3rd Joint Advanced Student School (JASS 2005)

Course 3 - Ubiquitous Tracking for Augmented Reality

Existing Architectures & Systems: OpenTracker, DWARF, VRPN, Trackd

Georgi Nachev
nachev@in.tum.de
Technische Universität München

Content

- Introduction
- Important goals and requirements
- OpenTracker
- DWARF
- VRPN
- Trackd
- Conclusion

Introduction

- Highly dependence of Augmented and Virtual Reality applications on accurate and precise tracking data.
- Combination of several trackers in order to minimize negative properties of one tracker by another – introducing sensor networks.
- Hardly reusable customized solutions to this problem.
- No standard interfaces between these technologies – inhibition of the development of large-scale sensor networks.
- Modular architecture in some current commercial VR systems – possibility of simultaneously application of different types of tracking systems and input devices. Limitation the extensibility and configuration options.
- Broad range of features in research systems. Limitation to their particular research domain and not for commercial use.

Important goals and requirements

- Requirement: Tools with high degree of customization, easy to use and to extend. Potentiality of features-mixing, simple creating and maintenance of complex tracker configurations.
- Device abstraction - fixed interface to the application for different devices and tracking systems.
- Support for distributed simulation – simultaneous users or better exploit of the available hardware. Multithreading, symmetric multiprocessing on the host, or cluster-approach.
- Complex dependancies and interactions between devices and subsystems.
- Model of AR applications as distributed systems from mobile and stationary components.
- Runtime reconfiguration and recalibration.
- No single point of failure.

Existing Architectures & Systems

- OpenTracker - static dataflow model for streams of sensors readings.

- DWARF - a framework for component-based peer-to-peer systems.

- VRPN - a static network-transparent abstraction between applications and pre-defined trackers.

- Trackd - a commercial system from VRCO Inc.

OpenTracker

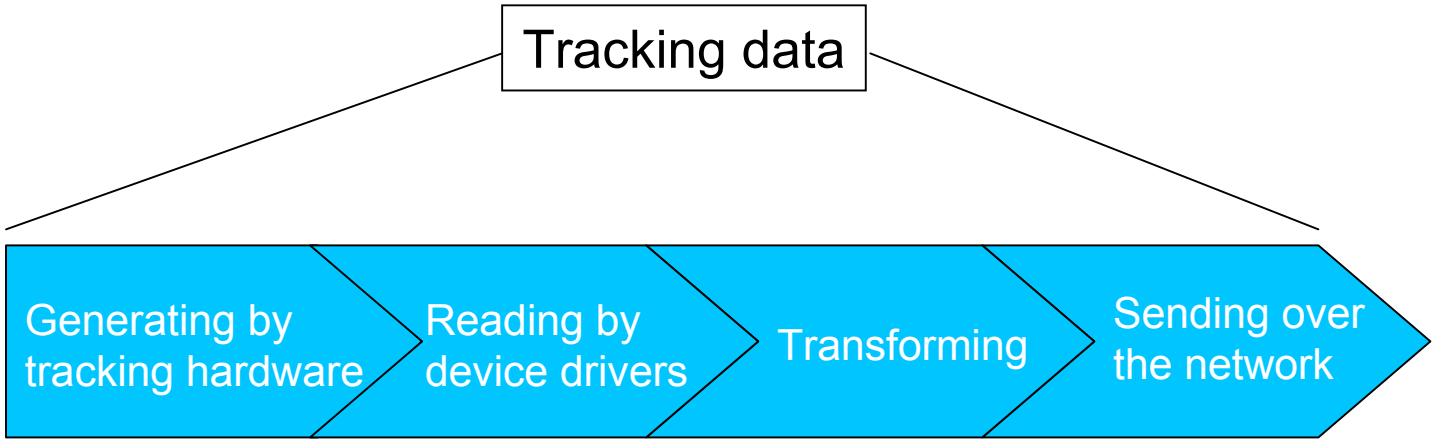
OpenTracker - Introduction

- Framework for the different tasks involved in tracking input devices and processing multi-modal input data in VR and AR applications.
- Eases the development and maintenance of hardware setups, using an object-oriented design based on XML.
- Based on a data flow concept for multi-modal events.
- Both a time-based and an event-based model, that can be used simultaneously, to serve a large range of applications.
- Extensive set of sensor access, filtering, fusion, and state transformation operations.
- Behavior specification by constructing graphs of tracking objects from user defined tracker configuration files.

OpenTracker - Advantages

- Encourages exploratory construction of complex tracking setups.
- End users can fully exploit their hardware without any custom programming.
- Developers can easily build test environments.
- Distributed and decoupled simulation by network transfer of event at any point in the graph structure.

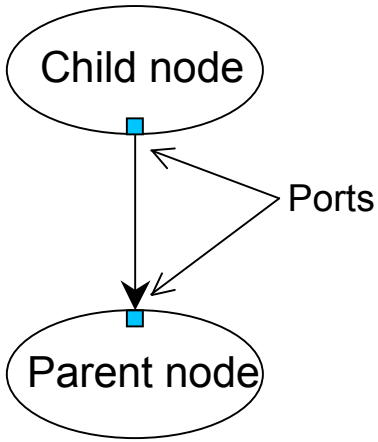
OpenTracker – Main Concept

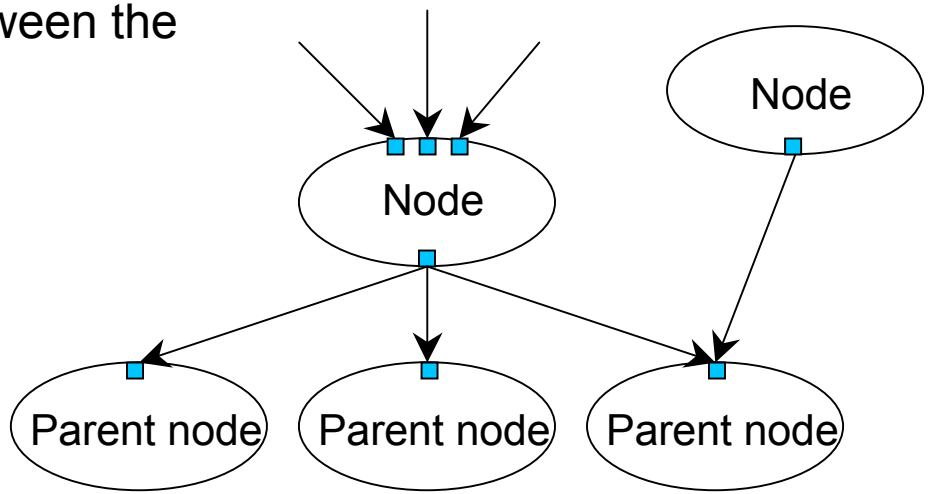


- Different subsets and combinations of the steps, but common individual steps among a wide range of applications.
- Break up the whole data manipulation into these individual steps and build a data flow network of the transformations.
  - Transformations as a nodes in a data flow graph.
  - Directed edges describe the direction of flow.

