Alma Mater Studiorum - Università di Bologna
CdS Laurea Magistrale in Ingegneria Informatica
II Ciclo - A.A. 2014/2015
Corso di Sistemi Mobili M (8 cfu)

Qualnet
Un simulatore di reti wireless

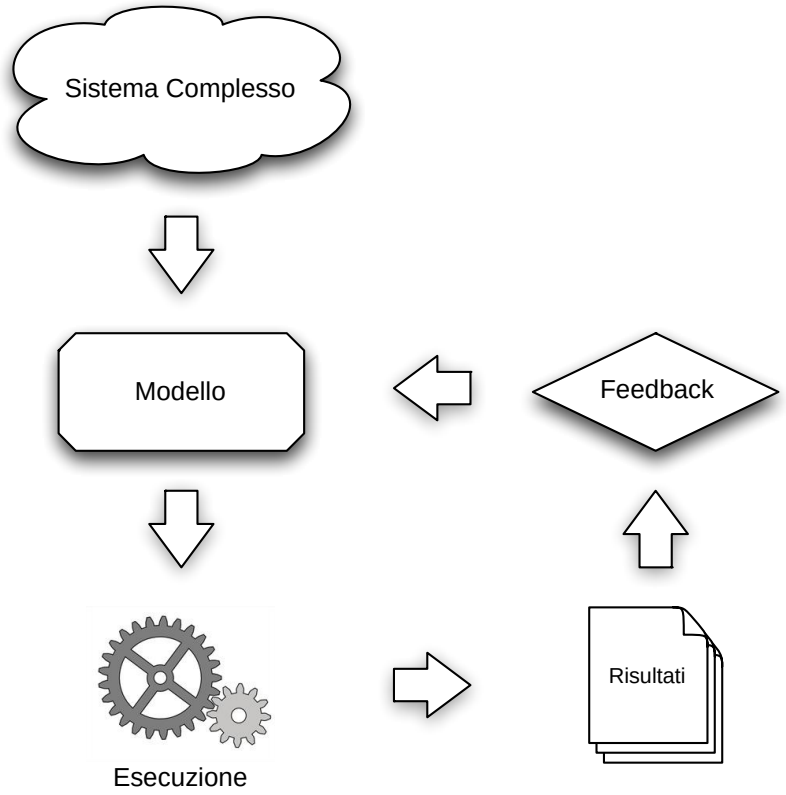
Andrea Reale & Paolo Bellavista

http://lia.disi.unibo.it/Courses/sm1415-info/
http://lia.disi.unibo.it/Staff/PaoloBellavista/

Simulazione e Qualnet

INTRODUZIONE

Simulazione



- *Riproduzione* del comportamento di un sistema complesso
- **Astrazione** da dettagli superflui
- Focus su parametri e aspetti d'interesse del sistema

Usi della simulazione

- Sistema reale ***troppo costoso*** da realizzare
- Livello di ***dettaglio massimo non necessario***
- Misurazione di parametri altrimenti difficili da osservare
- Necessità di ***rallentare/velocizzare*** il tempo di esecuzione

Classificazione di simulatori

- Statico vs Dinamico
  - **Statico**: il tempo non è una variabile importante del modello
  - **Dinamico**: considera il variare del tempo
- Continuo vs Eventi (Discreti)
  - **Continuo**: il tempo avanza ad intervalli costanti
  - **Ad eventi**: il tempo avanza solo quando accadono *eventi significativi* (ad es. ricezione di un pacchetto)

Validazione di un simulatore

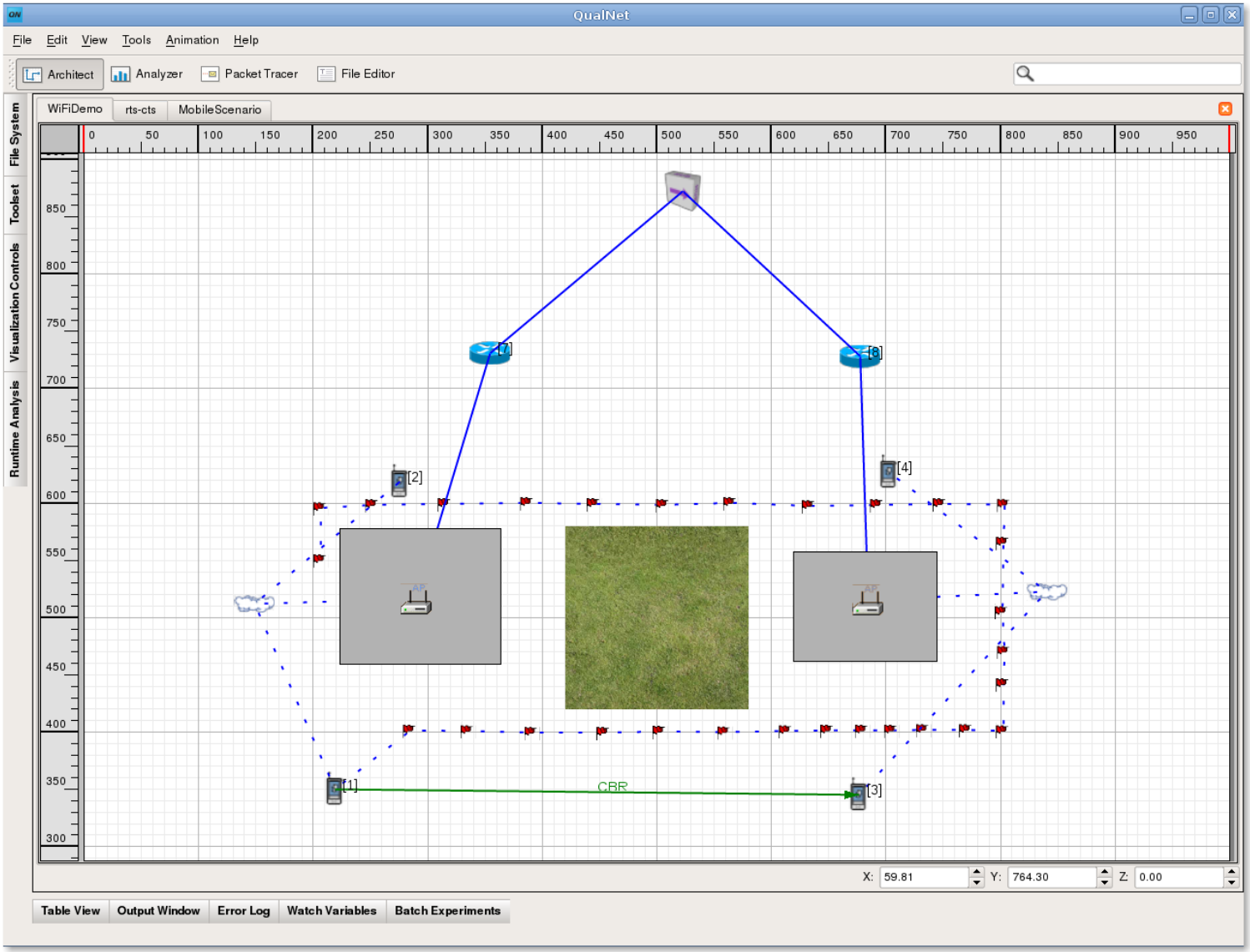
- Un modello usato in una simulazione ***può produrre risultati molto diversi da quelli reali***
  - Errori nel modello
  - *Trascurati dei dettagli importanti*

- **Validazione** di un modello
  - Confronto dei risultati di simulazione con risultati reali attesi

Simulatori di rete

- Modelli di reti di calcolatori
  - Nodi, router, switch, link, …
  - Protocolli di comunicazione tra nodi
- Esecuzione di scenari altrimenti costosi in termini di hw, tempo e praticità
  - Nuovi protocolli o varianti di protocolli esistenti
  - Topologie di rete complesse
  - Facilità di tuning di caratteristiche nodi e misura di parametri valutativi

Qualnet: simulatore di reti (wireless) ad eventi discreti



Qualnet: caratteristiche principali

- Prodotto commerciale
- Spin-off di un progetto opensource di UCLA
- Modelli accurati e veloci di protocolli di tutto lo stack di rete
  - Modelli di propagazione radio (pathloss, shadowing, fading)
  - 802.11, 802.15.4, 802.16, GSM/UMTS, …
  - RIP, DSR, AODV, …

