Fast and Secure Routing Algorithms for Quantum Key Distribution Networks†

Abhishek Sinha

Joint work with

Vishnu B

Dept. of Electrical Engg.

IIT Madras

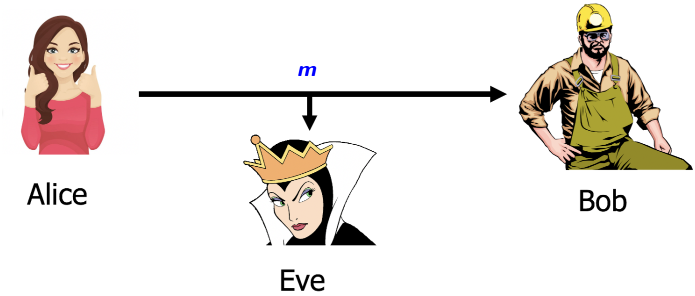
Thursday 11th November, 2021

† https://arxiv.org/abs/2109.07934

Goal of the talk

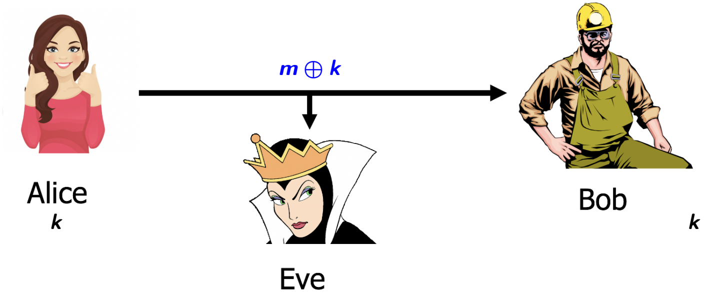
- We study the QKD problem from a high-level optimal resource allocation point-of-view.
- We will present a network architecture and a routing algorithm that achieves the capacity of a network while guaranteeing the full security of the transmitted messages in a standard system model
- Our scheme is general, can handle very general traffic flows, and does not depend on particular scheme used for either the quantum key generation or message transmission.
- The key idea is to suitably modify a throughput-optimal policy proposed by us in the past to take into account the availability of quantum keys

Secure message transmission



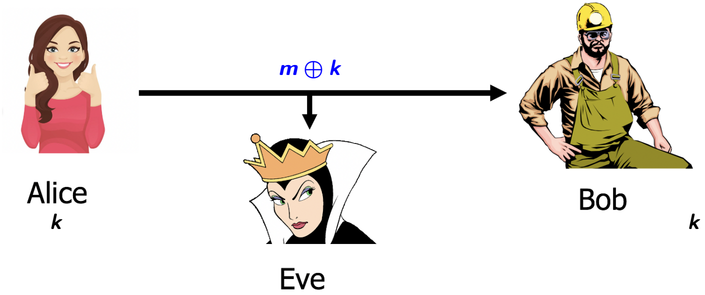
- Alice wants to send a secret (binary) message m to Bob over a public channel
- Unfortunately, a third-party, Eve can listen in to the transmission
- **Problem**: Design an encryption scheme so that Bob can correctly decode the message m, but Eve can not

Symmetric Key Encryption: One Time Pad



- Assume that **both** Alice and Bob hold a shared secret key k of the same length as the message m. Eve does not have the key.
- Alice transmits the message $m' = m \oplus k$ over the classical public channel.
- Bob decodes by XORing the received message with k $(m' \oplus k = (m \oplus k) \oplus k = m)$.
  - Information Theoretically secure.

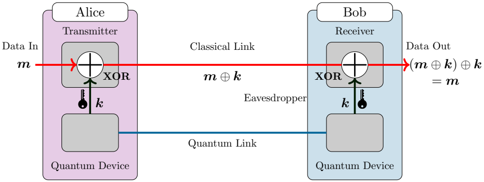
Symmetric Key Encryption: One Time Pad



- Assume that **both** Alice and Bob hold a shared secret key k of the same length as the message m. Eve does not have the key.
- Alice transmits the message $m' = m \oplus k$ over the classical public channel.
- Bob decodes by XORing the received message with k $(m' \oplus k = (m \oplus k) \oplus k = m)$.
  - Information Theoretically secure.

Problem: How to establish the shared secret key k?