OpenTracker – Nodes, Ports and Edges

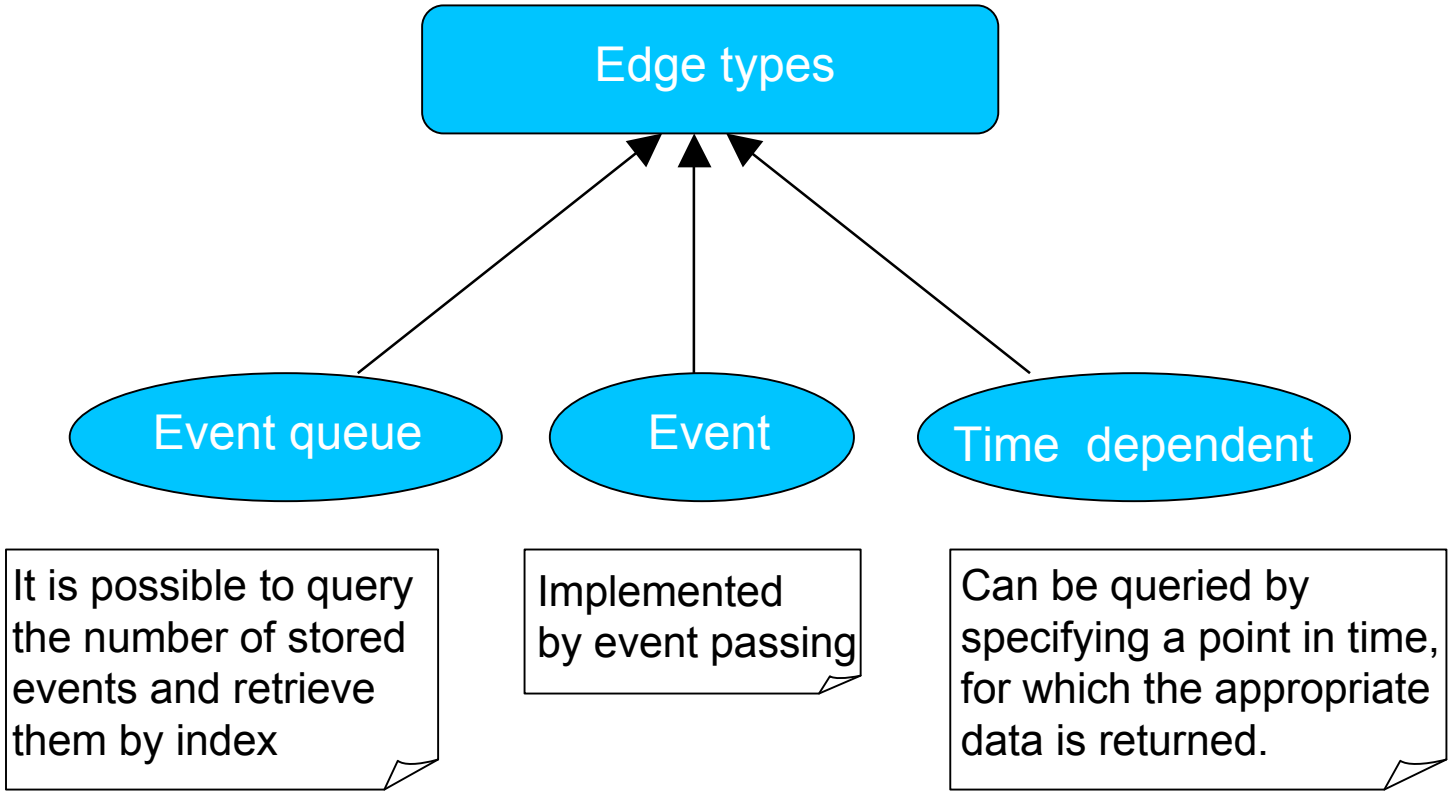
- Child node as the origin of the edge and parent node as the endpoint of the edge.
- Port - distinguish connection point for an edge.
- One or more input ports and a single output port
- Possibility to connect:
  - the output port to many input ports of a different nodes.
  - several output ports to the same input port, (no distinguishing between the actual children).





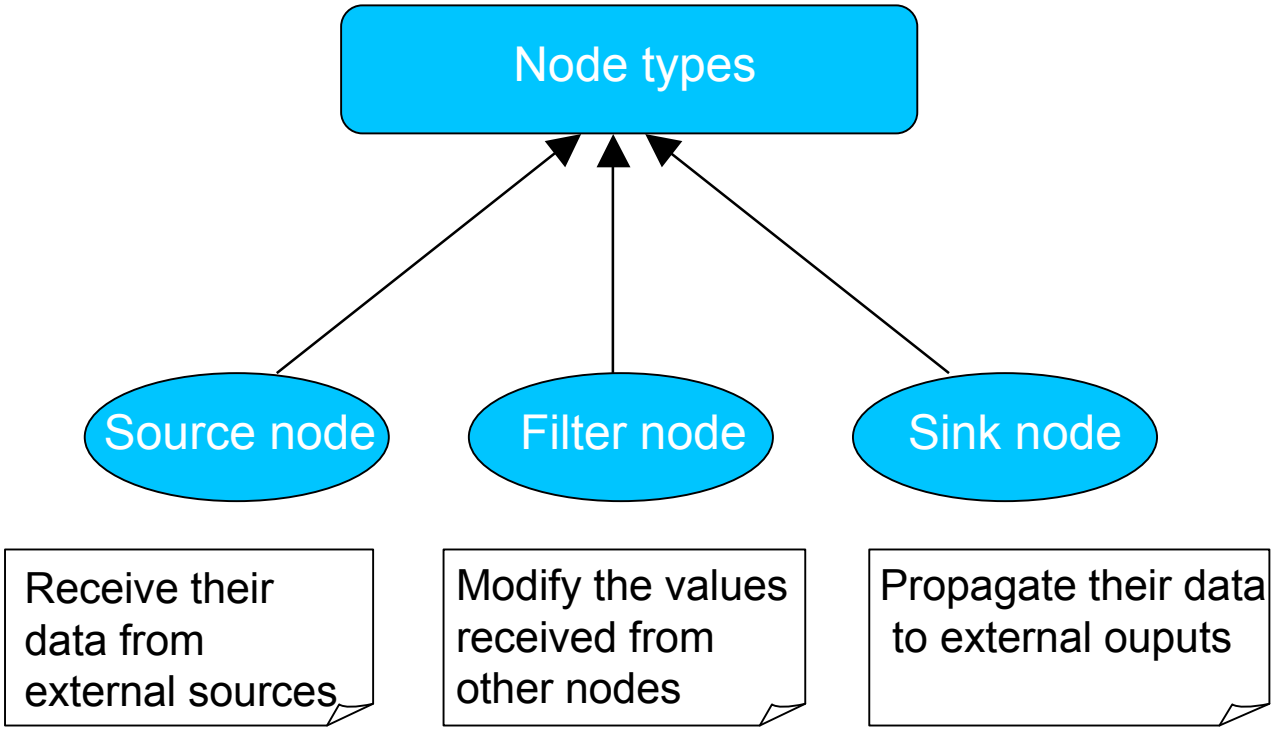
OpenTracker – Edge Types

- Different edge types are distinguished - typed by typing the ports of the nodes they connect.
- Only two ports of the same type can be connected and this type is equal to the type of the edge.



