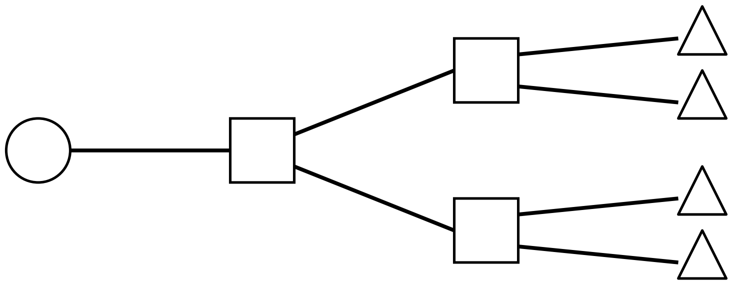
# Combinatorial Algorithm for Calculating Blocking Probabilities in Multicast Networks with Multiple Connection Classes

Samuli Aalto, Jouni Karvo & Jorma Virtamo
Laboratory of Telecommunications Technology
Helsinki University of Technology

samuli.aalto@hut.fi

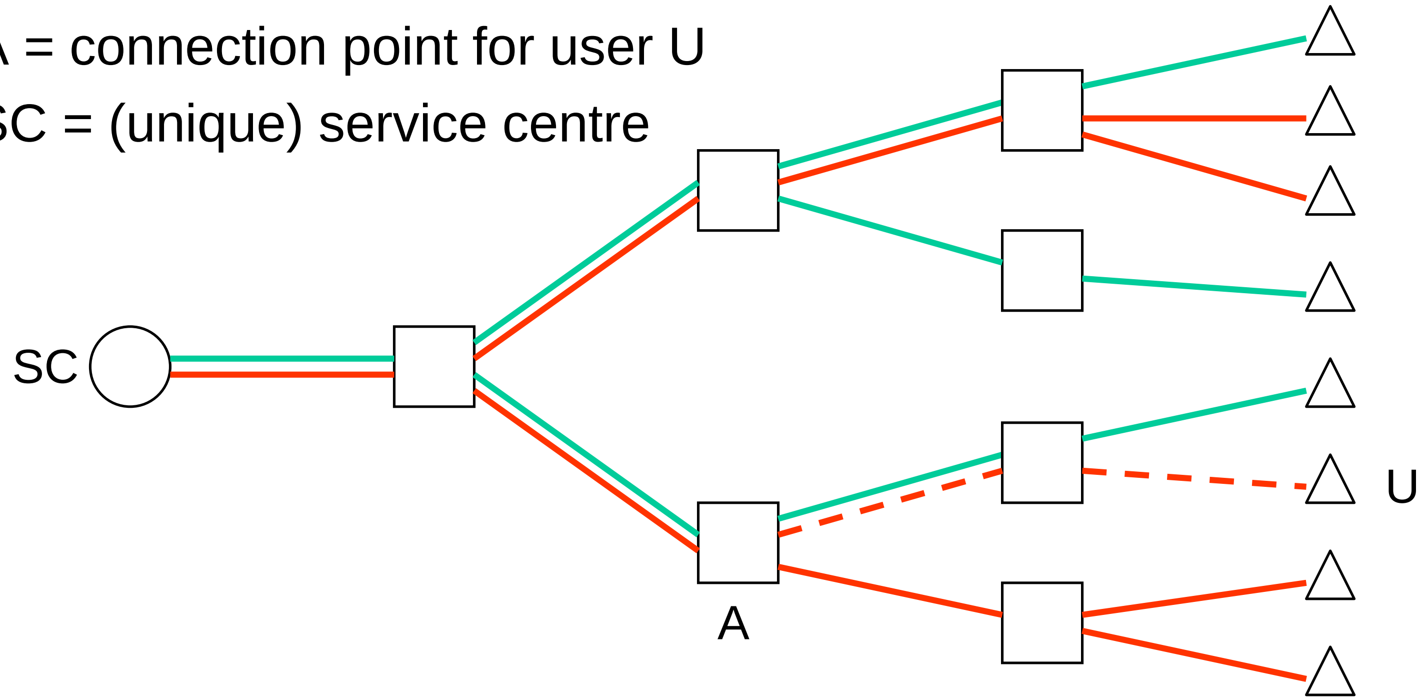
# Multicast network model

- Setup:
  - Unique **service center** offers a variety of **channels**
  - Each channel $i \in I$ is delivered by a **multicast connection** with **dynamic membership**
  - Each multicast connection uses the same **multicast tree**
    $\Rightarrow$ **fixed routing** of these multicast connections
  - **Symmetric** connections belong to the same **class** $k \in K$
  - Service center located at the **root node** of the multicast tree
  - Users $u \in U$ located at the **leaf nodes** of the multicast tree



