

# Building a Distributed Call Center

Leif Madsen

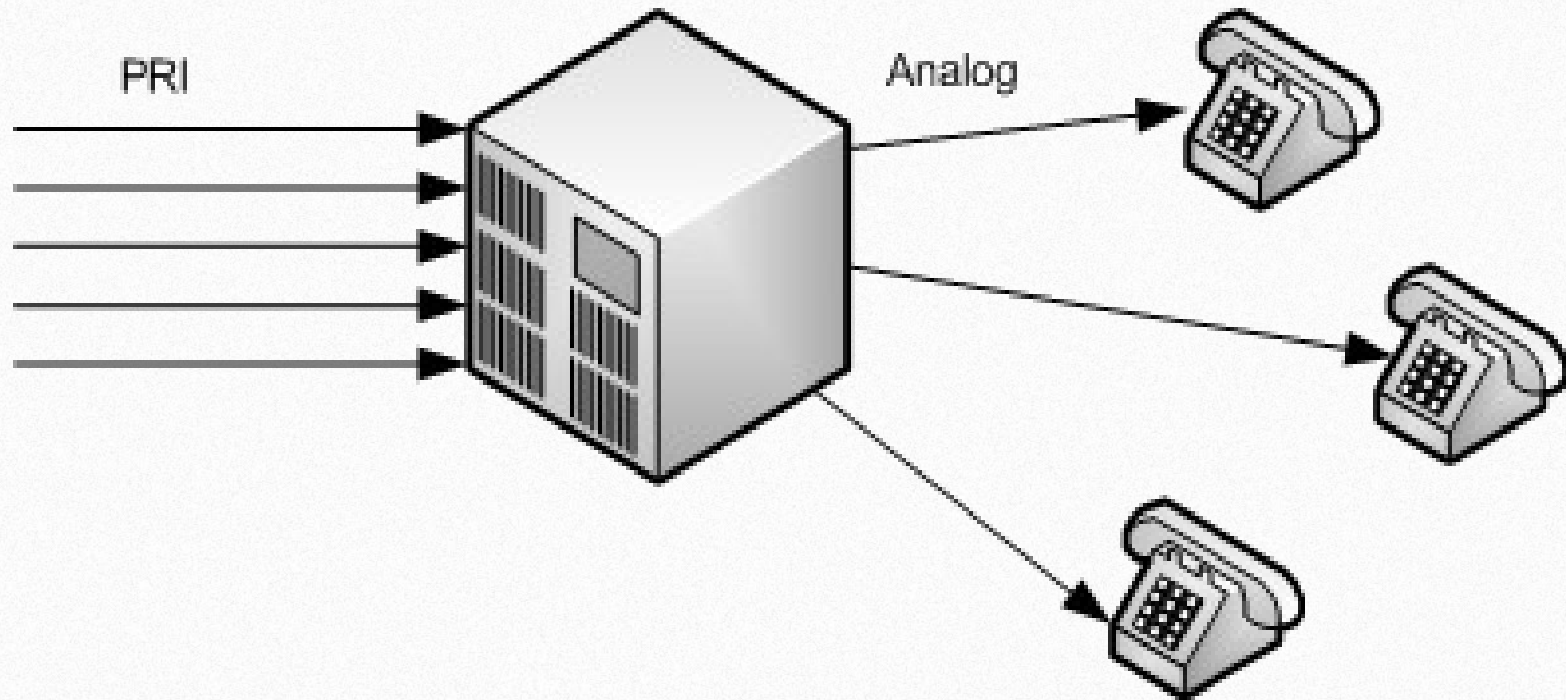
AstriCon 2010

- **Leif Madsen, Asterisk Consultant**
- **Co-author of Asterisk: The Future of Telephony (Asterisk 1.4 based)**
  - **Currently working on Asterisk: The Definitive Guide (Asterisk 1.8 based)**
- **Asterisk bug marshal**
- **Asterisk release manager**
- **Specialize in database integration and distributed call center implementations**

- Topologies
  - Describe various call center setups
  - Start simple; increasing complexity
  - Call center types
    - Traditional
    - Hybrid
    - Database driven
    - Distributed

