Infrastructure for Cross-Layer Designs Interaction

Zhijiang Chang, Georgi Gaydadjiev, Stamatis Vassiliadis
Computer Engineering Laboratory
Technical University of Delft, Mekelweg 4, 2628 CD Delft, the Netherlands
email: [zhijiangchang, georgi, Stamatis]@ce.et.tudelft.nl
Telephone: +31 15 278 6177

Abstract—The current system design of mobile ad hoc networks (MANET), derived from their traditional fixed counterparts, cannot fully meet the requirements inherent to the dynamic nature of such networks. Cross-layer (CL) designs, a modification of the classic protocol stack, are envisioned as a solution for this problem. Many CL design approaches are proposed, each for a different optimization purpose. Mobile terminals require a variety of optimizations that can be provided only by using different CL designs. Consequently, the coexistence and interaction of such designs needs attention. The lack of common interface and infrastructure among different CL designs, however, makes their interaction a significant problem. The proposed common interaction infrastructure is able to compensate for the negative effects introduced by particular CL design by using runtime information from all CL implementations involved in the system. This paper first presents an analysis of the weaknesses of standalone CL designs. Furthermore, this paper introduces a novel architecture to ensure optimal interaction of multiple CL designs according to different QoS requirements. A piggybacked signaling protocol using XML based internal messaging format is introduced. The proposed architecture and three CL designs were simulated for various network scenarios and compared in terms of delay and network efficiency (using the throughput percentage). The results show that the performance with the interaction framework is averagely within 3% to that of the best individual case and maybe up to 4.5% better. In addition, the QoS requirements are always respected.

Keywords: cross-layer design, cross-layer interaction, protocol stack architecture

I. INTRODUCTION

The cross-layer design approach is a promising new paradigm for ad hoc networks. This approach provides new possibilities to increase the performance and adaptability of MANET [1]. At the same time some complexity is introduced. The latter might degrade the stability and security of a communication system. Furthermore, the communication system may require multiple CL designs in order to achieve the overall quality of service (QoS) constrains such as bandwidth and energy [2]. The compatibility and interaction of such designs become a serious issue in wireless network system design. De facto, many organizations would introduce different CL designs into the system without the knowledge of each other. The state of the art in CL designs does not provide an analysis the compatibility and interaction among the coexisting CL designs. Instead, they either concentrate on individual design [3], [4], [5], or provide architectural improvements to enable multiple CL designs working simultaneously [6], [7]. In order to avoid adaptation loops, such architectures consider every CL design as a standalone component and consequently prevent them from interaction. This design principle makes overall QoS control impossible because each entity works alone and lacks a complete view of the QoS requirement matrix.

In this paper, we investigate how CL designs are coupled, in other words, whether the combination introduces some additional positive or negative effects on the overall QoS. Our analysis and the experimental results show that interaction among CL designs is essential for the overall system performance optimization. Our framework is able to optimize the overall system performance in order to benefit from the best CL combination in any particular situation.

The contributions of this paper are:
1. The shortcomings of non-interactive CL designs are analyzed and some general solutions are proposed;
2. A novel infrastructure for CL design interaction that implements the proposed solutions is described;
3. The proposed architecture is validated by simulation using three classical CL designs.

This paper is organized as follows. We outline the related CL designs in section II. The problems of standalone CL designs is analyzed in section III. In section IV we introduce an novel architecture to ensure CL designs interaction and avoid their negative effects on overall QoS. The simulation results are demonstrated in section V. We finally conclude the discussion in section VI.

II. RELATED WORK

This section outlines several well-acknowledged CL optimizations as well as some architectures that allow multiple CL designs coexistence. Previous work has been focusing not only on the CL designs definition, but also on modifying the protocol stack architecture in order to support such designs. In case of multiple CL designs architectures, specific guidelines and customized interfaces are provided in order to implement different cross-layer designs.