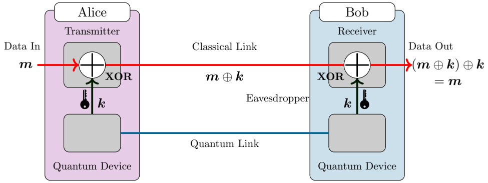
Sharing Secret keys: Quantum Key Distribution (QKD)



QKD link with OTP Protocol

- QKD allows remote communication parties to securely share symmetric keys $\boldsymbol{k}$
- Uses Quantum Entanglement or Photon Polarization to agree upon a secret key sequence in a provably secured fashion
    - Protocols: BB84, E91 *etc.*
- Two required channels
    - Key agreement takes place over an authenticated Quantum channel and
    - the encoded message is transmitted over a Classical channel (free space or optical fiber)

Sharing Secret keys: Quantum Key Distribution (QKD)



QKD link with OTP Protocol

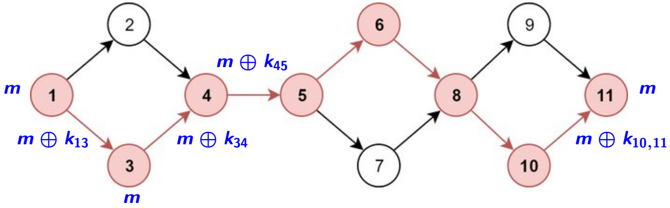
- QKD allows remote communication parties to securely share symmetric keys $\boldsymbol{k}$
- Uses Quantum Entanglement or Photon Polarization to agree upon a secret key sequence in a provably secured fashion
  - Protocols: BB84, E91 *etc.*
- Two required channels
  - Key agreement takes place over an authenticated Quantum channel and
  - the encoded message is transmitted over a Classical channel (free space or optical fiber)
- Distance Limited - Fidelity of entanglement drops exponentially with distance

Question: How to securely extend the one-hop QKD scheme to multi-hop networks?

Trusted Node QKD

We consider a trusted node QKD architecture used in the European **SECOQC** network and more recently in **Oak-Ridge-Los Alamos** QKD Network.

- The nodes are assumed to be secured; only the links can be compromised
- Packets are sequentially encrypted and decrypted *hop-by-hop* by each node along its path

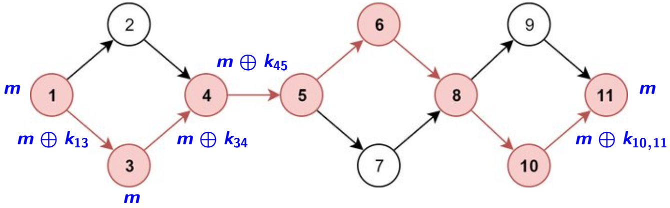


- A packet can be securely transmitted over a link only if *sufficiently many keys are available*.

Trusted Node QKD

We consider a trusted node QKD architecture used in the European **SECOQC** network and more recently in **Oak-Ridge-Los Alamos** QKD Network.

- The nodes are assumed to be secured; only the links can be compromised
- Packets are sequentially encrypted and decrypted *hop-by-hop* by each node along its path



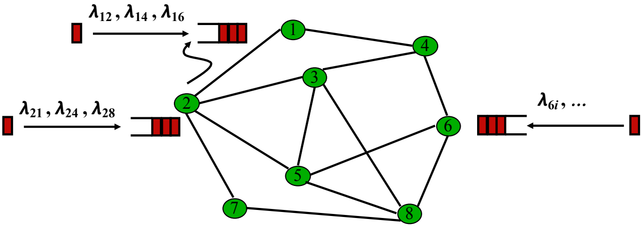
- A packet can be securely transmitted over a link only if *sufficiently many keys are available*.

Problem Statement (informal)

With the instantaneous key availability constraints, how to securely route packets to achieve the entire secured throughput region of the network?

System Model

- The network is represented by the graph $\mathcal{G}(V, E)$; V denotes the set of nodes. E denotes the set of edges.
  - Each edge contains a classical link and an overlay quantum link.
- Physical link capacity of the link e is γ_e. For simplicity, assume that the classical links don't interfere.
  - the algorithms that we are going to present can be extended to wireless networks.
- Quantum keys are generated over the link e according to a stochastic counting process at the rate of η_e.
- Packet transmissions are subjected to the key availability and the link capacity constraints.



