Novel Packet Switching for Green IP Networks

Jo, Seng-Kyoun

2017-04

Downloaded from Heldas, University of Helsinki institutional repository.

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.
Please cite the original version.
A green technology for reducing energy consumption has become a critical factor in ICT industries. However, for the telecommunications sector in particular, most network elements are not usually optimized for power efficiency. Here, we propose a novel energy-efficient packet switching method for use in an IP network for reducing unnecessary energy consumption. As a green networking approach, we first classify the network nodes into either header or member nodes. The member nodes then put the routing-related module at layer 3 to sleep under the assumption that the layer in the OSI model can operate independently. The entire set of network nodes is then partitioned into clusters consisting of one header node and multiple member nodes. Then, only the header node in a cluster conducts IP routing and its member nodes conduct packet switching using a specially designed identifier, a tag. To investigate the impact of the proposed scheme, we conducted a number of simulations using well-known real network topologies and achieved a more energy-efficient performance than that achieved in previous studies.

Keywords: Green network, Power consumption, IP network, Network node clustering, Packet switching.
such as Network Function Virtualization [11], the purpose of which is the use of IT virtualization-related technologies to virtualize entire classes of network node functions into building blocks that may be connected together to create a communication service.

This study may be considered as a green networking approach to energy-efficient packet switching. Specifically, we propose an algorithm to minimize the power consumption of a network during off-peak periods in an IP network. Assuming that the functions of each layer in a network node can be independently controlled, the proposed algorithm first selects header nodes (HNs) to manage general IP packet processing and manages subordinate member nodes (MNs) that will temporarily suspend IP packet processing at layer 3. In addition, instead of IP routing, each MN will conduct packet switching using a specially designed tag.

The reminder of this paper is organized as follows. Section II provides an overview of related works, and the problem formulation is presented in Section III. Section IV provides the network configuration and packet switching algorithm. A performance analysis using a simulation is developed in Section V, and the results and conclusions are presented in Section VI.

II. Related Works

Energy efficiency is not a new concept, and it has been an important topic of wireless environments such as sensor networks, IoT, and mobile communication. A significant amount of research has been expended to extend the battery life of various devices. The authors in [12] analyzed the trade-off between spectrum efficiency and energy efficiency and introduced various studies within a framework of energy-efficient resource allocation in a 5G wireless network. In the area of MWSN, the authors in [13] designed a clustering algorithm to optimize the energy efficiency of mobile sensor nodes; this algorithm is employed to find the near-optimal threshold for residual energy and to reconfigure clusters. However, due to the recent exponential growth in the volume of network traffic, the spread of access to the Internet, and the expansion of new ICT services offered by service and network providers, energy efficiency has also become a high-priority goal of wired networks.

To analyze green technologies in a wired IP network, the authors of [14] introduced base approaches such as re-engineering, dynamic adaptation, and sleeping/standby. A large amount of research related to these issues has been performed during the last several decades.

As a re-engineering approach focusing on designing energy-aware elements, the authors in [15] introduced an investigation of the potential savings through power-aware network design and routing. The authors measured the power consumption of various configurations of widely used routers. Based on measurement results, they then created a general model for router power consumption to explore the potential impact of power-awareness in a simple space of example networks.

To achieve a dynamic adaptation, approaches have been formulated with an objective to modulate the capacities of packet processing engines and a network interface to satisfy actual traffic loads and requirements. The authors in [16] proposed a novel approach that involved switching off some portions of the Universal Mobile Telecommunication System core networks while still guaranteeing full connectivity and maximum link utilization. As an extension of [16], the authors of [2] proposed a simple algorithm to power off the links and full routers while maintaining the QoS constraints, such as maximum link utilization. As an example of dynamic adaptation, the GreenOSPF algorithm, presented in [17], extended the existing OSPF protocol slightly to share the shortest path tree of a specific node with neighboring nodes, and it solves the above problem by switching off network elements that are excluded by the shared shortest paths. The authors in [18] proposed reducing the router power consumption using a machine-learning technique to identify and quickly adapt to the changing traffic characteristics.