OpenTracker – Node Types(1)

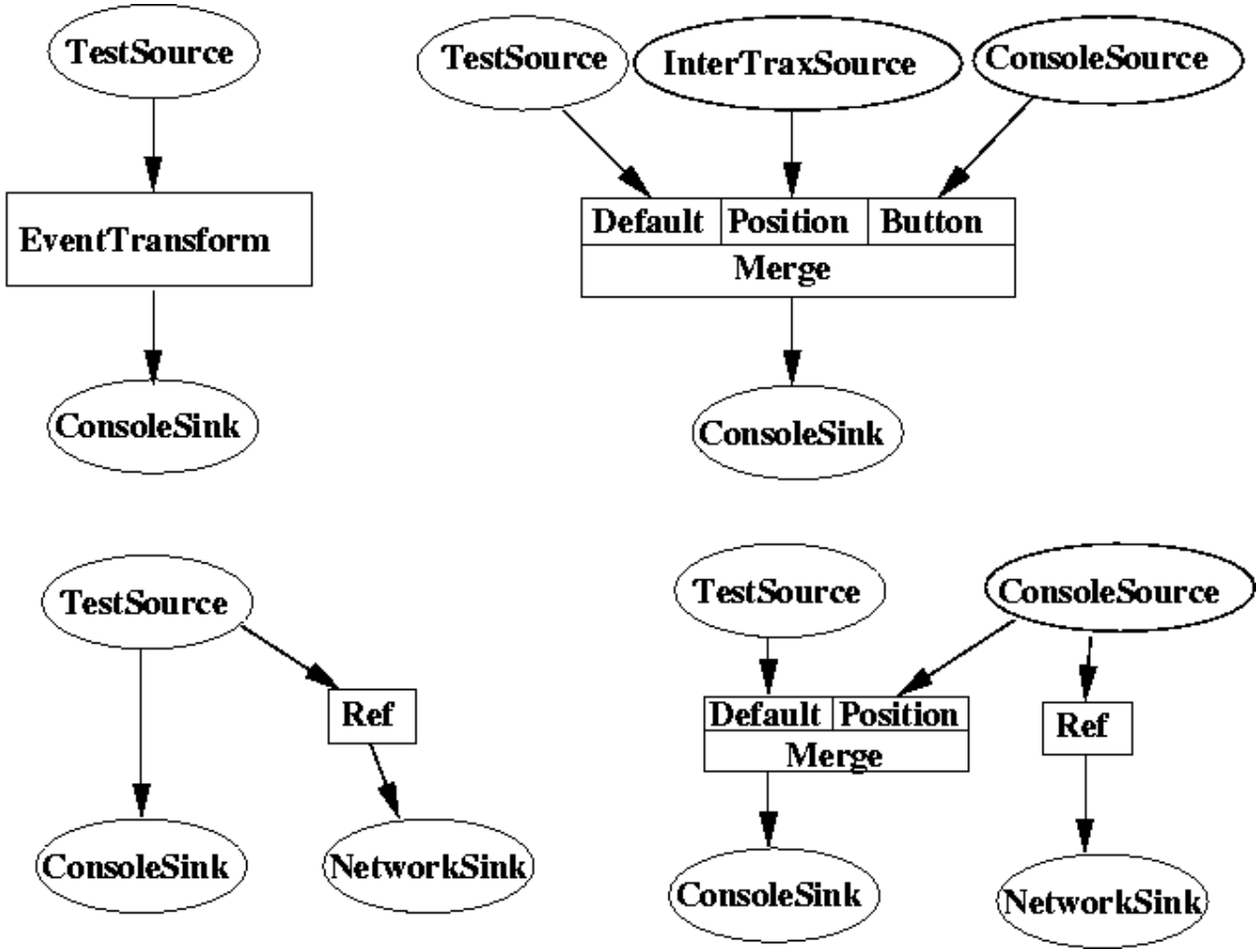
- Three types of nodes:



OpenTracker – Node Types(2)

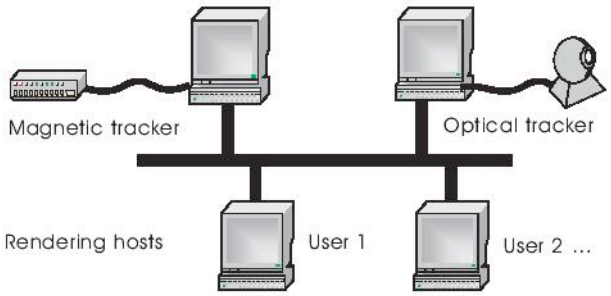
- Source nodes - device driver, tracker emulation via the keyboard, access network data, or simply responds with constant values.
- Filter nodes - receive data from one or more child nodes and compute their own state based on the collect data. Geometric transformation of the children's value, prediction to compensate for measuring lag, merging different parts of data values, conversion of one data type into another, etc.
- Sink nodes - output to network multicast groups, debugging output to an user interface, or threadsafe shared memory.