Network topology $\mathcal{G}(V, E)$

Traffic Classes

Generalized Flow: Traffic Class c has arrival rate λ_c, source node S^c, destination node(s) D^c, where

- Unicast: Single source, single destination

$$\mathcal{S} = \{s\}, \mathcal{D} = \{d\}$$

Traffic Classes

Generalized Flow: Traffic Class c has arrival rate λ_c, source node S^c, destination node(s) D^c, where

- Unicast: Single source, single destination

$$\mathcal{S} = \{s\}, \mathcal{D} = \{d\}$$

- Multicast: Single source, multiple destinations

$$\mathcal{S} = \{s\}, \mathcal{D} \subset V \setminus \{s\}$$

Traffic Classes

Generalized Flow: Traffic Class c has arrival rate λ_c, source node S^c, destination node(s) D^c, where

- Unicast: Single source, single destination

$$\mathcal{S} = \{s\}, \mathcal{D} = \{d\}$$

- Multicast: Single source, multiple destinations

$$\mathcal{S} = \{s\}, \mathcal{D} \subset V \setminus \{s\}$$

- Broadcast: Single source, all destinations

$$\mathcal{S} = \{s\}, \mathcal{D} = V \setminus \{s\}$$

Traffic Classes

Generalized Flow: Traffic Class c has arrival rate λ_c, source node S^c, destination node(s) D^c, where

- Unicast: Single source, single destination

$$\mathcal{S} = \{s\}, \mathcal{D} = \{d\}$$

- Multicast: Single source, multiple destinations

$$\mathcal{S} = \{s\}, \mathcal{D} \subset V \setminus \{s\}$$

- Broadcast: Single source, all destinations

$$\mathcal{S} = \{s\}, \mathcal{D} = V \setminus \{s\}$$

- Anycast: Single source, choice of one among multiple alternative destinations

$$\mathcal{S} = \{s\}, \mathcal{D} = v_1 \oplus v_2 \oplus \ldots \oplus v_k$$

Problem Statement

Let $R_\pi^{(c)}(T)$ denote the number of encrypted packets received by all destinations of class c up to time T under the policy π. The policy π is said to securely support an arrival vector $\boldsymbol{\lambda}$ if :

$$\lim\inf_{T \nearrow \infty} \frac{R_\pi^{(c)}(T)}{T} = \lambda^c, \ \forall c \in \mathcal{C} \quad \text{w.p. } 1.$$

Problem Statement

Let $R_\pi^{(c)}(T)$ denote the number of encrypted packets received by all destinations of class c up to time T under the policy π. The policy π is said to securely support an arrival vector $\boldsymbol{\lambda}$ if :

$$\lim \inf_{T \nearrow \infty} \frac{R_\pi^{(c)}(T)}{T} = \lambda^c, \ \forall c \in \mathcal{C} \quad \text{w.p. } 1.$$

1. Definition (Stability region of a policy)

$$\boldsymbol{\Lambda}_\pi(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma}) = \{\boldsymbol{\lambda} : \pi \text{ securely supports } \boldsymbol{\lambda}\}$$

2. Definition (Secure Capacity Region)

$$\boldsymbol{\Lambda}(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma}) = \bigcup_{\pi \in \Pi} \boldsymbol{\Lambda}_\pi(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma})$$

Problem Statement

Let $R_\pi^{(c)}(T)$ denote the number of encrypted packets received by all destinations of class c up to time T under the policy π. The policy π is said to securely support an arrival vector $\boldsymbol{\lambda}$ if :

$$\lim \inf_{T \nearrow \infty} \frac{R_\pi^{(c)}(T)}{T} = \lambda^c, \ \forall c \in \mathcal{C} \quad \text{w.p. } 1.$$

1. Definition (Stability region of a policy)

$$\boldsymbol{\Lambda}_\pi(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma}) = \{\boldsymbol{\lambda} : \pi \text{ securely supports } \boldsymbol{\lambda}\}$$

2. Definition (Secure Capacity Region)

$$\boldsymbol{\Lambda}(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma}) = \bigcup_{\pi \in \Pi} \boldsymbol{\Lambda}_\pi(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma})$$

Problem (Throughput-Optimal Secured Routing)

Find a policy $\pi^* \in \Pi$ s.t.

$$\boldsymbol{\Lambda}_\pi(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma}) \supseteq \text{int}(\boldsymbol{\Lambda}(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma})).$$

Characterizing the Secure Capacity Region

Consider the weighted graph $\mathcal{G}_\omega$ such that the edge e has capacity:

$$\omega_e = \min(\gamma_e, \eta_e), \qquad \forall e \in E.$$

Characterizing the Secure Capacity Region

Consider the weighted graph $\mathcal{G}_\omega$ such that the edge e has capacity:

$$\omega_e = \min(\gamma_e, \eta_e), \qquad \forall e \in E.$$

- The long-term rate of encrypted packet flow over any edge is limited by the quantum key generation rates and the capacity of the communication link.

