A Programmable Structure for Pervasive Computing

Lars Brenna

Joint work with Ingar M. Arntzen and Dag Johansen

Department of Computer Science
University of Tromsø
Norway

Outline

1. Problems

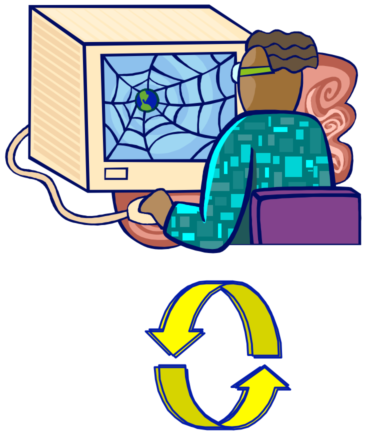
2. The WAIF approach

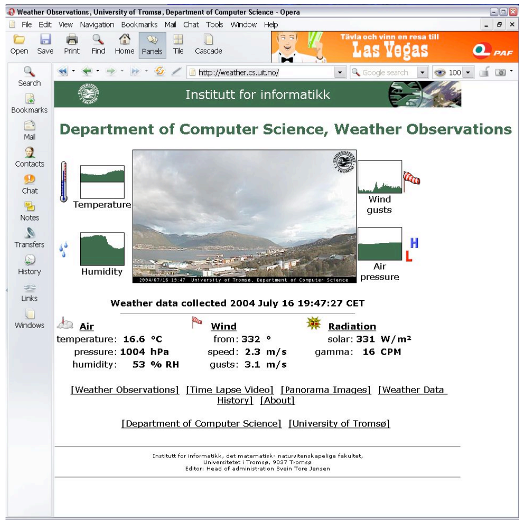
3. WAIF implementations

4. Lessons learned

1. Problems

- *Inherited structure* from the initial Internet: client-server.





Weather Observations, University of Tromsø, Department of Computer Science - Opera

http://weather.cs.uit.no/

Institutt for informatikk

Department of Computer Science, Weather Observations

2004/07/16 19:47 University of Tromsø, Department of Computer Science

Weather data collected 2004 July 16 19:47:27 CET

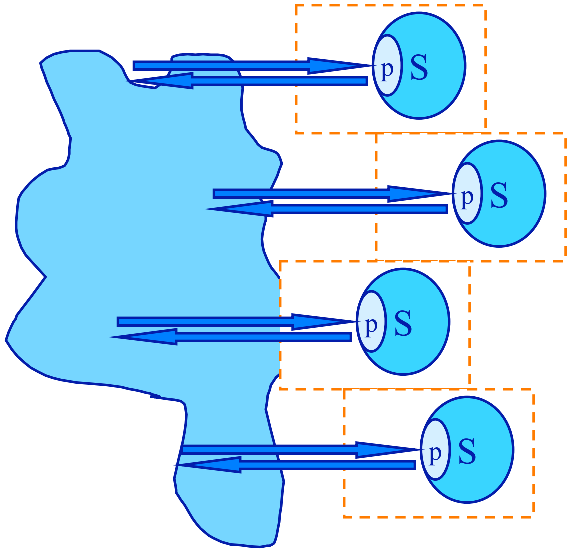
Air		Wind		Radiation	
temperature:	16.6 °C	from:	332 °	solar:	331 W/m²
pressure:	1004 hPa	speed:	2.3 m/s	gamma:	16 CPM
humidity:	53 % RH	gusts:	3.1 m/s		

[Weather Observations] [Time Lapse Video] [Panorama Images] [Weather Data History] [About]

[Department of Computer Science] [University of Tromsø]

Institutt for informatikk, det matematisk- naturvitenskapelige fakultet,
Universitetet i Tromsø, 9037 Tromsø
Editor: Head of administration Svein Tore Jensen

File Edit View Navigation Bookmarks Mail Chat Tools Window Help
Open Save Print Find Home Panels Tile Cascade
Tävla och vinn en resa till Las Vegas
PAF
Google search
100
Search
Bookmarks
Mail
Contacts
Chat
Notes
Transfers
History
Links
Windows
Temperature
Humidity
Wind gusts
Air pressure
H
L



- Fact: interaction model that *takes (user) time*.

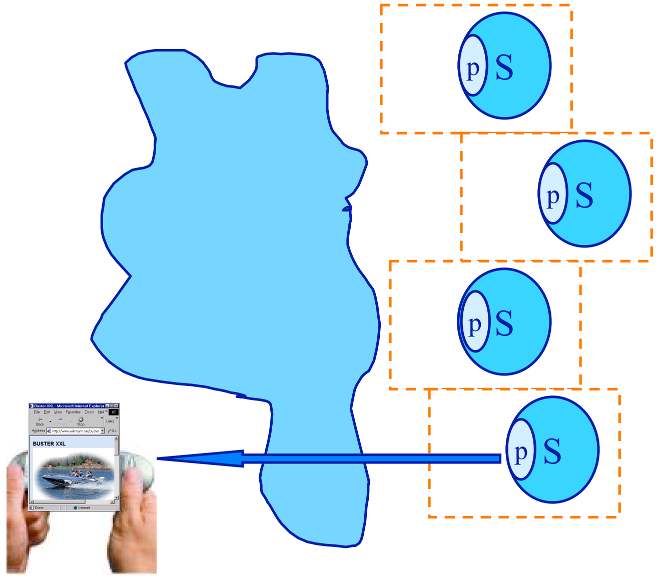
1. Towards a Proactive Internet

- We conjecture that the Web's next paradigm shift will include a much more *proactive* computing model.

- This will transform a passive web being searched by users, to information and service providers *searching actively* for users.

1. Proactive Internet

- The web works autonomously on your behalf and notifies you.



<u>Goals</u>:

- High recall.
- Extreme precision.
- Context-aware.
- Real-time.

2. WAIF (Wide Area Information Filtering)

- Problem:

  ”*How to structure the next generation web.*”

- International cooperation:

  University of Tromsø, Cornell University, and UC San Diego.

