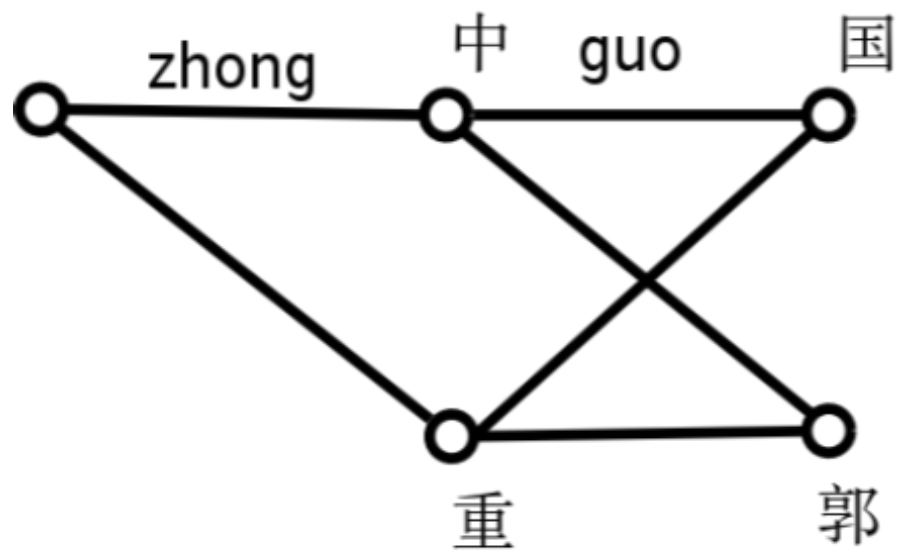
Viterbi Decode Process

- Take “ 中国人民 ” for example ,
- Input pinyin string “zhongguorenmin”,
- Will output “ 中国人民 ”