OpenTracker – Examples of data flow graphs



OpenTracker – Shared Data

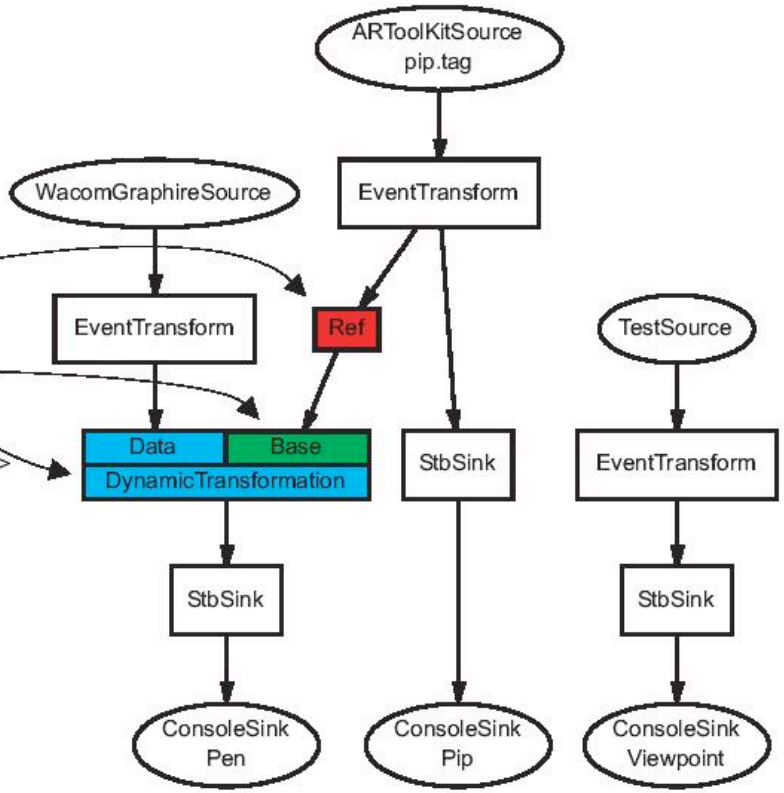
- Network support:
  - makes it easy to span multiple operating systems(if a specific tracking device or service is only available at one particular host).
  - achieves some degree of load balancing. Multi-processing based on inexpensive PCs becomes possible with little configuration effort.
  - makes the implementation of distributed virtual environments possible.
- Multiple senders and receivers of tracker data communicate asynchronously and independent through the network using IP multicasting(decoupled simulation).



- Single host as a sender and receiver at the same time.