2. WAIF Principles

- Approaches:
  1. *Proactive computing* combined with high precision:

     → humans not in the loop, but above the loop.

  2. Use the network as a *personal* computer:

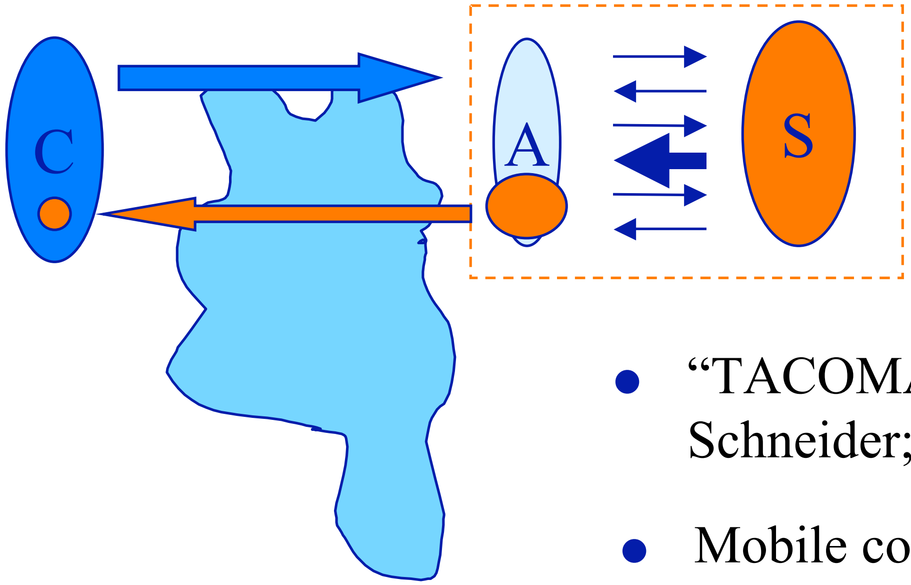
     → a single user should have his private push-based network.

  3. Mobile users in a pervasive computing environment:

     → design for mobility.

2. Extensible Servers

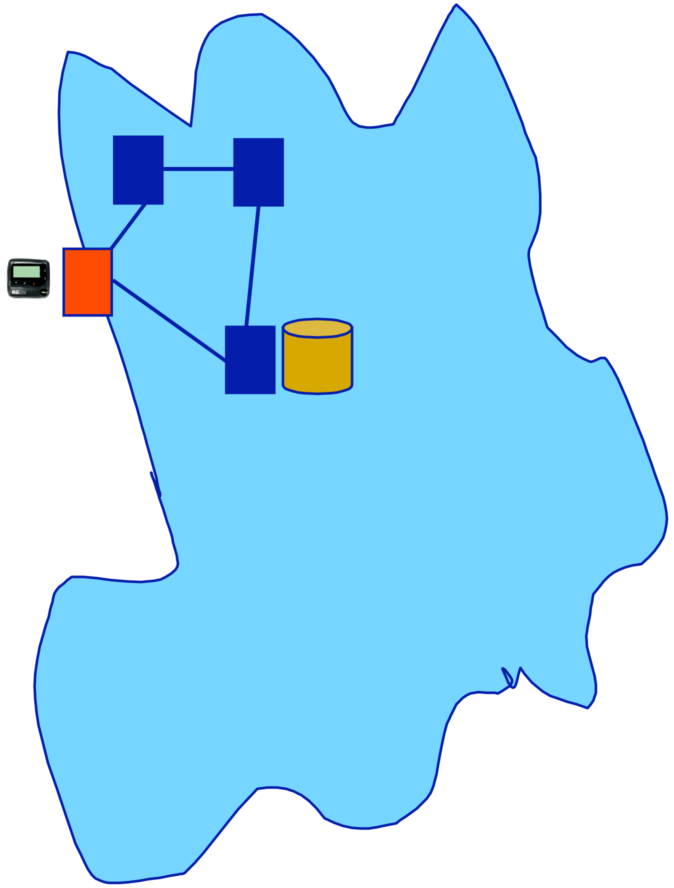
- Mobile code: suitable for run-time software installation _ *extensible servers.*



- “TACOMA”; Johansen, van Renesse & Schneider; 1994.
- Mobile code: program and install autonomous code (A: Python, C, Perl, Tcl, Java, Scheme) and data at remote servers.
- Current WAIF servers use the TOS kernel, http://tos.sourceforge.net/

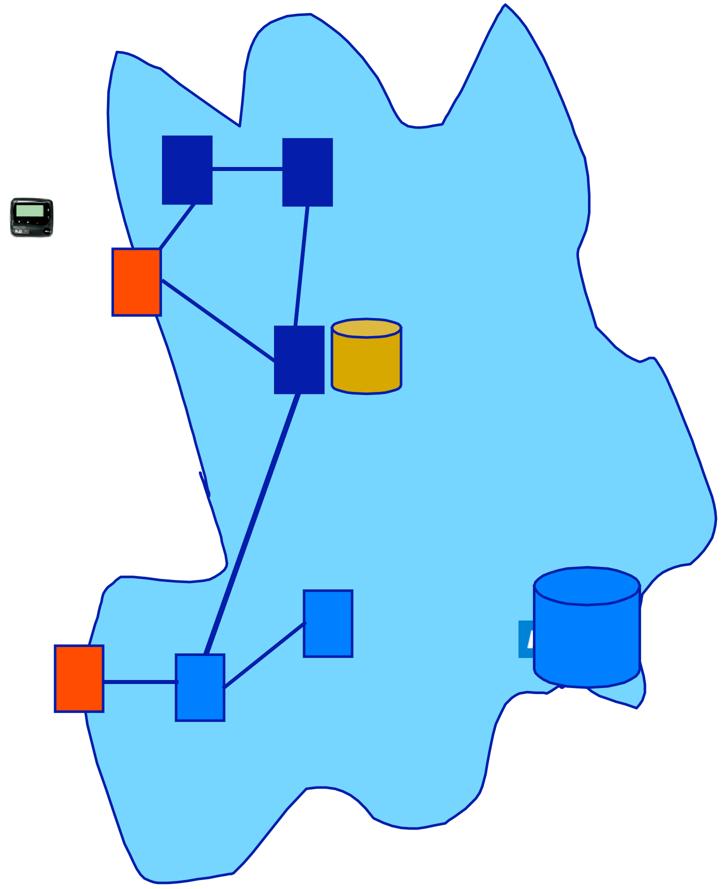
2. Software Architecture

- Pervasive computing: environment saturated seamlessly with computers, sensors and communication facilities.

