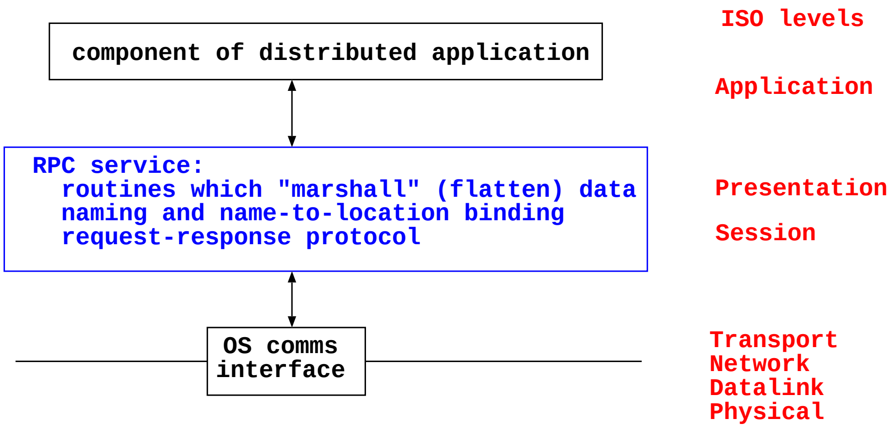
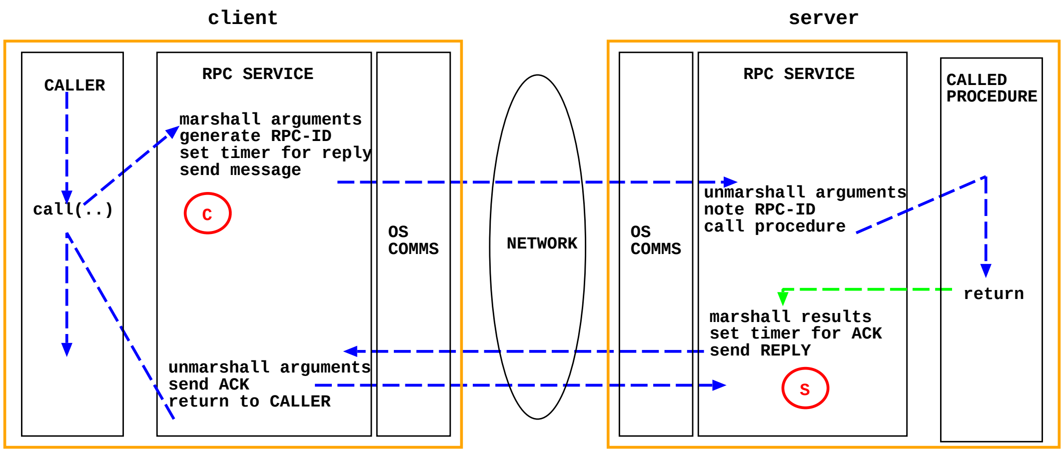
Remote Procedure Call (RPC)



examples: Mayflower/CCLU RPC, SUN RPC, ANSA RPC, MSRPC
Xerox Courier over XNS (SPP, Ethernet)
ISO-ODP, OSF DCE

RPC Request-Reply Acknowledge (RRA) protocol



C client timer

S server times

RPC semantics

recall that client, server and network may be congested or may fail independently of each other

RPC systems may offer AT MOST ONCE or EXACTLY ONCE semantics

C if the client timer expires:

AT MOST ONCE semantics:

exception return to the application
it is likely to repeat the call but this is not detectable
i.e. it will have a new RPC-ID