A. Cross-layer Designs

We selected the following three CL designs because they use quite different optimization approaches. They are, however, envisioned as the minimal set that is representative for all previous CL proposals.
1) *TCP Congestion Control Optimization:* In the traditional wired communication world, due to the fact the bit error rate (BER) of the link can be neglected, the TCP layer assumes that packet loss indicates network congestion. In the wireless scenario the package loss is mainly caused by errors on wireless link instead of congestion. Many papers have analyzed this problem and proposed different solutions [5], [8]. The general idea is to use MAC layer feedback such as link connectivity to notify the TCP protocol whether network congestion really happens. This can be defined as higher layer optimization using lower layer information.

2) *WIDENS:* Vania Conan et al. proposed in [4] the WIDENS architecture. The WIDENS project is a interoperable Public Safety system. It aims at an efficient self-organized communications infrastructure that anticipates, targets and responds to the future needs of emergency applications and services. In the WIDENS architecture, shared communication channels are established among the data link control layer (DLC), MAC layer and the physical layer of a wireless node. In the IP layer the optimized link state routing (OLSR) protocol is enhanced to update the lower layer state information via the DLC layer protocol and hence has the ability to adapt the routing scheme to the changing wireless link status. This architecture emphasizes the low-level protocol integration by virtually providing a super low layer that combines the DLC, MAC and PHY layers. Special communication channel between the network layer and the integrated MAC/PHY layer is also established in order to support hard QoS routing in MANET. The WIDENS architecture is a low layer optimization regarding wireless characteristics that are interpreted in IP layer and below.

3) *CROSSTALK:* In [3], Rolf Winter et al. proposed the CrossTalk architecture. The CrossTalk is a network-wide cross-layer design focusing on overall throughput optimization. Unlike the above two designs, the CrossTalk architecture emphasizes cross-node cooperation as well as cross-layer design within individual nodes. A key characteristic is that standard, not modified, layered protocols can still work within the framework of CrossTalk and CrossTalk enabled nodes are able to run in a classical layered mode, since the required optimization information is not always available. In other words, the implemented protocols are not dependent on the cross-layer information availability.

### B. Cross-layer Architectures

The cross-layer architectures focus on the coexistence issues, such as adaptation loop avoidance and protocol stack improvement. Well defined architectures are also essential for rapid and systematic deployment of existing and new CL optimizations. This is becoming an important issue because of the increasing diversity and complexity of cross-layer designs that makes the protocol stack modifications very difficult.

1) *ECLAIR:* In [6], the authors introduced a framework named ECLAIR. The ECLAIR architecture provides a function call based solution for CL design implementation. This architecture prohibits adaptation loops by exhaustive listing all the functions that may be called by CL designs and providing registration scheme to control the function calls. ECLAIR is operating system (OS) dependent due to the fact that function implementations are not the same in different operating systems. In [6], the authors verified their architecture with Receiver Window Control. As analyzed in section III, a cross-layer architecture should provide mechanism for multiple designs to interact. The ECLAIR architecture does not clearly state this issue or provide any solution.

2) *Mobileman:* The Mobileman [7] provides a solution that breaks the basic layered architecture. The Mobileman architecture introduces an additional protocol stack component named Network Status. This component is a centralized repository provided for network information sharing among the layers. The Mobileman recommends replacing the standard protocol layers with a redesigned network status-oriented protocol, so that the protocol can interact with the Network Status. This, however, would lead to increasing implementation and maintenance efforts. Furthermore, the protocols need modification if the Network Status is enhanced. Efficiency would be lower and complexity would be higher in architecture. Consequently, Mobileman is not suitable for current system development for pragmatic reasons.