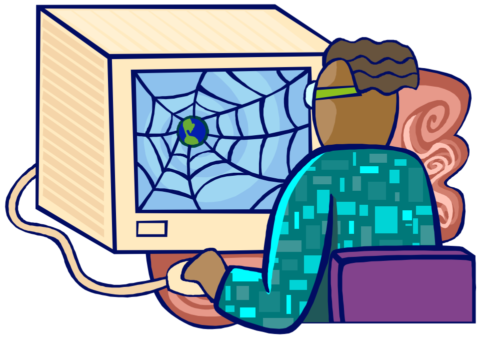


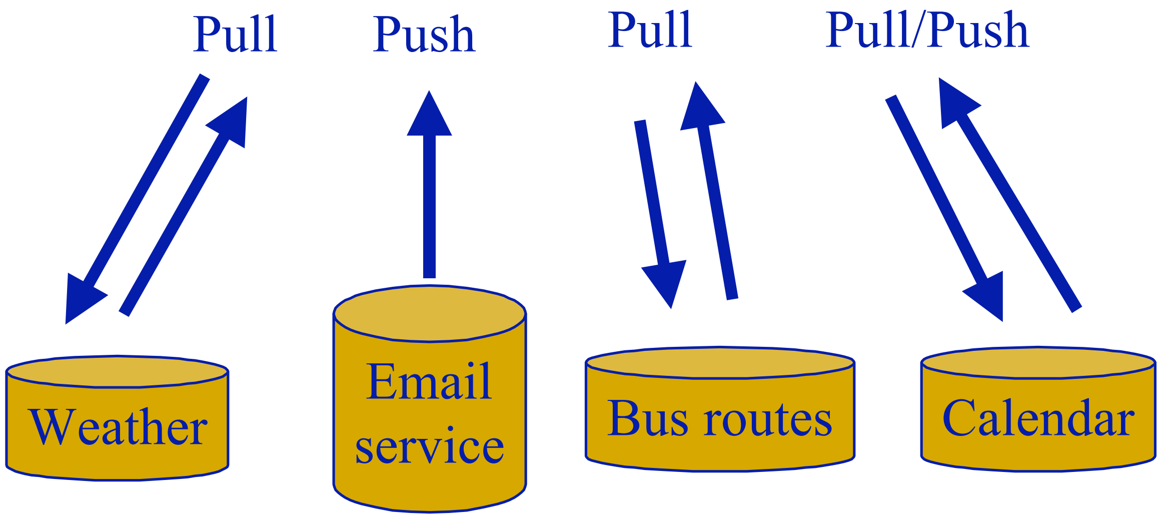
2. Software Architecture

- Pervasive computing: environment saturated seamlessly with computers, sensors and communication facilities.
- Mobile agent lessons: Install software components remotely. Single-hop agents the normal case, multi-hop the special case.
- Run-time configuration renting services from the environment (3'rd parties).