# Multicast connections with dynamic membership

- U = new user of channel ‘red’
- A = connection point for user U
- SC = (unique) service centre



# Link states

- Consider first a network with infinite link capacities
- Let

$$Y_{ji} = 1\{\text{connection } i \text{ active on link } j\}$$

- Detailed link state (for any link $j \in J$)

$$\mathbf{Y}_j = (Y_{ji}; i \in I) \in S := \{0,1\}^I$$

- Classwise link state (for any link $j \in J$)

$$\mathbf{N}_j = (\sum_{i \in I_k} Y_{ji}; k \in K) \in S := \{0,1,\ldots,I_1\} \times \ldots \times \{0,1,\ldots,I_K\}$$

- Link state (for any link $j \in J$)

$$N_j = \sum_{i \in I} Y_{ji} \in \{0,1,\ldots,I\}$$

# Stationary state probabilities in a network with infinite link capacities

- Assume that the probabilities of the detailed **leaf link states** (which depend on the user population model adopted) are known, and denote them by

$$\pi_u(\mathbf{y}) := P\{\mathbf{Y}_u = \mathbf{y}\}$$

  - where $\mathbf{y} \in \{0,1\}^I$

- Due to infinite link capacities and independent behaviour of the user populations, it follows that the probabilities of the detailed **network states** are also known:

$$\pi(\mathbf{x}) := P\{\mathbf{X} = \mathbf{x}\} = \prod_{u \in U} P\{\mathbf{Y}_u = \mathbf{y}_u\} = \prod_{u \in U} \pi_u(\mathbf{y}_u)$$

  - where $\mathbf{x} = (\mathbf{y}_u; u \in U) \in \{0,1\}^{U \times I} =: \Omega$

# Stationary state probabilities in a network with finite link capacities

- If the **Truncation Principle** applies (which depends on the user population model adopted), then

$$\tilde{\pi}(\mathbf{x}) = \frac{\pi(\mathbf{x})}{\sum_{\mathbf{x}\in\tilde{\Omega}} \pi(\mathbf{x})}$$

  – where $\mathbf{x} = (\mathbf{y}_u;\, u \in U) \in \tilde{\Omega}$ and

$$\tilde{\Omega} \quad = \text{set of allowed network states}$$

# Blocking probability

- $B^t_{ui}$ = **time blocking** for user population $u$ and connection $i$
  = stationary probability of such network states in which
  a new request originating from user population $u$ to join
  connection $i$ would be rejected due to lack of link capacity

- How to calculate $B^t_{ui}$?

# Calculation of blocking probabilities (1)

- 1st possibility: closed form expression

$$B_{ui}^{t} := 1 - \sum_{\mathbf{x} \in \tilde{\Omega}_{ui}} \tilde{\pi}(\mathbf{x}) = 1 - \frac{\sum_{\mathbf{x} \in \tilde{\Omega}_{ui}} \pi(\mathbf{x})}{\sum_{\mathbf{x} \in \tilde{\Omega}} \pi(\mathbf{x})}$$

  – where

$$\tilde{\Omega}_{ui} = \text{set of nonblocking network states for } (u,i)$$
$$\tilde{\Omega} \quad = \text{set of allowed network states}$$

- **Problem**: computationally extremely complex
  – exponential growth both in $U$ and $I$

# Calculation of blocking probabilities (2)

- 2nd possibility: recursive algorithm **exact**

$$B_{ui}^{t} = 1 - \frac{\sum_{\mathbf{y} \in S} Q_J^{ui}(\mathbf{y})}{\sum_{\mathbf{y} \in S} Q_J(\mathbf{y})}$$

  - where probabilities $Q_j^{ui}(\mathbf{y})$ and $Q_j(\mathbf{y})$ can be calculated recursively (from the common link $J$ back to leaf links $u$)

- **Problem**: computationally complex
  - linear growth in $U$ but (still) exponential growth in $I$

# Calculation of blocking probabilities (3)

- 3rd possibility: **new** recursive algorithm **combi**

$$B_{ui}^{t} = 1 - \frac{\sum_{\mathbf{n}\in S} Q_J^{ui}(\mathbf{n})}{\sum_{\mathbf{n}\in S} Q_J(\mathbf{n})}$$

  – where probabilities $Q_j^{ui}(\mathbf{n})$ and $Q_j(\mathbf{n})$ can be calculated recursively (from the common link $J$ back to leaf links $u$)