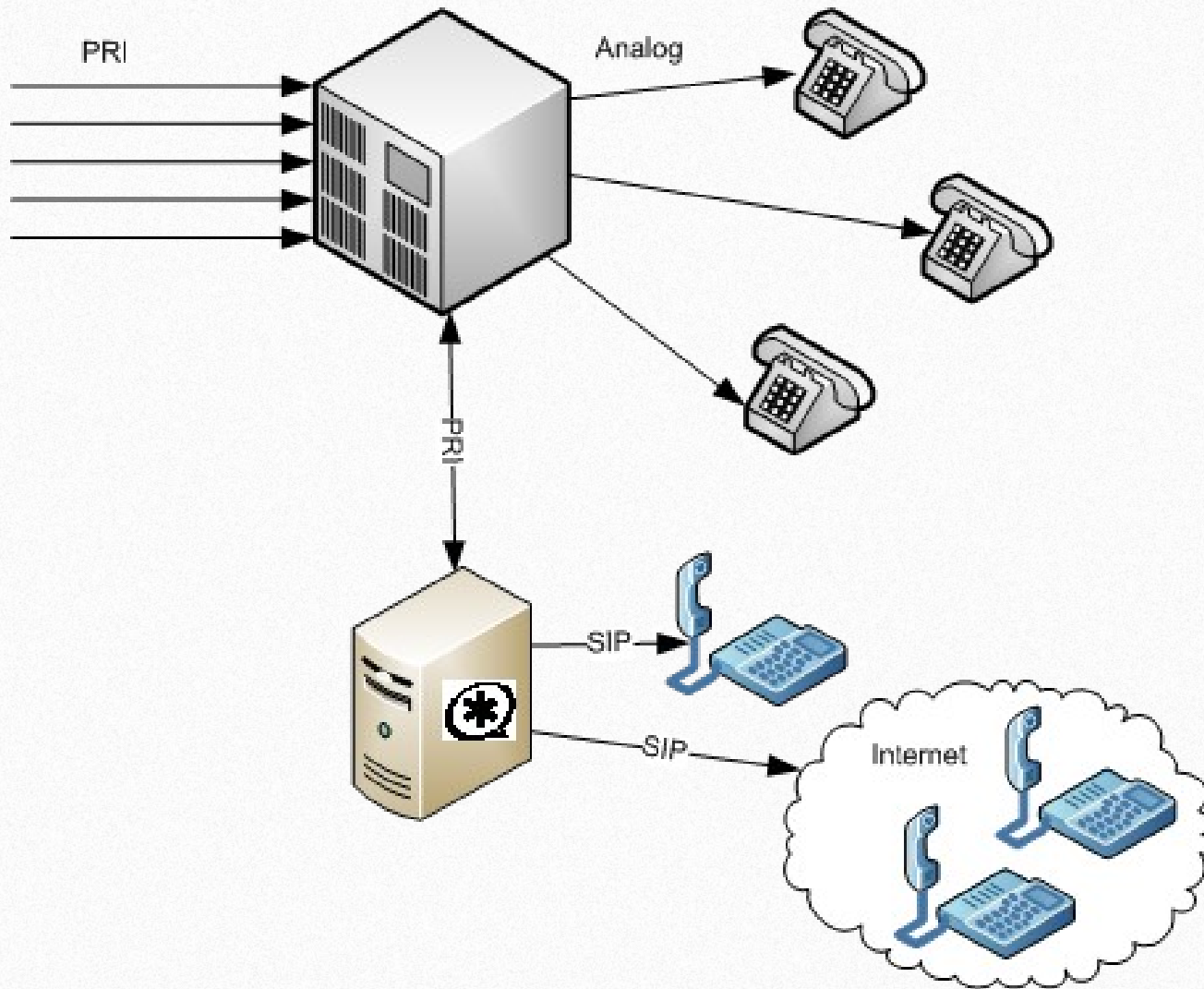
# Topologies and Discussion

# Traditional Call Center



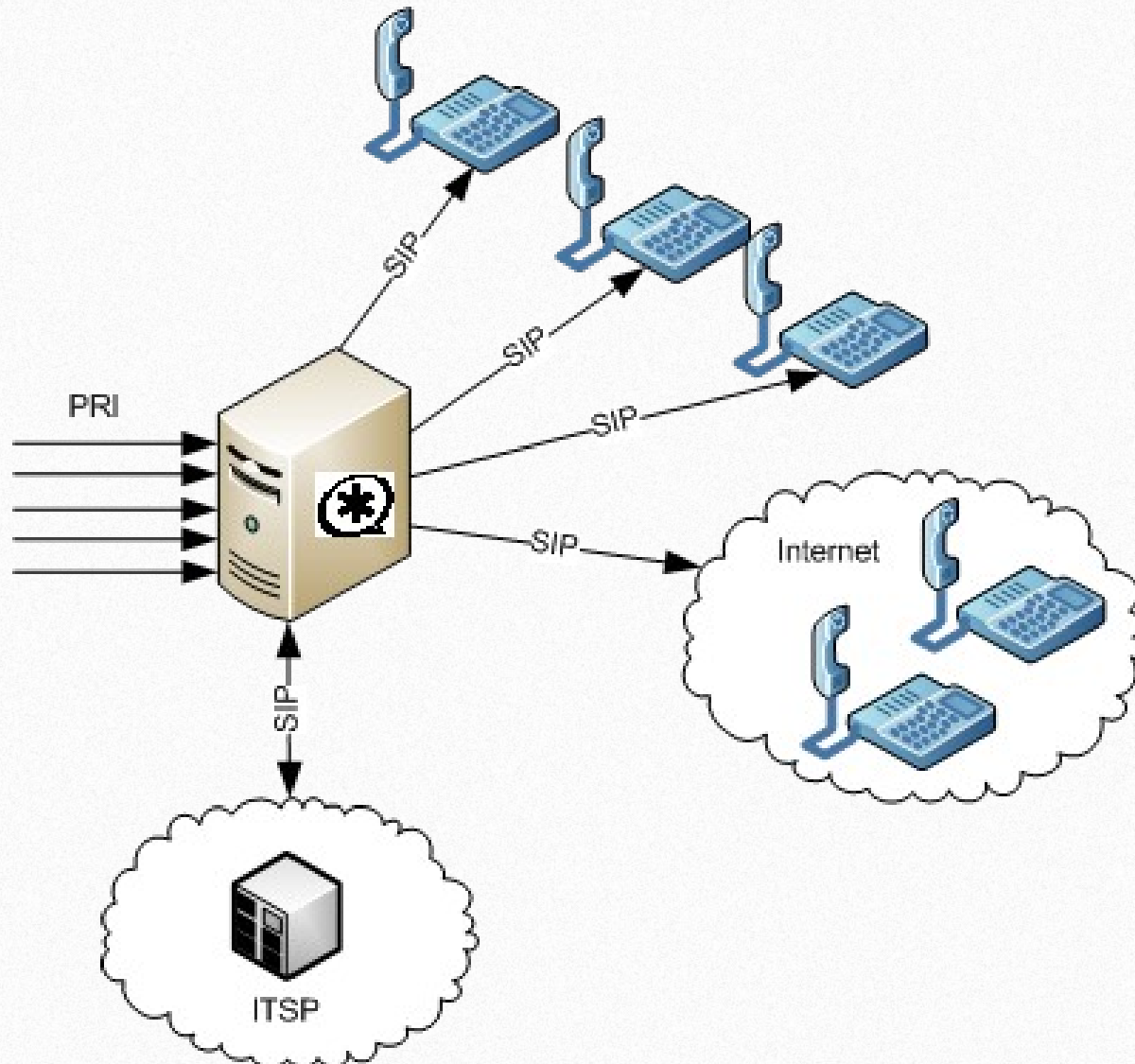
- Calls come from PRI into traditional PBX
- Delivered to agents via standard ACD (Automatic Call Distribution) rules
- Not distributable; hard (impossible?) to deliver calls to remote employees
- Typically need to be physically connected to the phone system