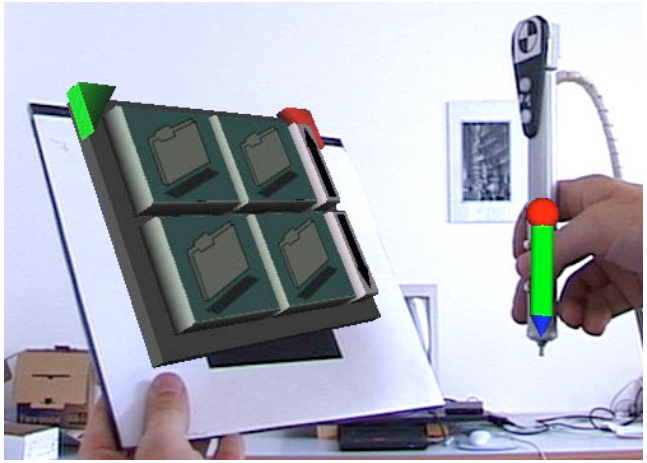
OpenTracker – Example configfile

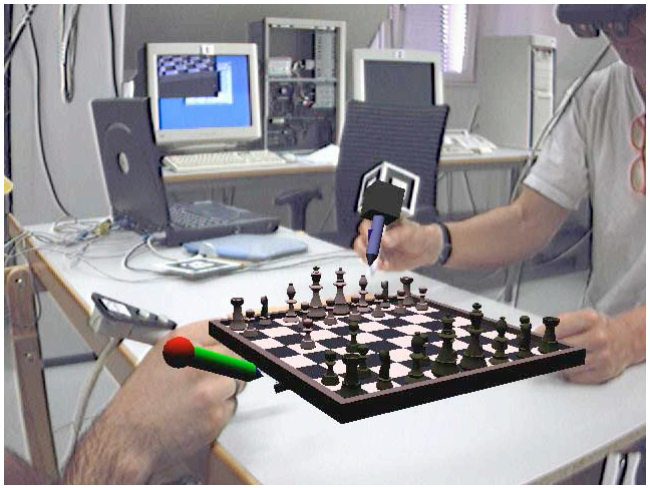
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OpenTracker SYSTEM "opentracker.dtd">
<OpenTracker>
    <configuration>
        <ARToolKitConfig camera-parameter="camera_para.dat"/>
    </configuration>
    <ConsoleSink comment="Pip">
        <StbSink station="0">
            <EventTransform DEF="Camera" scale="0.001 0.001 0.001">
                <ARToolKitSource tag-file="pip.tag" />
            </EventTransform>
        </StbSink>
    </ConsoleSink>
    <ConsoleSink comment="Pen">
        <StbSink station="1">
            <EventDynamicTransform>
                <TransformBase>
                    <Ref USE="Camera"/>
                </TransformBase>
                <EventTransform scale="-2.1 -2 0" translation="0.14 0.1 -0.01">
                    <WacomGraphireSource device="1"/>
                </EventTransform>
            </EventDynamicTransform>
        </StbSink>
    </ConsoleSink>
    <ConsoleSink comment="Viewpoint">
        <StbSink station="2">
            <EventTransform rotation="1 0 0 0">
                <TestSource frequency="25"/>
            </EventTransform>
        </StbSink>
    </ConsoleSink>
</OpenTracker>
```



OpenTracker – Example Integrating

- Studierstube Augmented Reality Project‘s „Pen-and-pad interface“
- Vision tracking approach(ARToolkit) for the pad and a magnetic tracker(Ascension Flock of Birds) for the pen are combined.
- Two separate servers for video and magnetic tracking send their measurment over the network to a rendering host, where the combined data is picked up by an OpenTracker component.





DWARF

DWARF – Introduction

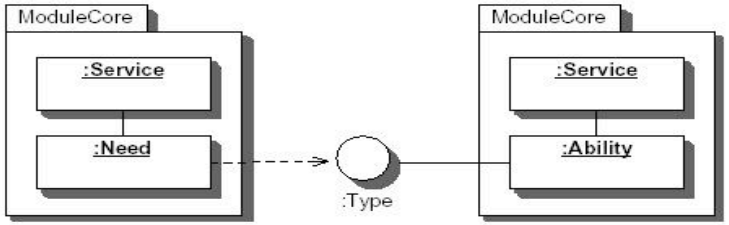
- DWARF stands for Distributed Wearable Augmented Reality Framework.
- Framework for component-based peer-to-peer systems.
- AR applications - modeled as distributed systems which form spontaneously from mobile and stationary components.
- Reconfiguration itself at runtime by exchanging components and changing component configurations.
- Network of collaborating distributed services, which expose their requirements, called Needs, and offers called Abilities.
- No central management-component, but a service manager on each network node - control their local services, maintain descriptions of them and set up connections over the network to other services.
- Service description, written in XML with Abilities, Needs and communication protocols, called cannectors.

DWARF - Advantages

- Platform independance and heterogeneity
- Modularity and Distribution
- Easily integration of third-party hard- and software
- Remote Monitoring

DWARF – Abilities/Needs

- Both the Abilities and the Needs have a set of attributes describing the quality of service parameters of the service, that are offered respectively expect.
- The service manager select abilities, that can provide a sufficient quality of service to satisfy a given need, or to ensure at runtime, that the desired quality of service is still provided.