### III. PROBLEMS OF NON-INTERACTIVE CROSS-LAYER DESIGNS

Non-interactive cross-layer designs have their weaknesses. When trying to improve the performance in some aspects using cross-layer designs, however, we may also lose performance in other aspects. Overheads are inevitably introduced into the original system by deploying cross-layer designs. Furthermore, an incautious design may sometime damage the original communication protocol functionality. The problems of non-interactive cross-layer designs and their general solutions can be categorized as follows:

1) *Faulty assumptions:* The cross-layer design also has assumptions as the traditional layered designs. These assumptions may be wrong as well. For instance, the TCP fake congestion avoidance only uses local link information from its MAC layer. If the wireless links of the neighboring nodes are also unstable, the local information is not enough to reflect the real network situation. Consequently by maintaining the speed of sending packages, real congestion is more likely to happen. This problem is inherent for all non-interactive cross-layer designs. Interaction among the cross-layer designs can prevent such problems by sharing their information to provide better picture of the network for each design.

2) *Shared cross-layer data:* The cross-layer data is not only readable but also writable. Stability estimation can be derived by observing the dependency graph. If two entities try to update the same variable on different timescales, they conflict with each other. This problem is inevitable when the number of CL designs introduced into the system is growing. A priority scheme is required to handle this problem. Entities with higher priority can override variable controlled by entities with lower priority. An expiration timer can be attached to the
variable so that the low level entities are still able to update the information.

3) **Dependency loops**: The CL designs often cause adaptation loops that are parts of different protocols to interact with each other. This problem is complicated and its prevention requires significant efforts during design. Designers of cross-layer architectures need to take this into account. One solution is that the cross-layer designs follow some design guidelines so that in each round of data manipulation a cross-layer design is only invoked once. The disadvantage of this approach is that the cross-layer designs may react slower to external environmental changes.

4) **Additional overhead**: CL designs introduce internal overhead within a node as well as external overhead on the network. Additional overhead is introduced when one tries to improve the accuracy of the cross-layer information. The external overhead is calculated as the number of bytes added to normal packages to carry cross-layer information. The internal overhead can also be evaluated in terms of the size of cross-layer information. These overheads can be explicitly measured. On the other hand, some overhead introduced by cross-layer designs is difficult to define and evaluate. For example, some functions may be invoked by cross-layer designs and consume system resources. Without a system that supervises all the CL designs, no design has a clear picture of the overall resource consumption. There should be an interaction control middleware that supervises the tradeoff between the accuracy of information and system overheads and ensures the correctness of each design.

IV. NOVEL CROSS-LAYER DESIGN INTERACTION INFRASTRUCTURE

This section introduces a novel interaction infrastructure that focuses on solving the problems of standalone CL designs described in the previous section. Our proposal enables not only interaction among different CL designs but also interaction between CL designs and the proposed control middleware. This optimally prevents the faulty assumption problem. Priority control mechanism supported in the architecture solves the shared cross-layer data problem. The architecture uses piggybacked signaling to minimize the internal and external overheads. The CL designs can avoid dependency loops by following the design guidelines provided by the architecture. The eXensible Markup Language (XML) format is used to describe the different CL designs and their interactions.

**A. Proposed Interaction Architecture**

We propose a centralized control architecture to solve the CL compatibility and interaction problems. This architecture introduces two additional system components, centralized control middleware and interaction interfaces at each layer. After studying many cross-layer designs, we found out that most of the information flows are from lower layers such as IP, MAC and PHY to upper layers such as TCP and applications. The interactions of cross-layer designs mostly happen in MAC and network layers. Therefore, our control middleware is located in the MAC layer, which has a well-defined interface with the above network layer. The control middleware is made aware of the required parameters by each CL design through a dedicated registration procedure by using the regular data propagation. Thereafter the combined parameter list is sent to all other protocol stack layers again using the regular data traffic. The additional interaction interfaces at each layer configure and start adding their specific parameters only to the packages sent from that particular layer. The cross-layer information (registration and run time data) is piggybacked at the end of the regular packages and propagated among the layers by using the normal layered messaging procedure. Therefore, the proposed cross-layer architecture does not violate the OSI model, which makes our proposal backward compatible with the standard protocol stack. In addition, such standard interaction interface will reduce the complexity of cross-layer designs that need access to different parameters at different layers.

