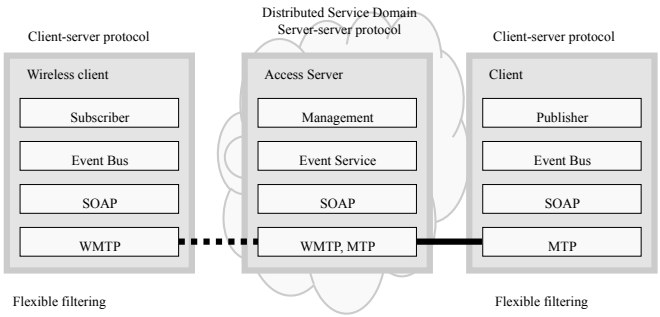
Fuego Core

Event-based systems: Rendezvous-Notify

Sasu Tarkoma

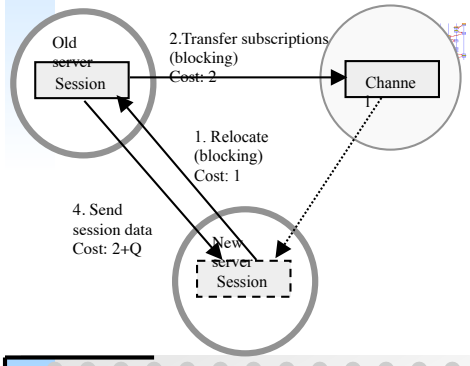
Event-based Systems: Rendezvous-Notify

- Scalable distributed event framework for mobile computing based on a distributed data structure
- Addresses terminal and user mobility issues
- Mobility requires moving subscriptions from one server to another and that events are stored for disconnected clients
- Subscription management operations need to efficient and propagate fast in order to ensure timely handovers
- Support for disconnected operation and mobility. Efficient event session handover between event servers
- Cost model for accessing event servers and using sessions. Simulation and formal verification is used to validate the proposed approach



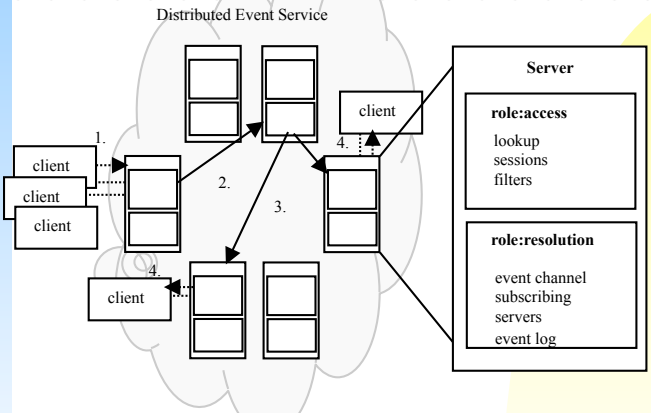
Handover Protocol

- Move session from old access server to the new access server
- Update event channel subscription information and buffer events
- Similar to the Wireless CORBA forward handoff



Architecture

- Two server roles: access servers, which are responsible for event sessions, and resolution servers, which manage event channels
- Event channels are rendezvous points for subscriptions and may be connected, for example, in hierarchies
- Event channel maintains subscription tables for access servers (indirection), assigns sequence numbers for channel-specific total order, and performs authentication and logging
- Constant or near constant cost in terms of messages for event channel subscription and management using linear hashing
- Different mechanisms for efficient lookup: directory lookup, linear hashing on event type, LH*, peer-to-peer algorithms (SCRIBE)



(1) A client publishes an event to the access server using the client-server protocol,

(2) the message is forwarded to the resolution server responsible for the event channel,

(3) the server filters (filter phase I) and a notification is multicast to interested access servers,

(4) access servers then associate the notification with an active client session (filter phase II), and notify the client or buffer the notification.

Flexible filtering

- Distribution of filters
- Custom XML <name,type,value> tuple-based filtering language
- Frequent publishers require client-side filtering
- Adaptive filtering strategies between clients and servers
- Pattern matcher, language for complex temporal pattern detection filters

Contact: sasu.tarkoma@hiit.fi

Web: www.hiit.fi/fuego

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY
TEKES
NOKIA CONNECTING PEOPLE
sonera
elisa
ERICSSON
MoreMagic
CREANOR