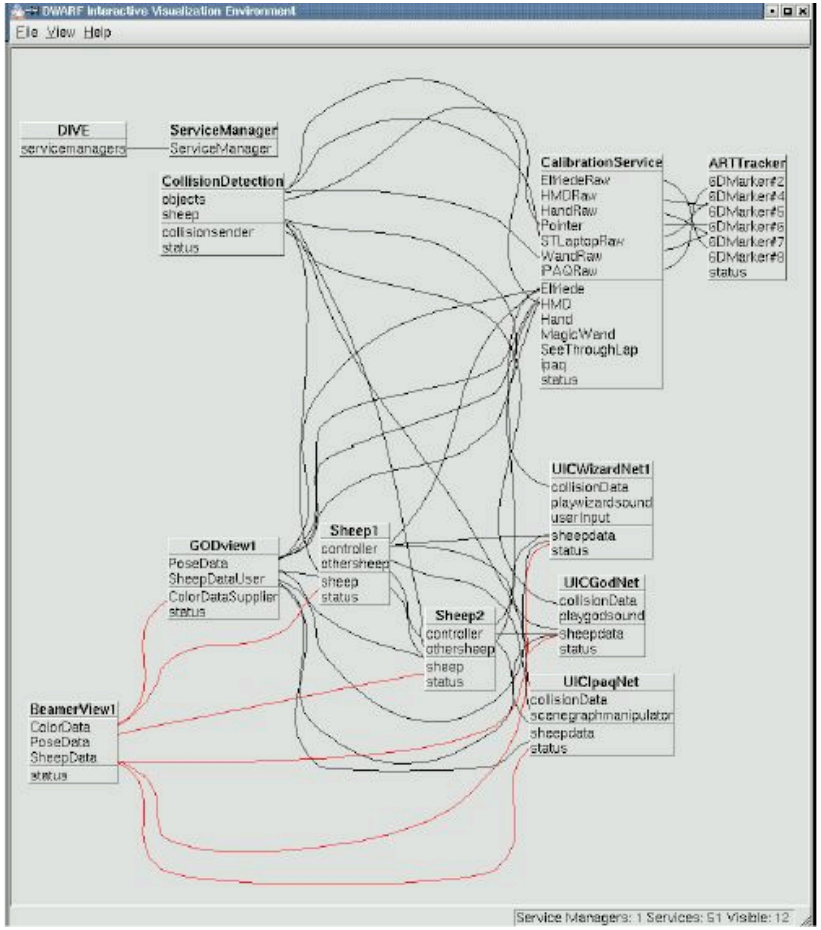


DWARF - Subsystems

- Services grouped in subsystems, which are responsible for specific tasks. Some important of their are:
- Presentation Subsystem - provides an Viewer where models can be loaded and registered with Needs for position data. Based on the OpenInventor implementation Coin by Systems in Motion.
- Tracking Subsistem - any tracking hardware is encapsulated in appropriate services, also a calibration service for tuning purposes are provided.
- Application Subsystem - the place where the application developer can put its own services.
- Input Subsystem - user interaction and input devices services have to be implemented here.

DWARF - Interactive Visualization Environment

- Graphical view of the network of interconnected services that dynamically changes with system configuration changes.

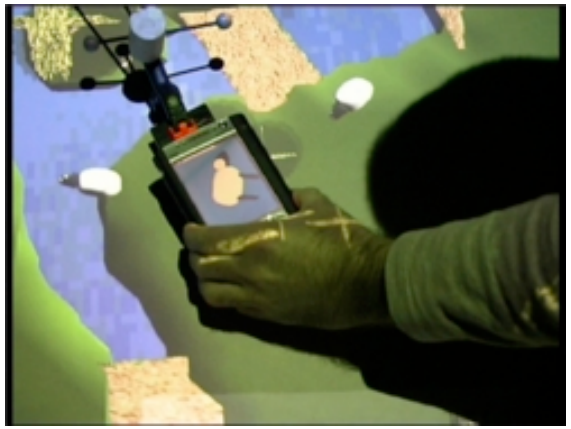


DWARF – SHEEP(1)

- The Shared Environment Entertainment Pasture(SHEEP) - an example project made by Augmented Reality Research Group using DWARF
- A multiplayer sheepherding game explore the possibilities of multimodal, multiuser interaction with wearable computing in an intelligent environment.

DWARF – SHEEP(2)

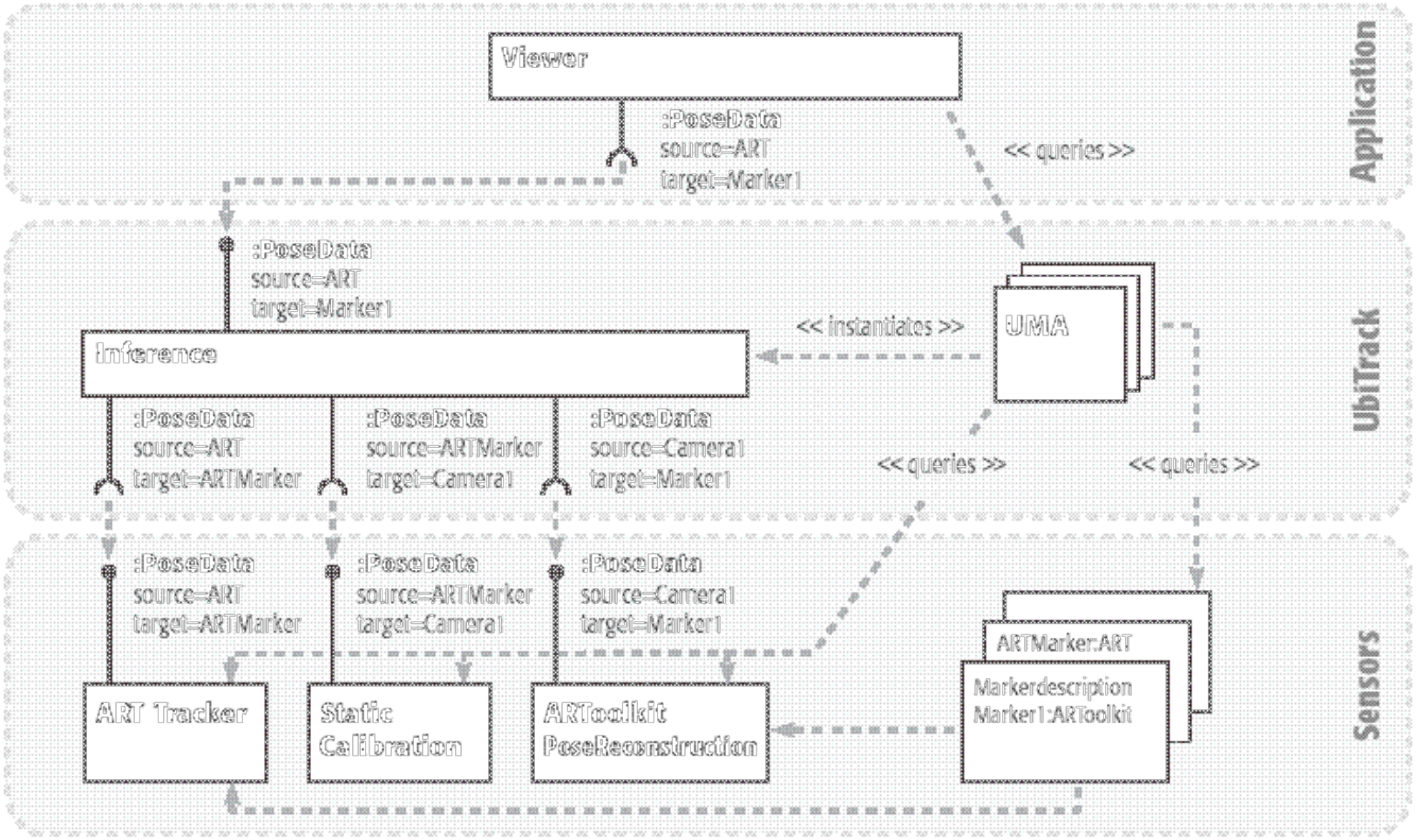
- The game is centered around a table with a beamerprojected pastoral landscape.
- Players can use different intuitive interaction technologies (beamer, screen, HMD, touchscreen, speech, gestures) offered by the mobile and stationary computers.
- Building on the DWARF framework, the system uses peer-to-peer, dynamically cooperating services to integrate different mobile devices into the game.



DWARF - Ubiquitous Tracking

- Ubitrack Middleware Agent(UMA), started on each host.
- Connection to the local ServiceManager - receiving information about all local objects and measurement between them.
- Possibility to compute and store all paths between arbitrary local objects together with the corresponding set of attributes.
- Grouping all local objects in a supernode with cached paths.
- New object in the tracking range of a local tracker is stored as an anonymous object until it is identified.
- UMA tries to connect to other UMAs which are in close range and requests for information about their local objects.
- If the anonymous object can be identified, a communication channel to its UMA is set up.
- The new object is now stored in both UMAs. If the object leaves the local tracking range, it is deleted from the local UMA as well as the communication channel.

DWARF - Ubitrack Middleware Agent



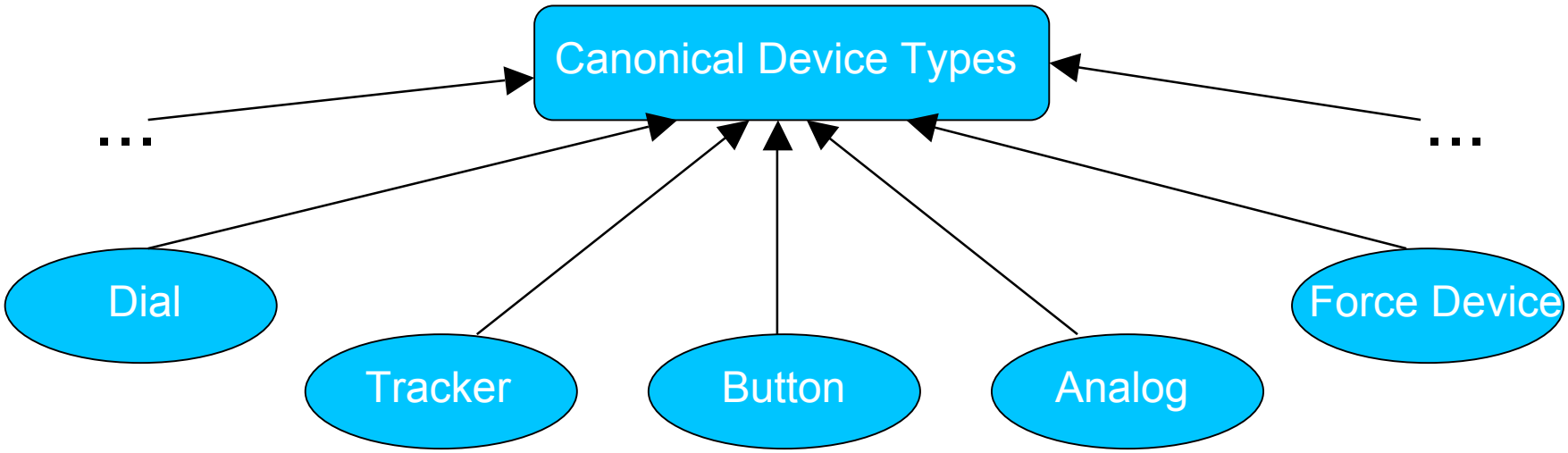
VRPN

VRPN - Introduction

- The Virtual Reality Peripheral Network(VRPN) provides:
- A device independent and network-transparent interface between applications and several physical devices and systems(e.g. Tracking) used in a VR-system.
- Multiple simultaneous connections to devices.
- Automatic reconnection to failed remote servers.
- Time stamps for all messages.