**B. Interaction and Priority Control**

The proposed architecture supports interaction among CL designs and the control middleware. The interaction mechanism uses priority control to handle the shared cross-layer data problem. Two kinds of interaction are supported in the proposed architecture:

1) **Interaction between CL designs**: Because some optimizations have better information about the network situation, they should control the behavior of other CL designs. For instance, any optimization that supervises the link quality of neighboring nodes should be able to control the classic TCP fake congestion avoidance optimization that only holds the local link information. The reason is that if link condition of neighboring nodes is not good, errors are more likely to happen on the route.

2) **Interaction between the control middleware and a CL design**: The control middleware supervises the overall QoS. A certain CL design may break some QoS requirements when trying to achieve optimization in other aspects. Furthermore, each design introduces overhead. The control middleware should suppress the cross-layer design in case that the overhead cannot be tolerated.

The current system supports two interaction commands: *suppress* and *resume*. The suppress command means one CL
design or the control middleware can suppress some other
designs in order to match QoS requirements or to avoid
negative impact caused by them. The resume command means
the control middleware or some designs can ask the suppressed
CL design to resume action. The control middleware provides
interfaces for cross-layer designs to interact, while holding the
right to make the final decision. In order to achieve the above
interaction protocol, the control middleware and cross-layer
designs are categorized into the following four priority levels
(listed in order of importance):

Level 1: The control middleware: The middleware can
access all the information provided by each individual cross-
layer design as well as the overall QoS requirements. Conse-
quently it has the highest priority in the architecture.

Level 2: CL designs with global (network) knowledge: The
global or the network knowledge describes the most important
information for network communication. The network opti-
mizations have higher priority than local ones because we try
to optimize the overall behavior of the network instead of a
single node only.

Level 3: Local lower layer CL designs. The lower layer
designs optimize the point-to-point performance. As we know,
the MAC and PHY layer protocols supervise the behavior of
one-hop wireless communication. We also include IP layer
optimization in this level because the dynamic topology is
also the nature of MANET. Furthermore, the wireless channel
condition heavily affects the performance of routing protocols.

Level 4: Local higher layer CL designs. This refers to
TCP and application layer optimizations. At this level the
CL design considers the underlying network transparent. The
major source of problems in wireless networks, e.g. unstable
link quality and dynamically changing topology, are taken care
of by CL designs located in the lower (IP, MAC and PHY)
layers. Therefore, the end-to-end optimizations have the lowest
priority.

C. Cross-layer Communication

The proposed architecture uses the signaling approach to
pass information among layers. The information is piggy-
backed at the end of internal packages and propagated together
with the package headers and original payloads. This approach
utilizes the standardized layered structure, which is not op-
erating system dependent. Therefore, the The CL designs at
each layer can understand the information regardless of their
different implementations.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Comparison of Communication Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>function call</td>
</tr>
<tr>
<td>format</td>
<td>NA</td>
</tr>
<tr>
<td>overhead</td>
<td>low</td>
</tr>
<tr>
<td>access method</td>
<td>API</td>
</tr>
<tr>
<td>complexity</td>
<td>high</td>
</tr>
<tr>
<td>delay</td>
<td>low</td>
</tr>
</tbody>
</table>

Two previously reported propagation approaches: the func-
tion calls and the local profiling approaches were compared
with the signaling approach [6], [7] in table 1 using five
criteria. The function call method has smaller overhead. But
this approach is operating system dependent, a feature that
all communication systems try to remove. The advantage of
the profiling approach is that no signaling is needed. The
disadvantages are: 1) complex read/write control must be
applied in order to avoid dead lock; 2) delay is considerable if
the profile is stored in low speed memory. The packages used
in the signaling approach, on the other hand, are always in
memory. Because the information carried by packages travels
from layer to layer, the signaling approach (and hence the one
proposed here) has longer delay compared to the function call
approach. This difference, however, cannot be easily simulated
because for the best of our knowledge all simulators assume
the internal package propagation delay as zero. A possible
mathematical evaluation of this problem is presented hereafter.
The total delay \( D \) of messages passing from layer 1 (the
source) to layer \( n \) (\( n \leq 5 \), the destination) is the sum of all
delays at all layers and interfaces involved. For each layer
the delay can be represented by two parts, the propagation
delay \( D_l \) from layer \( i - 1 \) to layer \( i \), and the internal processing
delay \( D_{pi} \). Assuming the propagation delays between any two
layers are the same \( D_l \), the total delay can be represented as:
\( D = (n - 1)D_l + \sum_{n=1}^{n-1} D_{pi} \). Considering the fact that \( n \) is a
small integer (\( n \leq 5 \)) and that usually \( D_l \) is very low, the total
processing delay will be the dominating factor. In case of a
state of the art mobile node (as known in 2007), this overhead
is expected to be negligible.