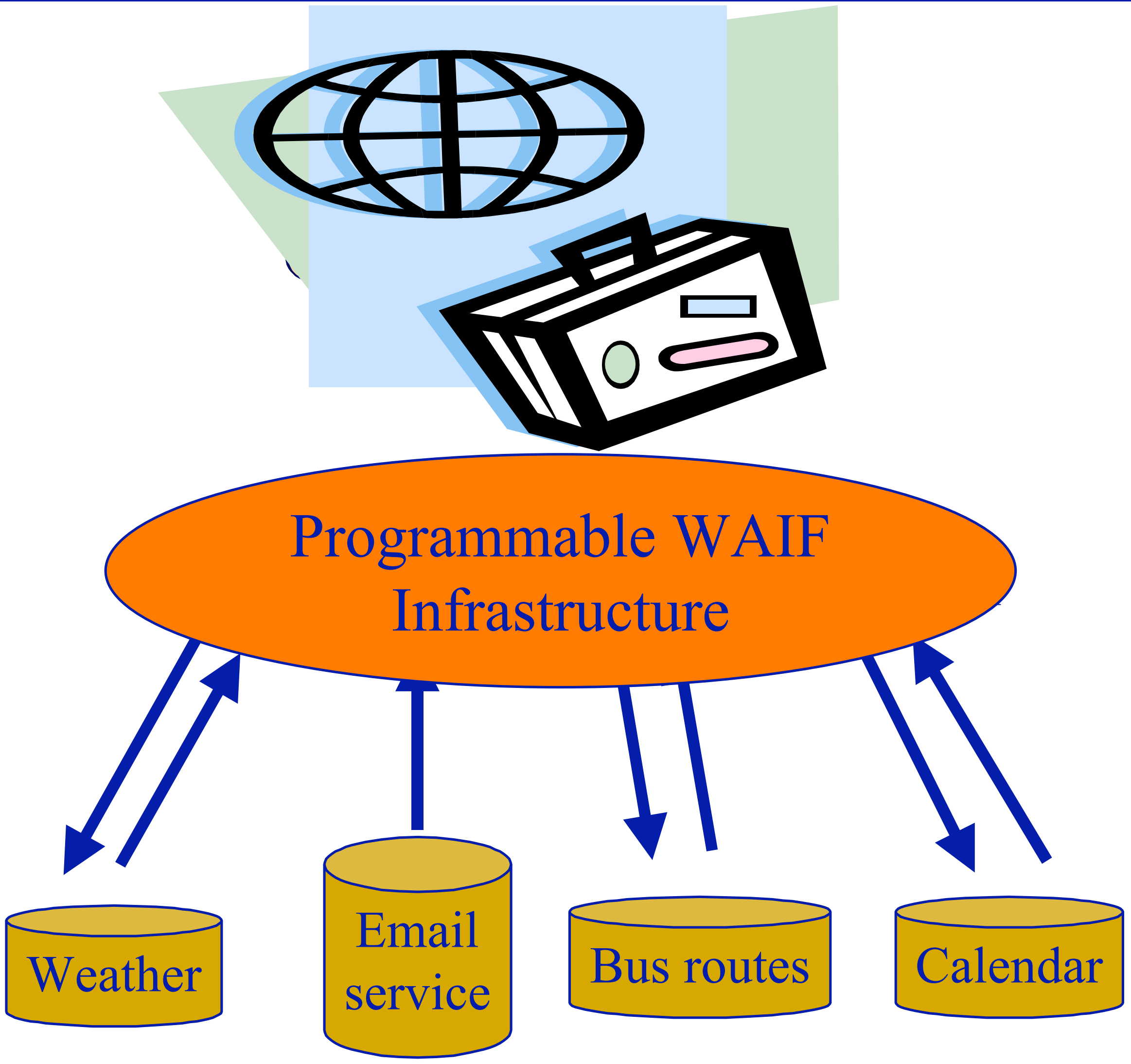


3. Today's Interaction Model





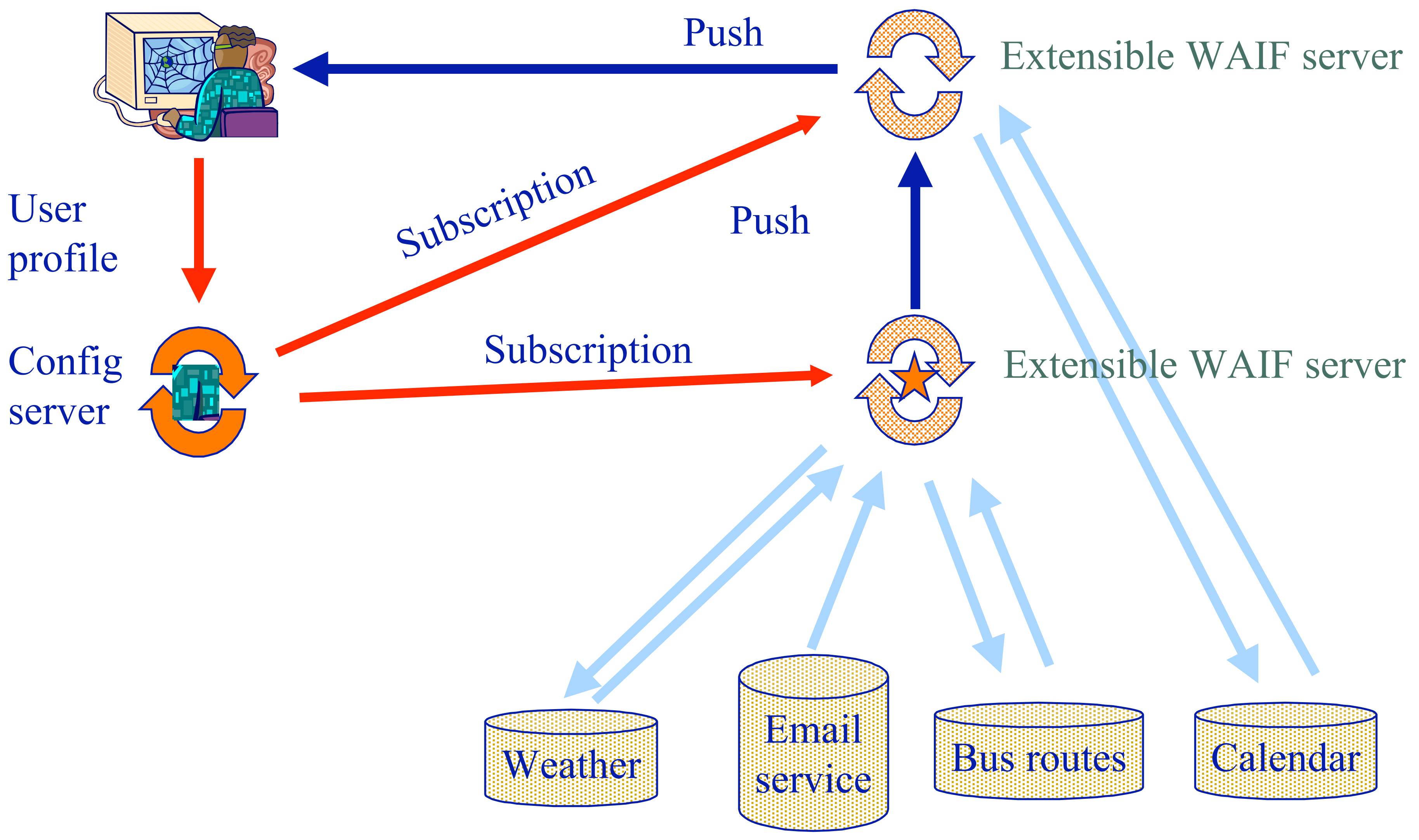
3. The WAIF Approach



3. Programming a WAIF Infrastructure

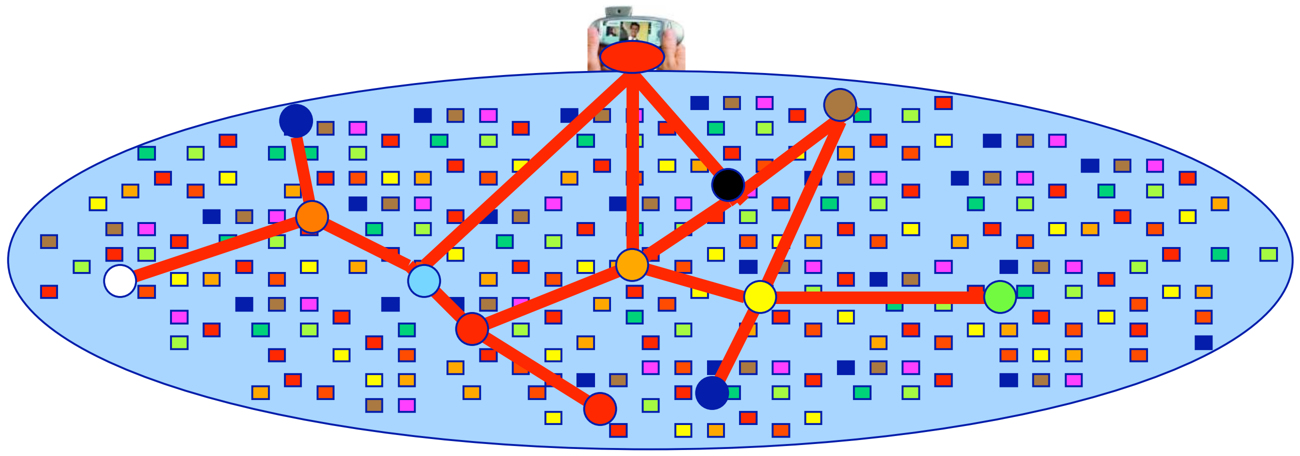
- Extensible and programmable servers.
- Programmed by expressive mobile code (filters).
- Filters extend server functionality, either user-specific or globally.
- No explicit programming required for novice Internet users.

3. Programming a WAIF Infrastructure

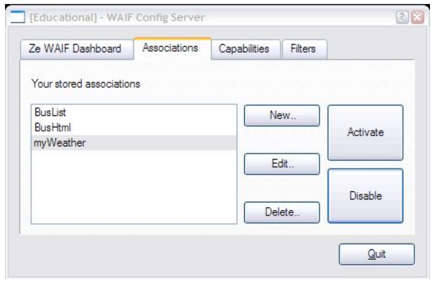


3. ”Personal Overlay Network System”

- Locate extensible servers and create your personal distributed system as an overlay network (*PONS*).



