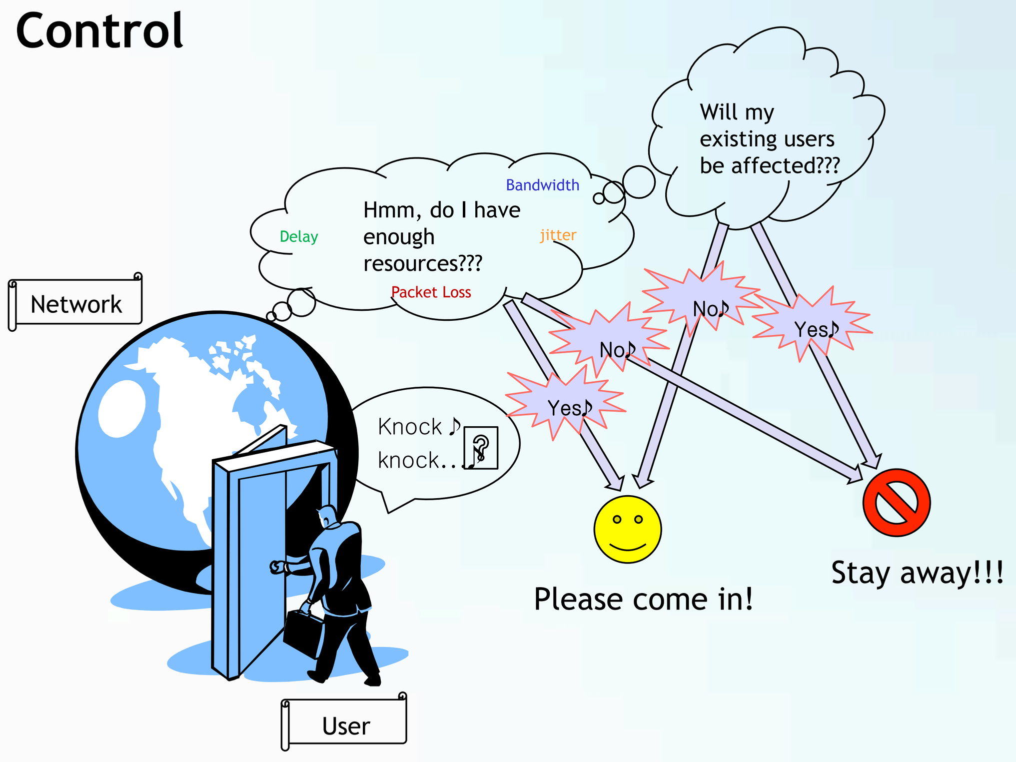
A MULTIPLE CRITERIA ADMISSION CONTROL MECHANISM FOR SELF-AWARE NETWORKS

Imperial College London

SAN Self-Aware Networks

Georgia Sakellari and Erol Gelenbe

Admission Control



Self-Aware Networks (SAN)

Packet networks that collect data at different distributed points, by probing the network with special packets.
According to the QoS objective that those packets are pursuing, a SAN adaptively takes corrective action in order to address QoS and provide reliable service to its users.
The SANs use adaptive packet routing protocols, such as the **Cognitive Packet Network (CPN)** which is designed to perform Self-Improvement by using Random Neural Networks (RNN) with Reinforcement Learning (RL) and Genetic Algorithms

A Multiple Criteria Measurement-Based AC Algorithm for SANs

- Our algorithm constantly gathers data on the ongoing users QoS in the network by exploiting the network's Self Awareness.
- This data is available in one or more locations in the network where the decision is taken.
- This does not require any special mechanism since the CPN already collects QoS information on all links and paths that the special packets have explored and on all paths that any user is using in the network.
- It is a multiple criterion AC algorithm in which the QoS criteria can be specified by the user.

Contribution of the Algorithm

- ✓ User specified QoS constraints.
- ✓ Probing at a small rate.
- ✓ Novel algebra of QoS metrics which investigates whether there is a feasible path which can accommodate the new request without affecting the ongoing connections.