In contrast to the other two approaches, the signaling
mechanism (used also in our proposal) is the standard for
network communication, for terminal nodes such as mobile
devices and for intermediate nodes such as routers. This
makes the signaling approach universally applicable despite of
the differences among operating systems and the differences
between software and hardware of various wireless system
nodes.

D. Information and the Encoding

Two types of information are described in our architecture:
variables and special commands. The special commands are
the control signals in our architecture that support registration
of the CL designs and their interaction. The properties of a
variable include name, the layer it comes from, the last entity
that updates it (owner) and the priority level of the owner.
In addition, each variable value has an expiration time. This
mechanism ensures the lower priority designs have the chance
to control the variable after the old value expires even if it
was updated by a design with higher priority. The properties
of a special command include name of the command such
as \textit{register} and \textit{suppress} and \textit{resume}, object of the command,
name of the command sender (owner) and the owner’s level.
The object parameter of a command contains the command
sender name in case of a registration request, or the destination
CL design identifier in case of an interaction command.

The proposed infrastructure uses the eXtensible Markup
Language (XML) format to describe the layer parameters,
the CL designs, and their interactive relationships. XML is
the dominating formatting pattern in the Internet nowadays for its universal readable nature and capability to describe metadata as well as profile data. XML provides an application independent way of sharing data using Document Type Definition (DTD) to define the legal building blocks of an XML document. The independent entities can use a common DTD for interchanging data. Therefore, the XML format is suitable for our interaction infrastructure that supports such information exchange among CL designs.

Therefore, the full descriptions of a variable and a control command in XML format are:

```
<var> <varName>value</varName>
<owner>level0Owner</owner> </var>
<cntl> <command>object</command>
<owner>level0Owner</owner> </cntl>
```

This information encoding also enables the designs to observe each other’s behaviors. Therefore any new CL design is able to “cooperate” with the already installed ones based on their runtime information.

The possible weaknesses of this design are the need of XML parser and system overheads to transmit the ASCII based information. In fact, the Java 2 Micro Edition (J2ME) and Microsoft Windows CE platforms already have XML supported on handheld products [9], [10]. According to [10], the size of XML parsers for J2ME varies from 6KB to 14KB, which is negligible compared to most mobile applications found in an average mobile node existing in 2007. In our architecture, the XML information is transmitted only inside the node, which means it does not occupy any network resource. Furthermore, if we have a unified data presentation and handling method, such as XML, the system resource can be reused.

### E. Guidelines for Cross-layer Designs

In order to work in the interaction architecture and avoid adaptation loops, the CL designs need to follow some guidelines. The guidelines are made based on the principle that each design first reads parameters and performs local calculation, then writes parameters that change the system performance. Each design behaves according the following three phases on each round of data manipulation:

**Phase 1:** Registration phase. If not registered yet, the CL design first registers to the control middleware using a registration request. The request also describes the required parameters. The middleware checks a set of criteria in order to grant this request. The DTD file that explains the information format is sent to the cross-layer design if the request is granted.

**Phase 2:** Decision making phase. The CL design collects information and performs local calculation, instead of writing any parameter that is visible to the layers or has any effect on the performance.

**Phase 3:** Action phase. The CL design first checks if it is suppressed. If not, the design updates the parameters that affect the optimization if 1) it has higher priority than the current owner of the parameter or 2) the value of the parameter already expires. In this phase, if the design decides to interact with other designs or the control middleware, a special request is sent to the middleware. The middleware can either grant the request by generating an interaction command to the desired CL design, or consume the request according to global QoS requirements.