Characterizing the Secure Capacity Region

Consider the weighted graph $\mathcal{G}_\omega$ such that the edge e has capacity:

$$\omega_e = \min(\gamma_e, \eta_e), \qquad \forall e \in E.$$

- The long-term rate of encrypted packet flow over any edge is limited by the quantum key generation rates and the capacity of the communication link.
- Now consider the set of arrival rates $\overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}(\mathcal{G}, \boldsymbol{\gamma}, \boldsymbol{\eta})$ for which a feasible flow-decomposition on $\mathcal{G}_{\boldsymbol{\omega}}(\mathcal{G}, \boldsymbol{\gamma}, \boldsymbol{\eta})$ exists, *i.e.*, $\boldsymbol{\lambda} \in \overline{\boldsymbol{\Lambda}}_{\boldsymbol{\omega}}(\mathcal{G}, \boldsymbol{\gamma}, \boldsymbol{\eta})$ iff there exist a non-negative scalar $\lambda_i^{(c)}$, associated with the i^{th} admissible route $T_i^{(c)} \in \mathcal{T}^{(c)}, \forall i, c$, such that

$$\lambda^{(c)} = \sum_{i: T_i^{(c)} \in \mathcal{T}^{(c)}} \lambda_i^{(c)}, \quad \text{(Multipath flow decomposition)} \tag{1}$$

$$\lambda_e \overset{\text{(def.)}}{=} \sum_{\substack{(i,c): e \in T_i^{(c)}, \\ T_i^{(c)} \in \mathcal{T}^{(c)}}} \lambda_i^{(c)} \leq \omega_e, \quad \forall e \in E. \quad \text{(Feasibility)} \tag{2}$$

Characterizing the Secure Capacity Region

Main Theorem

The network-layer secured capacity region $\mathbf{\Lambda}(\mathcal{G}, \boldsymbol{\eta}, \boldsymbol{\gamma})$ is given by the set $\overline{\mathbf{\Lambda}_\omega}$, *i.e.*

1. **[Converse]** $\mathbf{\Lambda} \subseteq \overline{\mathbf{\Lambda}_\omega}$.
2. **[Achievability]** $\mathrm{int}(\overline{\mathbf{\Lambda}_\omega}) \subseteq \mathbf{\Lambda}$ and there exists an admissible policy which achieves any rate in $\mathrm{int}(\overline{\mathbf{\Lambda}_\omega})$.

- The proof of the converse follows from the fact that the long-term rate of transmission of packets over a link e is upper bounded by $\omega_e = \min(\eta_e, \gamma_e)$.
- For proving the achievability part, we design a new dynamic key management and packet routing policy - our main focus of this talk.

Recap: Universal Max-Weight Policy (UMW) [Sinha and Modiano 2017]

UMW is a throughput-optimal routing and scheduling policy for generalized traffic classes. However, it does not consider any key availability constraints as in QKD.

Recap: Universal Max-Weight Policy (UMW) [Sinha and Modiano 2017]

UMW is a throughput-optimal routing and scheduling policy for generalized traffic classes. However, it does not consider any key availability constraints as in QKD.

- Competitive against the back pressure policy [Tassiulas, Ephremides, '92]. But more general as it can route broadcast and multicast flows as well.

Recap: Universal Max-Weight Policy (UMW) [Sinha and Modiano 2017]

UMW is a throughput-optimal routing and scheduling policy for generalized traffic classes. However, it does not consider any key availability constraints as in QKD.

- Competitive against the back pressure policy [Tassiulas, Ephremides, '92]. But more general as it can route broadcast and multicast flows as well.
- Instead of taking control actions based on queue lengths $\boldsymbol{Q}(t)$ (closed-loop control), UMW is oblivious to the queues (semi open-loop control).

Recap: Universal Max-Weight Policy (UMW) [Sinha and Modiano 2017]

UMW is a throughput-optimal routing and scheduling policy for generalized traffic classes. However, it does not consider any key availability constraints as in QKD.

- Competitive against the back pressure policy [Tassiulas, Ephremides, '92]. But more general as it can route broadcast and multicast flows as well.
- Instead of taking control actions based on queue lengths $\boldsymbol{Q}(t)$ (closed-loop control), UMW is oblivious to the queues (semi open-loop control).
  - It maintains a virtual queue-length $\tilde{\boldsymbol{Q}}(t)$ vector at the source, which are used for making routing and scheduling decisions.