Probing Stage - Estimation of the impact of a new flow

Before Probing the Network

Let ***u*** be a new user requesting admission for a connection from a source ***s*** to destination ***d*** carrying a traffic rate ***X*** and with QoS value $q_v(u)$ with constraints $q_v(u) \in C_v(u)$, $q_v \in R$, $v=1, \ldots m$, and also ***z*** users currently occupying the network, with QoS constraint $q_w(z) \in C_v(u)$.
The CPN algorithm explores the network and collects QoS data about the parts of the network that are being currently used, or which have been explored by smart packets. This data is available in one or more locations in the form of *nxn* link QoS matrices Q_v with elements:

$Q_v(i,j)=r$ where $r \geq 0$ is a real number representing the QoS of link (i, j) which has been measured at some recent enough time, and

$Q_v(i,j)=$***unknown*** if a SP has not explored the link for QoS metric *v* or if this happened so long ago that the value could be inaccurate.

After Probing the Network

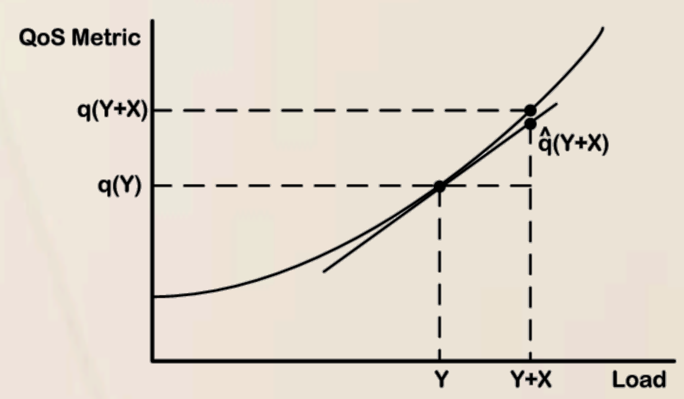
For a new user with requested traffic rate **X**, it sends probe packets at rate **x** and estimates the link QoS matrices in the case that the new call is accepted.

$$\hat{Q}_v(i,j) = Q_v(i,j) + X\,\hat{q}_v(i,j)$$, for all concerned links

$$\hat{Q}_v(i,j) = Q_v(i,j)$$, for unconcerned links

Estimation of the QoS Values

The estimation of the QoS values is based on the fact that Every QoS metric can be considered as a value which increases as the "traffic load" increases.
The addition of a new connection will increase the load of the paths it may be using, and therefore it is assumed that the value taken by the QoS metrics will increase. Let us consider some link (i, j). A small increase x in the load that is obtained in a controlled manner, generates an estimate of the manner in which the QoS metric q varies around the current load point *Y*. The impact of a new flow with total traffic rate *X* can then be evaluated by using the measured derivative without having to know the initial load *Y*.



$$\hat{q}' = \frac{q(Y+x) - q(Y)}{x}$$

$$\hat{q}(Y+X) = q(Y) + \hat{q}'X$$

Decision Stage

From the **link matrices** $\hat{Q}_w$ of all the QoS metrics of interest, including $w=v$, we can compute:
- The set of known (explored) paths $P(i, j)$ from i to j. If $P(s, d)$ for the source **s** and destination **d** of the new user is empty, the algorithm sends SPs to discover paths. If that is unsuccessful, the new request is rejected.
- The **path QoS matrices** $\hat{K}_v$, where $\hat{K}_v(i,j)$ = *the best estimated value of the QoS metric v* for any path going from i to j if such a path exists and if the links on the path have known entries in the link QoS matrices, or $\hat{K}_v(i,j)$ = ***unknown*** otherwise.
- If $\boxed{\hat{K}_v(s,d) \in C_v(u) \;\; AND \;\; \hat{K}_w(s',d') \in C_w(z)}$ for the user ***u*** requesting to enter the network from source ***s*** to destination ***d*** with QoS requests q_v and for all the existing users **z** with source-destination pairs ***(s', d')*** and QoS metrics q_w, then accept ***u***. Else the request is rejected.