### V. Validation And Results

In order to validate our interaction architecture, we implement three CL optimizations as well as the interaction architecture in the Network Simulator 2 (ns-2) version 2.28 [11]. Three different CL optimizations were selected corresponding to the three priority levels as introduced in section IV-B with the exception of Level 1 that belongs to the control middleware. The selected optimizations have one-to-one interaction relationship. In addition, each of them collaborates with our control middleware.

**Optimization 1:** The integrated MAC/PHY layer. This is local lower layer optimization. The integrated layer provides better information about the wireless link, and consequently provides better information to other CL designs that use MAC layer information. By introducing references to physical layer functions in MAC layer, the MAC layer knows information such as SNR and number of packages that fall into noise level. This optimization provides information to optimization 2 and 3. It is registered and activated from the beginning of the simulation.

**Optimization 2:** The TCP congestion optimization. This is local higher layer optimization. The idea is to use local MAC layer feedback to distinguish package loss caused by network congestion and link errors. The MAC to TCP layer feedback is generated if the retry count in function RetransmitDATA() or function RetransmitRTS() has exceeded their thresholds. The standard TCP is used in this experiment. On receipt of the MAC layer feedback, the function slowdown() in TCP layer does nothing in order to keep the congestion window size. This optimization is registered and activated from the beginning of the simulation. It can be suppressed by optimization 3.

**Optimization 3:** The global load balancing optimization using one-hop neighbor’s link information. This is global (network) optimization. The general idea is to avoid nodes with lower SNR or more collisions. The local view is represented by the received signal strength and degree of collision. These two parameters are used to determine the network density and the interference among the neighboring nodes in a single hop range. In the ns-2 simulator, collision can be detected if MAC function collision() is called. The number of packages that collide is also available. The local views are piggybacked at end of MAC layer messages and propagated to neighboring nodes. The global view is calculated using the intuitive weighted average method introduced in [12]. Overhearing is not implemented. The AODV and OLSR use the views in their route request procedure. If the local or global view indicates congestion or low SNR, the AODV delays processing the routing request; the OLSR sets the willingness to join an MPR to WILL NEVER. Consequently the nodes with stronger SNR and less interference are more likely to join the route. This optimization is registered and activated from the beginning of the simulation. This optimization can be
throughput suppressed by the control middleware. The control middleware tries to maintain the QoS requirement that the overall network overhead introduced by CL designs is smaller than 7%.

In our experiments we used Ad hoc On-demand Distance Vector (AODV) [13] and Optimized Link State Routing (OLSR) [14] as the routing protocols in order to investigate the behavior in both cases, when proactive or reactive RPs are used. The NS-2 OLSR patch as provided in [15] was used. For the simulation we used the following network scenario:

- mobility model: random waypoint [16];
- number of nodes: 50;
- area: 500m × 500m;
- node speed: 0 to 20m/s;
- data sources: 25 FTP on TCP, and 10 CBR on UDP;
- test duration: 600s, pause time is the time when the nodes are stationary;

Four different scenarios: three different CL design combinations and the no CL optimization case were studied. NO CL representing the absence of CL optimizations, COMBINATION 1 that involves optimizations 1 and 2, COMBINATION 2 that uses optimizations 1 and 3, and finally OUR PROPOSAL that involves all three optimizations together with the proposed interaction infrastructure.

The traffic patterns and node movement are generated using CMU’s traffic and scenario generating scripts in ns-2 [11]. All results reported are the arithmetic mean of the results from 100 different simulations.

Our goal was not to evaluate the performance of any individual CL design, rather to validate the proposed interaction mechanism. In addition, complete (using the full property matrix) QoS driven control algorithm as reported in [1] is not considered in this work.