QualNet®
Building Smarter Networks

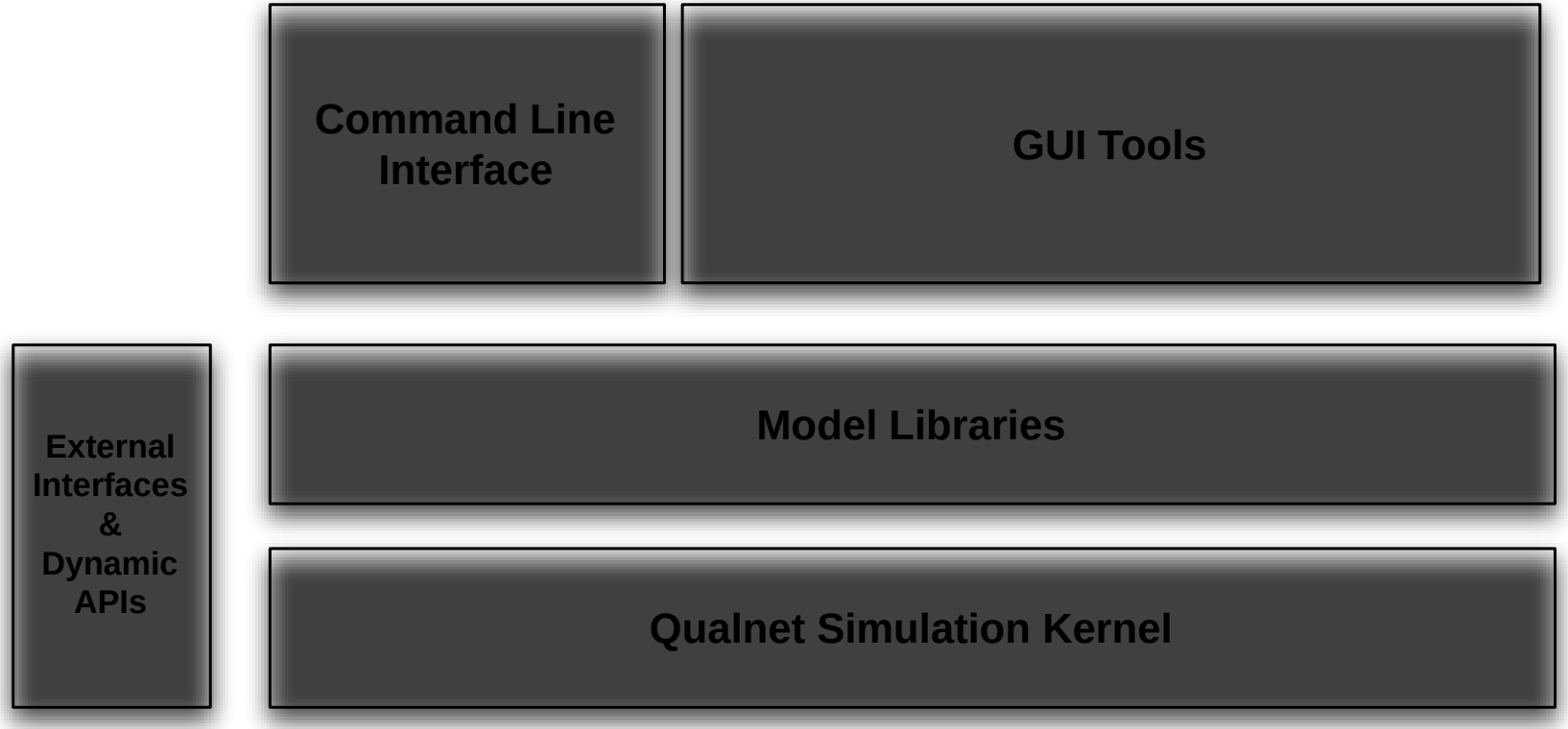
Qualnet: punti di forza

- **Scalabilità**
  - Esecuzione parallela su più di un processore (o su più di una macchina, con opportuna licenza)
- **Tradeoff dettaglio/prestazioni configurabile**
  - Modelli molto dettagliati
  - Possibilità di astrarre da dettagli non desiderati per velocizzare l'esecuzione
- **Portabilità**
  - Linux, MacOs X, Windows (32 e 64 bit)
- **Estensibilità**
  - Possibilità di sviluppare modelli ad ogni livello
  - Interfacciabile da software esterno

Crezione di scenari e analisi dei risultati

SIMULARE RETI CON QUALNET

Qualnet: architettura

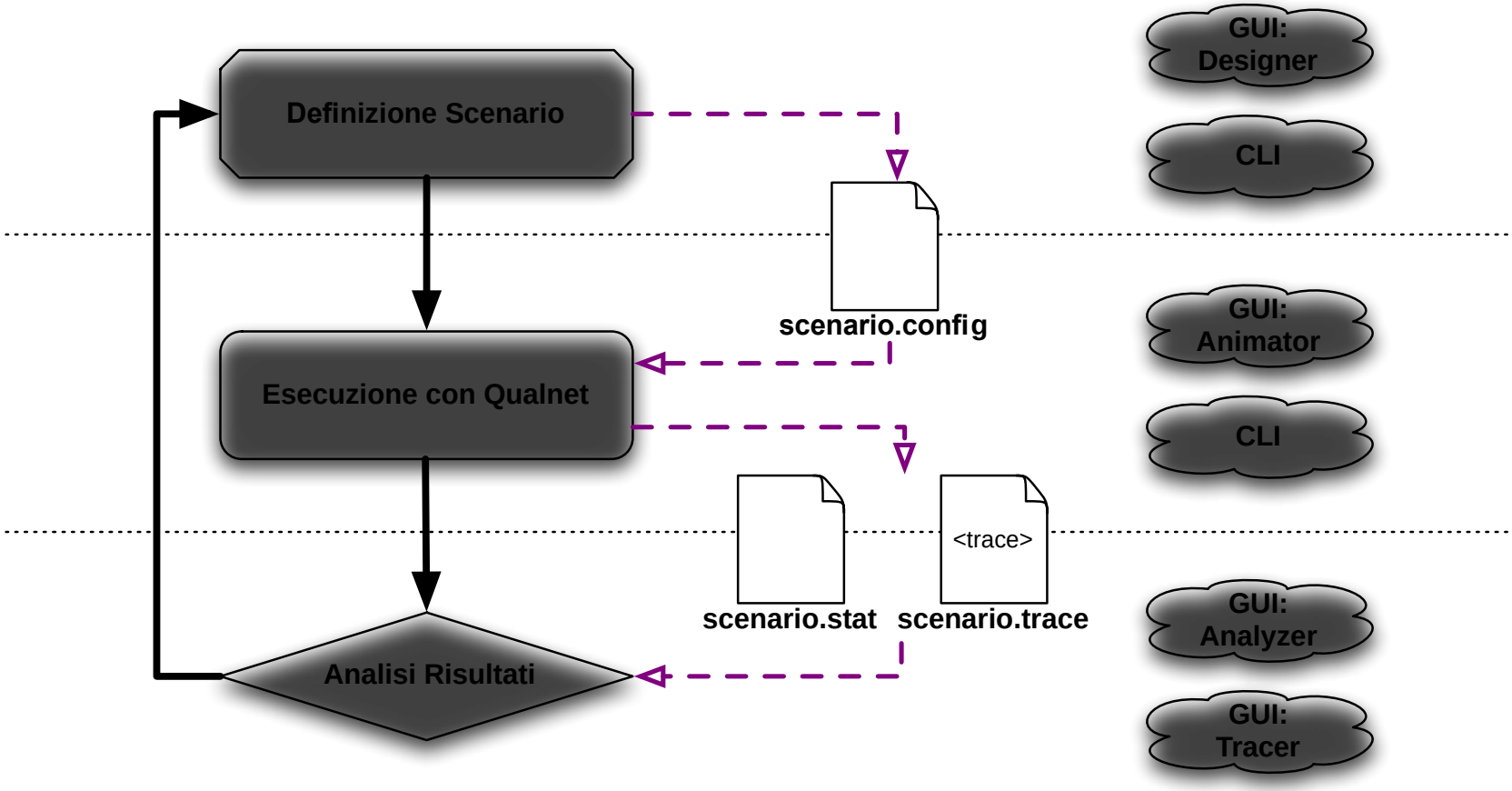


Scenario

Scenario: descrizione di una rete eseguibile dal simulatore. Comprende:

- Topologia: nodi, collegamenti fisici tra nodi
- Interfacce di rete (numero, tipo, modello fisico)
- Protocolli per ciascuna interfaccia (fisico, MAC, network, routing)
- Modelli di mobilità dei nodi
- Modello “ambientale”: caratteristiche del terreno, caratteristiche meteorologiche, …
- Batterie, modelli di consumo energetico, …
- Modello delle singole antenne
- …

Workflow di simulazione



Input: file .config

- Descrizione completa dello **scenario** da simulare
- Può includere altri file
  - **.app**: descrive i protocolli applicativi sui nodi
  - **.nodes**: descrive il posizionamento e la mobilità dei nodi
- Creazione
  - Tramite *GUI* con Qualnet Designer
  - *A mano,* utile soprattutto per scenari più complessi (ad es. con tanti nodi)

Input: file .config

Ogni riga del file imposta un parametro della simulazione

Formato:

[<scope>] <param> [<index>] <value>

- **scope**: specifica a quali oggetti della simulazione deve essere applicato il parametro. Un parametro può essere definito per uno di questi scope: global, node, subnet, interface
- **param**: nome identificativo del parametro
- **index**: per parametri che possono avere più istanze, specifica a quale istanza si riferisce (opt)
- **value**: valore del parametro. Il formato ed il tipo possono variare

example.config
Canali Radio

...