- **Remark**: computationally reasonable if …
  … only few connection classes

# Basic results (1)

- Connections symmetric among a class $\Rightarrow$
  - Whenever there are $n$ connections (belonging to class $k \in K$) active on any leaf link $u \in U$, each possible index combination $\{i_1,\ldots,i_n\}$ (where $i_1,\ldots,i_n \in I_k$) is equally probable

- This and the independence of the user populations $\Rightarrow$
  - Whenever there are n connections (belonging to class $k \in K$) active on any link $j \in J$, each possible index combination $\{i_1,\ldots,i_n\}$ (where $i_1,\ldots,i_n \in I_k$) is equally probable

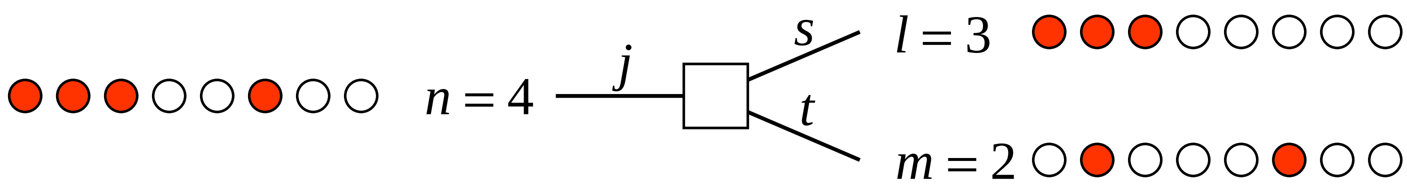
# Basic results (2)

- If link $j$ has two downstream neighbouring links $(s,t)$, then

$$\max\{N_s, N_t\} \le N_j \le \min\{N_s + N_t, I\}$$

- Assume (here only) that all connections belong to the same class and that $N_s = l \ge m = N_t$. Then

$$P\{N_j = n \mid N_s = l, N_t = m\} = \frac{\binom{l}{m-(n-l)}\binom{I-l}{n-l}}{\binom{I}{m}}$$



## Algorithm (1)

- Define (for all $j \in J$):

$$Q_j(\mathbf{n}) = P\{\mathbf{N}_j = \mathbf{n}; N_{j'} \le C_{j'}, \forall j' \in M_j\}$$

$$Q_j^{ui}(\mathbf{n}) = P\{\mathbf{N}_j^{(i)} = \mathbf{n}; N_{j'}^{(i)} \le C_{j'} - 1, \forall j' \in M_j \cap R_u;$$

$$N_{j'} \le C_{j'}, \qquad \forall j' \in M_j \setminus R_u\}$$

- Then time blocking probability for pair $(u,i)$ is

$$B_{ui}^t = 1 - \frac{P\{\mathbf{X} \in \tilde{\Omega}_{ui}\}}{P\{\mathbf{X} \in \tilde{\Omega}\}} = 1 - \frac{\sum_{n \in S} Q_J^{ui}(\mathbf{n})}{\sum_{n \in S} Q_J(\mathbf{n})}$$

# Algorithm (2)

- Recursion 1 to calculate $Q_j(\mathbf{n})$:

$$Q_j(\mathbf{n}) = \begin{cases} T_j[\pi_j](\mathbf{n}), & j \in U \\ T_j[\bigotimes_{j' \in N_j} Q_{j'}](\mathbf{n}), & j \notin U \end{cases}$$

  – where $\pi_j(\mathbf{n}) = P\{\mathbf{N}_j = \mathbf{n}\}$ depend on the chosen user population model

- Truncation operator 1:
  – Let $f$ be any real-valued function defined on $S$
  – Then define

$$T_j[f](\mathbf{n}) = f(\mathbf{n}) \cdot 1\{n_1 + \ldots + n_K \le C_j\}$$

# Algorithm (3)

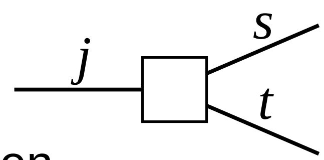
- Definition of operator ⊗:
  - Let $f$ and $g$ be any real-valued function defined on $S$. Then define

$$[f \otimes g](\mathbf{n}) = \sum_{l_1=0}^{n_1} \sum_{m_1=n_1-l_1}^{n_1} s_1(n_1 \mid l_1, m_1) \ldots$$
$$\sum_{l_K=0}^{n_K} \sum_{m_K=n_K-l_K}^{n_K} s_K(n_K \mid l_K, m_K)$$
$$\times f(\mathbf{l}) g(\mathbf{m})$$

