Cross-layer Stack Design Framework in OMNeT++
OMNeT++ Community Summit

Doğanalp Ergenç

Wireless Systems, Networks and Cybersecurity Laboratory
Department of Computer Engineering
Middle East Technical University
Ankara Turkey

September 6, 2018
Outline

Cross-layer Stack Architecture
   Main Types
   Related Work
   Motivation

Implementation of Cross-layer Framework in OMNeT++
   Definition of the management layer
   Extension of other layers
   Creation of a new node

A Cross-layer Clustering Algorithm for Ad-hoc Networks
   General architecture of PCA
   Flow Chart

Conclusion and Future Work

References
Cross-layer Stack Architecture

- Layered design
  - Self-containment + Abstraction
  - Independent layers + Inter-layer relationships
  - Stacked in an order

- Cross-layer architecture as an inter-layer relationship concept
  - Comprehensive information through overall architecture
  - Different directions through different modules
Main Types

- **Application Layer**
- **Transport Layer**
- **Network Layer**
- **Link Layer**
- **Physical Layer**

**Shared Storage** to **Interface**

**Management Layer**
- **Application Layer**
- **Transport Layer**
- **Network Layer**
- **Link Layer**
- **Physical Layer**
## Related Work

<table>
<thead>
<tr>
<th>Study</th>
<th>Goal</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Massin <em>et al.</em> [1]</td>
<td>Radio access and resource allocation</td>
<td>xLayer 1-2</td>
</tr>
<tr>
<td>Feeney [4]</td>
<td>Information sharing modification</td>
<td>XML content</td>
</tr>
</tbody>
</table>
Motivation

- Present a guideline for the fundamentals of cross-layer structure
- Implement a framework for general use
- Show its actual implication
Implementation of Cross-layer Framework in OMNeT++

- Definition of the management layer
  - Layer-specific parameters and gates in NED files
  - Packet-handling scheme for inter-layer communication in C++
- Extension of other layers
- Creation of a new node
Definition of the management layer

Layer-specific parameters and gates in NED files

```nml
simple ManagementLayer
{
    parameters:
    @display("i=block/buffer");

    gates:
    input appIn;
    output appOut;

    input transIn;
    output transOut;

    input networkIn;
    output networkOut;

    input linkIn;
    output linkOut;

    input phyIn;
    output phyOut;
}
```
Definition of the management layer

Packet-handling scheme for inter-layer communication in C++

```cpp
class ManagementLayer : public cSimpleModule {
    protected:
        cGate *appInGate = nullptr;
        cGate *appOutGate = nullptr;

        cGate *transInGate = nullptr;
        cGate *transOutGate = nullptr;

        cGate *networkInGate = nullptr;
        cGate *networkOutGate = nullptr;

    virtual void initialize() override;
    virtual void handleMessage(cMessage*) override;
    virtual void finish() override;

    void ManagementLayer::initialize()
    {
        appInGate = gate("appIn");
        appOutGate = gate("appOut");
        transInGate = gate("transIn");
        transOutGate = gate("transOut");
        networkInGate = gate("networkIn");
        networkOutGate = gate("networkOut");
    }
};
```
Definition of the management layer