3. PONS Configuration

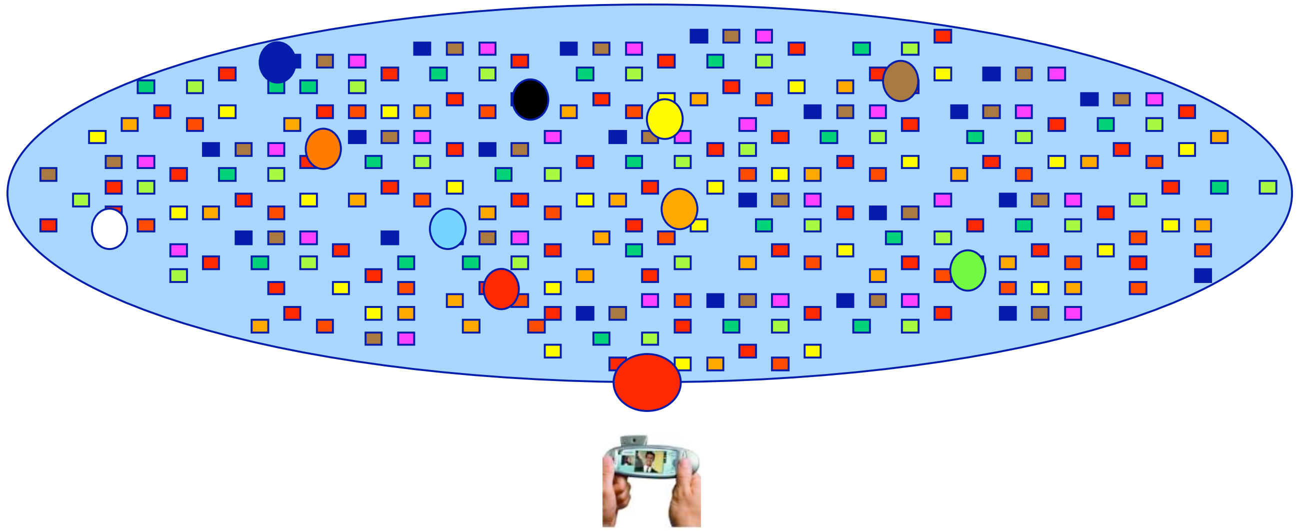

[Educational] - WAIF Config Server

Ze WAIF Dashboard | Associations | Capabilities | Filters

Your stored associations

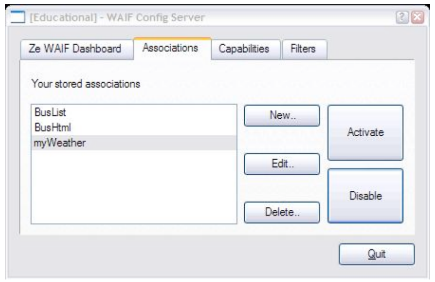
BusList
BusHtml
myWeather

New.. | Edit.. | Delete.. | Activate | Disable | Quit



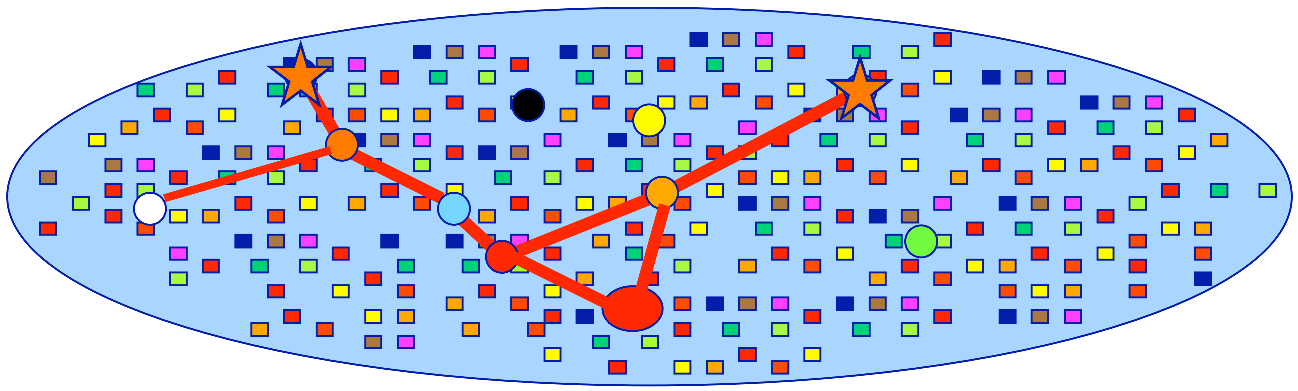
- No programming required for novice internet users
- User profile automatically mapped to overlay network structure

3. PONS Event Filtering and Routing

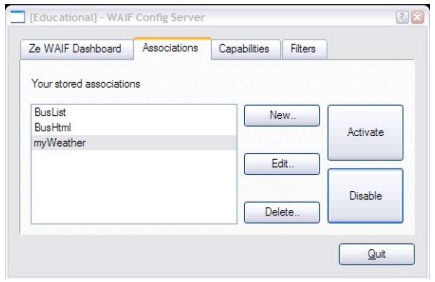

[Educational] - WAIF Config Server

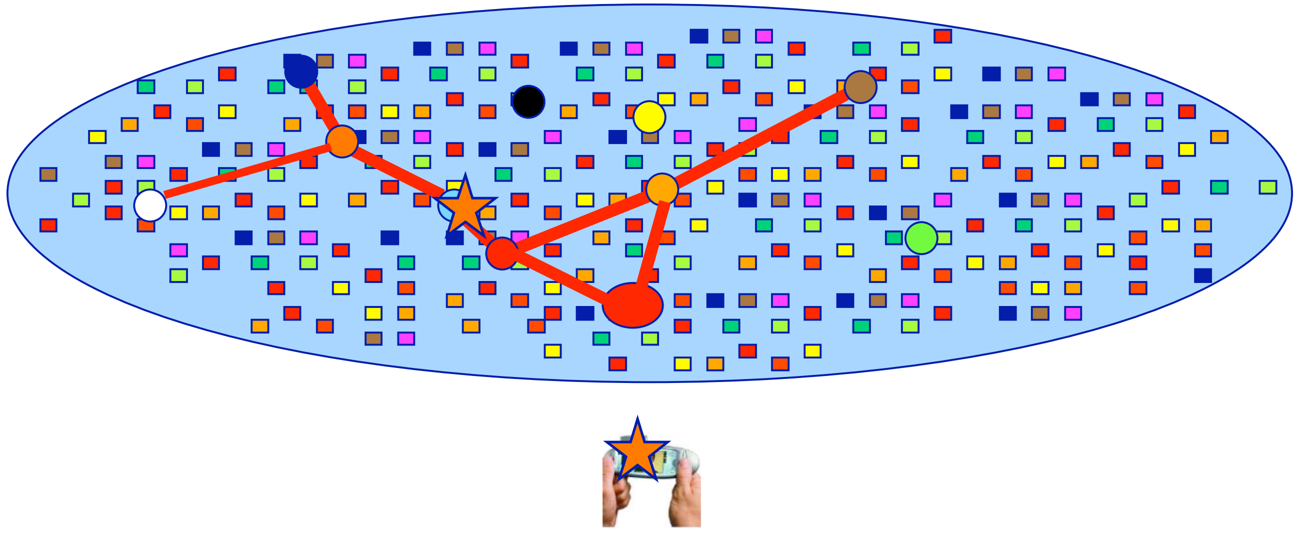
Your stored associations

BusList
BusHtml
myWeather



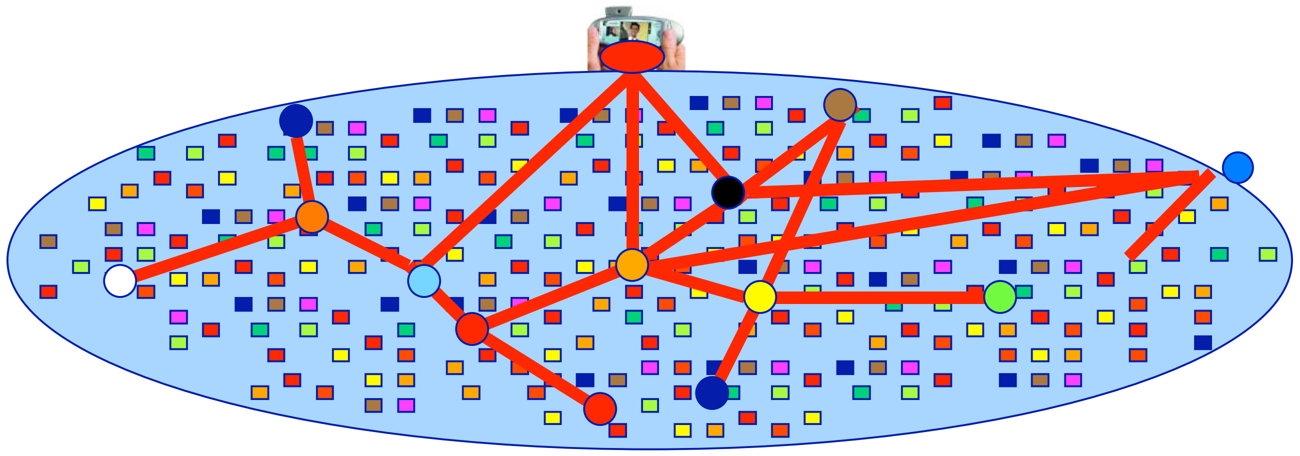
3. PONS Event Filtering and Routing





3. Supporting Mobile Users

- Move *user environments* transparently along.