# Hybrid System



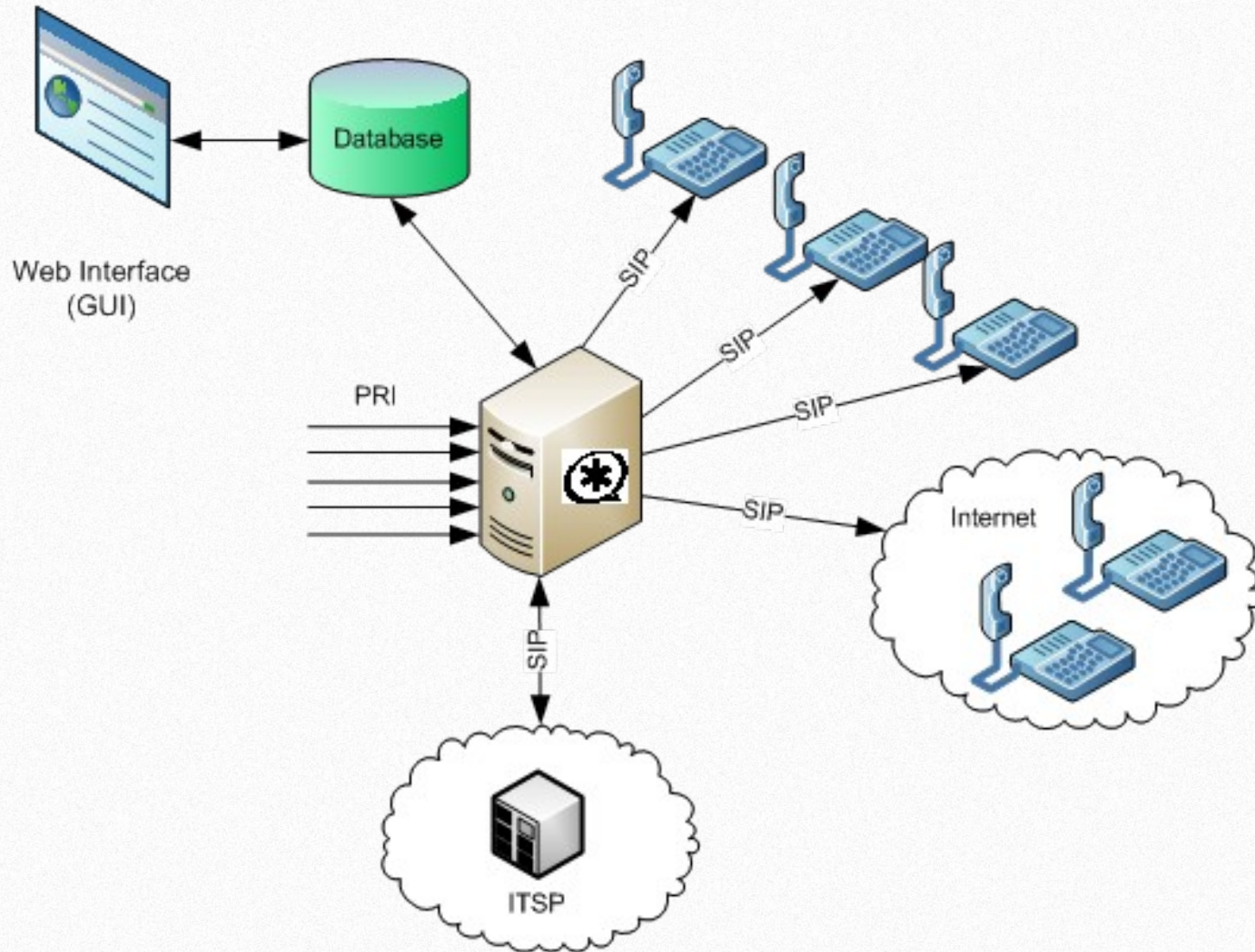
- Asterisk is the enabler; SIP end-points add new technologies to existing system
- Still limited by physical connections
- Typically expensive (PRI) or unreliable (analog)
- All hardware needs to reside at call center facility
- Allows remote employees (yay!)