SIMULATION-TIME 12H

COORDINATE-SYSTEM CARTESIAN

TERRAIN-DIMENSION (1500, 1500)

NODE-POSITION-FILE example.nodes

MOBILITY RANDOM-WAYPOINT

RADIO CHANNELS (un solo canale, indice 0)

PROPAGATION-CHANNEL-FREQUENCY[0] 2400000000

PROPAGATION-PATHLOSS-MODEL[0] TWO-RAY

PROPAGATION-SHADOWING-MODEL[0] LOGNORMAL

PROPAGATION-SHADOWING-MEAN[0] 4dB

PROPAGATION-LIMIT[0] -111.0 dB

PROPAGATION-MAX-DISTANCE[0] 0

example.config - Subnet

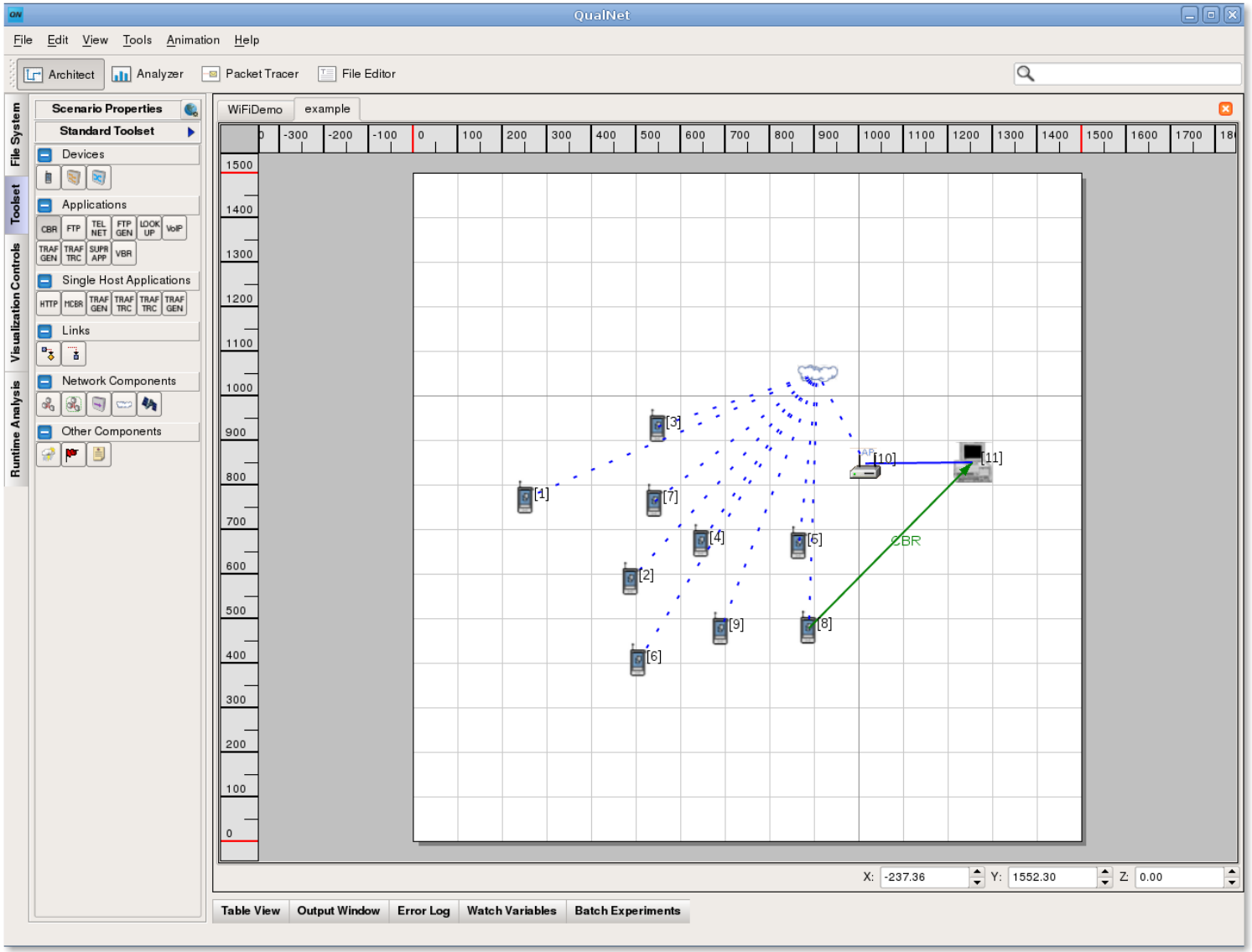
```
NETWORK-PROTOCOL IP
IP-QUEUE-TYPE[0] FIFO

SUBNET N8-1.0 { 1 thru 10} # crea nodi ed interfacce
[N8-1.0] PHY-MODEL PHY802.11b
[N8-1.0] MAC-PROTOCOL MACDOT11
[N8-1.0] ROUTING-PROTOCOL AODV

LINK N8-2.0 {10, 11}
[N8-2.0] LINK-PHY-TYPE WIRED
[N8-2.0] LINK-MAC-PROTOCOL MAC802.3 # MAC802.3|ABSTRACT
[N8-2.0] NETWORK-PROTOCOL IP
[N8-2.0] ROUTING-PROTOCOL BELLMANFORD

APP-CONFIG-FILE example.app
```

example.config
Risultato



Output: scenario statistics

File di testo **.stat**

Ogni riga ha il formato:

<id>, <addr>, <index>, <layer>, <proto>, <param> = <value>

- **id**: identificativo del nodo
- **addr**: indirizzo IP dell'interfaccia su cui gira il protocollo (opt)
- **index**: distingue istanze multiple di uno stesso protocollo (opt)
- **layer**: PHY, MAC, NETWORK, TRANSPORT, APPLICATION
- **proto**: nome del protocollo a cui si riferisce la statistica
- **param**: identificativo descrittivo della statistica
- **value**: misura della statistica

Output: Trace file

Tracce dei ***messaggi scambiati da singoli protocolli***

- Si possono selezionare selettivamente interi layer e/o singoli protocolli da tracciare
- File xml:

```
<rec>
        <rechdr></rechdr>
        <recbody></recbody>
</rec>
```

  - <rec> Entry per un singolo pacchetto tracciato
  - <rechdr> Header della registrazione
  - <recbody> header del pacchetto (uno o più se incapsulati)

example.trace

```
<trace_file>
 <head> … </head>
 <body>
  …
  <rec>
   <rechdr> 3 0 3.3584881 7 3 2 <action> 1 0 </action> </rechdr>
   <recbody>
    <udp> 519 519 36 0 </udp>
   <recbody>
  </rec>
  …
 </body>
</trace_file>
```