- Clock synchronization between client and server on different machines.
- Storage and replay of interactive sessions.

VRPN - Canonical device types

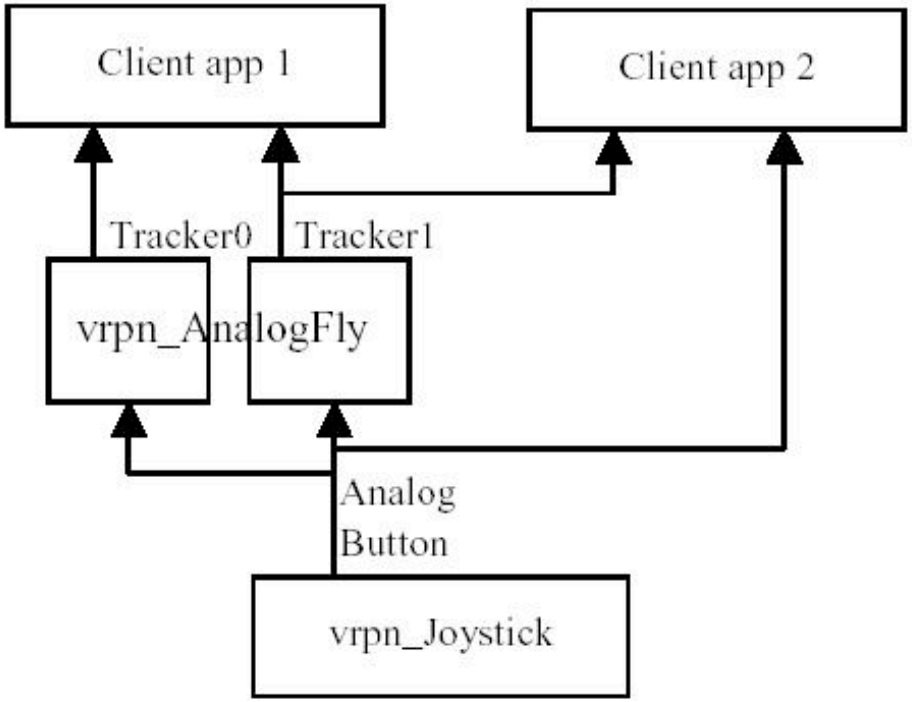
- Particular device is of one or more canonical device type and implements the interface.



VRPN - Features

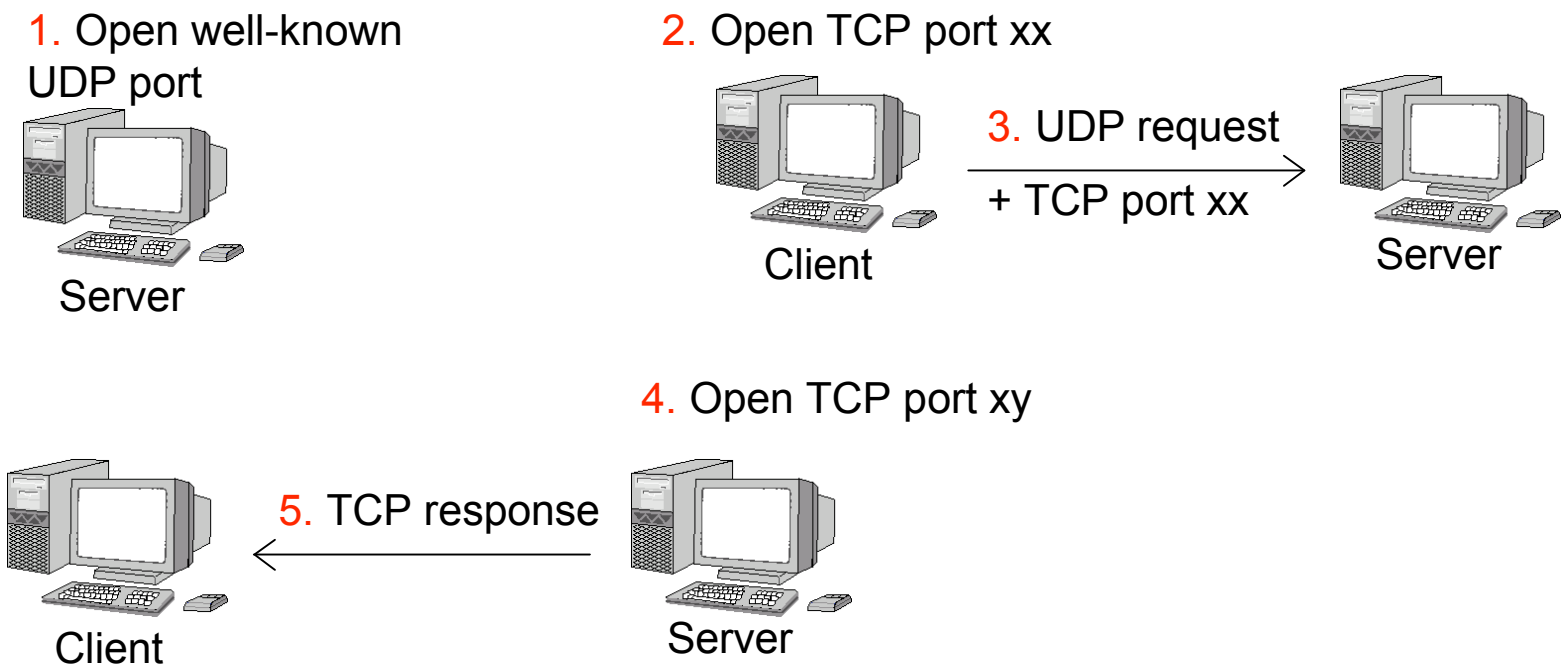
- Factoring devices based on their functions: Device with more than one function - interfaces for multiple device types are exported under the same device name.
- Mapping devices to connections: Communication efficiency through internally mapping of multiple interfaces for a single device to the same network connection.
- Enabling devices to export multiple interfaces: The same physical device may act as different device types at different times.
  - Both interfaces exported under different names.
  - A special case - layered device. A higher-lever behavior is built on top of an existing device. Application code can attach to either or both devices.

VRPN - Layered device



VRPN – Connection Establishing(1)

- Each device driver in VRPN may have a server- and client-side, wich communicate over a Connection object.
- No dependence on particular port on the server and on the client for this object.



VRPN – Connection Establishing(2)

- Once the reliable TCP chanel is established, it is used to establish a separate unreliable UDP chanel between the hosts - also for version checking and clock synchronization.
- In case of dropped connection - a message is sent to any interested objects indicating the drop.
- Attempt from the client to re-establish the link using the same algorithm - robust and very useful in cases of long start up times.

VRPN - Performance

- Performance of the system in comparison to a locally-connected device using device-specific driver.
- Well time-costs measuring values. Significantly less time to read a message using a remote VRPN server than that of a locally-connected device.
- Timing information and latency-reducing optimizations.
- Network latency tests between SGI and Linux box within a switched Ethernet with average one-way times of 3,3ms. Includes all overheads from the operating systems network layers.
- Slightly lower times from Linux client to a Windows 98 server.
- Average one-way times of 1,7ms between SGI client and Windows 98 server.

Trackd

Trackd – Introduction

- Comercial device software from VRCO Inc. for VR applications in the immersive display industry.
- Small deamon application that takes information from a variety of tracking and input devices and makes that information available for other applications to use.
- Used by all VRCO products and is available with most commercially installed systems.
- Possibility to share information from tracking and input devices to different graphics machines.
- Networking capabilities and support for a variety of operating systems.

Trackd – Design

- Introduction of Generic Application Programmers Interface(TrackdAPI).