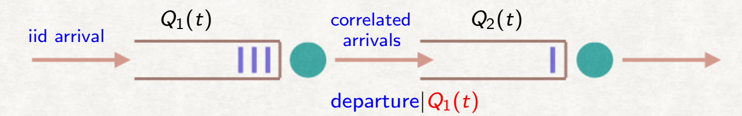
Recap: Universal Max-Weight Policy (UMW) [Sinha and Modiano 2017]

UMW is a throughput-optimal routing and scheduling policy for generalized traffic classes. However, it does not consider any key availability constraints as in QKD.

- Competitive against the back pressure policy [Tassiulas, Ephremides, '92]. But more general as it can route broadcast and multicast flows as well.
- Instead of taking control actions based on queue lengths $\boldsymbol{Q}(t)$ (closed-loop control), UMW is oblivious to the queues (semi open-loop control).
    - It maintains a virtual queue-length $\tilde{\boldsymbol{Q}}(t)$ vector at the source, which are used for making routing and scheduling decisions.
- The virtual queues $\tilde{\boldsymbol{Q}}(t)$ correspond to a precedence-relaxed network.
- UMW uses some standard combinatorial algorithms (e.g., Shortest Path, MST, Steiner Tree, MCDS) on a graph weighted by the virtual queues as a subroutine.
- Instead of making routing decisions for each packet hop-by-hop, UMW dynamically chooses the routes of each packet at the sources itself.
    - Unlike BP, chosen routes are acyclic, which leads to significant delay reduction.

Design of UMW: Motivation and Insight

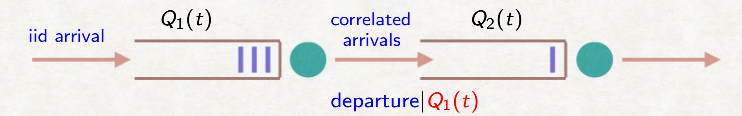
- **Observation:** Because of interdependencies, networked queues are harder to analyze and control.



IID arrivals to Q_1 causes correlated arrivals to Q_2

Design of UMW: Motivation and Insight

- **Observation:** Because of interdependencies, networked queues are harder to analyze and control.



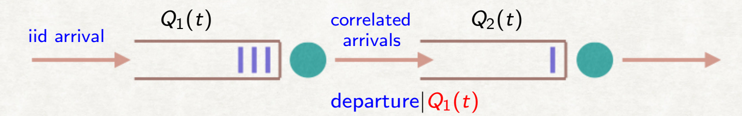
IID arrivals to Q_1 causes correlated arrivals to Q_2

- This motivates us to obtain a relaxed system, which is easier to analyze, yet, preserves some fundamental characteristics we are interested in (e.g., stability).

Question: How to obtain a good relaxation? Which constraints to relax?

Design of UMW: Motivation and Insight

- **Observation:** Because of interdependencies, networked queues are harder to analyze and control.



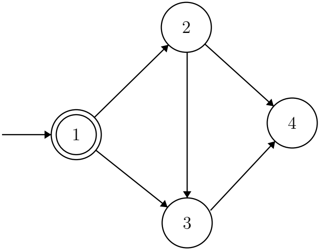
IID arrivals to Q_1 causes correlated arrivals to Q_2

- This motivates us to obtain a relaxed system, which is easier to analyze, yet, preserves some fundamental characteristics we are interested in (e.g., stability).

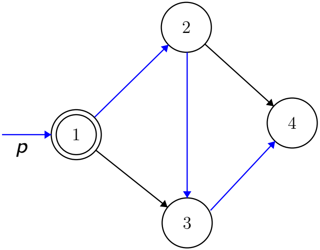
Question: How to obtain a good relaxation? Which constraints to relax?

Ans: The Precedence Constraints!

Precedence Relaxation: Example



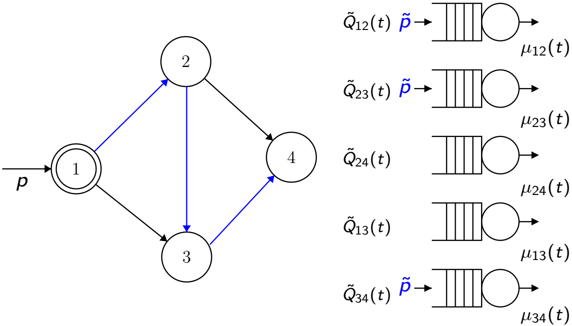
Precedence Relaxation: Example



$\texttt{path}^* = \{\{1,2\},\{2,3\},\{3,4\}\}$

Precedence: The packet p cannot be physically transmitted over the link $2-3$, until it has bee transmitted over the link $1-2$.