Finally, sleep/standby approaches are used to selectively drive network equipment into low standby modes, and to wake them up only when necessary. The authors in [19] investigated the effect of network connectivity proxy on energy saving by allowing idle hosts to enter a low-power sleep state while maintaining a full network presence.

Some interesting methods, focused on specific applications such as content delivery, have recently been proposed to reduce the power consumption in networks. The authors in [20] proposed an energy-efficient framework for content delivery in networks to support multicast transport and in-network caching. In [21], the authors also provided a survey of energy-efficient caching techniques in an information-centric network from the place, content placement, and request-to-cache routing aspects for a green ICN.

III. Problem Formulations

1. Network Energy Consumption Problem

An energy-consumption minimization network problem can be formulated using linear programming [22]. Let us consider a direct graph \( G(V, L) \), where \( V \) and \( L \) are the set of nodes and directed links, respectively. A link that connects node \( i \) to its neighbor node \( j \) is denoted by \( l_{ij} \). The set of neighbors of node
2. Reformulation of Network Energy Consumption Problem

In order to reformulate the original energy-consumption minimization network problem to be feasible, we introduce several primitives related to network clustering and a lemma. Our goal is to minimize the total power consumption of a network; we begin by comparing the power consumption computed prior to clustering with the power consumption computed after clustering. Prior to clustering, all nodes and links should be turned on, and in this case, the power consumption prior to clustering is as follows.

\[
PC = \sum_{i} e'(i) + \sum_{j} e'(l_j). \tag{4}
\]

On the other hand, based on the traffic change, several nodes and links can be switched off after clustering and the power consumption \(PC_{\text{clustering}}\) can be less than \(PC\). As a result, we can reduce the power consumption by reducing the difference between the power consumption before clustering and the power consumption after clustering. This implies that our goal can be referred to as “maximize the power consumption reduced by clustering,” which is defined as \(\max(PC - PC_{\text{clustering}})\).

Let us now suppose that the network consists of \(K\) clusters and \(l^K\) is a set of nodes that consists of one HN and one or more MNs.

\[
V^1 \cup V^2 \cup \ldots \cup V^K = V.
\]

Then, the number of HNs and the total power consumption of cluster \(k\) \((PC_k)\) is as follows.

\[
n^{\text{HN}} = N - \sum n_{\text{MN}}^k, \quad (5)
\]

\[
PC_k = e'(i) + n_{\text{MN}}^k e'(i) + \sum_{t=1}^{l_{\text{MN}}} \sum_{j \in l_{\text{MN}}} \gamma e'(l_j), \tag{6}
\]

where \(N\) is the number of nodes in the network and is equal to \(n^{\text{HN}} + n_{\text{MN}}^k e'(i) : V \rightarrow R^+\) is the power consumption of an MN in which the IP-routing related operation at layer 3 is temporarily put to sleep; \(n^{\text{HN}}, n_{\text{MN}}^k\) is the sum of the number of HNs in the network and MNs in cluster \(k\), respectively; and, \(\gamma\) is a binary variable \(\{0, 1\}\) if \(l_{\text{MN}}\) which is initialized to 1, is powered on.

In order to calculate the effect of clustering, we divide the power consumption of the network into the power consumption of the nodes and links. First, from the node energy perspective, the power consumption of nodes \((PC^v)\) before clustering is as follows.

\[
PC^v = \sum_{i} e'(i) = Ne'(i). \tag{7}
\]

The power consumption of the nodes after clustering \((PC^v_{\text{clustering}})\) is as follows.

\[
PC^v_{\text{clustering}} = \sum_{i} e'(i) + \sum_{j \in l_{\text{MN}}} \gamma e'(l_j). \tag{8}
\]