# Pure Asterisk, Non-Distributed



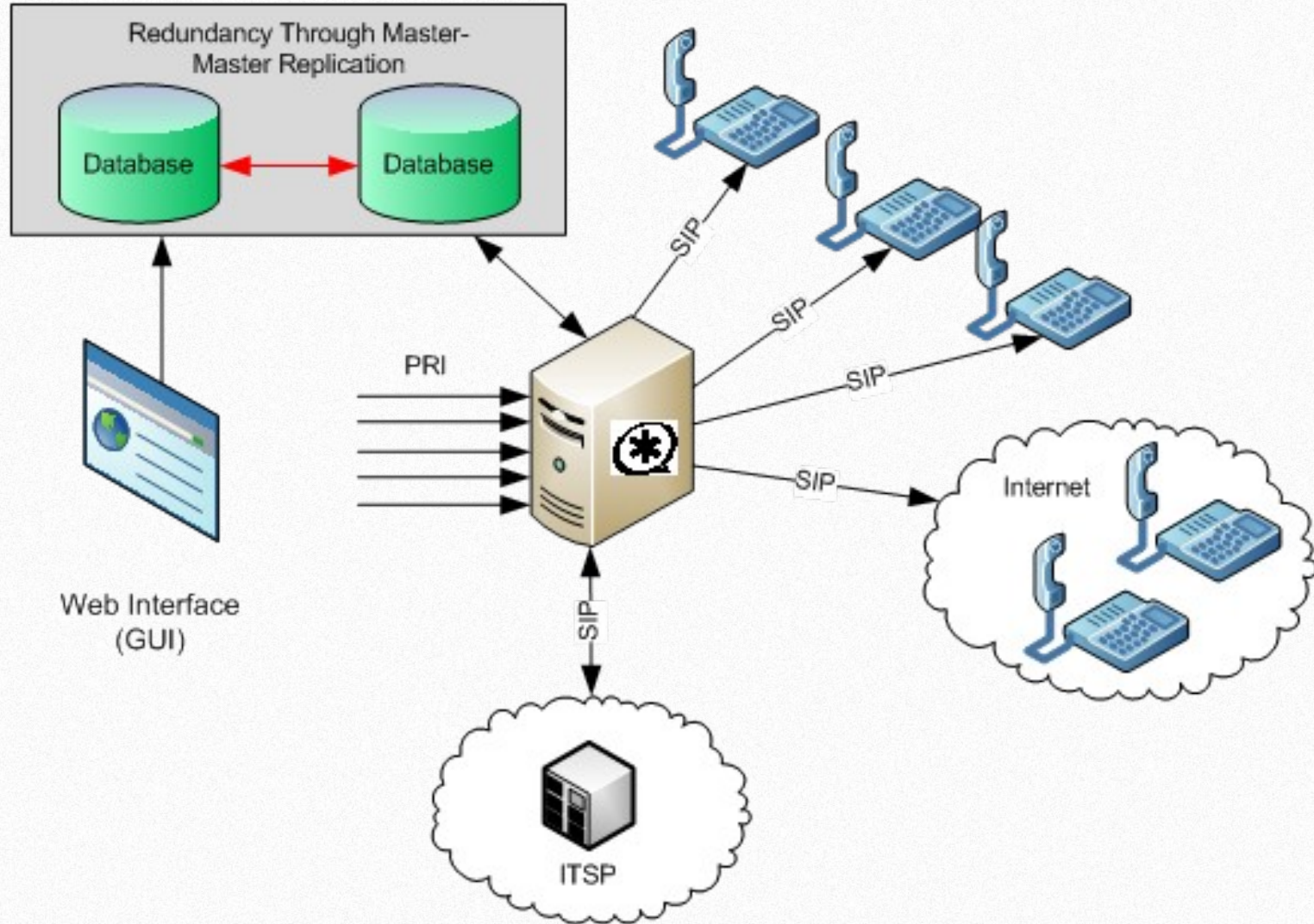
- No limitation on number of lines (within reason)
- No need for expensive lines at all; could be entirely SIP based
- Remote employees easily added to system
  - Save on electricity costs
  - Increase employee moral
  - Serve more timezones
- Simple and efficient, but limited in expandability