Precedence Relaxation: Example



$\texttt{path}^* = \{\{1,2\},\{2,3\},\{3,4\}\}$

Virtual Queues

Virtual Net: Packets are replicated to the virtual queues as soon as they arrive to the sources.

Virtual Queues: Operation

Formally,

1. Associate a virtual queue $\tilde{Q}_l(t)$ with each link l of the graph.
2. Upon packet arrival:
   - Determine a *route* $T^*(t)$ (e.g., path, tree, . . .) for each packet
   - Immediately inject a new virtual packet to each virtual queue along the route
     - This amounts to incrementing the queue counters along the route
3. Serve the virtual packets as long as the corresponding virtual queues are non-empty
   - Don't care whether the physical queue is empty or not.

Virtual Queues: Operation

Formally,

1. Associate a virtual queue $\tilde{Q}_l(t)$ with each link l of the graph.
2. Upon packet arrival:
   - Determine a *route* $T^*(t)$ (e.g., path, tree, . . .) for each packet
   - Immediately inject a new virtual packet to each virtual queue along the route
     - This amounts to incrementing the queue counters along the route
3. Serve the virtual packets as long as the corresponding virtual queues are non-empty
   - Don't care whether the physical queue is empty or not.

Question: How to design a throughput-optimal routing policy: $T^*(t)$?

Dynamics of the Virtual Queues $\tilde{\boldsymbol{Q}}(t)$

- Denote the (controlled) arrival to the VQ $\tilde{Q}_e$ by $\tilde{A}_e(t)$. Then, the virtual queues evolve as:

$$\tilde{Q}_e(t+1) = \left(\tilde{Q}_e(t) + \tilde{A}_e(t) - c_e(t)\right)^+, \quad \text{(Lindley recursion)} \tag{3}$$

- Note that, the arrivals to the virtual queues $\left(\tilde{A}_e(t), e \in E\right)$ are explicit control variables at the source.
- Unlike the original system, given the controls, the virtual queues are independent of each other. This makes the problem tractable.

Routing Policy to Stabilize the Virtual Queues

- Define a Quadratic Lyapunov (potential) function

$$L(\tilde{\boldsymbol{Q}}(t)) \stackrel{\text{def}}{=} \sum_{e \in E} \tilde{Q}_e^2(t)$$

- The one-slot drift of $L(\tilde{\boldsymbol{Q}}(t))$ under any admissible policy π may be computed to be

$$\begin{aligned} \Delta^{\pi}(t) &\stackrel{\text{def}}{=} L(\tilde{\boldsymbol{Q}}(t+1)) - L(\tilde{\boldsymbol{Q}}(t)) \\ &\leq B + 2\Big(\underbrace{\sum_{e \in E} \tilde{Q}_e(t) A(t) \mathbb{1}(e \in T^{\pi}(t))}_{(a)} - \sum_{e \in E} \tilde{Q}_e(t) c_e(t) \Big) \end{aligned}$$

Where $T^{\pi}(t) \in \mathcal{T}$ and $\boldsymbol{\mu}^{\pi}(t) \in \mathcal{M}$ are routing and activation control variables chosen for slot t.

Optimal Routing Policy

Minimizing the term (a), we get the following optimal routing policy.

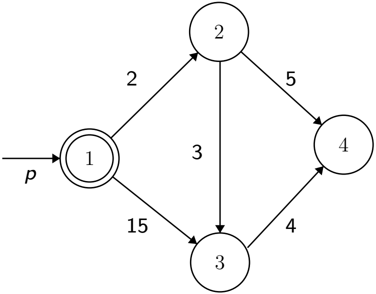
Optimal Routing : $\boldsymbol{T}^*(t)$

$$T^*(t) \in \arg\min_{T \in \mathcal{T}} \sum_{e \in E} \tilde{Q}_e(t) \mathbb{1}(e \in T)$$

Examples:

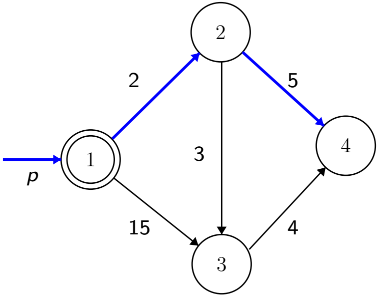
- For the unicast problem : $T^*(t)$ is the Shortest $s \to t$ path in the weighted graph $\mathcal{G}(V, E, \tilde{\boldsymbol{Q}}(t))$.
- For the broadcast problem : $T^*(t)$ is the Minimum Weight Spanning tree (MST) in the weighted graph $\mathcal{G}(V, E, \tilde{\boldsymbol{Q}}(t))$.
- For the multicast problem : $T^*(t)$ is the Minimum Weight Steiner tree in the weighted graph $\mathcal{G}(V, E, \tilde{\boldsymbol{Q}}(t))$ connecting the source nodes to the destination nodes.