The simulation results in figures 4, 5, 2 and 3 show that the performance of multiple CL designs under our architecture lies within 2.9% of the best cases on average. In some cases, the performance is up to 4.5% better and in the worst case can become 20% worse. The internal data overhead introduced by our proposal is 5% (maximum) and was estimated based on the ratio between the size of the complete packages and the additional piggybacked information. The following facts are supported by the simulation results:

1) The interaction between two CL designs according to the proposed priority scheme is validated: As shown in figure 5 when pause time \( \leq 60 \) s, the optimization 2 has negative impact on delay. This is because the proactive OLSR already introduces a lot of routing control messages to the network, which leads to more collision and consequently more retransmissions on the MAC layer. As shown in figure 3, the optimization 2 also has negative impact of throughput percentage when pause time \( \leq 30 \) s. This is because the route break during transmission is serious with OLSR, which leads to more fail delivery. With our proposed interaction architecture, the optimization 3 suppresses the optimization 2 in both cases based on the collision and channel information collected from
the TCP source’s neighbors. Consequently, the result of our proposal is better than that of optimization 2.

2) The interaction between the control middleware and CL designs according the QoS requirements is validated: As shown in figure 3, the overhead of optimization 3 increases as more packages, such as MAC control messages RTS/CTS and periodical routing messages of the OLSR, are in the network.

The situation is deteriorated when the network becomes stationary (pause time $\geq 120$s) and the throughput significantly increases. The control middleware successfully suppresses this optimization when learning that the overhead is higher than 7%. This information is calculated by optimization 3 and delivered to the interaction system. The simulation results also show that because the control middleware suppresses the optimization 3 for a period of time to decrease the overhead, the performance of our proposal is normally between those of the combination 1 and 2.

3) The architecture does not degrade the performance of the original protocols or individual CL designs: In figure 2, it can be observed that the AODV performs almost equivalently with or without CL designs in terms of throughput percentage when network topology is changing quickly (pause time $\leq 30$s). This is because the hop-by-hop routing nature of AODV already prevents most route break during transmission. Therefore, optimization 3 is not very useful in this situation.

When the network becomes more stationary, the optimization 3 improves the delivery percentage by balancing the load as more data packages are in the network. From figure 4 and 5, we learn that the optimization 3 always has good performance on delay optimization for both reactive AODV and proactive OLSR. This is an effect of a more evenly distributed load in the network, which reduces the average queuing delay of each package.

4) The proposed architecture has particular weaknesses: The worst case with our interaction architecture is shown in figure 5, when pause time is 30 seconds. The performance of our proposal is 20% worse than the best case (combination 2). In this case, because the network situation is alternating between stationary and not stationary frequently, the adaptation algorithm used in optimization 3 may not be quick enough to detect those changes. Therefore, the optimization 2 is not suppressed every time it should be. The negative impact of the optimization 2 is not fully removed. However, the accumulated performance under our architecture is still better than that without any CL design.

VI. CONCLUSION AND FUTURE WORK

This paper presents a novel infrastructure that supports the interaction and priority handling among CL designs. The proposal improves the system performance when considering the QoS requirements. The interaction control middleware and CL designs with higher priority are given the control over lower priority designs. By doing so, some negative effects of CL designs are detected and timely avoided. The standard signaling approach used for data distribution and the XML control commands make our proposal universal, OS independent and highly modular. In addition, our scheme will provide all CL designs with the complete system information. Therefore newly introduced CL designs can be compatible with the rest of the system. A set of design guidelines to prevent adaptation loops are also provided in this paper.

In the future, we would focus on integrating more complicated CL designs into the proposed architecture, for example, energy reservation designs that involve all layers in the protocol stack. Another research point is to provide a quantitative approach to generate interaction commands for individual cross-layer design using the global QoS requirements. There is also an activity to implement the architecture on the Linux kernel so that the interaction system could be further validated and evaluated.

ACKNOWLEDGMENT

This work is supported by the Dutch Government as part of the decree on subsidies for investments in the knowledge infrastructure (Bsing) program. This work is done within the Micro Satellite (MISAT) project.

REFERENCES