Accordingly, the amount of power consumption of the nodes reduced by clustering can be obtained as follows by using (5), (7), and (8).

\[
Ne'(i) - (n^{\text{HN}} e'(i) + n_{\text{MN}}^k e'(i)) = n_{\text{MN}}^k (e'(i) - e'(i)). \tag{9}
\]

Now, from the link energy perspective, we investigate the difference between the power consumption prior to clustering and after clustering. The total power consumption of the links prior to clustering is as follows.

\[
PC^l = \sum_{i} e'(l_j). \tag{10}
\]

Using (6), the total power consumption of links within cluster \(k\) after clustering is

\[
PC^l_{k} = \sum_{i=1}^{l_{\text{MN}}} \sum_{j \in l_{\text{MN}}} \gamma e'(l_j), \tag{11}
\]

where \(\gamma\) is also a binary variable \(\{0, 1\}\), the value of which specifies whether \(l_{\text{MN}}\) is powered on or off. Note that (11) includes only links within the cluster. In order to consider the power consumption of the link after clustering, we need to consider the links that connect the clusters; this can be expressed as follows:

\[
PC^l_{\text{clustering}} = \sum_{k} \left( PC^l_{k} + \sum_{j \in l_{\text{MN}}} \delta e'(l_j) \right), \tag{12}
\]
Then, the number of links among the clusters can be aggregated on the link \( l_{ij} \) first, and then switching off the other links, \( l_{ij} (\neq l_{ij}) \). Then, the number of links among the clusters can be reduced by switching off \( l_{ij} (\neq l_{ij}) \).

Accordingly, our goal to minimize the original problem in (1) is to reduce the number of active links among clusters that maximizes (15) while satisfying the flow conservation constraints described in (2) and (3).

IV. Network Node Clustering and Packet Switching Algorithm

1. Partitioning of an Entire Network into Clusters

The objective of our research—the creation of a packet switching algorithm—is presented in this section along with several detailed procedures. First, we classify all network nodes into either HNs or MNs. An HN is a general IP router for IP packet processing, and an MN is a special node (or router) that can perform IP packet switching using a tag whenever its function at layer 3 is in the sleep mode. Routing-related functions at layer 3 include routing and packet forwarding engines, such as IP header processing and quality of service queueing. Figure 1 illustrates the logic for energy-efficient packet switching, including the node architecture of the HNs and MNs.

A. Header Node Selections

We will begin with the simple method described in Algorithm 1 that describes the selection of HNs and configuration of clusters using the selected HNs and MNs. Regarding the selection of HNs, various criteria can be considered to define the selection policies, such as the power consumption of the network node, the network nodality, and end-to-end delay. In this study, the traffic carried on \( l_{ij} (\neq l_{ij}) \), to be aggregated on the link \( l_{ij} (t \in l_{ij}) \) first, and then switching off the other links, \( l_{ij} (\neq l_{ij}) \). Then, the number of links among the clusters can be reduced by switching off \( l_{ij} (\neq l_{ij}) \).

Accordingly, our goal to minimize the original problem in (1) is to reduce the number of active links among clusters that maximizes (15) while satisfying the flow conservation constraints described in (2) and (3).

IV. Network Node Clustering and Packet Switching Algorithm

1. Partitioning of an Entire Network into Clusters

The objective of our research—the creation of a packet switching algorithm—is presented in this section along with several detailed procedures. First, we classify all network nodes into either HNs or MNs. An HN is a general IP router for IP packet processing, and an MN is a special node (or router) that can perform IP packet switching using a tag whenever its function at layer 3 is in the sleep mode. Routing-related functions at layer 3 include routing and packet forwarding engines, such as IP header processing and quality of service queueing. Figure 1 illustrates the logic for energy-efficient packet switching, including the node architecture of the HNs and MNs.