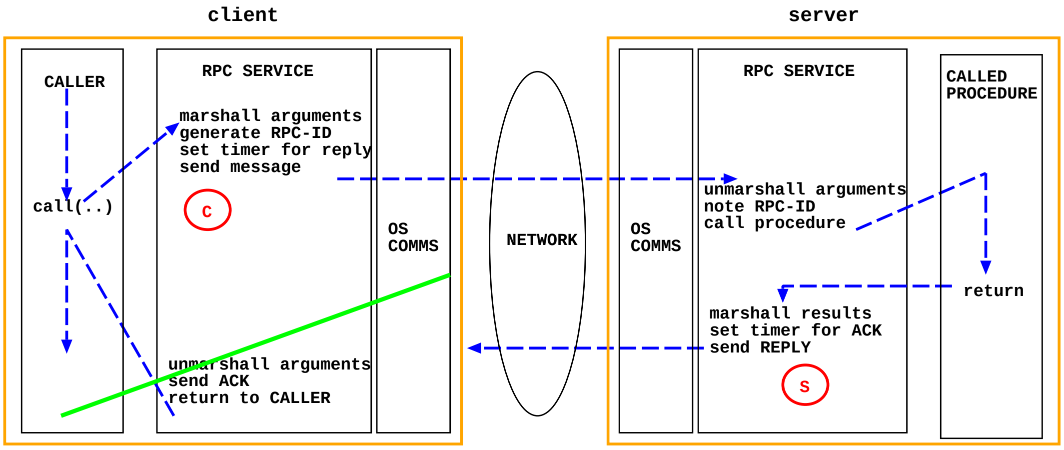
EXACTLY ONCE semantics:

retry a few times
RPC-ID means that the server can detect repeats
if no reply, exeption return to client

S if the server timer expires:

resend reults
RPC-ID means that the client can detect repeats

RPC client crash

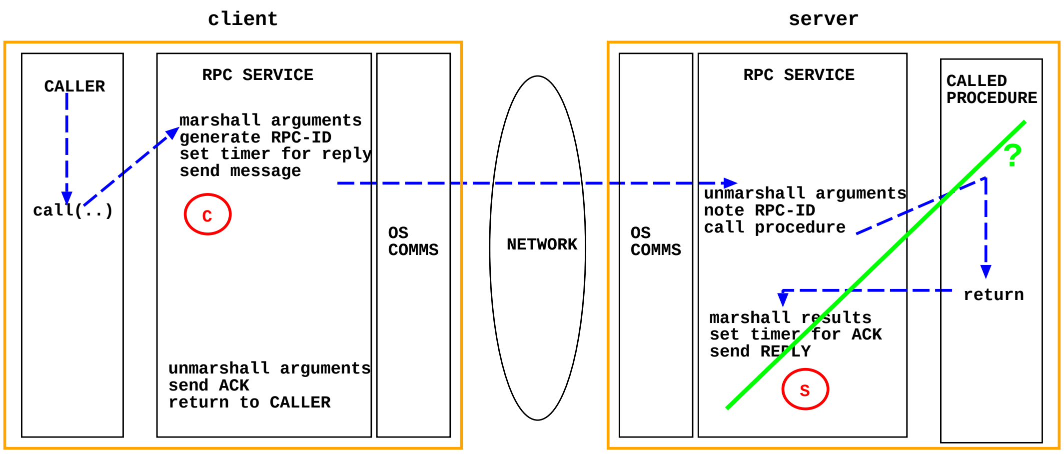


results are sent to crashed machine, are not acknowledged, S
and server timer expires repeatedly on resend

persistent state may have been changed by the procedure call
- should this be handled by RPC service?

NO - application-level transaction semantics (commit/abort) should be used.

RPC server crash



the server fails at some stage during the call
results are not sent and the client timer expires repeatedly C

persistent state may or may not have been changed by the procedure call
 - should this be handled by RPC service?

NO - application-level transaction semantics (commit/abort)
should be used.