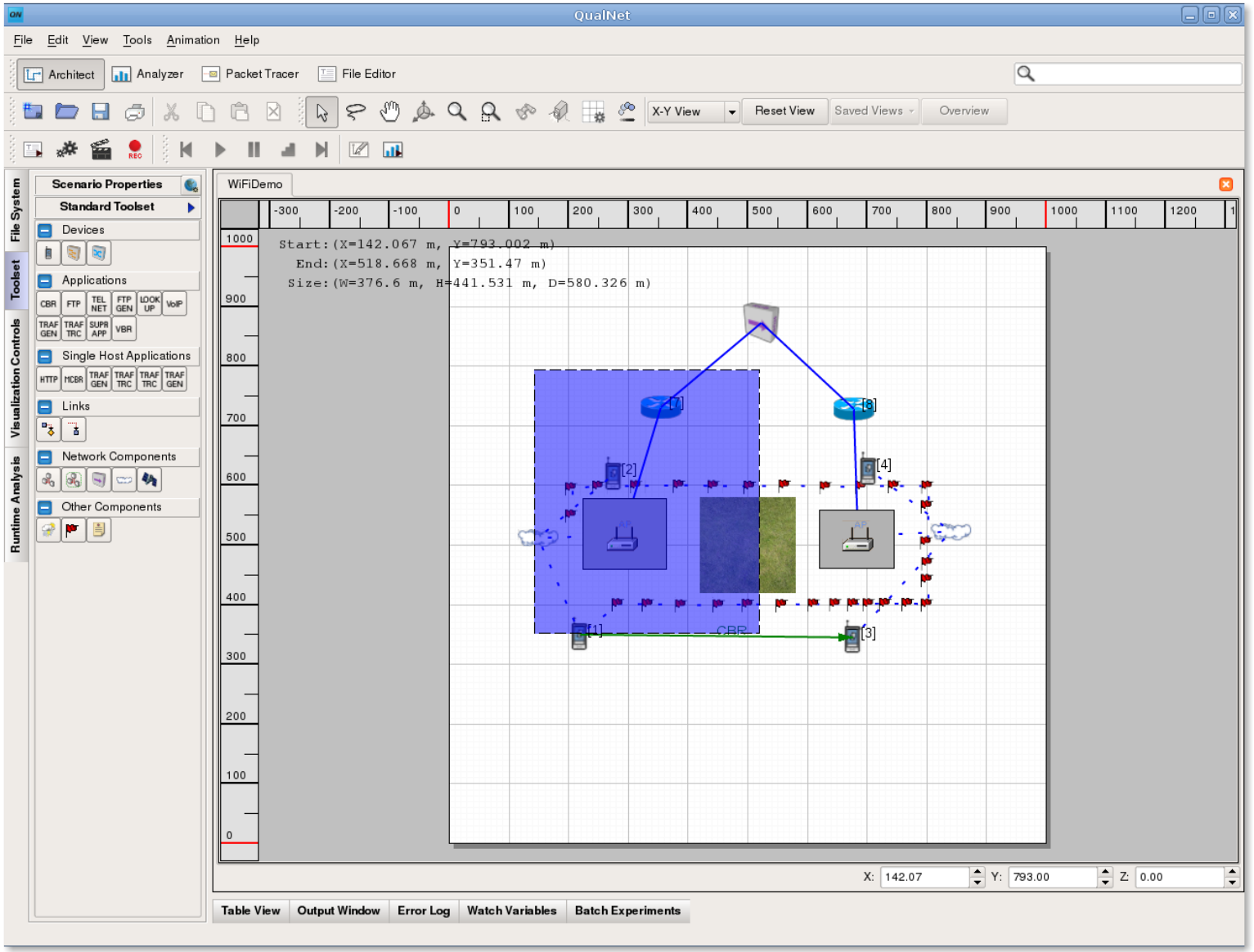
Il nodo 3 invia al nodo 0 un messaggio originato dal protocollo 4 (cbr) al secondo di simulazione 3.35. Il messaggio è tracciato dal protocollo 2 (udp) e l'azione associata è la 1 (send)

Porta sorgente: 519
Porta destinazione: 519
Length: 36 bytes
Checksum: N/A

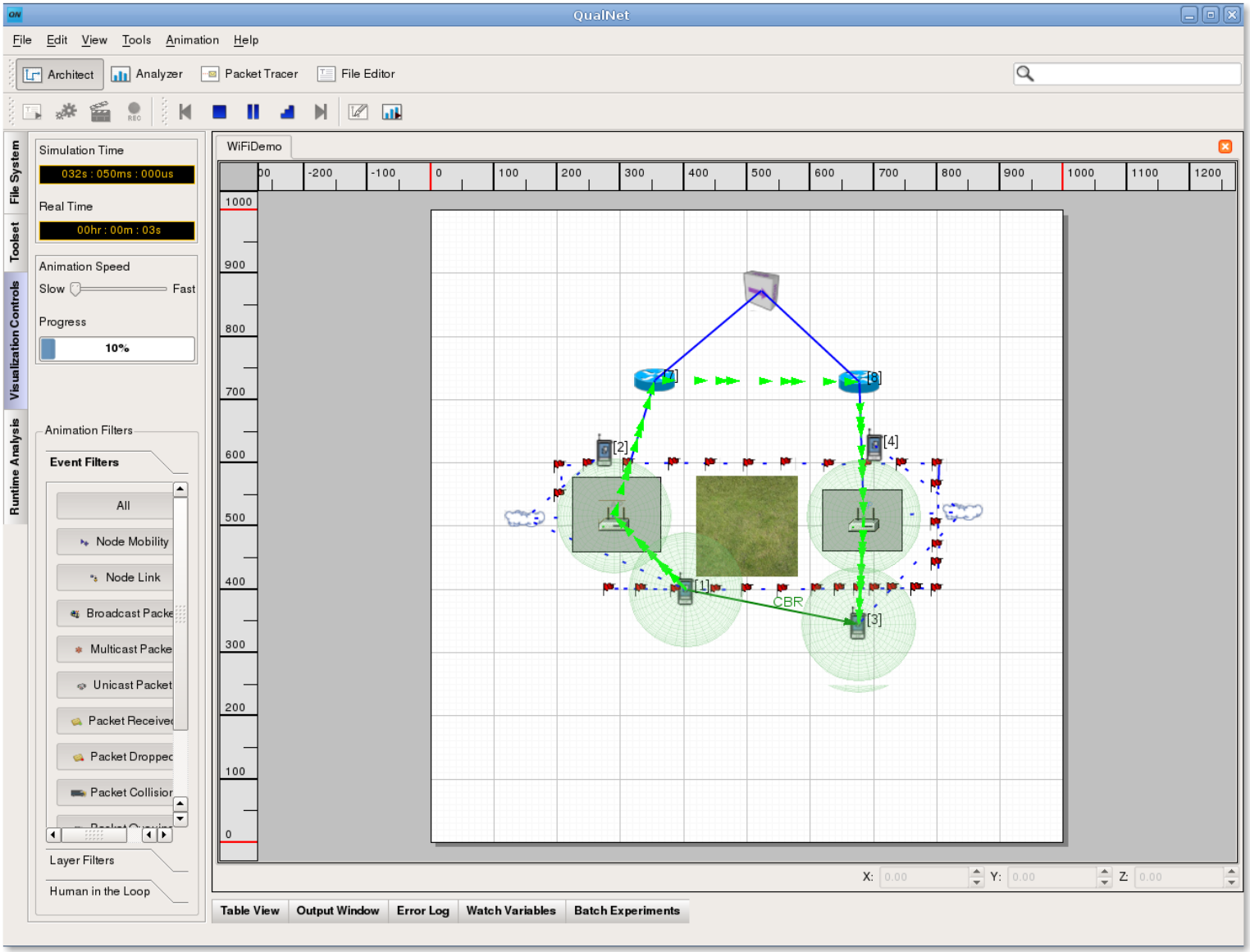
GUI Tools: Qualnet Architect

- Due modalità d'uso
- **Design Mode**
  - Creazione di scenari tramite *drag&drop* di componenti visuali
  - Generazione dei *file di testo* di configurazione
- **Visualize Mode**
  - *Esecuzione* di scenari
  - *Animazioni* e visualizzazione di (alcuni) parametri in tempo reale

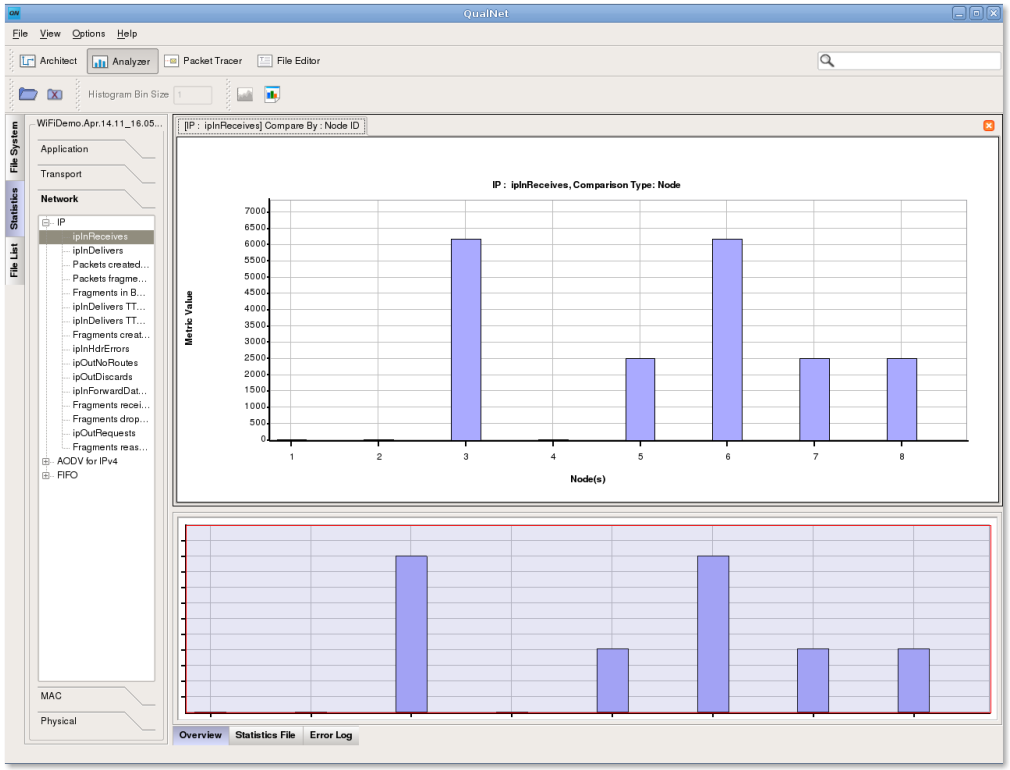
GUI Tools: Qualnet Architect (Design Mode)