A. Header Node Selections

We will begin with the simple method described in Algorithm 1 that describes the selection of HNs and configuration of clusters using the selected HNs and MNs. Regarding the selection of HNs, various criteria can be considered to define the selection policies, such as the power consumption of the network node, the network nodality, and end-to-end delay. In this study, the traffic carried on \( l_{ij} (\neq l_{ij}) \), to be aggregated on the link \( l_{ij} (t \in l_{ij}) \) first, and then switching off the other links, \( l_{ij} (\neq l_{ij}) \). Then, the number of links among the clusters can be reduced by switching off \( l_{ij} (\neq l_{ij}) \).

Accordingly, our goal to minimize the original problem in (1) is to reduce the number of active links among clusters that maximizes (15) while satisfying the flow conservation constraints described in (2) and (3).

IV. Network Node Clustering and Packet Switching Algorithm

1. Partitioning of an Entire Network into Clusters

The objective of our research—the creation of a packet switching algorithm—is presented in this section along with several detailed procedures. First, we classify all network nodes into either HNs or MNs. An HN is a general IP router for IP packet processing, and an MN is a special node (or router) that can perform IP packet switching using a tag whenever its function at layer 3 is in the sleep mode. Routing-related functions at layer 3 include routing and packet forwarding engines, such as IP header processing and quality of service queueing. Figure 1 illustrates the logic for energy-efficient packet switching, including the node architecture of the HNs and MNs.

A. Header Node Selections

We will begin with the simple method described in Algorithm 1 that describes the selection of HNs and configuration of clusters using the selected HNs and MNs. Regarding the selection of HNs, various criteria can be considered to define the selection policies, such as the power consumption of the network node, the network nodality, and end-to-end delay. In this study, the traffic carried on \( l_{ij} (\neq l_{ij}) \), to be aggregated on the link \( l_{ij} (t \in l_{ij}) \) first, and then switching off the other links, \( l_{ij} (\neq l_{ij}) \). Then, the number of links among the clusters can be reduced by switching off \( l_{ij} (\neq l_{ij}) \).

Accordingly, our goal to minimize the original problem in (1) is to reduce the number of active links among clusters that maximizes (15) while satisfying the flow conservation constraints described in (2) and (3).

IV. Network Node Clustering and Packet Switching Algorithm
we assume that the utilization of network nodes is applied to the HN selection and node clustering, as shown in Fig. 2.

Algorithm 1 describes the HN selection procedure. First, each node sends a message including the utilization information of a node and link to its neighbor nodes. Upon receiving this message, each recipient node inserts the neighbor information into the neighbor table. After sending and receiving this message, each node calculates the difference in utilization between nodes. Finally, nodes are selected to be either HNs or MNs through a comparison with the threshold. The threshold is a predefined value determined by the network operator and is used as the basis for an HN decision. A lower value of the threshold will result in a greater number of HNs.

