Cross-layer Stack Design Framework in OMNeT++

OMNeT++ Community Summit

Doğanalp Ergenç

WINS

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

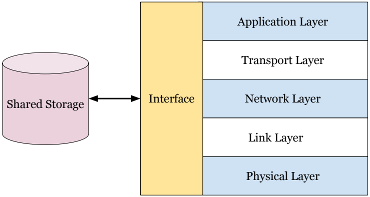
Conclusion and Future Work

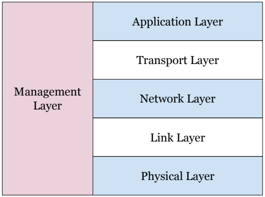
References

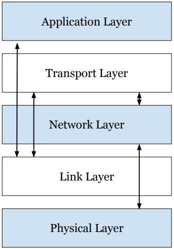
Cross-layer Stack Architecture

- 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


Shared Storage
Interface
Application Layer
Transport Layer
Network Layer
Link Layer
Physical Layer


Management Layer
Application Layer
Transport Layer
Network Layer
Link Layer
Physical Layer


Application Layer
Transport Layer
Network Layer
Link Layer
Physical Layer

Related Work

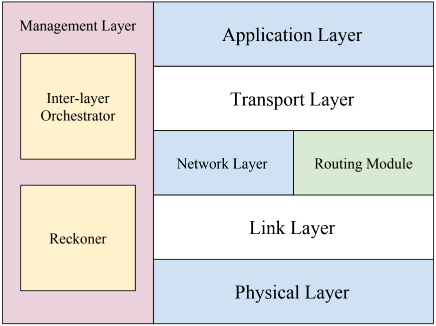
Study	Goal	Solution
Massin *et al.* [1]	Radio access and resource allocation	xLayer 1-2
Lebreton, Murad [2]	Wake-up radio optimization	xLayer 1-2
Mohaghegh *et al.* [3]	Latency in packet processing	xLayer 2-5
Feeney [4]	Information sharing modification	XML content

Motivation

- Present a guideline for the fundamentals of cross-layer structure
- Implement a framework for general use
- Show its actual implication

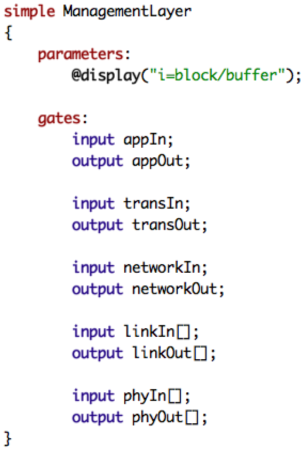
Implementation of Cross-layer Framework in OMNeT++

- 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

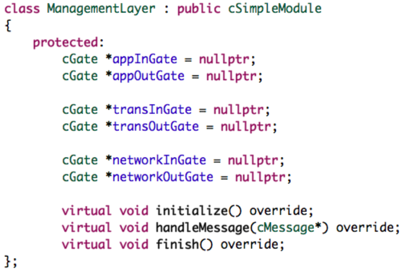
Layer-specific parameters and gates in NED files



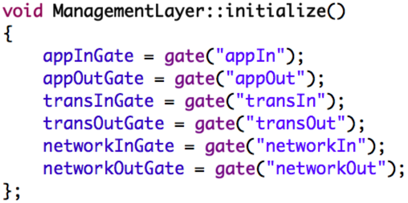
```
simple ManagementLayer
{
    parameters:
        @display("i=block/buffer");

    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++



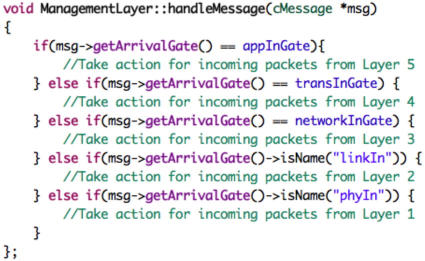
```
class ManagementLayer : public cSimpleModule
{
    protected:
        cGate *appInGate = nullptr;
        cGate *appOutGate = nullptr;

        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;
};
```



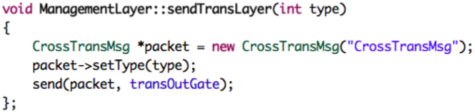
```
void ManagementLayer::initialize()
{
    appInGate = gate("appIn");
    appOutGate = gate("appOut");
    transInGate = gate("transIn");
    transOutGate = gate("transOut");
    networkInGate = gate("networkIn");
    networkOutGate = gate("networkOut");
};
```

Definition of the management layer

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

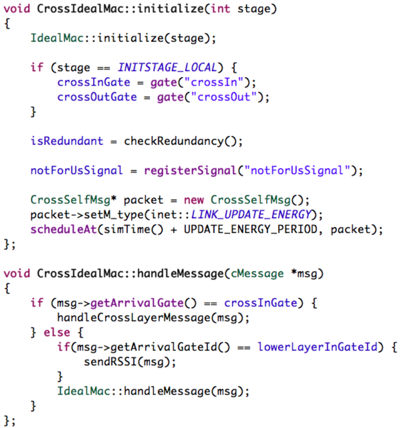


```
void ManagementLayer::handleMessage(cMessage *msg)
{
    if(msg->getArrivalGate() == appInGate){
        //Take action for incoming packets from Layer 5
    } else if(msg->getArrivalGate() == transInGate) {
        //Take action for incoming packets from Layer 4
    } else if(msg->getArrivalGate() == 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
    }
};
```

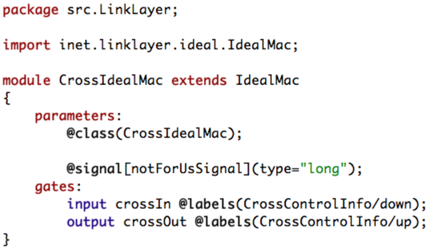