GUI Tools: Qualnet Architect (Visualize Mode)

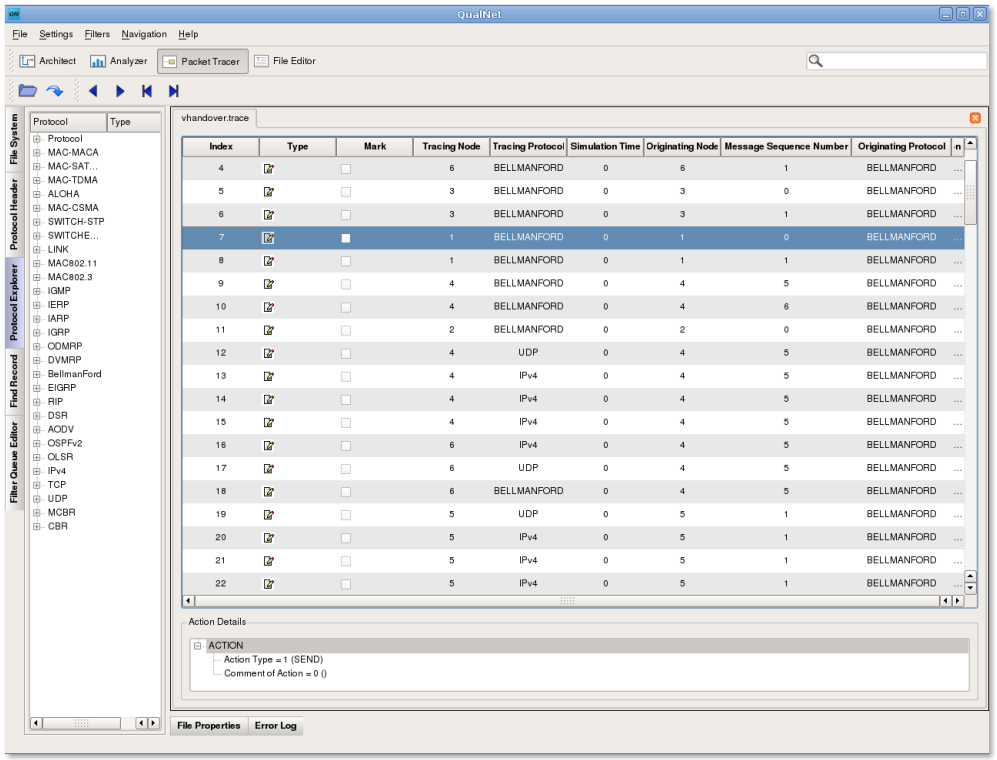


GUI Tools : Qualnet Analyzer



- *Parsing* dei file statistiche (.stat) prodotti da esperimenti
- *Visualizzazione grafica* delle statistiche
- Semplici strumenti di **aggregazione e sintesi**
- *Personalizzazione* dei grafici

GUI Tools: Qualnet Packet Tracer



- *Parsing* dei file .trace prodotti da esperimenti
- *Visualizzazione tabulare* delle informazioni nelle tracce
- Ricerca e filtraggio basati su criteri multipli
  - e.g., porta d'origine, TTL, nodo di destinazione, protocollo

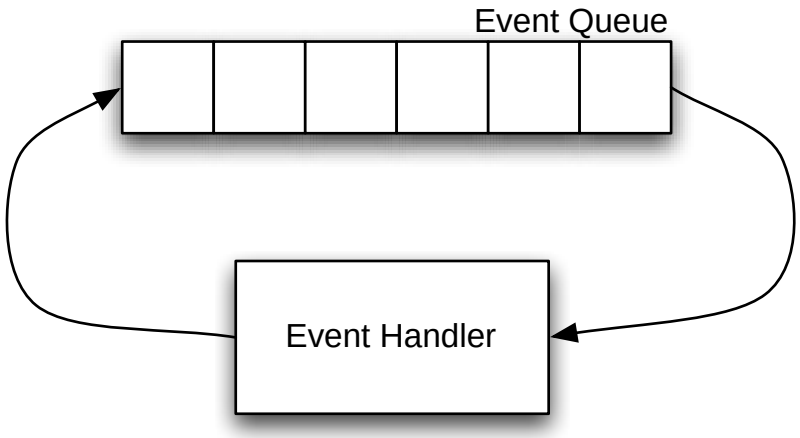
Implementare nuovi modelli e protocolli

SVILUPPO IN QUALNET

Qualnet: Ambiente di sviluppo

- Qualnet Kernel e Model Libraries scritte in **C++**
  - In realtà, object orientation del C++ è *scarsamente utilizzata*
  - Conoscenza di linguaggio C sufficiente per comprendere (e scrivere) gran parte del codice
- Sorgenti di parti del kernel e dei protocolli delle Model Libraries disponibili
  - Possibilità di ***modificare protocolli esistenti*** per adattarli alle proprie esigenze
  - Utili linee guida di sviluppo di nuovi modelli

Simulazione ad eventi: reprise



- **Coda di eventi**, ordinata secondo timestamp crescente dell'evento
- Ad uno ad uno gli eventi vengono **prelevati dalla coda e processati**
- L'elaborazione di un evento può **generare altri eventi**, che sono inseriti *ordinatamente* nella coda

Tipi di evento in Qualnet

- **Packet Event**
  - Usati per simulare *scambio di pacchetti* tra nodi distinti
  - Usati per simulare passaggio di pacchetti attraverso layer protocollari diversi di uno stesso nodi
- **Timer Event**
  - Usati per simulare *time-out o allarmi* (ricorrenti nella gestione di protocolli)
- Timer Event e Packet Event in Qualnet sono rappresentati da un'unica struttura dati: **Message**
  - Vengono gestiti dal kernel del simulatore in maniera *omogenea*
  - A livello terminologico, qualsiasi evento può essere chiamato "Messaggio" in Qualnet