- WAIFARER: Task migration for legacy applications
  - move the desktop around http://waifarer.sourceforge.net

3. Example Filter Programs

- Python code:

```
event = self.in.get()                              #upon msg: mk event
if self.check_importance(event) >= self.alertlevel:
    self.out.push(event)                                 #deliver ICPS
event
else:
    self.buffer.put(event)                         #wait for new orders
```

- Key:value dictionary:

  { userID='rharaty',
  datatype='ICPS2004',
  filter='myICPSFilter',
  alertlevel = HIGH
  }

3. WAIF Server Internals

- Python SOAP-RPC
  - Synchronous delivery, asynchronous handling.
  - Fault tolerance mechanisms.
  - Could perhaps also use JXTA

- Servers are instances of the downloadable `WAIFService` Python package.
  - http://waif.cs.uit.no/downloads

3. WAIFService Package

1. Init package.
2. Register custom event handlers.
3. `self.run()`

- Example custom event handler:

```
def busroute_handler(self, event):
   unpack (event)
   profile = self.users(user)
   self.push( address, subID,
              self.getroute(profile, event)
            )
```

3. WAIF Server Exported API

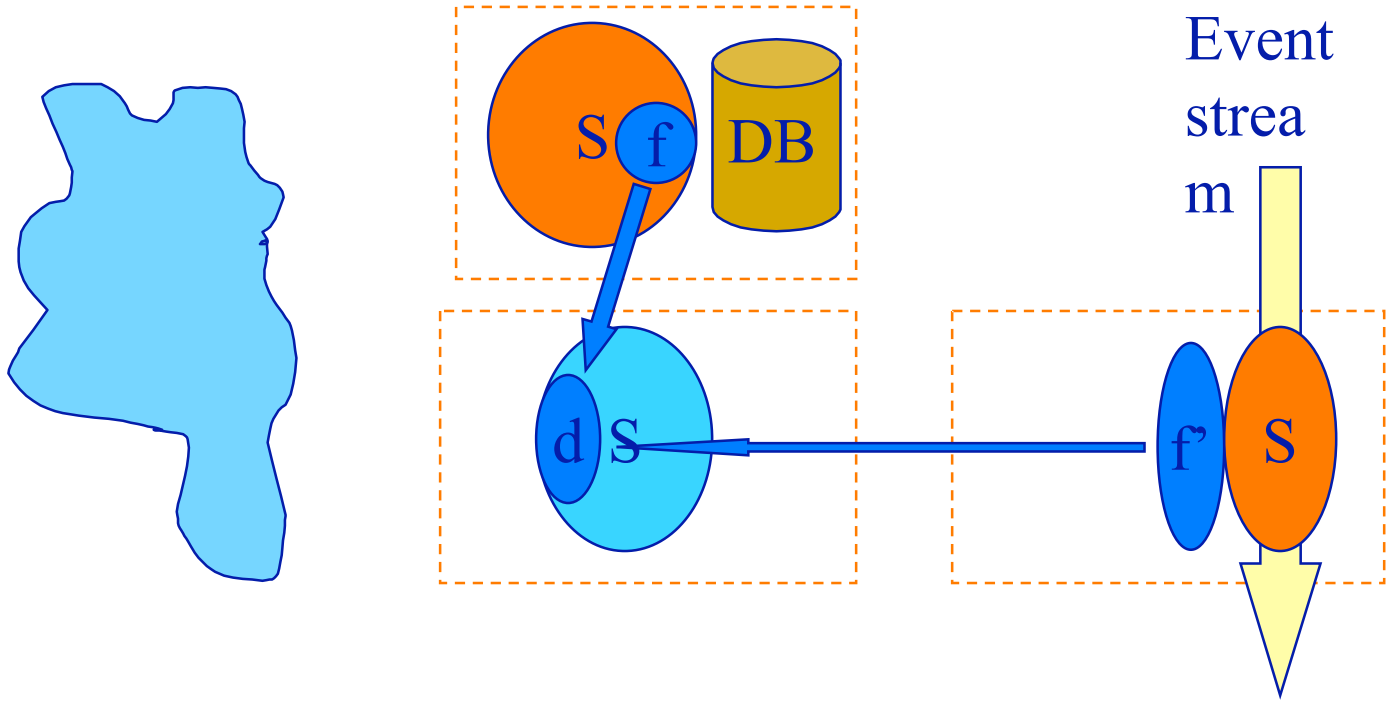
```
● subID subscribe   (waifID, taddr, params)
● subID unsubscribe (waifID, subID)
●       dispatch    (waifID, subID, event)

● Example params: { 'threshold' ='updates',
                   'datatype'  ='busroute' }

● Example event: {'weather':{ 'temp':85,
                              'wind':1.9 },
                 'busroute':{'busnr': 20,
                             'time': 08:23 }
                 }
```

3. WAIF Service Implementations

- Operational:
  - Bus route service
  - StormCast weather (http://weather.cs.uit.no)
  - Time alerts
  - Custom filter server
- In progress:
  - Concerts and events
  - RSS news feed