Computing the QoS matrices

The well known **Warshall's algorithm** determines for each **i, j** ⊂ **N** whether there is a path from node *i* to node *j* by computing a Boolean matrix, the transitive closure of the graph's adjacency matrix, in less than n^3 Boolean operations.

$$K = \bigcup_{k=1}^{n} Q^k \quad or \quad K^k[i,j] = K^{k-1}[i,j] \vee (K^{k-1}[i,k] \wedge K^{k-1}[k,j])$$

Floyd's algorithm extends Warshall's algorithm to obtain the cost of the "smallest cost path" between any pair of vertices in the form of a real-valued matrix.

$$K = \min_{k=1}^{n} Q^k \quad or \quad K^k[i,j] = \min\{\, K^{k-1}[i,j],\;\; K^{k-1}[i,k] + K^{k-1}[k,j] \,\}$$

If the QoS metric is additive then we can use the Floyd-Warshall algorithm to get the smallest value of the QoS metric among all known paths from any node *i* to any node *j*.

Generalisation of the Floyd-Warshall Algorithm for non-additive metrics

Consider the matrix Q_v, whose entries are the QoS values $r \geq 0$ over links **(i,j)** whenever such a link exists, or ***unknown*** otherwise. The matrix K_v provides us with the "best QoS value" for every path between every pair of vertices **(i,j)**.

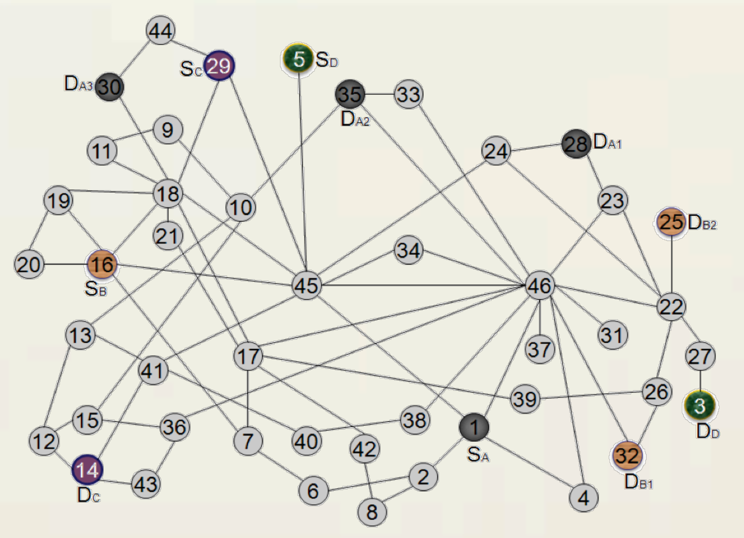
$$K_v = \bigoplus_{k=1}^{n} Q_v^k \quad or \quad K_v^k[i,j] = K_v^{k-1}[i,j] \otimes K_v^{k-1}[i,k] \oplus K_v^{k-1}[k,j]$$