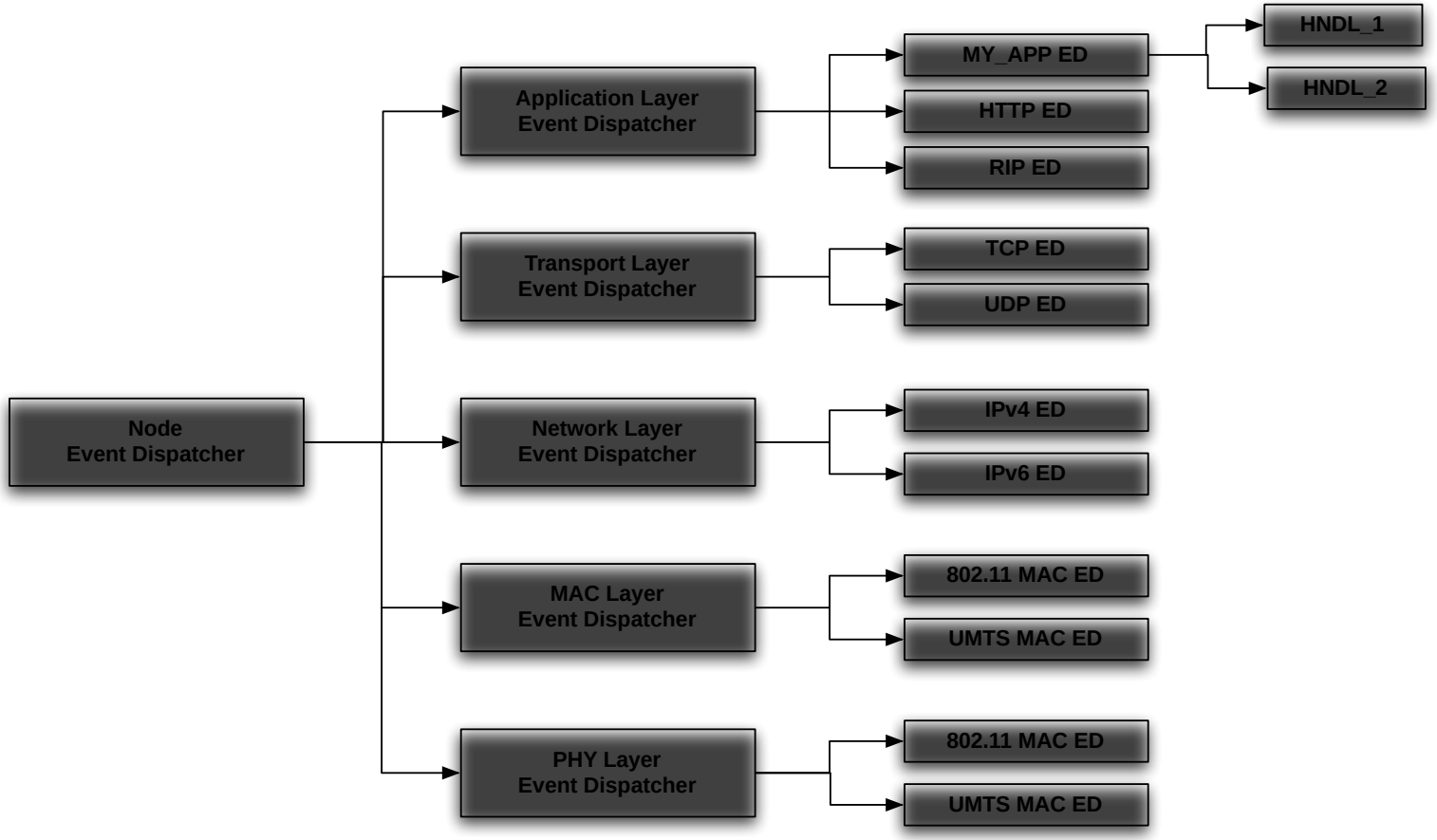
Message Data Structure

```
typedef struct message_str {
    …
  short layerType;          // e.g. APPLICATION_LAYER
  short protocolType;       // e.g. CBR
  short instanceId;         // e.g. 1515 (port)
  short eventType;          // e.g. MSG_APP_CBR_NEXT_PKT
  …
  char *packet;             // Payload (anche NULL)
  int virtualPayloadSize; // eg. 42
  …
  std::vector<MessageInfoHeader> infoArray;
  …
} Message;
```

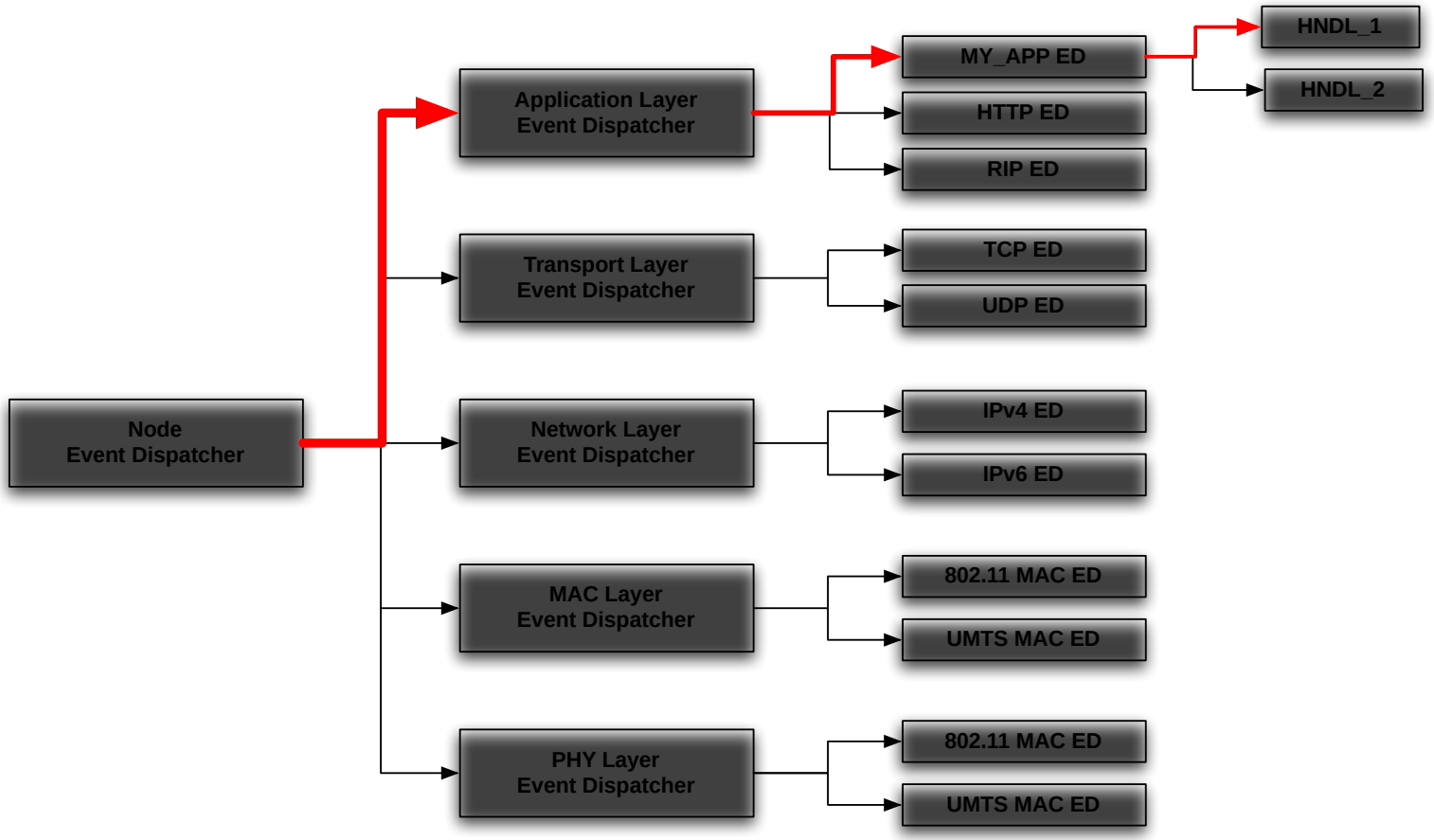
Layering

- Qualnet implementa una gestione dei messaggi basata su **Layer**
  - I layer sono quelli più bassi del modello OSI
- *Ogni messaggio ha un suo tipo, che è definito da un determinato protocollo*
- *Ogni protocollo appartiene ad uno dei layer*
- Viene fatto il **dispatching** di ogni messaggio sulla base di:
  - Tipo di messaggio
  - Protocollo che lo definisce
  - Layer a cui il protocollo appartiene
  - Nodo destinatario del messaggio