Integration of Programming Languages and RPC (1)

* some early RPC systems aimed for complete distribution transparency

e.g. Xeroc PARC, Mesa language, Courier RPC

a preprocessor detects which calls are not to local procedures
and replaces them by calls to RPC support

problem of incorrect procedure names that don't exist anywhere
problem of call semantics for some arguments

* Cambridge Mayflower system, CCLU RPC - made distribution explicit

the compiler was changed
different syntax for definition and call of procedures
that can be called remotely

BUT - this was still for a single language, CCLU

some RPC systems restricted the argument types
e.g. SUN RPC: C base-types only

CCLU RPC: most types including procedure names defined as remote
user supplies marshalling and unmarshalling routines
for constructed types (recursive descent)

Integration of Programming Languages and RPC (2)

* ANSA RPC, was initially developed for C
  but later also supported C++ and Modula3 - a very early heterogeneous system
  - defined a Distributed Programming Language (DPL)
  - DPL statements are embedded in the programming language, and tagged
  - a preprocessor detects these statement, replaces them with calls to
    RPC service

All RPC systems automatically generate marshalling and unmarshalling routines
to flatten call and return arguments into packet format suitable for
transmission, and unpack them on receipt.
These routines are programming-language-specific.

Now assume that we wish to support a number of different programming languages,
i.e. components written in different languages can interoperate

* the standard approach (ANSA, ISO-ODP, OSF-DCE), O-O platforms
  - define an Interface Definition Language (IDL)
  - provide mappings for programming language's type systems onto IDL
  - (internally) define the transfer syntax for IDL types
  - IDL compilers generate marshalling and unmarshalling routines
    appropriate for the programming languages involved.

    (CORBA calls the invoker's marshalling routine a STUB
    and the invoked object's unmarshalling routine a SKELETON)

Integration of Programming Languages and Middleware

* how do platforms that support objects and object invocation
  differ from the RPC schemes described above?
  (as above for IDL and STUB/SKELETON generation)

RPC systems name and identify interfaces and procedures

e.g. ANSA IDL has base and constructed data types and the
InterfaceRef type, an instance of which is a reference
to a loaded and running instance of a service's interface

O-O systems name and invoke objects

Externally invocable objects must be registered with the platform,
an object-ID is returned (and may be recorded in a name service)
The object becomes known globally and may be invoked remotely

Object-IDs are first-class values which may be passed as arguments

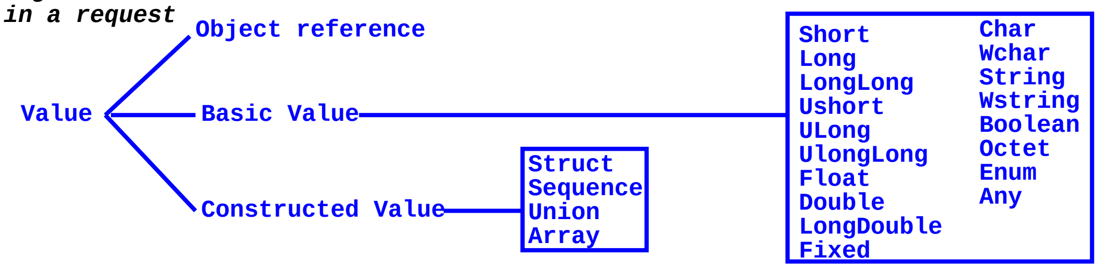
example: CORBA IDL

object type — members are object references

basic types
- 16, 32, 64 bit signed and iunsigned 2's complement integers
- single (32), double and double-extended floating point
- fixed-point decimal
- characters
- boolean
- 8-bit opaque, NOT converted on transfer between systems
- enumerated types
- string
- any (container)
- wide characters and wide character strings

constructed types
- record (struct) ordered set of (name,value) pairs
- discriminated union
- sequence
- array
- interface type - specifies the set of operations which an instance of that type must support

legal values in a request



CORBA http:www.corba.org/ some notes:

Programming language object refs and IDL object refs are NOT interchangeable.

The application designer must decide which objects are to be externally invocable and where they are to be located.

Such objects are registered with the ORB and stubs/skeletons are generated.

In versions of CORBA up to 2.2 such objects could not move.
Any changes required led to application recoding.