# Asterisk + Database



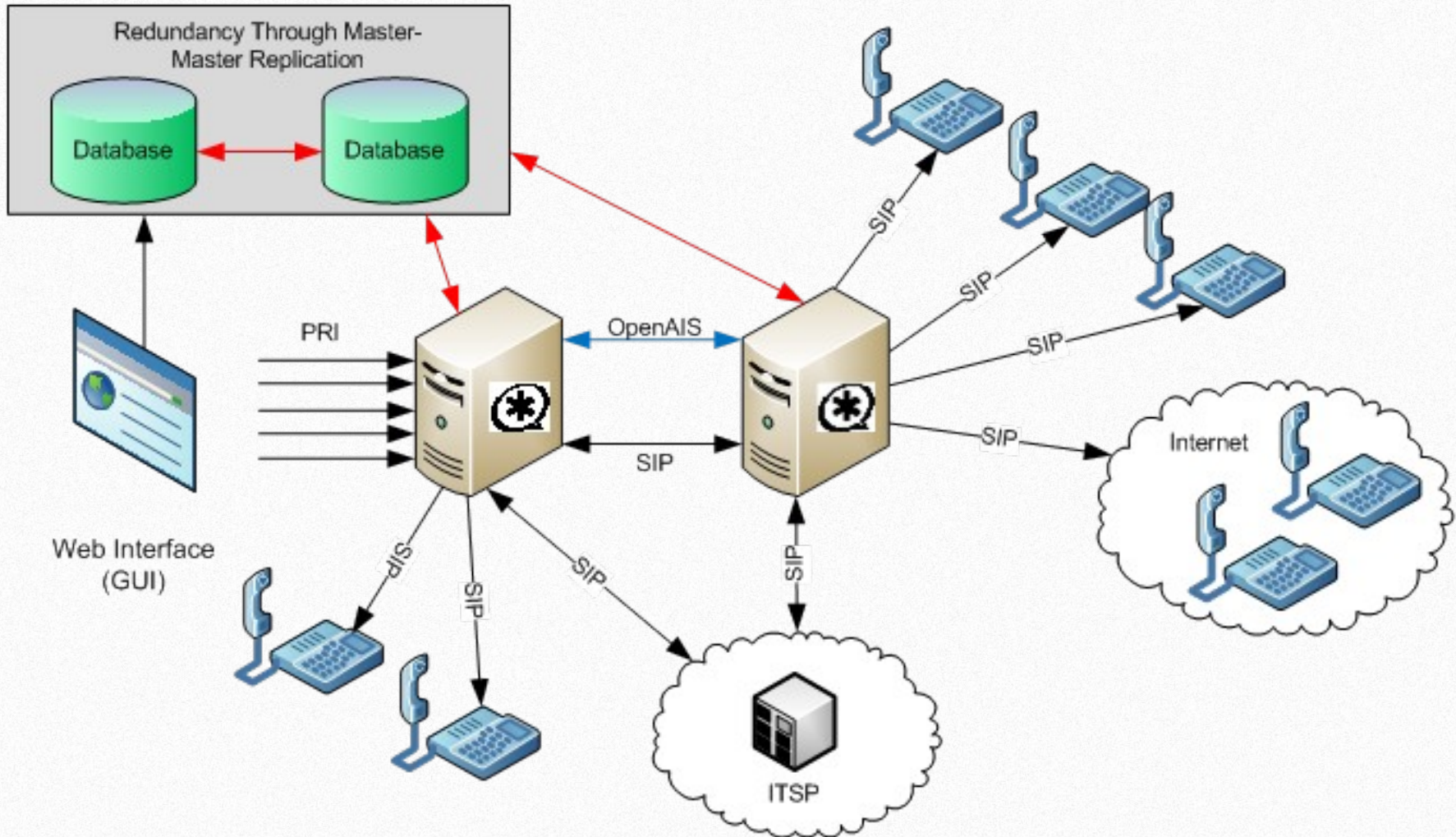
- Slightly more complex; but easier to update (realtime)
- Create own GUI system; need not be complex
  - `func_odbc` recommended; dynamic data, static dialplan
  - also like `func_curl`
- Start clustering; hot-swap via LinuxHA
- Keep `/etc/asterisk` in (r)sync
- Changes are nearly *realtime*