Dispatcher e Layering



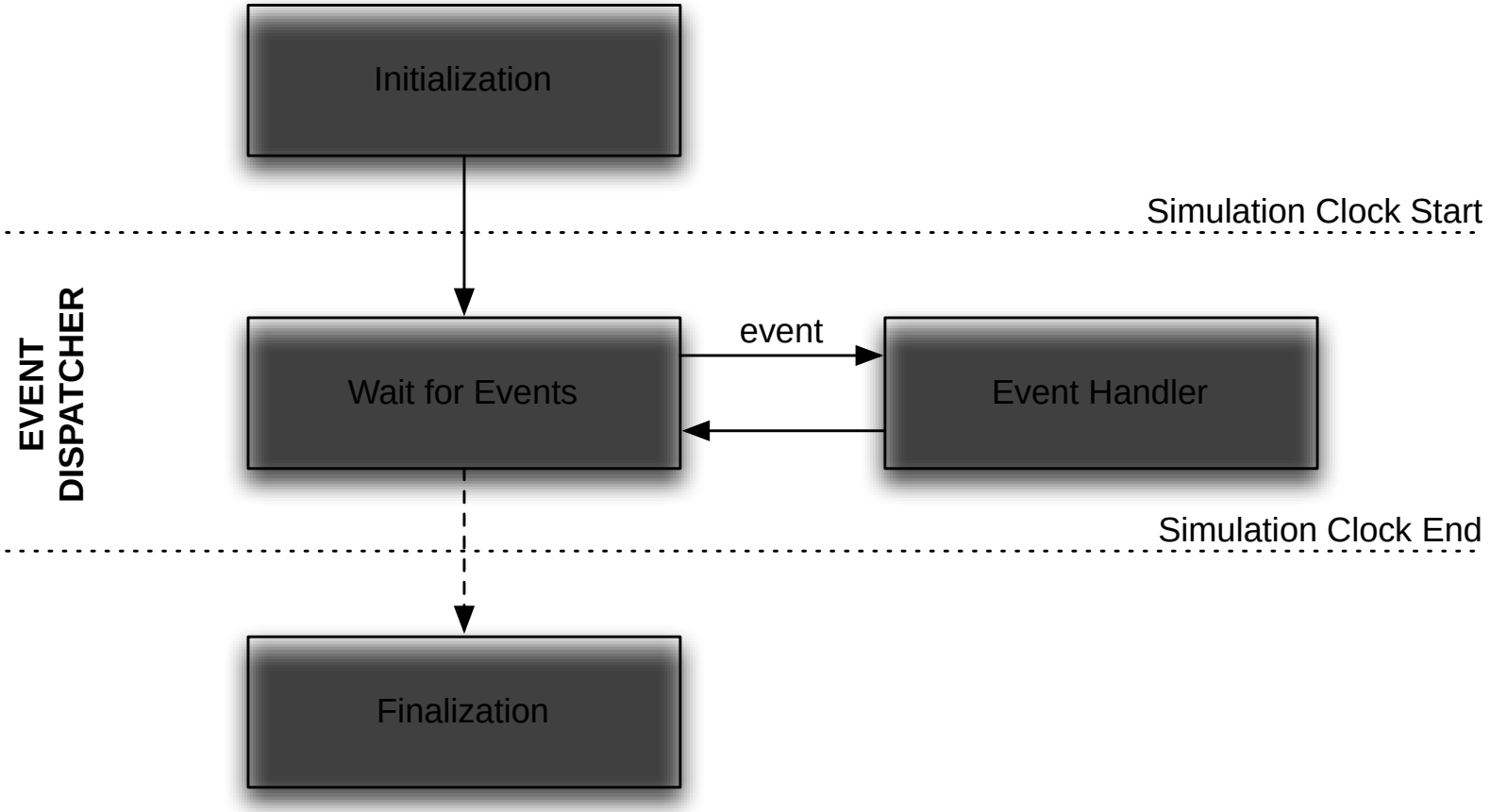
Dispatcher e Layering



Struttura di un protocollo

- Ogni protocollo *definisce un insieme di eventi* che si dichiara in grado di gestire
- Uno o più **handler** gestiscono il verificarsi degli eventi definiti
- Un "**Event Dispatcher**", con il ruolo di richiamare handler appropriato all'arrivo di nuovi messaggi

Protocolli come macchine a stati

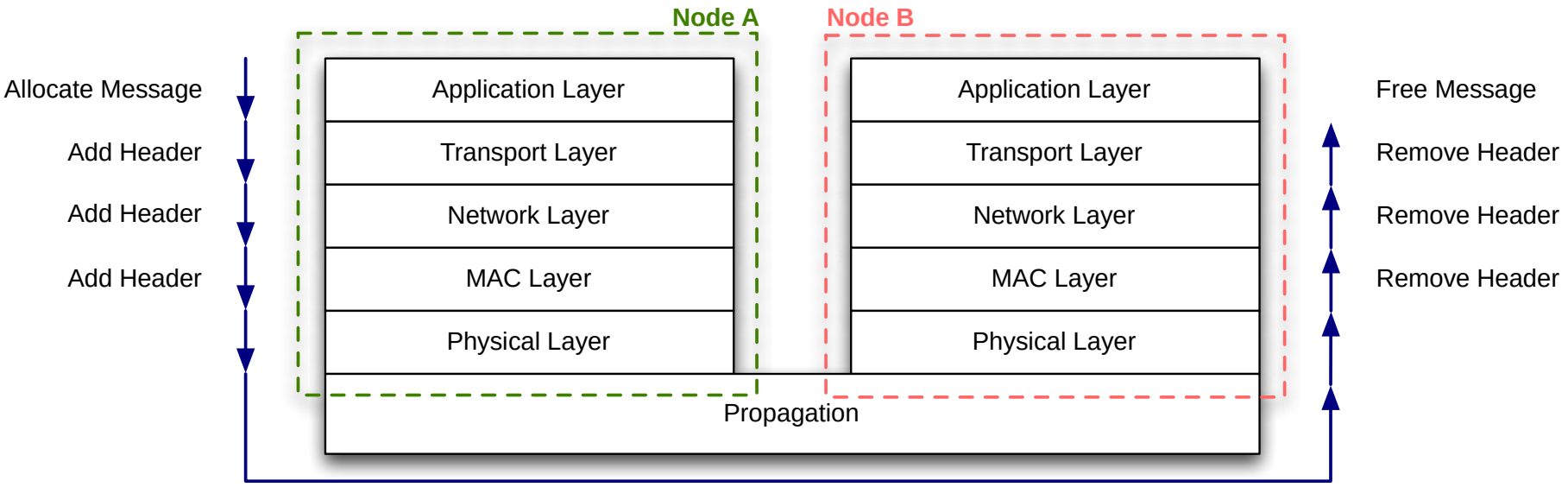


Packet Event e comunicazione inter-layer

- Ogni protocollo può generare ed inviare messaggi
  - Diretti ad un protocollo sullo stesso (o un altro) nodo *(più comune per Timer Event*)
  - Diretti ad un altro nodo, **passando per i layer direttamente sottostanti** (come in OSI)

- Ogni livello *prende in consegna il messaggio, lo elabora, aggiunge un suo header e lo propaga al livello più in basso*
  - Ad ogni passaggio possono eventualmente essere aggiunti opportuni delay

Packet Event: Gestione a Layer



Qualnet Message API

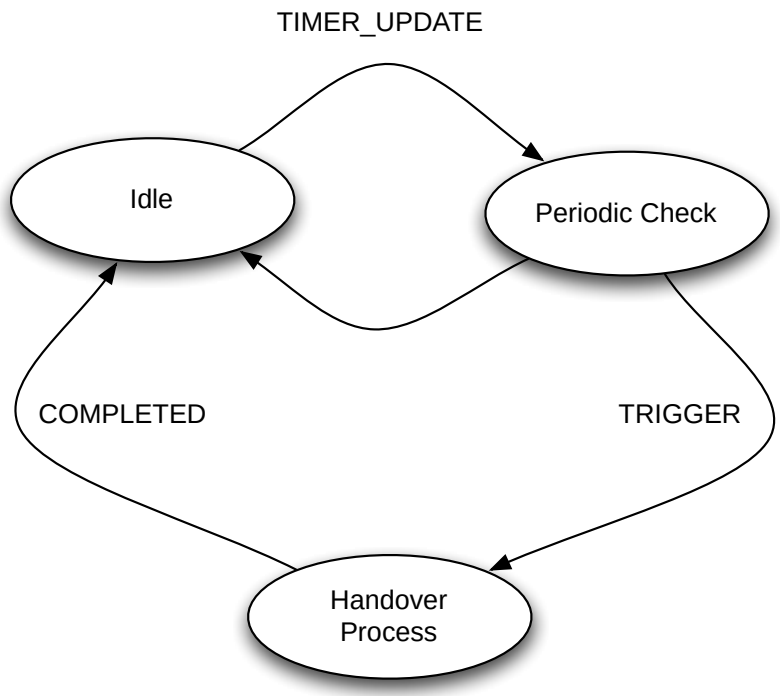
- API generiche per manipolare eventi:
  - **MESSAGE_Send**(Node*, Message*, clocktype)
  - MESSAGE_Alloc(Node*, int, int, int)
  - MESSAGE_Free(Node*, Message*)
  - MESSAGE_AddInfo(Node*, Message*, int, short)
  - MESSAGE_AddHeader(Node*, Message*, int, TraceProtocolType)
  - MESSAGE_RemoveHeader(…)
- API specifiche per APPLICATION Layer
  - APP_UdpSendNewData(…)
  - APP_UdpNewHeaderVirtualDataWithPriority(...)