Example of Optimal Routing: Unicast



Network Weighted by the Virtual Queue Lengths

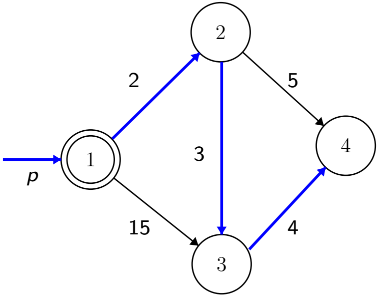
Example of Optimal Routing: Unicast



`Shortest 1-4 path` $= \{\{1,2\},\{2,4\}\}$

Network Weighted by the Virtual Queue Lengths

Example of Optimal Routing: Broadcast



MST rooted at 1 $= \{\{1,2\},\{2,3\},\{2,4\}\}$

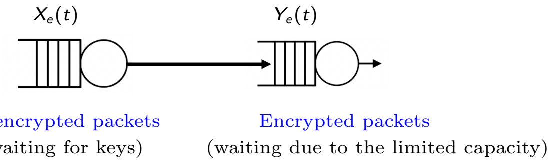
Network Weighted by the Virtual Queue Lengths

Back to QKD: Tandem Queue Decomposition

- The main difficulty in applying the UMW policy in the QKD setting is that not all packets waiting in the queue can be scheduled for transmission over the classical links
  - Only the encrypted packets can be transmitted while the unencrypted packets must wait till additional quantum keys become available

Back to QKD: Tandem Queue Decomposition

- The main difficulty in applying the UMW policy in the QKD setting is that not all packets waiting in the queue can be scheduled for transmission over the classical links
  - Only the encrypted packets can be transmitted while the unencrypted packets must wait till additional quantum keys become available
- This leads to the following natural queueing architecture for each link e
  - Divide the packets waiting to cross the link e in two tandem queues: X_e and Y_e.
  - The first group of packets (in X_e) are unencrypted and wait for the keys to be available
  - The second group of packets (in Y_e) are encrypted and wait due to the limited link capacity



TQD architecture for the link e

Virtual Queue Dynamics

- Let $\kappa_e(t)$ be the number of keys available for encoding packets for link e
  - Note that $\kappa_e(t)$ depends on the routing policy. Hence, the routing policy must balance between the available keys and capacities.

The one-step evolution (Lindley recursion) of the virtual queue processes $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}}$ can be written as:

$$\tilde{X}_e(t+1) = \big(\tilde{X}_e(t) + A_e^{\pi}(t) - \kappa_e(t)\big)^+, \quad \forall e \in E \tag{4}$$

$$\tilde{Y}_e(t+1) = \big(\tilde{Y}_e(t) + A_e^{\pi}(t) - \gamma_e\big)^+, \quad \forall e \in E. \tag{5}$$

Since a packet is encrypted as soon as the keys become available, we have

$$\tilde{X}_e(t)\kappa_e(t) = 0.$$

- $\kappa_e(t)$ is otherwise a complex process.

TQD Policy (informal)

Apply the UMW policy in the above transformed network with twice as many queues.

Stabilizing Routing Policy

Similar to the UMW policy, the drift of the quadratic Lyapunov function of the virtual queues may be bounded as:

$$\begin{aligned}\Delta^{\pi}(t) &\equiv \mathbb{E}\Big(L(\tilde{Q}(t+1)) - L(\tilde{Q}(t))|\tilde{\boldsymbol{Q}}(\boldsymbol{t})\Big) \\ &\leq B + 2\sum_{e\in E}(\tilde{X}_e(t)+\tilde{Y}_e(t))\mathbb{E}(A_e^{\pi}(t)|\tilde{\boldsymbol{Q}}(\boldsymbol{t})) \\ &\quad - 2\sum_{e\in E}\tilde{X}_e(t)\eta_e - 2\sum_{e\in E}\tilde{Y}_e(t)\gamma_e, \end{aligned} \tag{6}$$

where B is a finite constant that depends on the upper bounds of the packet arrival and quantum key generation rates.