Recent versions have defined objects-by-value (OBV)

(In Java RMI all parameter-passing is by copy (like CCLU))

Copying of objects leads to more semantic problems
e.g. consistency and integrity of references.

Mobility is only supported at a coarse grain - of services

Many problems remain.....

CORBA systems   http:www.corba.org/ ...../vendors

comprise an ORB, system services and domain services

free: e.g. OmniORB (AT&T), Orbacus

on licence: e.g. Orbix (IONA)- ex Esprit Comandos project
.... large number

research: e.g. TAO (real-time) Univ Washington
e.g. reflective TAO - Univ Illinois Roy Campbell
e.g. OpenORB reflective - Univ Lancaster

ref: http://computer.org/dsonline/ ..../middleware/MWprojects.htm

Where does XML fit in?   http://www.w3.org/XML

SGML - standard generalised markup language
1980's (earlier?) document standard

XML - document standard (W3 consortium) compatible with SGML
DTD - document type description
- tag types - graph-structured document

XSL - style sheets indicate how to display the document e.g. in HTML
HTML - hypertext markup language
embedded tags are about how to display

XML is becoming widely used as a transfer syntax
- for documents - as expected
- for general typed messages
(all types reduced to strings - external form)

SOAP - simple object access protocol
object invocation defined with call and return arguments as XML types

wide interest for message oriented and database access middleware

Cambridge Event Architecture (CEA)

- compatible with any style of middleware

- use standard data typing for named, parametrised events
  e.g. IDL -> ODL, XML?

- event sources publish the events they will notify

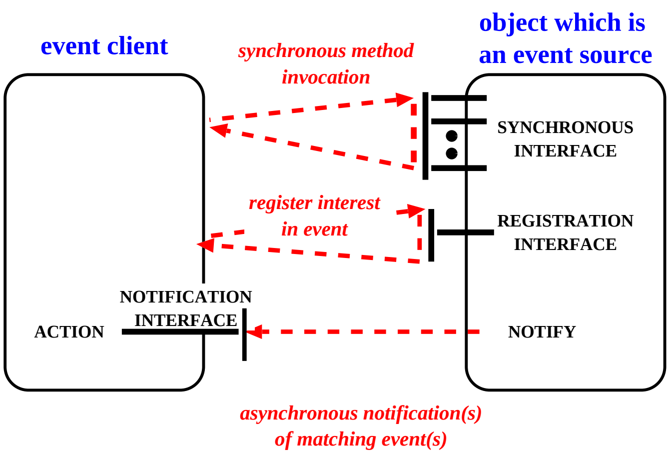
- clients register interest in events with sources
  indicating parameter values or wildcards

- sources notify clients with the stream of matching events

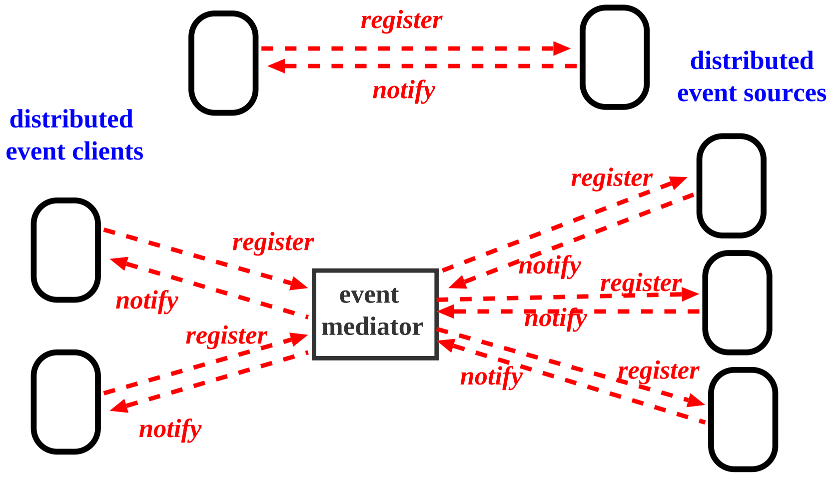
- event stores can be clients e.g. to log events
  note compatible transmission and storage technology