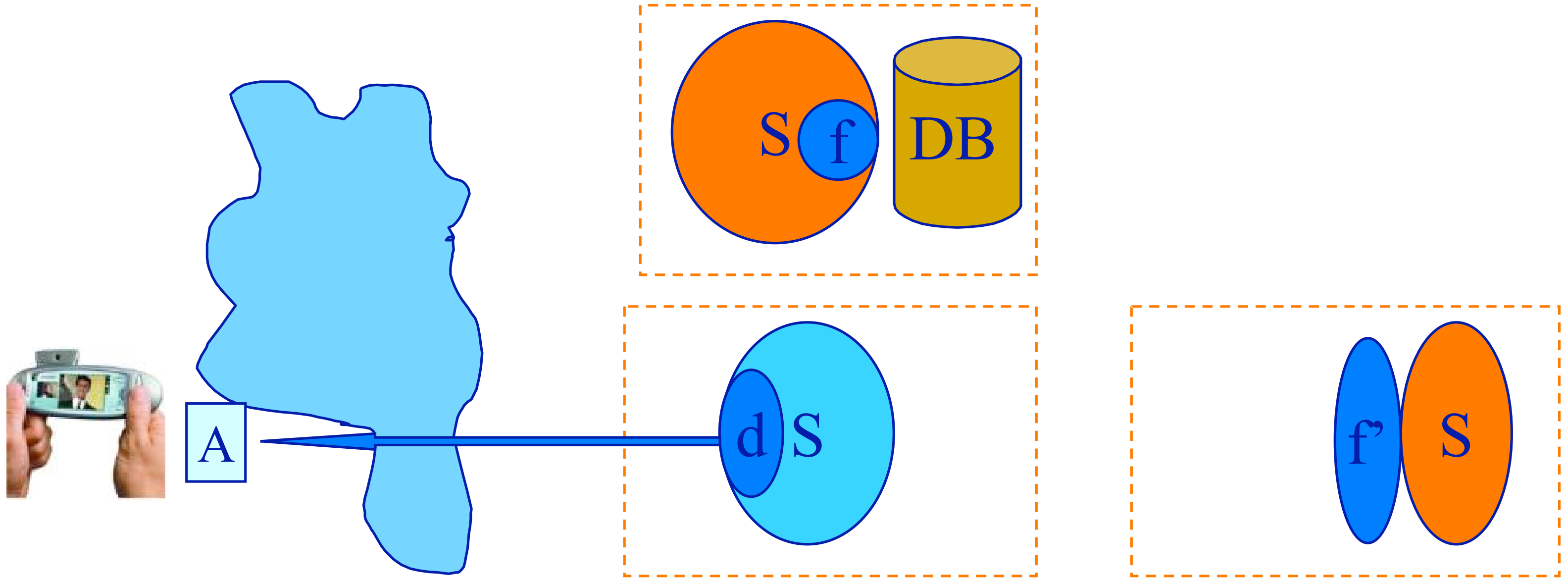
4. Lessons Learned

- Apply personalized filters on streams of real-life events.



4. Lessons Learned

- High expressiveness gives high-precision alerts.

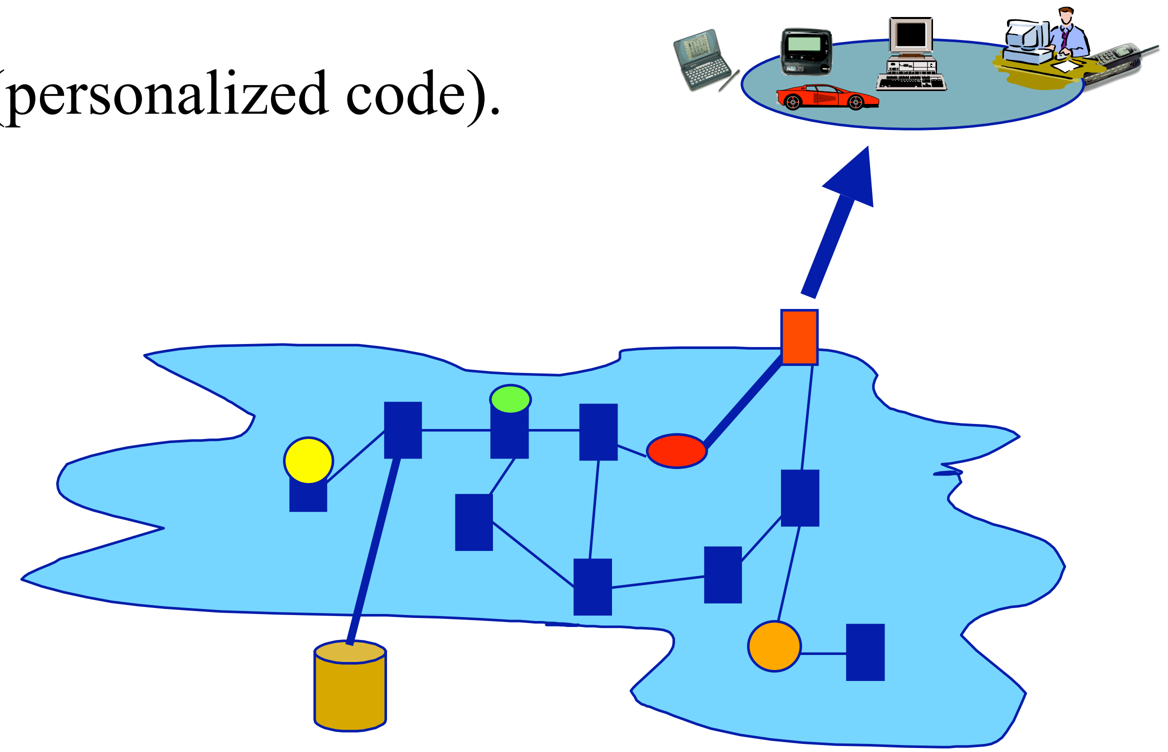


4. Related work

- Web services (Microsoft, IBM, BEA).
- Haystack and Oxygen (MIT).
- Oceanstore (Berkeley).
- Spinglass (Cornell).
- Semantic Web (W3C).
- Autonomic Computing Initiative (IBM).
- Pervasive computing, Pastry (DHT), Scribe (Microsoft Research).
- Aura (Carnegie Mellon University).
- Project JXTA (Sun)

4. Concluding Remarks

- Next generation Internet:
  - Pervasive.
  - Extensible (personalized code).
  - Push based.



http://www.waif.cs.uit.no

Questions?