**Algorithm 1.** Pseudo code for header node selection

```plaintext
Algorithm 1: Pseudo code for header node selection

\[ u_i \] network utilization of node \( i \)
\[ u_{ij} \] network utilization of link from \( i \) to \( j \)
\[ \text{node} \] adjacent node of node \( i \)

1: for \( i = 1 \) to \( N \) do
2:    sendToNeighbor (\( \text{node}_{i \rightarrow j} \), \( u_i \), \( u_{ij} \))
3:    receiveFromNeighbor (\( \text{node}_{j \rightarrow i} \), \( u_j \), \( u_{ji} \))
4:    insertNeighborTable (\( \text{node}_{i \rightarrow j} \), \( u_i + u_{ij} - u_j + u_{ji} \))
5: end for
6: CalUtilGap (\( i \)) = \sum_k (u_i + u_{ij} - u_j + u_{ji})
7: if \( \text{CalUtilGap (i) } \geq u_{TH} \)
     \( u_{TH} \) is pre-defined threshold value by network provider
    then
     node \( i \) is selected to be a Header Node(HN)
    else
     node \( i \) is selected to be a Member Node(MN)
8: end if

B. Clustering with One HN and Multiple MNs

After the selection of the HNs is complete, each HN sends advertising messages to its adjacent nodes. Once neighbor nodes receive a message from an HN, the HN updates its neighbor table and selects the HN that is the least utilized among multiple HNs. Once the nodes receive an advertising message from some HN, they then discard subsequent advertising messages. The clustering procedure is described in Algorithm 2.

**Algorithm 2.** Pseudo code for clustering

1: for header node \( i = 1 \) to \( i = N \) do
2:    sendToMN (\( \text{node}_{i \rightarrow j} \), \( u_i \), \( u_{ij} \))
3:    MNreceiveFromHN (\( \text{node}_{j \rightarrow i} \), \( u_j \), \( u_{ji} \))
4:    updateNeighborTable (\( \text{node}_{i \rightarrow j} \), \( u_i + u_{ij} \))
5: end for
6: MNjoinsHN (\( \text{node}_{i} \), \( \text{node}_{j} \))

Member node \( j \) selects the least utilized header node \( i \).

C. Tag Allocation and Distribution

After clustering, the MNs send routing information to their HN, and the HN adds this routing information to the routing entry to identify the arrival reachability through the MNs. In this step, there can be several routing entries that support the same next hop. After adding the MN information, the HN selects the shortest and non-overlapping next hop among multiple paths targeted for the same destination, and it updates the routing entry. The HN then assigns tags to the next hop that can be reached through the MNs, and the tags are distributed to cluster the MNs, as illustrated in Fig. 3.

Rather than using IP routing, we use a tag, which is a specially designed identifier within a cluster, for switching that includes the outgoing node interface. Even though the label defined in the Multi-Protocol Label Switching (MPLS) architecture [23] can be used for packet switching, we propose a new and simple identifier in order to solve the problem defined in Section I. Upon receiving tags from an HN, the MNs temporarily put the routing-related function in layer 3 to sleep and perform packet switching using a tag. If the MNs receive packets without a tag, the packets are directly forwarded to the cluster HN.

**Algorithm 2.** Pseudo code for clustering

1: for header node \( i = 1 \) to \( i = N \) do
2:    sendToMN (\( \text{node}_{i \rightarrow j} \), \( u_i \), \( u_{ij} \))
3:    MNreceiveFromHN (\( \text{node}_{j \rightarrow i} \), \( u_j \), \( u_{ji} \))
4:    updateNeighborTable (\( \text{node}_{i \rightarrow j} \), \( u_i + u_{ij} \))
5: end for
6: MNjoinsHN (\( \text{node}_{i} \), \( \text{node}_{j} \))

Member node \( j \) selects the least utilized header node \( i \).

C. Tag Allocation and Distribution

After clustering, the MNs send routing information to their HN, and the HN adds this routing information to the routing entry to identify the arrival reachability through the MNs. In this step, there can be several routing entries that support the same next hop. After adding the MN information, the HN selects the shortest and non-overlapping next hop among multiple paths targeted for the same destination, and it updates the routing entry. The HN then assigns tags to the next hop that can be reached through the MNs, and the tags are distributed to cluster the MNs, as illustrated in Fig. 3.

Rather than using IP routing, we use a tag, which is a specially designed identifier within a cluster, for switching that includes the outgoing node interface. Even though the label defined in the Multi-Protocol Label Switching (MPLS) architecture [23] can be used for packet switching, we propose a new and simple identifier in order to solve the problem defined in Section I. Upon receiving tags from an HN, the MNs temporarily put the routing-related function in layer 3 to sleep and perform packet switching using a tag. If the MNs receive packets without a tag, the packets are directly forwarded to the cluster HN.