Cambridge Event Architecture (CEA)

CEA 1. The *publish-register-notify* paradigm



CEA 2. Direct and mediated notification



CEA 2

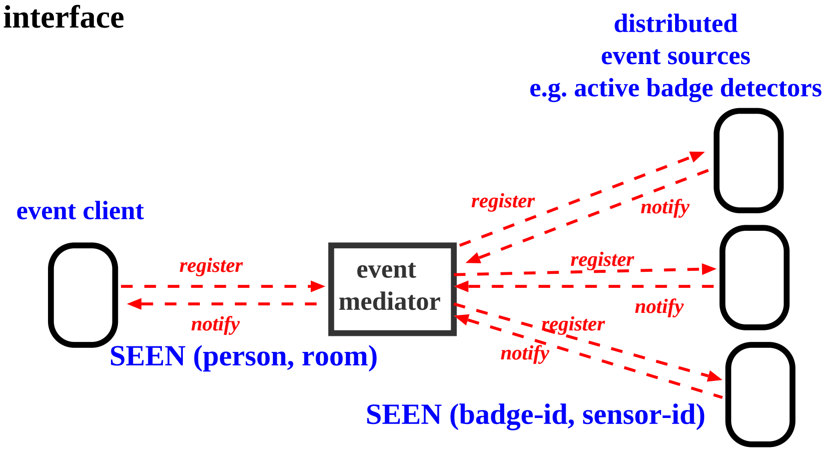
* avoid overload on primitive event sources

* decouple event source and client

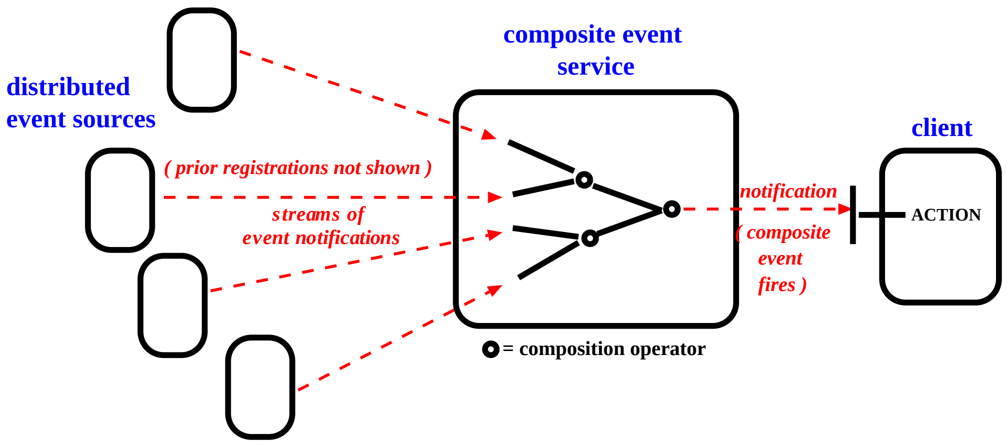
* one-to-many and many-to-many communication

  - multicast protocol may be exploited
    at event source or mediator

* mediated communication can be used to provide
  a higher-level interface



CEA 3. Event Composition - Composite event detection



CEA3 event composition operators

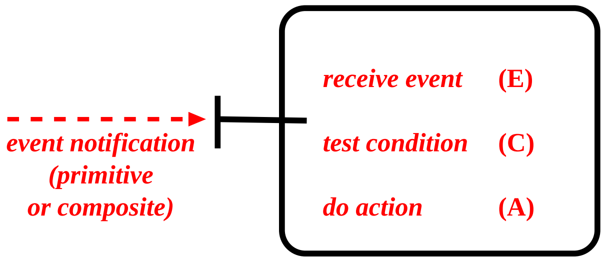
Without	**A - B**	*yields stream matching A until B occurs*
Sequence	**A ; B**	*yields stream matching B after A occurs*
Or	**A | B**	*yields stream matched by A or B*
And	**A&B**	*yields stream matched by both A and B*
First	**First(A)**	*yields the first event that matches A*

- need to be tested in practical applications

- precise meaning? .... consumption policy?

