A Middleware Solution to QoS and QoP Support for Multimedia Applications in Wireless Networks

Wenbo He, Klara Nahrstedt

**Goal:** To Provide tunable performance (QoS) and security (QoP) support at middleware layer

**Assumptions:**
Application decides to use the set of user acceptable levels of QoS requirements on delay \(d_1 < d_2 < \ldots < d_m\), and the security levels \(s_1, s_2, \ldots, s_k\).

Requirement Adaptor will dynamically selects the requirement pair \(\{d_i, s_j\}\) based on delay and priority.

**Architecture of the Middleware Solution**

The **Priority Adaptor** adjusts the priority of the application with the change of network delay to fulfill the requirements of the application.

If the priority already reaches the maximum, but the delay still larger than required delay \(d_i\), the **Requirement Adaptor** will determine the new required delay and security level report to the application for further response.

**Middleware Design:**

\[
\begin{align*}
p(t) &= \text{priority at time } t \\
d^* &= \text{specified delay bound} \\
s^* &= \text{specified security level} \\
P_{\text{max}} &= \text{maximum value of } p(t) \\
d &= \text{observed delay value}
\end{align*}
\]

**Priority Adaptor**

**Goal:** update \(p(t+1)\) according to \(p(t), d^*, \text{ and } d\).

**Algorithm:**

1. If \(p(t) = P_{\text{max}}\) and \(d > d^*\), let \(d^* = d_{i} \), let \(s^* = s_{j}\) unchanged.
2. If \(p(t) = P_{\text{max}}\) and \(d > d^* = d_{m}\), downgrade \(s^* \) from \(s_j \) to \(s_k \); if \(s^* = s_k\), then drop the application.
3. If \(p(t) < P_{\text{max}}\) and \(d < d_{i-1}\), upgrade \(d^* \) from \(d_{i} \) to \(d_{i-1}\) and \(s^* \) from \(s_i \) to \(s_{i-1}\) until \(d^* = d_1\).

Else \(p(t) < P_{\text{max}}\) and \(d_i > d \geq d_{i-1}\), upgrade \(s^* \) from \(s_i \) to \(s_{i-1}\) and keep \(d^* \) unchanged until \(s^* = s_1\).

**Requirement Adaptor**

**Goal:** Given \(\{d^*, s^*\}\) pair according to \(d\) and \(p(t)\), where \(d^* = d_i\); \(\{d_1 < d_2 < \ldots < d_m\}\) and \(s^* = s_j\); \(\{s_1, s_2, \ldots, s_k\}\).

**Algorithm:**

1. If \(p(t) = P_{\text{max}}\) and \(d > d^* \neq d_m\), downgrade \(d^* \) from \(d_i\) to \(d_{i+1}\), let \(s^* \) unchanged.
2. If \(p(t) = P_{\text{max}}\) and \(d > d^* = d_m\), downgrade \(s^* \) from \(s_j \) to \(s_k \); if \(s^* = s_k\), then drop the application.
3. If \(p(t) < P_{\text{max}}\) and \(d < d_{i-1}\), upgrade \(d^* \) from \(d_{i} \) to \(d_{i-1}\) and \(s^* \) from \(s_i \) to \(s_{i-1}\) until \(d^* = d_1\).

Else \(p(t) < P_{\text{max}}\) and \(d_i > d \geq d_{i-1}\), upgrade \(s^* \) from \(s_i \) to \(s_{i-1}\) and keep \(d^* \) unchanged until \(s^* = s_1\).