- Allows applications to support a range of hardware without having to modify source code, or write specific hardware drivers.
- Changing, replacing or upgrading of components without having to replace their software, or lose functionality.
- Combination of two applications - Trackd Server and Trackd Daemon.
- In addition various modules, which are compiled as shared object libraries to support various devices. One module for each separate device.

Trackd – Server and Deamon

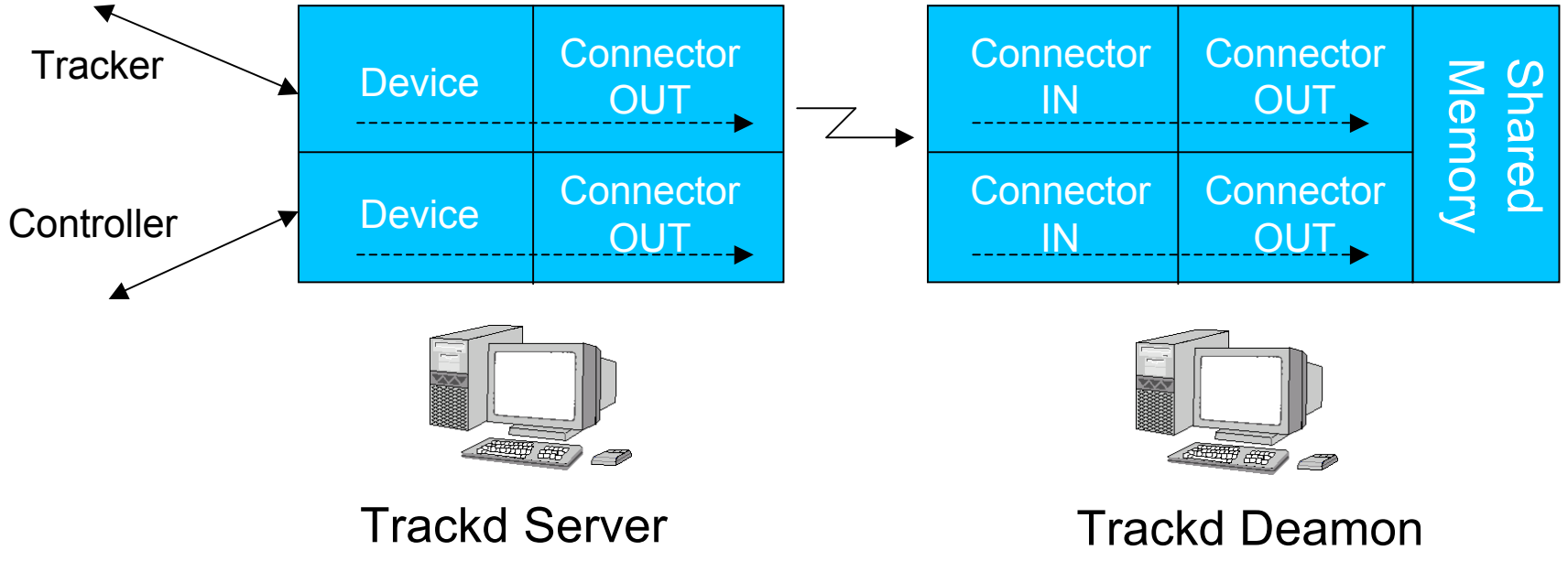
- Trackd Server:
  - opens up a connection and "talks to" tracker or controller devices.
  - Sends the devices' data through the network or serial port to a Trackd Daemon, or to another Trackd Server.
- Trackd Daemon:
  - collects data from any connected devices via the serial port, or reads data coming across a connector from the Trackd Server.
  - is responsible for storing device data in shared memory.
  - Once in shared memory, the data can be accessed by applications that use the trackdAPI.
- Both require configuration files. Each tracker or input controller have to be configured in one of these files.

Trackd – Device and Connector

- Trackd uses two terms to describe objects that do the actual work within Trackd Deamon and Trackd Server - device and connector.
- Device - object that communicate with a physical input device.
  - Knows the protocol for talking with that hardware device.
  - Each supported device has its own shared object library written for it.
- Connector - responsible for transferring, shipping and storing data.
  - Three types - shared memory, serial cable and network(UDP).
  - Two "flavours„ - one for tracker data and one for controller data.
  - can receive or transmit data for either a tracker or a controller, but not both.
- Both are defined in the configuration files.
- Unique set of variables that can be configured for each of them.

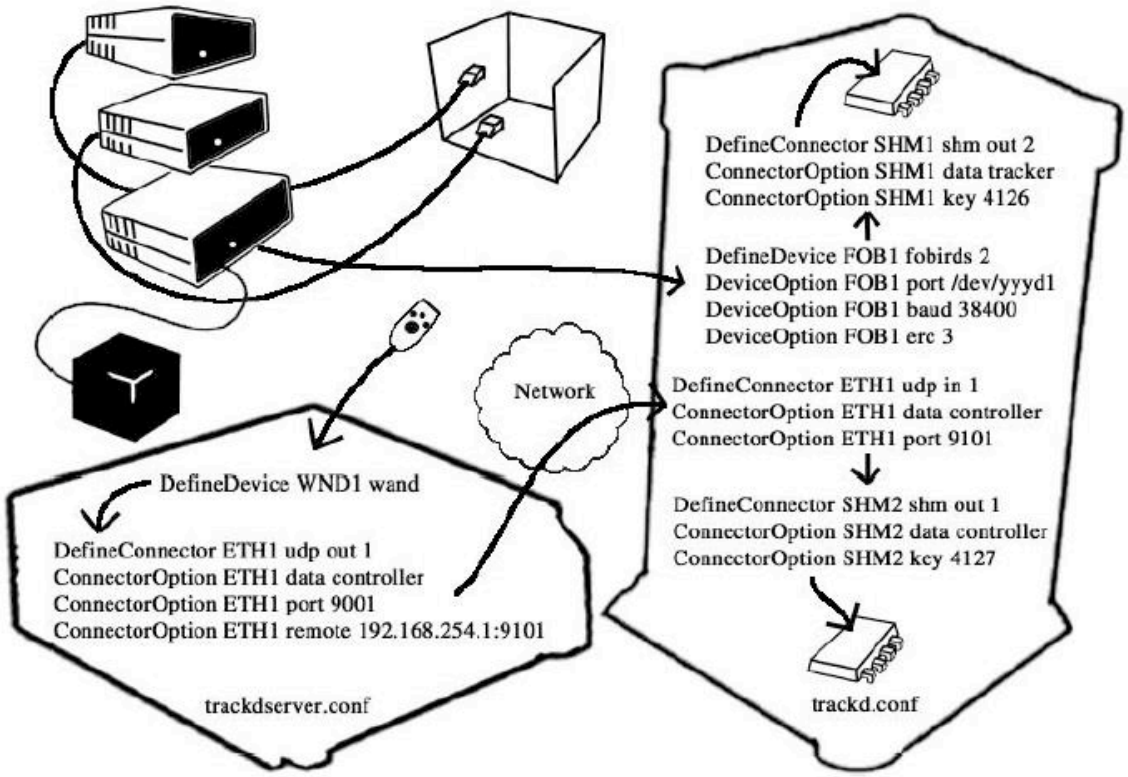
Trackd – Data Transfer

- Automatically data transfer
  - From device objects to outgoing connector objects(of the same type), within a Trackd Daemon or a Trackd Server.
  - From input connectors to output connectors of the same data type.
- Two different devices, one a tracker and the other a controller - two different shared memory output connectors.



Trackd - Example

- One of the devices is plugged into the local machine, and another into an auxiliary machine with Trackd Deamon running on the local machine.



Conclusion

- Sensor networks minimize negative properties of one tracker by another.
- Middleware-systems are developed, which form ubiquitous tracking environments consisting of these networks.
- Keywords: Device abstraction, Modular architecture, Standard interfaces, Distributed services, Simultaneously application of different device types, Easy to use and to extend, High degree of customization.
- But different main focuse.
- Combination of these systems can resolve the problems of research or commercial immersive systems, using diverse tracking and input devices.