CEA 4. Active programming: *event-condition-action*

(Component composition)



event composition embodies some conditions

the receiver may also impose conditions

*** events are the glue for composing distributed software components**

- **active office, home, airport, city (sensor-rich environments)**
- **virtual reality, augmented reality**

Distributed Implementation:

any component or connection may fail at any time

Uncertainty:

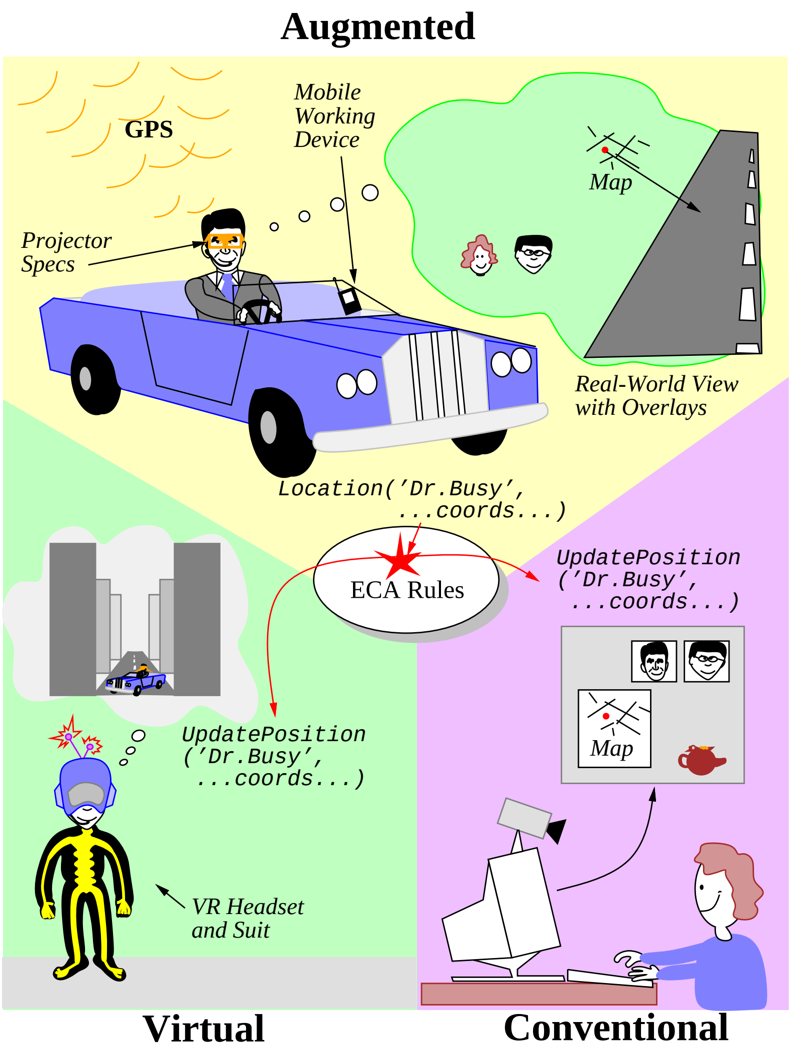
either - the alarm has not been sent, all is well