2. Packet Switching Using a Tag

To demonstrate the operation of packet switching from a source to a destination, we consider a simple network topology with 3 clusters \{a, c, d\}, \{b\}, and \{e, f\}, as shown in Fig. 4. Each HN of each cluster is expressed in bold. When node a receives packets without tags and forwards them to node d, as an operating assumption, the MN (that is, node a) forwards packets to the HN if there is no tag information in the routing entry of the MN. Upon receiving packets, node d looks up the routing entry, encapsulates it with the tag, and forwards all of the packets to node e. After tag switching, node e forwards the packets to the next hop, node f. Finally, packets with a tag arrive at node f, which removes the tag and routes the packets to the destination.

V. Performance Analysis

To investigate the performance of the proposed algorithm, we conducted several simulations using an NS-2 simulator configured with the well-known NSFNet topology with 14 nodes and 42 links, as shown in Fig. 5(a). Additionally, as described in Figs. 5(b) and 5(c) and Table 1, we include real network topologies from the Rocketfuel project given in [24]; the Exodus (3967) and Ebene (1755) networks are composed of 79 nodes and 294 links, and 87 nodes and 322 links, respectively.

For an evaluation of the energy consumption, we assume that the consumed power of the node is 600 W; 420 W are used for IP routing, whereas the residual is used for tag switching [6]. In addition, the power consumption of a link is 235 W according to [14].

Under these environments, we measured the energy efficiency, defined as the ratio of the number of links to be switched off to the total number of all links. In particular, we compared the energy efficiency of the proposed algorithm with those of the GreenOSPF [17] and ILP [2] algorithms.

Figure 6 shows the results obtained from the evaluation of the performance of the proposed algorithm under various topologies. As the offered load increases, energy efficiency is reduced in all network topologies. However, in the cases of Exodus and Ebene, both of which are scale-free networks, the energy efficiency rapidly decreases when the offered load is greater than 0.3. On the other hand, we observe less reduction in a random NSF network as the load increases. This behavior is caused by traffic congestion at hot spot nodes in the Exodus and Ebene topology with smaller offered load. In an NSF network, less congestion occurs because the nodal degree is uniformly distributed between 1 and 4. Thus, the proposed algorithm shows better performance in random networks than scale-free networks regardless of the offered load. Overall, the proposed algorithm can be applied to both random and scale-free network topologies as our objective is the reduction of power consumption during off-peak periods.

Table 1. Simulation environments.

<table>
<thead>
<tr>
<th>AS</th>
<th># of nodes</th>
<th># of links</th>
<th>Avg. nodal degree</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Degree = 1</td>
<td>Degree &gt; 1</td>
</tr>
<tr>
<td>Ebene</td>
<td>87</td>
<td>322</td>
<td>3.72</td>
<td>9.3</td>
</tr>
<tr>
<td>Exodus</td>
<td>79</td>
<td>294</td>
<td>3.75</td>
<td>8.7</td>
</tr>
<tr>
<td>NSF</td>
<td>14</td>
<td>42</td>
<td>3.00</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 7 summarizes the energy efficiency when the offered load is 0.2. All previous algorithms attempt to reduce the energy consumption by turning off the nodes and links. Therefore, the energy efficiency of an ILP is much higher than GreenOSPF because the ILP determines the nodes and links to be turned off through a complicated optimization technique. Generally, turning off the network equipment allows incoming flows, passing by the elements, to be rerouted to other candidate paths, which results in paths that are much longer than the shortest paths.

The behavior described above causes the traffic to be
aggregated on the links that still turn on, and hence, the average link load for such links naturally increases, as shown in Fig. 7. If some of the nodes begin to experience congestion as a result of the rerouting, the algorithm may no longer support the energy consumption, which is why the energy efficiency of an ILP stops at 25%. Contrary to previous approaches, the proposed algorithm simply turns off the L3 routing function while maintaining the network connectivity to the HN. Accordingly, the candidate paths are no longer than the original shortest paths, thereby providing an opportunity to reduce the energy consumption much more than previous methods.