Packet-handling scheme for inter-layer communication in C++

```cpp
void ManagementLayer::handleMessage(cMessage *msg)
{
    if(msg->getArrivalGate() == appInGate)
        //Take action for incoming packets from Layer 5
    else if(msg->getArrivalGate() == transInGate) {
        //Take action for incoming packets from Layer 4
    }
    else if(msg->getArrivalGate() == networkInGate) {
        //Take action for incoming packets from Layer 3
    }
    else if(msg->getArrivalGate()->isName("linkIn")) {
        //Take action for incoming packets from Layer 2
    }
    else if(msg->getArrivalGate()->isName("phyIn")) {
        //Take action for incoming packets from Layer 1
    }
}

void ManagementLayer::sendTransLayer(int type)
{
    CrossTransMsg *packet = new CrossTransMsg("CrossTransMsg");
    packet->setType(type);
    send(packet, transOutGate);
}
```
Extension of other layers

```cpp
void CrossIdealMac::initialize(int stage)
{
    IdealMac::initialize(stage);

    if (stage == INITSTAGE_LOCAL)
    {
        crossInGate = gate("crossIn");
        crossOutGate = gate("crossOut");
    }

    isRedundant = checkRedundancy();

    notForUsSignal = registerSignal("notForUsSignal");

    CrossSelfMsg* packet = new CrossSelfMsg();
    packet->setM_type(inet::LINK_UPDATE_ENERGY);
    scheduleAt(simTime() + UPDATE_ENERGY_PERIOD, packet);
};

void CrossIdealMac::handleMessage(cMessage *msg)
{
    if (msg->getArrivalGate() == crossInGate) {
        handleCrossLayerMessage(msg);
    } else {
        if (msg->getArrivalGateId() == lowerLayerInGateId) {
            sendRSSI(msg);
        }
        IdealMac::handleMessage(msg);
    }
};

package src.linkLayer;

import inet.linklayer.ideal.IdealMac;

module CrossIdealMac extends IdealMac
{
    parameters:
        @class(CrossIdealMac);

        @signal[notForUsSignal](type="long");

    gates:
        input crossIn @labels(CrossControlInfo/down);
        output crossOut @labels(CrossControlInfo/up);
};
```
Creation of a new node

```c++
module AdhocNode extends WirelessHost {
    parameters:
        forwarding = default(true);
        string crossType = default("LowestIDClustering");

    submodules:
        cross: <crossType> like ICrossLayer {
            @display("p=527,287");
        }

    connections allowunconnected:
        cross.app0Out --> udpApp[0].crossIn;
        udpApp[0].crossOut --> cross.appIn;

        cross.transOut --> udp.crossIn;
        udp.crossOut --> cross.transIn;

        cross.networkOut --> networkLayer.crossIn;
        networkLayer.crossOut --> cross.networkIn;

        for i=0..sizeof(radioIn)-1 {
            cross.linkOut++ --> wlan[i].XmacIn;
            wlan[i].XmacOut --> cross.linkIn++;

            cross.phyOut++ --> wlan[i].XphyIn;
            wlan[i].XphyOut --> cross.phyIn++;
        }
}
```
A Cross-layer Clustering Algorithm for Ad-hoc Networks

- General architecture
- Flow chart

- Clustering in ad-hoc network for distributed and dynamic management
- Cross-layer architecture to manage leader selection
- Probabilistic Clustering Algorithm (PCA)
General Architecture of PCA
A Cross-layer Clustering Algorithm for Ad-hoc Networks

Flow Chart

Phases

Bootstraping Phase

Maintenance Phase

PCA Flow

Broadcast MAC frames

Discover neighborhood

Calculate probability for being cluster head

Is cluster head

Collect information for neighbor clusters

Select the best cluster to join

Arrange routing information

Announce itself as a cluster head

Update application layer for role

Update network layer for new cluster head

Cross-layer Communication

MAC frames forwarded to management layer

Reckoner evaluates probability

Update management layer with neighbors’ information

Update application layer for role

Update network layer for new cluster head

Link to Management

App. to Management

Management to App.

Management to Network
Conclusion and Future Work

- Explained implementation steps of a generic cross-layer framework
- Presented an illustrative use case

*Easy to implement, but..*

- Comparison with other inter-layer communication techniques
- Alternatives in other simulation environments
Questions

Thank you for listening

Cross-layer Stack Design Framework in OMNeT++
OMNeT++ Community Summit

presented by Doğanalp Ergenç

WINS

September 6, 2018
References

OMNeT++-Based Cross-Layer Simulator for Content Transmission over Wireless Ad Hoc Networks. 

Implementation of a Wake-up Radio Cross-Layer Protocol in OMNeT++, MiXiM. 

Cross-layer optimisation for quality of service support in wireless sensor networks. 

Managing cross layer information in OMNeT++ network simulations.