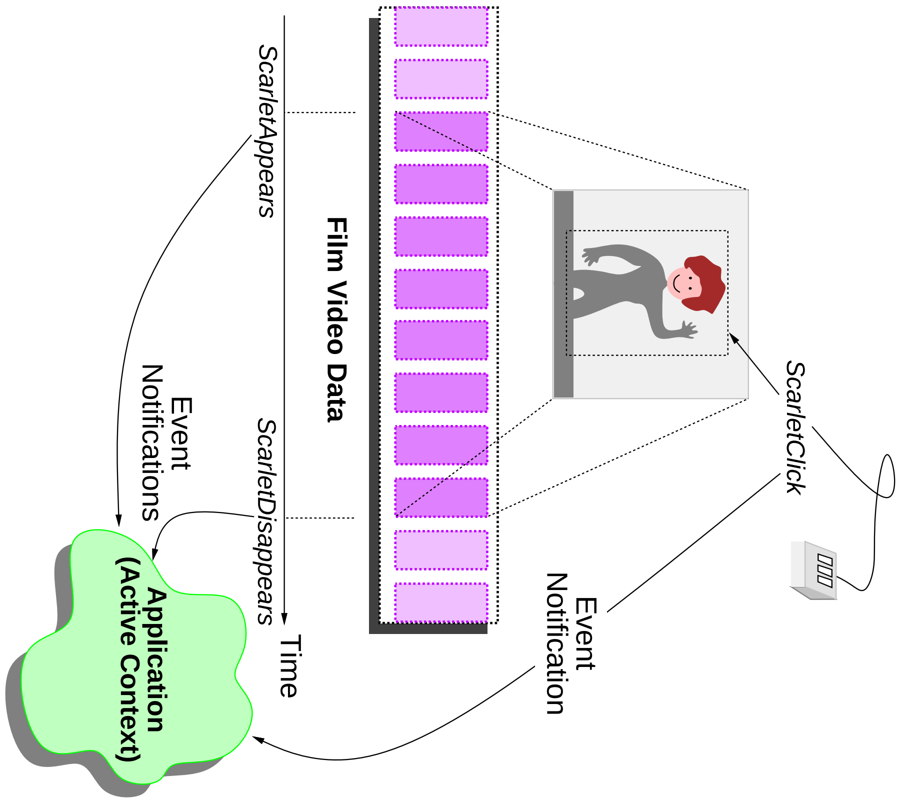
or - the component that raises the alarm has failed, the crisis may or may not have occurred

Solution - use a heartbeat protocol for failure tolerance

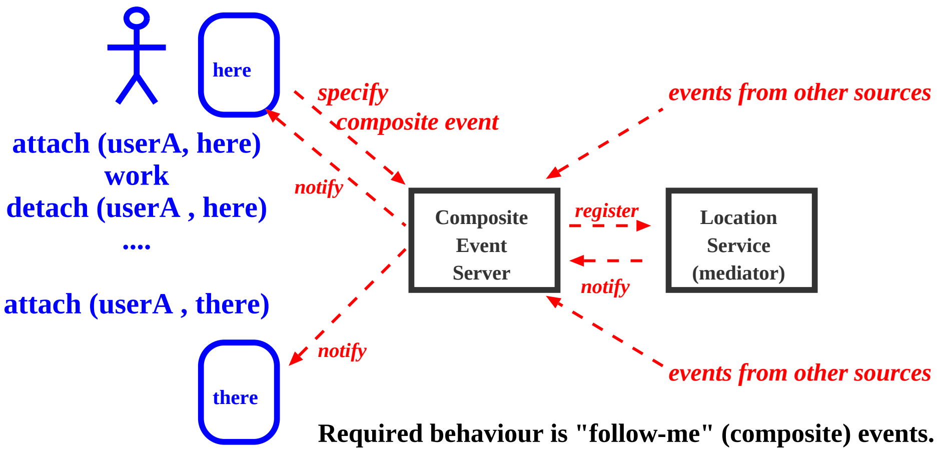
heartbeat rate
is tuneable pairwise between event sources and clients

client policy
for non-occurrence of either event or heartbeat


Augmented
GPS
Mobile
Working
Device
Map
Projector
Specs
Real-World View
with Overlays
Location('Dr.Busy',
...coords...)
ECA Rules
UpdatePosition
('Dr.Busy',
...coords...)
UpdatePosition
('Dr.Busy',
...coords...)
Map
VR Headset
and Suit
Virtual
Conventional