# Asterisk + Replicated Database



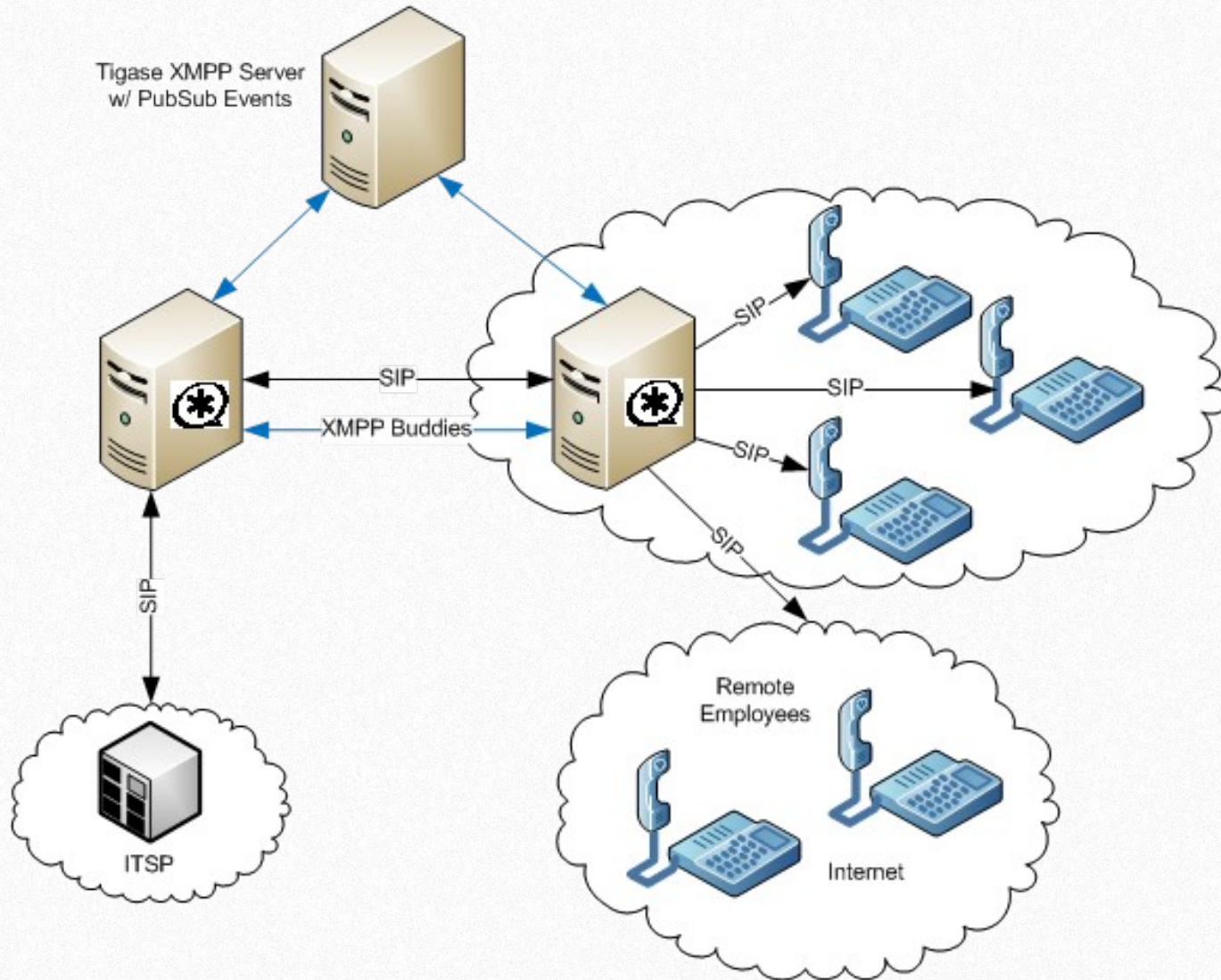
- Master-Master Replication with MySQL
- Perform load balancing
- Help prevent downtime with redundant, live databases
- Reasonably easy to setup and maintain
- Failover with *pen*; `res_odbc` also fails over