Where the operator $\oplus$ between two real valued matrices B, C is defined as

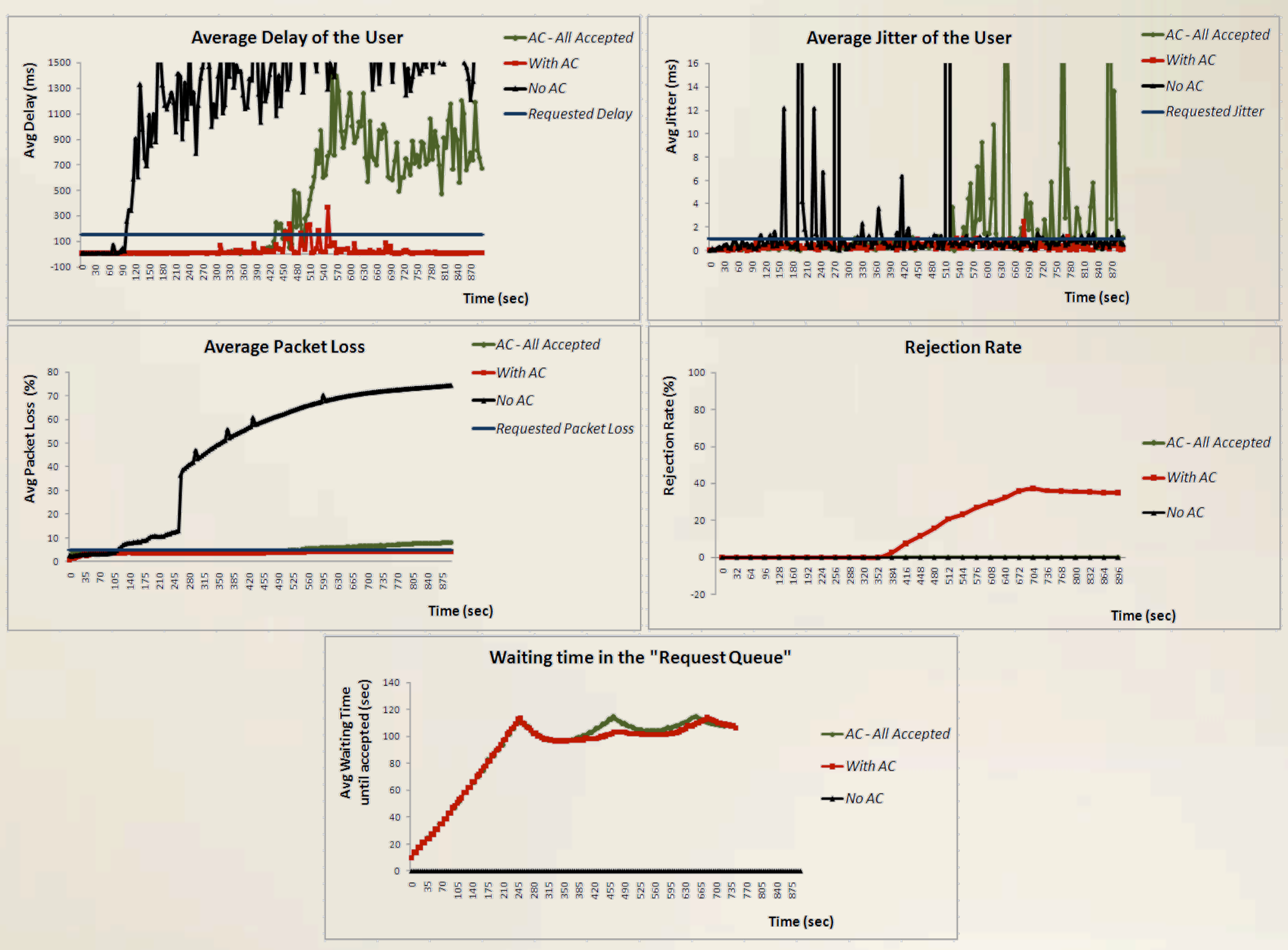
$$D_v(i,j) = \bigotimes_{t=1}^{n} B_v(i,t) \oplus C_v(t,j)$$

The operator $\oplus$ between two QoS parameters depends on the QoS metric that is being considered and can be the addition (+) for delay and variance, the minimum (min) for bandwidth etc. The $\otimes$ is also an operator that depends on the specific QoS metric q, and selects the "best value" among the elements on which it operates, e.g. in case of the delay, loss or variance metric it will obtain the minimum value, while for bandwidth it will select the maximum value, for all paths going from *i* to *j*.

Experiments



46-node test-bed (Switch-LAN network topology), with 10 Mbps capacity links.
Current QoS metrics:
- Requested **Delay ≤ 150 msec.**
- Requested **Jitter ≤ 1 msec.**
- Requested **Loss Rate ≤ 5%.**
- Total arrival rate λ=28 rqsts/min.
- Probe rate equal to 40% of the user's rate.
- Probing time 2s.
- When a call is accepted the source will generate a constant bit rate UDP traffic at 1Mbps that last for 600s.



Symposium on Perspectives in Modeling and Performance Analysis of