```
void ManagementLayer::sendTransLayer(int type)
{
    CrossTransMsg *packet = new CrossTransMsg("CrossTransMsg");
    packet->setType(type);
    send(packet, transOutGate);
};
```

Extension of other layers

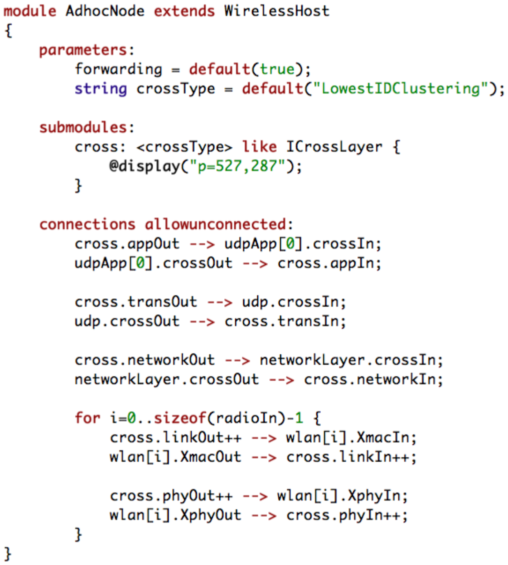


```
void CrossIdealMac::initialize(int stage)
{
    IdealMac::initialize(stage);

    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



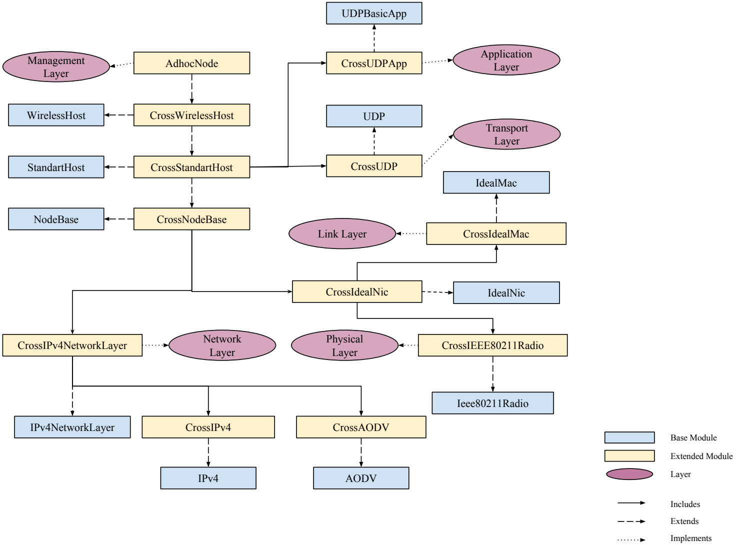
```
module AdhocNode extends WirelessHost
{
    parameters:
        forwarding = default(true);
        string crossType = default("LowestIDClustering");

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

    connections allowunconnected:
        cross.appOut --> 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

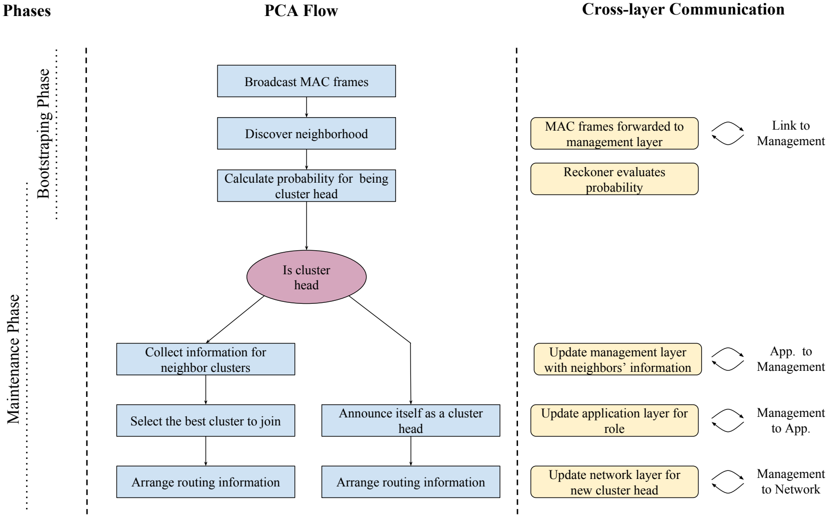
- General architecture
- Flow chart

- **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



Flow Chart


Phases
PCA Flow
Cross-layer Communication
Bootstraping Phase
Maintenance Phase
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
Arrange routing information
MAC frames forwarded to management layer
Link to Management
Reckoner evaluates probability
Update management layer with neighbors' information
App. to Management
Update application layer for role
Management to App.
Update network layer for new cluster head
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

September 6, 2018

?

References

[1] R. Massin, C. Lamy-Bergot, C. J. Le Martret, and R. Fracchia.
OMNeT++-Based Cross-Layer Simulator for Content Transmission over Wireless Ad Hoc Networks.
EURASIP Journal on Wireless Communications and Networking, 2010(1):502549, Jan 2010.

[2] Jean Lebreton and Nour Murad.
Implementation of a Wake-up Radio Cross-Layer Protocol in OMNeT++, MiXiM.
CoRR, abs/1509.03553, 2015.

[3] M. Mohaghegh, C. Manford, and A. Sarrafzadeh.
Cross-layer optimisation for quality of service support in wireless sensor networks.
In *Proc. of the IEEE 3rd International Conference on Communication Software and Networks*, pages 528–533, May 2011.

[4] Laura Marie Feeney.
Managing cross layer information in OMNeT++ network simulations.