# Asterisk + Database w/ Distributed Device State



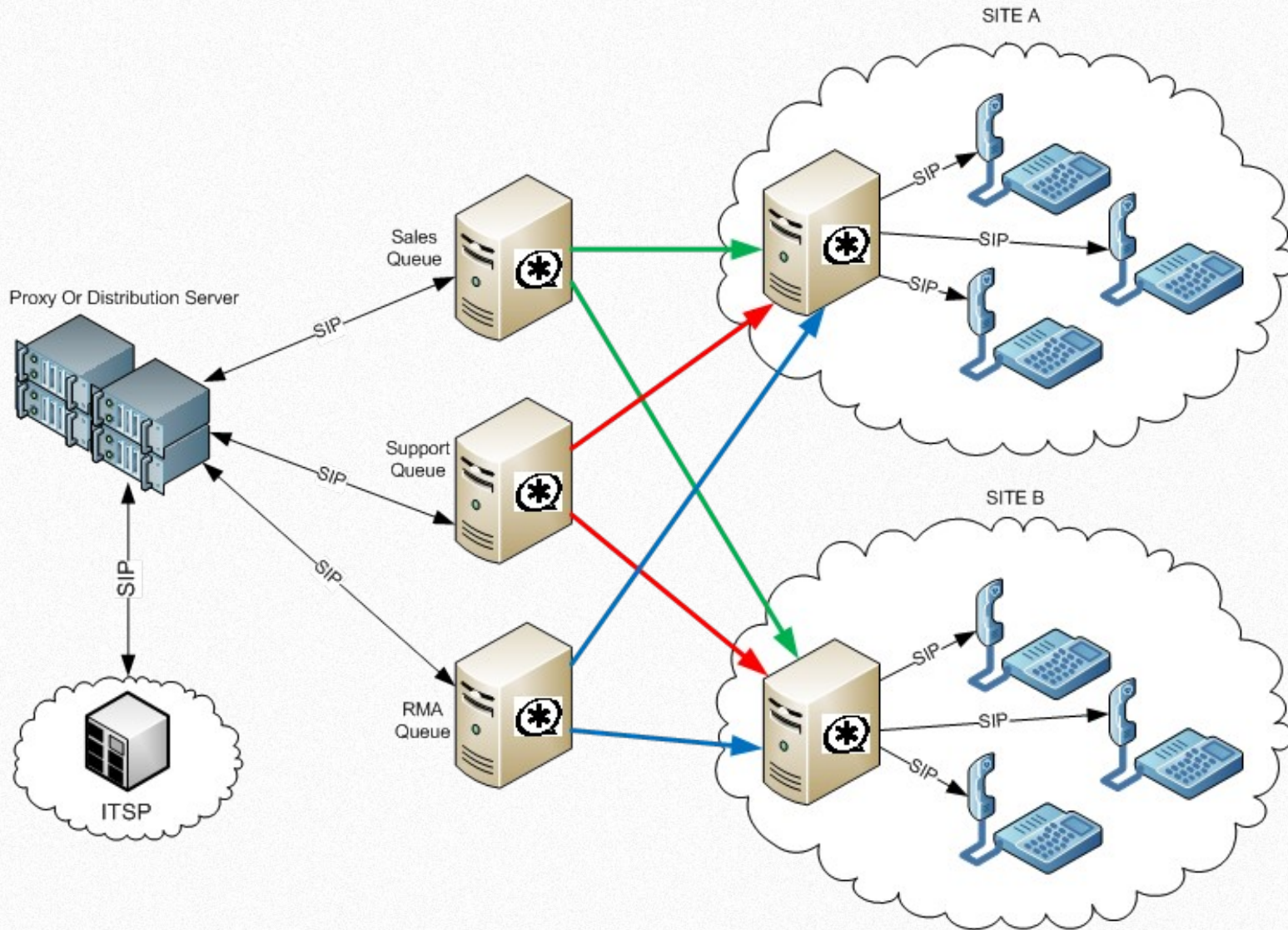
- Multiple systems know status of devices remotely
- Allows agents to reside on multiple Asterisk systems
- Can login to queues across the cluster
- Status of agents are known on other servers; no multiple ringing or excessive attempts
- With OpenAIS you can only use over a low latency link (e.g. LAN)

# Distributed Device State Over WAN



- Asterisk 1.8 allows device state to be distributed over WAN links
- Uses Tigase XMPP server due to PubSub support
- Allows agents to reside in different physical locations and answer calls from queues across the cluster
- Can provide redundancy by failing over calls from PSTN or ITSP to other servers

# Multi-Queue, Multi-Site



- Load distributed across multiple systems
- Useful where queues are heavily utilized
- Expand number of agents by adding additional servers
- Agent servers can perform other tasks
- Save processing power on queue servers for just queues

- For large queues, must be on the same system
- No ability to share queue position
  - Does it matter?
- Will callers know they are the 3<sup>rd</sup> calls on one system vs. multiple-systems?

- Started simple; increased in design complexity
- Many ways to grow as your company expands
  - Don't go more complicated than necessary
  - If you're only 40 seats, don't build a solution for 500 seats
- Building a simple system now, with good forward planning, can be expanded
- Increase employee moral and save on hardware and electricity costs with remote agents
  - Requires planning and testing to verify connection is suitable for voice

# Questions?

# Leif Madsen

<http://ofps.oreilly.com>  
(Asterisk: The Definitive Guide, Public Review)

Twitter: **leifmadsen**