ScarletAppears
Film Video Data
Event
Notifications
ScarletDisappears
Time
Application
(Active Context)
ScarletClick
Event
Notification

Examples: mobile user, fixed infrastructure
mobile user with mobile device



Required behaviour is "follow-me" (composite) events.
- specify as a composite event:

Deliver the stream of events matching this pattern to me
- to where I'm attached
- when I'm attached
If any occur while I'm detached, buffer them
and deliver them to me when I next attach

Fault detection in communications networks

what event patterns indicate which problems ?

- in real-time
- in event logs, e.g. fraud detection
- high volume

Active Badge (electronic tag) Technology, Sensor-rich Environments

e.g. active house, office, hotel, airport, city

?✓ locked doors will open

✓ the nearest computer will fetch your environment,
video streams, email, news, buffered events

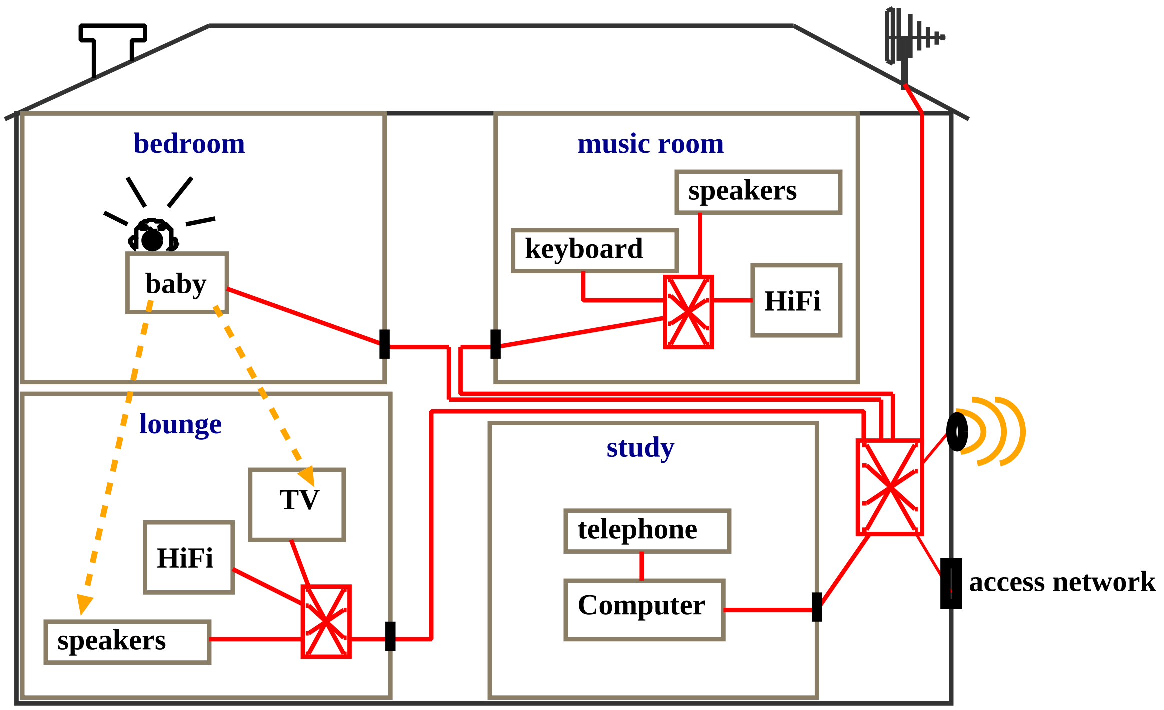
✓ equipment can be tagged for security - movement raises an alarm

✓ mobile objects can be tracked (buses, cars, taxis, ambulances)

? people can be tracked, meetings can be detected

-> access control needed on registration and notification

An Active Home



......but we can be monitored .....