    - where

$$s_k(n \mid l, m) = \frac{\binom{\max\{l,m\}}{l+m-n}\binom{I_k - \max\{l,m\}}{n-\max\{l,m\}}}{\binom{I_k}{\min\{l,m\}}}$$

# Algorithm (4)




- Key result:
  - If link $j$ has two downstream neighbouring links ($s$,$t$), then

$$
\begin{aligned}
P\{\mathbf{N}_j = \mathbf{n}\} = &\sum_{l_1=0}^{n_1} \sum_{m_1=n_1-l_1}^{n_1} s_1(n_1 \mid l_1, m_1) \ldots \\
&\sum_{l_K=0}^{n_K} \sum_{m_K=n_K-l_K}^{n_K} s_K(n_K \mid l_K, m_K) \\
&\times P\{\mathbf{N}_s = \mathbf{l}\} P\{\mathbf{N}_t = \mathbf{m}\}
\end{aligned}
$$

  - In other words,

$$
\pi_j(\mathbf{n}) = [\pi_s \otimes \pi_t](\mathbf{n})
$$

  - Proved by a “sampling without replacement” argument!

# Algorithm (5)

- Recursion 2 to calculate $Q_J^{ui}(\mathbf{n})$ :

$$Q_j^{ui}(\mathbf{n}) = \begin{cases} T_u^{\circ}[\pi_u^{(i)}](\mathbf{n}), & j = u \\ T_j^{\circ}[Q_{D_u(j)}^{ui} \odot \bigotimes_{j' \in N_j \setminus R_u} Q_{j'}](\mathbf{n}), & j \in R_u \setminus \{u\} \end{cases}$$

  - where $\pi_u^{(i)}(\mathbf{n}) = P\{\mathbf{N}_u^{(i)} = \mathbf{n}\}$ depend on the chosen user population model
- Truncation operator 2:
  - Let $f$ be any real-valued defined on $\{0,1,\ldots,I\}$
  - Then define

$$T_j^{\circ}[f](\mathbf{n}) = f(\mathbf{n}) \cdot 1\{n_1 + \ldots + n_K \le C_j - 1\}$$

# Algorithm (6)

- Definition of operator $\odot$:
  - Let $f$ and $g$ be any real-valued function defined on $S$. Then define

$$[f \odot g](\mathbf{n}) = \sum_{l_1=0}^{n_1} \sum_{m_1=n_1-l_1}^{n_1} s_1^{(i)}(n_1 \mid l_1, m_1) \ldots$$

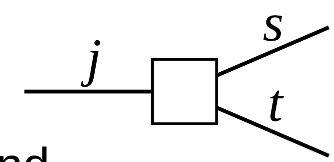
$$\sum_{l_K=0}^{n_K} \sum_{m_K=n_K-l_K}^{n_K} s_K^{(i)}(n_K \mid l_K, m_K)$$

$$\times f(\mathbf{l})[(1 - \frac{m_{k(i)}}{I_{k(i)}}) g(\mathbf{m}) + \frac{m_{k(i)}+1}{I_{k(i)}} g(\mathbf{m} + \mathbf{e}_{k(i)})]$$

- where

$$s_k^{(i)}(n \mid l, m) = \frac{\binom{\max\{l,m\}}{l+m-n}\binom{I_k^{(i)} - \max\{l,m\}}{n - \max\{l,m\}}}{\binom{I_k^{(i)}}{\min\{l,m\}}}$$

# Algorithm (7)




- Key result:
  - If link $j$ has two downstream neighbouring links $(s,t)$, and link $s$ belongs to the interesting route, i.e. $s = D_u(j)$, then

$$P\{\mathbf{N}_j = \mathbf{n}\} = \sum_{l_1=0}^{n_1} \sum_{m_1=n_1-l_1}^{n_1} s_1^{(i)}(n_1 \mid l_1, m_1) \ldots$$

$$\sum_{l_K=0}^{n_K} \sum_{m_K=n_K-l_K}^{n_K} s_K^{(i)}(n_K \mid l_K, m_K)$$

$$\times P\{\mathbf{N}_s = \mathbf{l}\}[(1-\frac{m_{k(i)}}{I_{k(i)}})P\{\mathbf{N}_t = \mathbf{m}\} + \frac{m_{k(i)}+1}{I_{k(i)}} P\{\mathbf{N}_t = \mathbf{m} + \mathbf{e}_{k(i)}\}]$$

  - In other words,

$$\pi_j^{(i)}(n) = [\pi_s^{(i)} \odot \pi_t](n)$$

  - Proved by a "sampling without replacement" argument!

# THE END