TQD Algorithm

Algorithm 1 Tandem Queue Decomposition (TQD) algorithm

1: **[Weight Assignment]** Assign each edge of the original graph $e \in E$ a weight $W_e(t)$ equal to $\tilde{X}_e(t) + \tilde{Y}_e(t)$, i.e

$$\boldsymbol{W}(t) \leftarrow \tilde{\boldsymbol{X}}(t) + \tilde{\boldsymbol{Y}}(t).$$

2: **[Route Assignment]** Compute a Minimum-Weight Route $T^{(c)}(t) \in \mathcal{T}^{(c)}(t)$ for a class c incoming packet in the weighted graph $\mathcal{G}(V, E)$.

3: **[Key Generation]** Generate symmetric private keys for every edge e via QKD and store them in the key banks.

4: **[Encryption]** Encrypt the data packets waiting in physical queue X_e with the available keys and move the encrypted packets to the downstream queue Y_e.

5: **[Packet Forwarding]** Transmit the encrypted physical packets from the queue Y_e according to some packet scheduling policy (*e.g.*, ENTO, FIFO etc).

6: **[Decryption]** Decrypt the data packets received at physical queue X_e for every edge e using the symmetric key generated earlier via the QKD process.

7: **[Queue Counter Updation]** Update the virtual key queues and virtual data queues assuming a precedence-relaxed system.

Strong Stability

Theorem 2 (Strong Stability of the Virtual Queues):

Under the TQD routing policy, the virtual queue process $\{\tilde{\boldsymbol{Q}}(t)\}_{t\geq 0}$ is strongly stable for any arrival rate vector $\boldsymbol{\lambda} \in \text{int}(\overline{\boldsymbol{\Lambda}})$, *i.e.*,

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{e\in E} \mathbb{E}(\tilde{X}_e(t) + \tilde{Y}_e(t)) < \infty$$

Strong Stability

Theorem 2 (Strong Stability of the Virtual Queues):

Under the TQD routing policy, the virtual queue process $\{\tilde{\boldsymbol{Q}}(t)\}_{t\geq 0}$ is strongly stable for any arrival rate vector $\boldsymbol{\lambda} \in \text{int}(\overline{\boldsymbol{\Lambda}})$, *i.e.*,

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{e\in E} \mathbb{E}(\tilde{X}_e(t) + \tilde{Y}_e(t)) < \infty$$

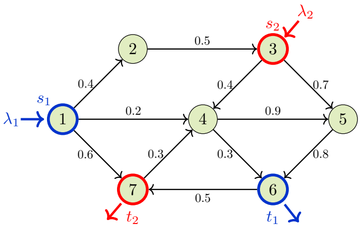
Theorem 3: Rate Stability of the physical queues

Under the action of a suitable packet scheduling policy, that physical queues are rate stable, *i.e.*,

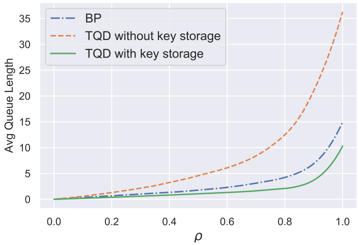
$$\lim_{t\to\infty} \frac{Q_e(t)}{t} = 0, \quad \forall e \in E, \qquad \text{a.s.}$$

Proof involves an adversarial queueing argument using the specific packet scheduling policy.

Simulation - Unicast Traffic

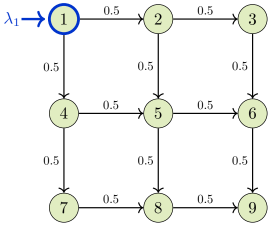


Graph used in the unicast setting
Multicommodity traffic with 2 classes of data packets.
$\lambda_1 + \lambda_2 \leq 1.1$

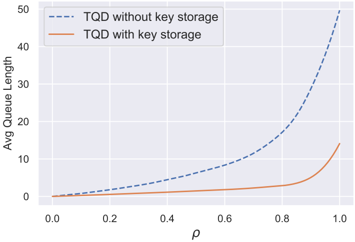


Performance comparison between the **TQD** policy (with and without key storage) and the **Back pressure** policy in the unicast setting

Simulation - Broadcast Traffic



Graph used in the broadcast setting
Broadcast capacity = 0.5



Delay Performance of the **TQD** policy for broadcast traffic.

Our team is currently building a realistic simulator using OMNeT++ platform.

Thanks

Thank You!

I am reachable at: abhishek.sinha at ee dot iitm dot ac dot in