Figure 8 illustrates the energy efficiency under various network topologies according to the change in HN ratio by adjusting the threshold value in the HN selection procedure when the offered load is 0.4. As the number of HNs increases, the energy efficiency in an NSF also increases. As explained in Figs. 7 and 8, a random NSF network has fewer hot spot nodes and less congestion caused by rerouting after clustering. On the other hand, congestion can easily occur at several hot spot nodes in a scale-free network. We can therefore observe the optimal conditions that can maximize the energy efficiency under the given traffic load and HN ratio.

VI. Conclusions

Energy efficiency is becoming a key factor in the green ICT industry, in addition, it is essential to realize trusted network infrastructure as ICT industries continue to grow. Our research for a green network provides an algorithm to minimize the power consumption in IP networks during off-peak periods. The proposed algorithm first configures clusters consisting of one HN and multiple MNs according to the HN selection method, and conducts packet switching using tags. The results obtained show that the proposed packet-switching scheme can obtain more energy efficient performance than previous works. In a future study, we will investigate a trade-off between the HN selection method and performance, as well as research optimal network conditions to maximize energy efficiency.

References


Seng-Kyoun Jo received his BS degree in electronic and information engineering from the Korea Aviation University, Goyang, Rep. of Korea, in 2004. He received his MSc degree from the Department of Information and Telecommunications Engineering from Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea, in 2006. Since 2006, he has been part of the senior engineering staff of the ETRI, Daejeon, Rep. of Korea. Further, he has been a Rapporteur of Q9/13 and co-Rapporteur of Q23/13 in ITU-T since 2013. His current research interests include green ICT, network optimization, and trust management in ICT.

Young-Min Kim received his MSc degree and PhD from the Korea Advanced Institute of Science and Technology, Daejeon, Rep. of Korea, in information and communication in 2006 and 2011, respectively. In 2011, he joined the ETRI, Daejeon, Rep. of Korea. His research mainly focuses on energy saving routing, QoS framework in telecommunication networks network, artificial intelligence, and machine learning. He is a member of IEEE.

Hyun-Woo Lee received his BS degree in electronics engineering from Korea Aviation University in 1993, Goyang, Rep. of Korea, and his MS degree and PhD in communication and information engineering from Korea Aviation University, in 1995 and 2005, respectively. Since 1995, he has been working as a principle researcher in the Broadcasting and Media Research Laboratory of ETRI, Daejeon, Rep. of Korea, where he is currently working as the assistant vice president of the Media Research Section. His current research interests include trust information framework, trust management in ICT, and smart media.
Jussi Kangasharju received his MSc degree from Helsinki University of Technology, Finland, in 1998. He received his Diplome d’Etudes Approfondies from the Ecole Superieure des Sciences Informatiques in Sophia Antipolis, France in 1998. In 2002, he received his PhD from University of Nice Sophia Antipolis/Institut Eurecom, France. In 2002, he joined Technische Universität Darmstadt, first as post-doctoral researcher, and from 2004 onwards as assistant professor. Since June 2007, he is a professor at the Department of Computer Science at University of Helsinki, Finland. Between 2009 and 2012, he was the director of the Future Internet research program at Helsinki Institute for Information Technology (HIIT). Jussi's research interests are information-centric networks, content distribution, opportunistic networks, and green ICT. He is a member of IEEE and ACM.

Max Mülhäuser received his PhD from the University of Karlsruhe, Germany, where he founded a research center for Digital Equipment (DEC). He is a Full Professor of Computer Science and head of the Telecooperation Lab at Technische Universität, Darmstadt, Germany. His Lab conducts research on smart ubiquitous computing environments for the ‘pervasive Future Internet’ in three research fields: middleware and infrastructures, novel multimodal interaction concepts, and human protection in ubiquitous computing (privacy, trust, & civil security). He is also a directorate member of CASED, the center for advanced security research, and deputy speaker of the Future Internet collaborative research center MAKI. Since 1989, he has worked as either professor or visiting professor at universities in Germany, Austria, France, Canada, and the US.