Esempio: gestione di Hard Vertical Handover

- **Scenario**
  - Dispositivo mobile con due interfacce wireless
  - Un'interfaccia IEEE 802.11 e una IEEE 802.16

- **Obiettivo**
  - Il dispositivo "a regime" utilizza *solo una delle due interfacce*
  - Quando la qualità del segnale ricevuto sull'interfaccia scende sotto una certa **soglia**, il dispositivo prova a ottenere connettività migliore dall'altra interfaccia

Diagramma degli stati



- **Idle**
  - Una interfaccia attiva
- **Periodic Check**
  - Periodicamente, viene controllata qualità del segnale ricevuto
- **Handover Process**
  - Vengono attivate entrambe le interfacce, e viene lasciata accesa la “migliore”

Simple Vertical Handover: alcune considerazioni

- Peculiarità del protocollo
  - Protocollo "*single host*": non comunica con altri nodi. Solo informazioni locali
  - Posizionato a livello Application, ma necessita di informazioni a livello MAC (RSSI dei segnali)
- Rompe parzialmente il layering di Qualnet

Definire un modello: cose da fare

- Definire
  - Identificativo di protocollo
  - Tipo degli eventi gestiti
  - Strutture dati utilizzate

  } COSTANTI
- Implementare
  - **Funzione di inizializzazione**: allocazione e inizializzazione delle strutture dati del protocollo
  - **Dispatcher del protocollo**: gestione degli eventi e invocazione degli handler appropriati
  - **Handler** degli eventi
  - **Funzione di finalizzazione**: calcolo e stampa di statistiche, e deallocazione delle strutture dati
- "Registrare" le funzioni sul gestore del layer appropriato

Definizione del protocollo

Aggiungere il nuovo protocollo nella lista dei protocolli applicativi disponibili

```
typedef enum { // include/application.h
    …
    APP_FTP_SERVER = 21,
        …,
    APP_HANDOVER_SIMPLE,
    APP_PLACEHOLDER
} AppType;
```

Definizione degli eventi

Includere i tipi di evento che il protocollo vuole gestire nella lista degli eventi possibili

```
enum { // include/api.h
    MSG_APP_CBR_NEXT_PKT,
    …
    MSG_APP_HANDOVER_UPDATE,
    MSG_APP_HANDOVER_TRIGGER,
    MSG_APP_HANDOVER_FINALIZE,
    …,
    MSG_DEFAULT
};
```

Inizializzazione e finalizzazione del protocollo

- Funzione di inizializzazione *invocata per ciascuna istanza del protocollo* (ad es. per ogni istanza su ogni nodo)
  - **Allocazione** ed inizializzazioni di strutture dati definite appositamente per il protocollo
  - **Salvataggio delle strutture** nello stato del nodo a cui si riferiscono
- Funzione di finalizzazione *invocata per ciascuna istanza del protocollo, a fine simulazione*
  - **Raccolta statistiche** e stampa sul file di output .stat
  - **Deallocazione** delle strutture dati del protocollo

Inizializzazione e Finalizzazione

```
void HandoverInit(Node *node) {
    HandoverData *hd;
    hd = (HandoverData*) MEM_malloc(sizeof(HandoverData));
    …
    APP_RegisterNewApp(node, APP_HANDOVER_SIMPLE, hd);
    timerMsg = MESSAGE_Alloc( node, APP_LAYER,

                              APP_HANDOVER_SIMPLE,
                              MSG_APP_HANDOVER_UPDATE);
    MESSAGE_Send(node, timerMsg, HANDOVER_UPDATE_PERIOD);
}

void HandoverFinalize(Node *node) {
    HandoverData *hd = _HandoverRetrieveData(node);
    if ( hd != NULL ) App_UnregisterAPP(node, data);
}
```

Dispatcher degli eventi

```
void HandoverProcessEvent(Node *node, Message *msg) {
    switch(MESSAGE_GetEvent(msg)) {
        case MSG_APP_HANDOVER_UPDATE:
            HandoverPeriodcStatusUpdate(node, msg);
            break;
        case MSG_APP_HANDOVER_TRIGGER:
            HandoverStartHandover(node, msg);
            break;
        case MSG_APP_HANDOVER_FINALIZE:
            HandoverMakeDecision(node, msg);
            break;
        default:
            ERROR_ReportError("Unexpected event");
    }
}
```

“Registrazione” delle callback

```
// main/application.cpp
void APP_InitializeApplications(Node *first, NodeInput *in)
{…
    for (i=0; i< in.numLines; i++) {
        …
        else if (strcmp(appStr, “CBR”) == 0) { … }
        else if (strcmp(appStr, “HANDOVER_SIMPLE”) == 0) {
            …
            node = MAPPING_GetNodePtrFromHash( … );
            if (node != NULL) {
                HandoverInit(node);
            }
        }
    }
}
```

“Registrazione” delle callback

```
// main/application.cpp
void APP_ProcessEvent(Node *node, Message *msg) {
    …
    switch(msg->protocolType) {
        case APP_ROUTING_BELLMANFORD:
        …
        case APP_HANDOVER_SIMPLE:
            HandoverProcessEvent(node, msg);
            break;
        …
    }
}

void APP_Finalize(Node *node) {
    // Il concetto rimane il solito…
}
```

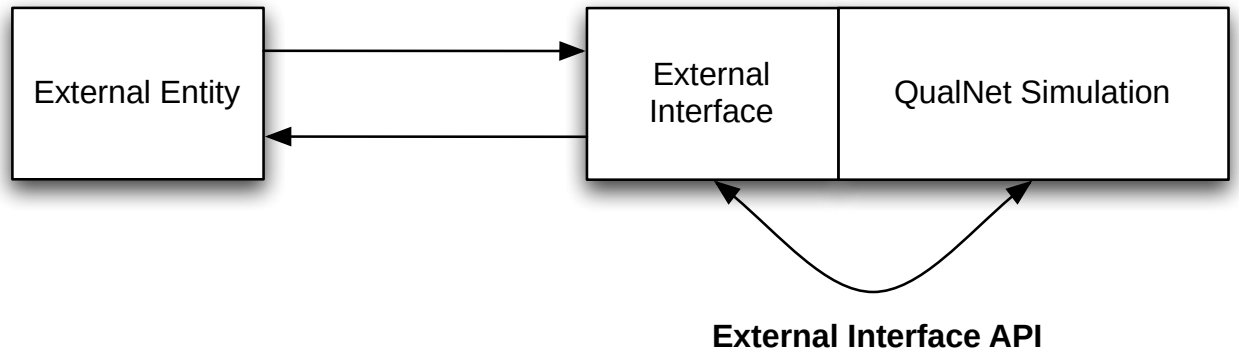
Dynamic API

- **Monitoring** e modifica **runtime** di variabili di simulazione (variabili dinamiche)
  - Ogni protocollo/modello deve definire esplicitamente le variabili che vuole esporre
- Servizio di **directory** (molto semplificato) per istanze di variabili dinamiche
  - Path di una variabile dinamica va esplicitamente registrato sul corrispondente servizio D_Hierarchy
  - Variabile dinamica va esplicitamente associata al path tramite una specifica funzione di libreria
  - /node/129/interface/192.168.0.1/aodv/numRequestsInitiated
- Componenti software **interni** possono registrare funzioni di *callback* per essere notificati di cambiamenti alle variabili dinamiche

External Interface API

- Consente la creazione di interfacce che permettano ***l'interazione runtime di software esterni*** con il simulatore

- QualNet mette a disposizione (su licenza aggiuntiva) tre diverse implementazioni di External Interface:
  - DIS (IEEE 1278), **HLA** (IEEE 1516), STK



QualNet: conclusioni

- **Pro**
  - Disponibili numerosissimi modelli molto precisi e dettagliati
  - Libertà di configurazione del tradeoff velocità/dettaglio delle simulazioni
  - Ottimi tool grafici
  - Buona manualistica introduttiva e forum di supporto online molto attivo
  - Codice ben scritto
- **Cons**
  - Necessario comprare tante licenze per usare diversi modelli
  - Manca documentazione dettagliata del codice (ad es. API doc)
  - Comunità di utilizzatori ridotta e meno “collaborativa” rispetto ad altre alternative (vedi prossima slide)

Simulatori di reti: aldilà di QualNet

- **NS-2 / NS-3**
  - Simulatori opensource (GPL) molto utilizzati in ambito accademico
  - NS-2 famoso ☹ per la sua *lenta* curva di apprendimento
  - NS-3 più usabile di NS-2, ma non altrettanto ricco di modelli già esistenti
- **OMNeT++**
  - Academic Public License: gratuito per scopi accademici
  - Non è un simulatore di reti, ma un framework generico per simulazione ad eventi discreti
  - Disponbili diversi package contenenti modelli per la simulazione di reti (INET)
  - Linguaggio abbastanza sofisticato (NED) per descrivere scenari di simulazione

Voglio provare QualNet!

- L'Università di Bologna dispone di una **licenza di QualNet** installata su un' unica macchina fissa
  - *Developer Model Library* (IEEE 802.3, IP, TCP, UDP, RIP, …)
  - *Wireless Model Library* (IEEE 802.11, MACA, DSR, …)
  - *Advanced Wireless Model Library* (IEEE 802.16)
  - *Sensor Networks Model Library* (IEEE 802.15.4)
  - *UMTS Model Library*

- **Possibilità di utilizzare da remoto quella macchina**
  - Con alcune limitazioni