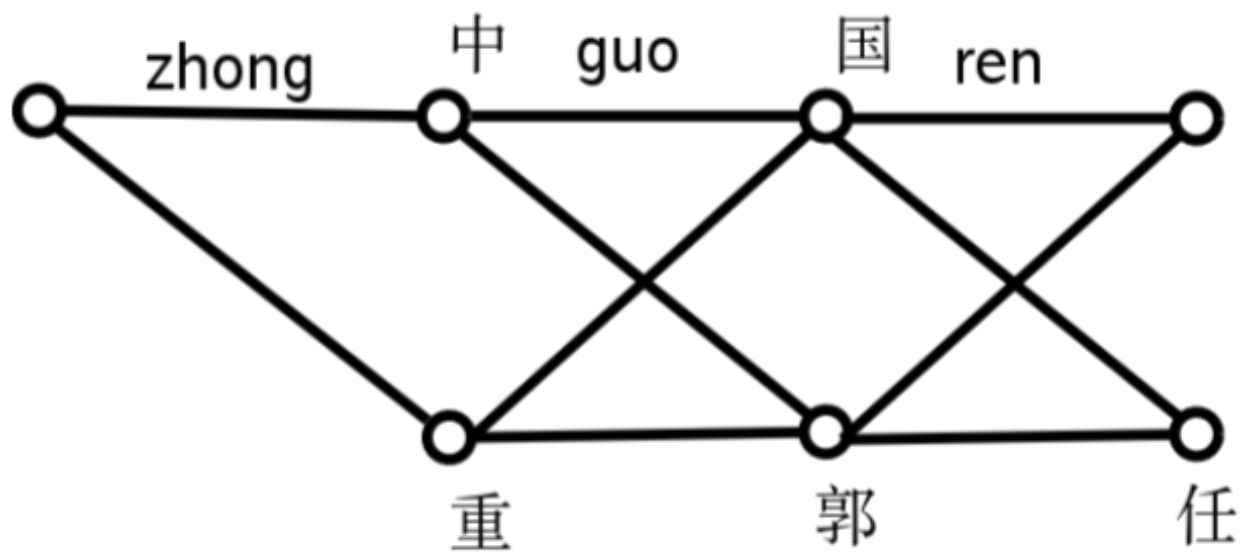
$k = 1$



$K=2$



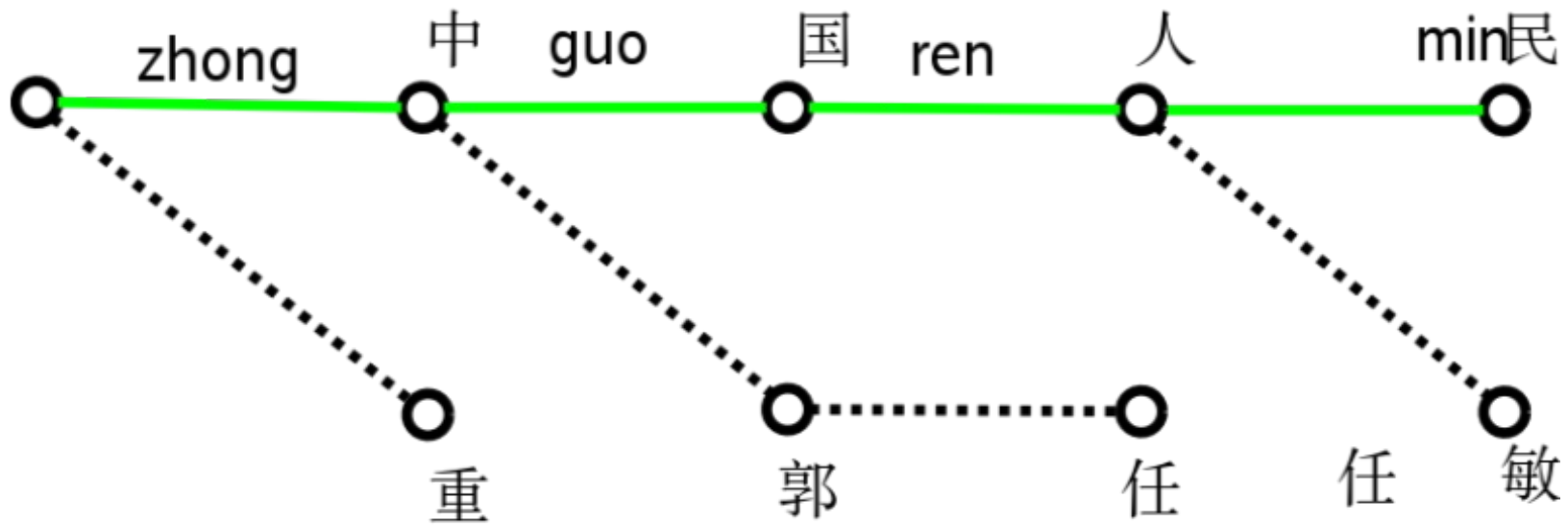
$K=3$



K=3 (Survival Path)



Final Result



Interpolation Smoothing

- Interpolation Formula
 - $P_{interpo} = \lambda * P_{bi-gram} + (1 - \lambda) * P_{uni-gram}$
 - But need more complicated EM algorithms to compute the λ parameter.
 - Refer to deleted interpolation
- The key difference between back-off and Interpolation.
 - Back-off would not take the lower gram, when explicit estimation exists.
 - Interpolation considers uni-gram even when bi-gram available.

Source Code

- include: include directory
- memory_chunk.h novel_types.h stl_lite.h
- lookup: pinyin lookup
- lookup.h pinyin_lookup.cpp winner_tree.*
- segment: word segment
- mmseg.cpp
- storage: binary storage
- ngram.* pinyin_base.* pinyin_phrase.h
- phrase_index.* pinyin_large_table.*
- training: training module
- estimate_interpolation.cpp gen_unigram.cpp
gen_ngram.cpp

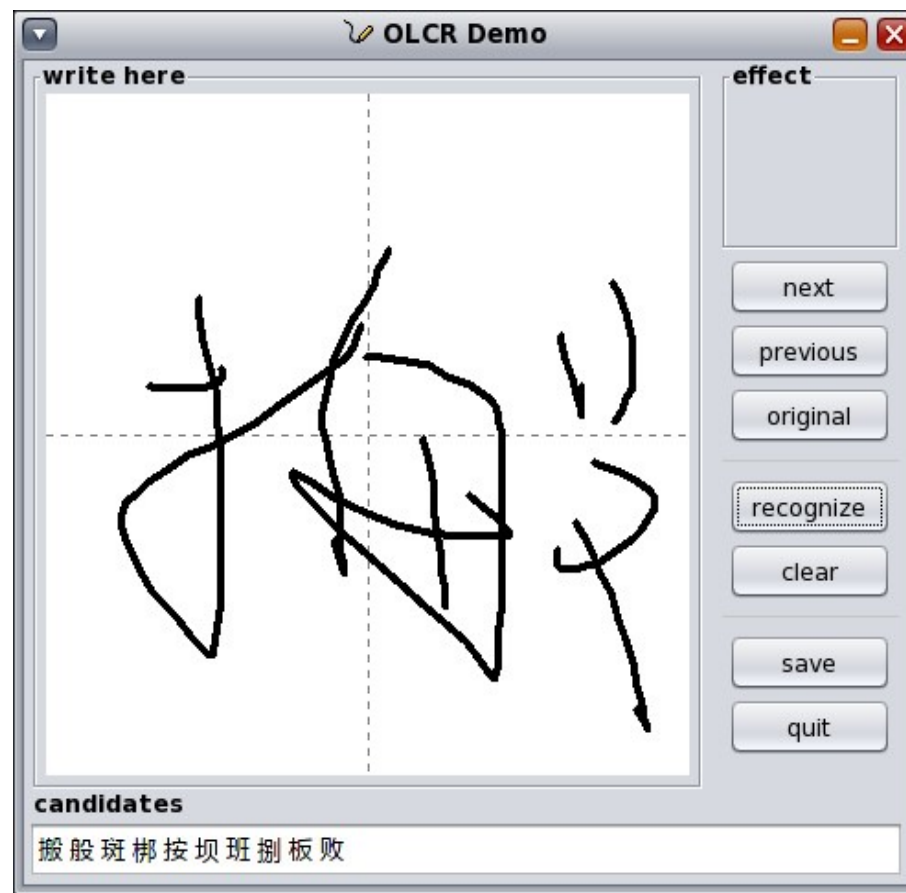
On-line Handwriting Recognition

- On-line character recognition input system can introduce a more comfortable Human-Computer interface for desktop users.
- Well-suited to text entry for large alphabets (like Asian languages).
- Lack of mature open source OLCR input method.

O-Pen: An OLCR System

- Developed by Sun engineer, [Feng Zhu](#), and would be opensource'd to [OS.o input-method](#) project in near future.
- A statistical recognition method for Asian languages like CJK (Chinese, Japanese and Korean).
 - In contrast to structural recognition method, it can recognize glyphs better in cases where the user writes haphazard characters, providing high accuracy rate.
 - Statistical recognition methods are more portable because they do not need a complex analysis of character structure features.
- Based on angles feature extraction.
- Training patterns with samples collected from hundreds of peoples.

Screenshots of O-Pen



Q&A